SAGE: Current Project Status Report

William Stein

November 8, 2006, San Diego

This talk is an (incomplete) overview of several **current** SAGE development projects, many of which started as coding sprints at SAGE Days 2.

Review: The Overall Structure of SAGE

- **Custom package management system** – 42 standard packages, and 29 optional ones. Automated upgrades.
- **Interactive command-line** interface – IPython.
- **Graphical user interface** – SAGE Notebook via web browser.
- **Fast underlying arithmetic** – built on mature robust C libraries (GMP, NTL, PARI, GSL). New code in SageX[1] and Python.
- **Interfaces with other software** using buffered **psuedo-tty**'s.
- **Special purpose components** – e.g., **genus2reduction**, **GMP-ECM**, **Sympow** (L functions), **Givaro** (finite fields), etc.
- **Mercurial revision control system** – included standard; makes it very easy for users to make changes, add docs, etc., and give them to me.

[1] A variant of Pyrex

```
cddlib-094b.spkg
clisp-2.40.spkg
conway_polynomials-0.1.spkg
cremona_mini-0.1.spkg
doc-1.4.3.alpha2.spkg
ecm-6.0.1.p0.spkg
examples-1.4.3.alpha2.spkg
extcode-1.4.3.alpha2.spkg
freetype-2.1.10.spkg
gap-4.4.8.spkg
genus2reduction-0.3.spkg
gfan-0.2.2.spkg
givaro-3.2.1.spkg
gmp-4.2.1.p1.spkg
gnuplotpy-1.7.p1.spkg
gsl-1.8.spkg
ipython-20061028.spkg
lcalc-2006.09.19.spkg
libpng-1.2.8.p0.spkg
matplotlib-0.87.6.spkg
maxima-5.10.0.spkg
mercurial-0.9.1.p2.spkg
mpfr-20061015.spkg
mwrank-20061107.spkg
networkx-0.32.spkg
ntl-5.4.1.spkg
numeric-24.2.spkg
pari-2.3.1.spkg
pexpect-2.0.spkg
pyrexembed-0.1.1.2006-05-17
python-2.5.p2.spkg
readline-5.0.1.spkg
sage-1.4.3.alpha2.spkg
sage_scripts-1.4.3.alpha2.s
sagex-20061103.spkg
singular-3-0-2-20061014.spk
sympow-1.018.1.spkg
tachyon-0.97.spkg
termcap-1.3.1.spkg
twisted-2.4.0.p1.spkg
zlib-1.2.3.p1.spkg
zodb3-3.6.0.spkg
```

Some Components of SAGE (by category)

Basic Arithmetic	**GMP, NTL, MPFR, PARI**
Command Line	**IPython**
Commutative algebra	**Singular** (libcf, libfactory)
Database	**ZODB**, Python Pickles
Graphical Interface	**jsmath, SAGE Notebook**
Graphics	**Matplotlib, Tachyon**
Group theory and combinatorics	**GAP**
Graph theory	**Networkx**
Interactive programming language	**Python** (mainstream !!!)
Networking	**Twisted**
Numerical computation	**GSL, Numeric, etc.**
Symbolic computation, calculus	**Maxima**

Interfaces

- SAGE **interfaces to**: Axiom, GAP, GP/PARI, Kash, Macaulay2, Magma, Maple, Mathematica, MATLAB, Maxima, Octave, Singular, etc.
- In progress: REDUCE (Bill Page), 4ti2 (Stein and Tristram Bogart), PHCpack (Stein).

Generator Names and Global Uniqueness

William Stein

- In SAGE all **parent structures** will (generally speaking) be **immutable**. In particular, variable names of polynomial rings, finite fields, power series rings, etc, must be specified at creation time and can't be changed later.
- Most **ring elements** are now immutable (e.g., integers, polynomials, power series, etc.)
- There is exactly one instance of each parent object.
- This and other low-level optimization helps **makes basic arithmetic much more efficient**.

