Sage-Symbolics
Making a new system for performing Calculus and Physics

Gary Furnish

August 6, 2008

Introduction

Introduction

- Sage needs a strong system for performing calculus in order to effectively compete with Mathematica and Maple.

Introduction

- Sage needs a strong system for performing calculus in order to effectively compete with Mathematica and Maple.
- Sage currently uses Maxima through a pexpect interface.

Introduction

- Sage needs a strong system for performing calculus in order to effectively compete with Mathematica and Maple.
- Sage currently uses Maxima through a pexpect interface.
- However pexpect is slow, especially for performing numerous small calculations.

Introduction

- Sage needs a strong system for performing calculus in order to effectively compete with Mathematica and Maple.
- Sage currently uses Maxima through a pexpect interface.
- However pexpect is slow, especially for performing numerous small calculations.
- It is hard to extend Maxima as it is written in lisp.

Design Goals

Design Goals

- Sage-Symbolics should be very fast.

Design Goals

- Sage-Symbolics should be very fast.
- Sage-Symbolics should be maintainable.

Design Goals

- Sage-Symbolics should be very fast.
- Sage-Symbolics should be maintainable.
- Sage-Symbolics should present a good platform to build more complicated symbolic algorithms off of.

Design Goals

- Sage-Symbolics should be very fast.
- Sage-Symbolics should be maintainable.
- Sage-Symbolics should present a good platform to build more complicated symbolic algorithms off of.
- Most importantly, it must be easy to use.

More Design Goals

- However, given an opportunity to start from scratch, we have a rare chance to do it right.
- Sage has a great mathematical type system (Coercion).
- It contains a built in knowledge of Rings, Modules, etc.
- We can use this to our advantage to design a significantly more powerful symbolic manipulation platform.

New Frontier

- Mathematica and Maple don't have something analagous to Coercion.
- Poor native differential geometry support in most general purpose CAS's.
- No easy way to do noncommutative symbolics.
- No way to add new operations as first class objects.

New Frontier

- Mathematica and Maple don't have something analagous to Coercion.
- Poor native differential geometry support in most general purpose CAS's.
- No easy way to do noncommutative symbolics.
- No way to add new operations as first class objects.
- Should we care?

Physics

- Quantum Field Theory – Indexed and Tensorial expressions
- Quantum Mechanics – Noncommutative Expressions
- General Relativity – Differential Geometry
- Needs are only served by special case programs or code.
- No general purpose enviroment for all needs.

More Design Goals

- Noncommutative symbolic manipulations the natural starting point.
- Commutative symbols are a special case
- Calculus is just a small fraction of what we have to support
- Support for arbitrary types of symbols... let X be a matrix
- Still has to be fast.

Progress

- Noncommutative operations “just work”
- So do most calculus operations
- Native support for unevaluated functions.
- Native derivation
- Symbolic Matricies (but no RREF yet)
- Global Non-recursive pattern matching
- Fast (But it could be even faster)

Maxima Interface

- Still using Maxima for complicated operations
- Integrals, Factorization, Summation, Laplace
- Assumptions don't work yet, but are getting there.
- The Maxima interface is faster then it used to be.
- Unevaluated functions work better

- f = 5*x*y*z +y**10*x
- expand(f+int(10000*random())) * (f+int(10000*random())))
- Sympy: 12.9 ms Symbolics: 4.25 ms Maxima: 57.4 ms

- f = 5*x*y*z +y**10*x
- expand(f+int(10000*random())) * (f+int(10000*random())))
- Sympy: 12.9 ms Symbolics: 4.25 ms Maxima: 57.4 ms
- expand(f*(f+1) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())))
- Sympy: 76.4 ms Symbolics: 41.4 ms Maxima: 47.1 ms

- expand(f*(f+1) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random()))
- Sympy: 379ms Symbolics: 113 ms Maxima: 93ms

- expand(f*(f+1) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random()))
- Sympy: 379ms Symbolics: 113 ms Maxima: 93ms
- expand(f*(f+1) * (f+int(10000*random())) * (f+int(10000 * random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())))
- Symbolics: 171ms Maxima: 126 ms Mathematica: 4ms

- expand(f*(f+1) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random()))
- Sympy: 379ms Symbolics: 113 ms Maxima: 93ms
- expand(f*(f+1) * (f+int(10000*random())) * (f+int(10000 * random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())) * (f+int(10000*random())))
- Symbolics: 171ms Maxima: 126 ms Mathematica: 4ms
- Maxima through Sage
- Sympy with caching enabled
- Sympy with caching disabled performs really poorly
- Sympy Core faster then symbolics right now*

Analysis

- Profilers: Memory initialization expensive
- Use pools, help some via TPALLOC
- Real problem is cython autogenerated TPNEW
- Modify Cython to emit better code and link symbolics at once
- Alternatively hand written C TPNEW functions

Analysis

- Noncommutative algebra detection code
- Solution: Seperate Noncommutative and Commutative multiplication classes

Analysis

- Noncommutative algebra detection code
- Solution: Seperate Noncommutative and Commutative multiplication classes
- Excessive memory creation:
- Change multiplication class for commutative rings to store constant seprately
- Change multiplication classes to store powers without an additional class

Analysis

- Not a flaw in the design – a consequence of wanting noncommutative symbolics from the start
- Can be fixed without too much trouble
- 1-3 order of magnitude speedup should be possible from these changes.

Near Term Goals (Next month)

Near Term Goals (Next month)

- Write enough of an assumption engine so that Maxima assumptions work again (necessary for many integrals)
- Finish trig default simplifications
- Minimal piecewise function support (to current level of support)
- Symbolic polynomials should use libSingular
- Seperate out noncommutative and commutative cases for multiplication
- Optimize memory creation overhead if necessary for merge
- Write doctests, start formal review process

Future Plans

- Switch all simplification to new pattern matching engine
- Full differential geometry support
- Optimize memory creation overhead (running theme)
- More advanced algorithms for addition/multiplication?
- Basic integration algorithms

Demoes

Now I will show some demoes of what I have done.

Questions

Questions?