

# Re: Symbolic Math as Computer Science

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math.symbolic/2005-12/msg00131.html>

---

- *From:* Hatto von Aquitanien <[abbot@xxxxxxxxxxxxxxxx](mailto:abbot@xxxxxxxxxxxxxxxx)>
  - *Date:* Thu, 08 Dec 2005 20:28:24 -0500
- 

Richard Fateman wrote:

> Rouben Rostamian wrote:

>> In article <o9OdnWTfmrYsqQzenZ2dnUVZ\_v-dnZ2d@xxxxxxxxxxxxxxxx>,

>> Hatto von Aquitanien <[abbot@xxxxxxxxxxxxxxxx](mailto:abbot@xxxxxxxxxxxxxxxx)> wrote:

>>

>>>Is there much in the way of academic computer scientific literature on

>>>the

>>>mechanisms underlying symbolic computing? I don't (believe I) mean

>>>\_Modern

>>>Computer Algebra\_ by von zur Gathen and Gerhard. I'm really thinking in

>>>terms of implementation along the lines of compiler theory, and formal

>>>language theory.

>>>

>>>My limited experience with such systems is pretty much exclusively with

>>>Mathematica. Does anybody happen to know of literature that reveals how

>>>Mathematica works at the fundamental level?

>>

>>

>> You will find the following useful:

>>

>> Peter Norvig: Paradigms of AI Programming.

>>

> You might also find Abelson/Sussman: Structure and INterpretation of

> Computer Programs, esp. the section on data abstraction and polynomials

> to relate "computer science" data strutures to symbolic math.

>

> There is also Knuth's Art of Computer Programming, volume 2.

>

> You will find lots of details in the book

> Algorithms for Computer Algebra by K. O. Geddes et al.

>

> I think you may also find some stuff of interest to you in the literature

> on the Axiom system, or Maxima. (which has an extensible parser for

> the compiler buffs).

>

> But mostly compiler theory and formal language theory have to do with only

> about 5% of the symbolic math system-building task. ..How to build a

> parser or maybe even a 2-d handwriting recognizer front end that

## Re: Symbolic Math as Computer Science

- > can handle some subset of "math". I assume you are also interested
- > in data structures and representation and maybe things like
- > object-oriented programming etc by way of "academic CS".
- >
- > Maybe you could clarify your question?

There are probably many levels worth exploring. The two which seem most relevant are the typesetting level, and the "internal" functional level.

At one level we have the "traditional form" of mathematical notation we find in the Mathematica Front End (what I'm calling the typesetting level.) That is, mathematics expressed using symbols such as +, -, =, /, etc., as well as superscripts, subscripts, overscripts, etc. All that is of secondary interest to me at this point.

I am more interested in strictly prefix functional representation of mathematics (Mathematica full form.) I believe that it is possible to transform any mathematical expression into such a form without loss of functionality. In this respect, I'm not necessarily concerned with Mathematica per se. I have the sense there is some minimal set of "axiomatic" constructs upon which all mathematics could be constructed. I know that statement begs for clarification and elaboration. But for now, I will leave it as it stands.

In my view there are really only a small number of basic ideas in mathematics. They are all founded in our naive notions of left-right, front-back, up-down, past, present, future, before, after, between, contig, measuring, true, and false. From such basic ideas, I believe we can determine a set of fundamental entities which can be used to construct all mathematical ideas.

- > If you want to see how to program Mathematica, you can look at an
- > open-source version of a mockup of it, the front end parser,
- > simplification, display, a few commands (e.g. D[], Integrate[]) in
- > MockMMA.

I may take a closer look at this. Ironically, my experience has been that Mathematica code tends to be easier to understand than (Emacs) Lisp.

- > If you want to know some of the "secrets" inside Mathematica,
- > see my review of it in J. Symbolic Computation (version is also online).

Am I correct in understanding that to be the paper dated November 1991? I've read part of it. I wouldn't describe it as flattering to Mathematica. I understand that it is a critical review, and can expect this to some extent. Some of the criticisms are a bit pedantic/nit-picky from the perspective of a "real user". OTOH, Mathematica is very difficult to learn to use well, and it certainly has its share of astonishing features.

I believe the reluctance to expose the concepts which make it work is a big contributor to the difficulty in learning to use the product.

—  
Nil conscire sibi  
.

---

- **Follow-Ups:**
  - ◆ **Re: Symbolic Math as Computer Science**  
◇ From: Richard J. Fateman
- **References:**
  - ◆ **Symbolic Math as Computer Science**  
◇ From: Hatto von Aquitanien
  - ◆ **Re: Symbolic Math as Computer Science**  
◇ From: Rouben Rostamian
  - ◆ **Re: Symbolic Math as Computer Science**  
◇ From: Richard Fateman
- Prev by Date: **Re: Maple Vs Mathematica debugging / cost to write a system**
- Next by Date: **Re: complexity of numerical software**
- Previous by thread: **Re: Symbolic Math as Computer Science**
- Next by thread: **Re: Symbolic Math as Computer Science**
- Index(es):
  - ◆ **Date**
  - ◆ **Thread**