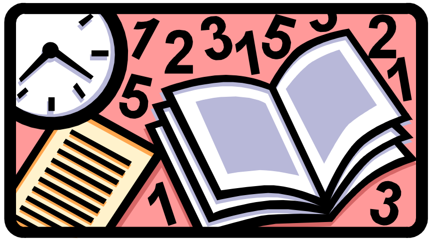
Algebraic Algorithms: CS 282
Spring, 2006

Lecture 1

The subject: “Symbolic Computation”

- Computer algebra systems (CAS) and their supporting algorithms for performing symbolic mathematical manipulation.
- Math surprises: Can you program, or make constructive, various more-or-less well-known symbolic computations?
- Computer Science tasks: Can you build a mathematical intelligence? Or at least a skilled assistant?
- How hard are these tasks (Asymptotic complexity, actual benchmarks)

An aside on your non-constructive education



In freshman calculus you learned to integrate rational functions. You could integrate $1/x$ and $1/(x-a)$ into logarithms, and you used partial fractions. Unless you've recently taken (or taught) a calculus course, you've forgotten the rest of details.

Here's an integration problem

$$\int \frac{1}{x^3 - 6\,x^2 + 11\,x - 6}\,dx$$

Fortunately you can factor the denominator this way (by guesswork, perhaps)

$$x^3 - 6\,x^2 + 11\,x - 6 = (x - 3)\,(x - 2)\,(x - 1)$$

And then do the partial fraction expansion

$$\frac{1}{x^3 - 6\,x^2 + 11\,x - 6} = \frac{1}{2\,(x-1)} - \frac{1}{x-2} + \frac{1}{2\,(x-3)}$$

And then integrate each term…

$$\int \frac{1}{2\,(x-1)} - \frac{1}{x-2} + \frac{1}{2\,(x-3)}\,dx =$$

$$\frac{\log(x-1)}{2} - \log(x-2) + \frac{\log(x-3)}{2}$$

Non-constructive parts

Do you really know an algorithm to factor any denominator into linear and quadratic factors?

- Can you do this one, say…

$$x^4 - q^2 + 2\,p\,q - p^2 = \left(x^2 - q + p\right)\,\left(x^2 + q - p\right)$$

- And if it does not factor (it need not, you know… how do you proceed to integrate the function?)

If the denominator doesn't factor

$$\int \frac{1}{x^3 - a}\,dx = -\frac{\log\left(x^2 + \sqrt[3]{a}\,x + a^{2/3}\right)}{6\,a^{2/3}} - \frac{\arctan\left(\frac{2\,x + \sqrt[3]{a}}{\sqrt{3}\,\sqrt[3]{a}}\right)}{\sqrt{3}\,a^{2/3}} + \frac{\log\left(x - \sqrt[3]{a}\right)}{3\,a^{2/3}}$$

And it gets worse … there is no guarantee that you can even <u>express</u> the roots of irreducible higher degree polynomials in radicals.

Moral of this story

- You were not taught how to integrate rational functions. Only some rational functions.
- Writing a program to (say) factor or integrate uses ideas you may have never seen before.

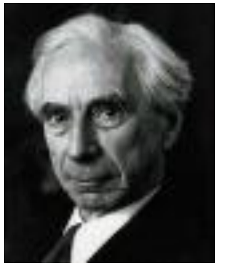
End of aside

Some History: Ancient

- Ada Augusta, 1844 foresaw prospect of non-numeric computation using Babbage's machines. Just encode symbols as numbers, and operations as arithmetic.

Some History: Less Ancient

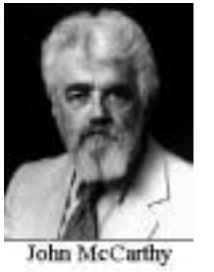
- Philosophers/Mathematicians, e.g. G. Frege, but best known: B. Russell, A.N. Whitehead (*Principia Mathematica* 1910-1913)



Some History: No, you can't do it all…

- Kurt Gödel, Alan Turing

Some History: New optimism?



1958-60 first inklings .. automatic differentiation, tree representations, Lisp,

- Minsky ->Slagle, (1961), Moses(1966); Is it AI?

Computer Algebra Systems : threads

- Three trends emerged in the 1960s:
  - AI / later…expert systems
  - Mathematics e.g. Berlekamp [factoring] , Liouville-> Risch [integration], computational group theory
  - Algorithms / Computer Science e.g. Knuth/Brown/Collins [polynomial GCD]

Some Historical Systems

- Early to mid 1960's - big growth period, considerable optimism in programming languages, as well as in computer algebra…
- PLs.. Fortran, Algol, Lisp, COBOL all rather new.
- - Mathlab, Symbolic Mathematical Laboratory,
- Formac, Formula Algol, PM, ALPAK, Reduce; Special purpose systems,
- Optimism about conquering all of math by coming up with the right programming formalism, and accumulating “facts”.

Mathematics' flirting with computing..

- Constructive algorithmic algebra was fashionable in the early 20th century (early editions of van der Waerden's classic "Modern Algebra" book), but existence proofs became more popular. Too bad. I think the tide is turning towards constructive approaches.

Some theory/algorithm breakthroughs

- 1967-68 algorithms: Polynomial GCD,
- Berlekamp's polynomial Factoring,
- Risch Integration "near algorithm",
- Knuth's Art of Computer Programming vol2
- 1967 - Daniel Richardson: interesting zero-equivalence results
- Emergence of Gröbner bases calculations for polynomial system solving and related probs.

Some well-known systems / timelines

- Computers got comparatively cheaper, so systems get more ambitious, more available (1968-78)
  - SAC-1, Altran, Macsyma, Scratchpad.
  - Mathlab 68, MuSimp/MuMath, SMP, Automath, others.
- Further development; new entrants of 1980's
  - Maple, Mathematica (1988), Derive, Axiom, Theorist, Milo,
- Consolidation: 1990s improving existing systems,
  - new experimental systems (theorem proving, niche math)
- 2000++ C++ overloaded libraries:
  - Faster
  - More like the systems of the 1960s.

Some support systems

- Common Lisp gets standardized.
- Scheme gets standardized too.
- C++ popularized as “the answer”
- Portability (UNIX™? Linux, Windows, Apple)
- Java
- HTML, XML, MathML and Browsers

The Marketing Blitz and shakeout

- Mathematica, NeXT, Apple, graphics.
- Maple comes out from under a rock.
- IBM/Scratchpad goes public as Axiom under NAG sponsorship, then is killed. (2001)
- MuPad at Univ of Paderborn, is free, then sold.
- Macsyma goes into hiding, parts come out free as Maxima on sourceforge.net – STUDENT PROJECTS?
- Openmath and MathML put "Math on the Web".
- Connections.
  - Links from Matlab to Maple,
  - Scientific Workplace to Maple or Mathematica.
- The arrival of network agents for problem solving.
  - Calc101, Tilu, TheIntegrator, Ganith, ...

Are there really differences in systems?

- What we see today in systems (crude characterizations):
  - Mathematica essentially takes the view that all mathematics is a collection of rules with a procedure for pattern matching; and that a system needs neat graphics for Marketing.
  - Axiom takes the view that a computer algebra system is an implementation of Modern Algebra.
  - Almost everyone concedes that good algorithms and data structures are necessary for effective, efficient computation; sometimes Math takes a back seat.

Next time

- What do these CAS and the many systems we haven't explicitly mentioned, have in common?
- Algebraic systems
  - Objects
  - Operations
  - Properties? Axioms?
  - Extensions to a base system (programming? Declarations?)
  - Underlying all of this: <u>efficient representations</u>