

si(x) in Maxima (Was: Mathematica vs. Lisp)

Source: <http://sci.tech-archive.net/Archive/sci.math.symbolic/2004-06/0196.html>

From: Albert Reiner (areiner_at_tph.tuwien.ac.at)

Date: 06/22/04

Date: 22 Jun 2004 11:45:22 +0200

To: Richard Fateman <rfateman@sbcglobal.net>

Richard Fateman <rfateman@sbcglobal.net> writes:

- > *There is a tradeoff in computer algebra systems "The good of the*
- > *one outweighs the good of the many." to reverse Spock's quote.*
- > *Do you want a/b to be (quotient $a b$) or do you want to minimize the*
- > *proliferation of "kernels" like quotient, and use (times a (expt b*
- > *-1)) ? which is wordier, but re-uses kernels that you can't really*
- > *get rid of.*
- > *Si(x), the sine integral is one of those kernels that you can pretty*
- > *much get rid of: just use the integral. In some cases you want to*
- > *pander to the common usage. e.g. sine and cosine could be expressed*
- > *in terms of each other or in terms of complex exponentials, but are*
- > *so familiar, they must all coexist somehow.*

Of course, but in my case si and ci (or actually some related functions called, I think, f and g in Abramowitz–Stegun) are just as basic as sin and cos, simply because I have an efficient numerical implementation for them to use in later processing.

- > *But I think the original question was really about the*
- > *programming language issues, not the mathematical capabilities*
- > *of lisp...*

Actually I am no longer talking about the OP's question but rather about something related to my own work (involving 3-dimensional Fourier transforms of interaction potentials in simple liquids with cutoffs in both k - and r -space), where the symbolic end result I need to get should be in terms of si, ci (or the f and g mentioned above) and I also need the small- k series expansions of those expressions, all of these written in a way suitable for numerical evaluation. The types of expressions that arise are pretty limited, and their numerical properties are well understood, so one basically knows what form one needs to produce.

In the past I just did all of this by hand, with my good old Gradstein–Ryzhik by my side, but this is barely feasible with the scheme to be considered now. Also, I would like to be able to have

those manipulations automated for the standard potentials in liquid state physics; note that I do not really need `integrate()` to do those things automatically, I only need to arrive at a procedure that will yield the end result in a usable form, checking the validity of the transformations on the way, and telling me if I need to expand that procedure.

If I were doing this in Mathematica, I wouldn't rely on `Integrate` either (branch cuts!) but rather work with replacement rules; but then again, Mathematica is terrible for scripting, and it seems I cannot use `mash`, <http://ai.eecs.umich.edu/people/dreeves/mash/>, at this place.

My idea was that by using Maxima I could get the basics of those manipulations, simple simplifications in particular, for free, that I would be able to do what I need with replacements (I see `subst` and `substpart` etc. in the manual), and that it would integrate well into the rest of the software if written in Common Lisp; actually, this was the primary reason for considering CL at all.

BTW, does anyone have an example of using a Maxima result for a lambda body? I am thinking of a function that returns a function for evaluating the maxima result numerically. E.g., given a representation of $\cos(x)$, such a function should compute the derivative and return

```
'(lambda (x)
  (declare (type long-float x))
  (- (sin x)))
```

On a related note, does anyone know how close the Maxima representation of algebraic expressions is to that of CL or Fortran?

Albert.