Arithmetic architecture and Coercion

William Stein, D. Harvey, M. Albrecht (Bremen grad)

- **OBJECT COERCION `_coerce_`:** Suppose a `_coerce_` map $R \to S$ is defined. Then:
  1. `R.category()` must be a **subcategory** of `S.category()`.
  2. The map $R \to S$ defined by coerce must define a **morphism** in `S.category()`.
  3. If `_coerce_` is defined in **both direction**, then the composition in both directions must be the identity maps.
  4. **Reflexive:** If `R is S` is True, then `_coerce_` must be the identity map.
  5. **Transitive:** If coercion from R to S is defined and coercion from S to T is defined, then coercion from R to T must also be defined, and must agree with the composition of the coercion from R to S with the one from S to T.
- **ARITHMETIC `__add__`, `__mul__`, ...:** When doing a binary operation, if the parents are not identical (in the sense of is), determine if **precisely one** `_coerce_` map is defined; if so, apply it and do the arithmetic operation. If **both** are defined, the parents are canonically isomorphic, so use the left one. If **neither** are defined, raise a TypeError.

Matrix algebra

W. Stein, R. Bradshaw (UW grad), D. Harvey

- Matrix classes **systematically structured** and completely **implemented in SageX**.
- Easy to create new optimized matrix classes (over specific rings).
- Much work is about organization and providing a wide range of functionality that is built on a couple of basic algorithms.
- Robert Bradshaw and David Harvey came up with and completely implemented optimized **asymptotically fast algorithms** for matrix multiplication and echelon forms in the general case (arbitrary size matrices). Tuning still needed.

Numerical mathematics

W. Stein, Josh Kantor (UW grad), Tom Boothby (UW undergrad)

- **Numerical computation** is **extremely important** for SAGE:
  - Numerical algorithms are deeply relevant to algebraic and geometry computation (it's a major current research trend),
  - There is a large numerical applied group at UW.
  - The Python community has a large mature package of numerical software (numpy, scipy).
- One can use **numpy and scipy** from SAGE easily now.
- We are creating native SAGE classes for numerical objects (e.g., matrices, vectors, ODE's, double precision real and complex numbers, etc.) built on top of **GSL** – the GNU Scientific Library.

Graph theory

Emily Kirkman (UW undergrad), Robert Miller (UW grad), Bobby Moretti (UW undergrad)

- Emily, Robert, and Bobby did a massive survey of all graph theory software they could find (both free and commercial).
- **Their rough conclusions**:
  - The **Maple** and **Mathematica** graph theory packages are slow.
  - **MAGMA** is *incredibly fast* at graph theory, and has a wide range of computational functionality. No visualization.
  - The best "all around" free package, at least for what most of our users wanted, is **NetworkX**, which is a Los Alamos project that is conveniently written in Python.
- The students are working on making Networkx **integrate nicely** with the rest of SAGE, e.g., graphs attached to matrices, groups, combinatorial structures, Hecke operators, etc.

Integer Factorization

William Hart, Robert Bradshaw (UW grad), Yi Qiang (UW undergrad)

- **Bill Hart** (a young Australian number theorist working in England) just finished writing an optimized quadratic sieve for integer factorization. He **GPL'd** it and is helping us **include it in SAGE**. (He is now working on core arithmetic optimization for SAGE, e.g., very fast polynomial arithmetic.)

  > *"It takes* ***15s*** *on SAGE.math [with my sieve] for a C61. Note that PARI on SAGE.math compiled against the latest 64 bit GMP takes* ***54s*** *for the same computation. MAGMA takes around* ***72s****, but I forgot, it spends some time in GMP-ECM. Around* ***63s*** *is spent in MPQS, which is not that far behind Pari I guess."*

- Robert Bradshaw and Yi Qiang: Improve integration of **GMP-ECM** into SAGE; make distributed computation using GMP-ECM from SAGE easy.

Modular Forms

Me, Ifti B. (USC), David K. (Sydney), Jordi Q. (sabbatical at UW)

- Ifti Burhanuddin, David Kohel, and I – implemented the **Mestre method of graphs**; need to optimize.
- Jordi Quer is visiting me at UW this quarter – will implement general congruence subgroups; extend **modular symbols computations**.

The SAGE Notebook

Alex Clemesha (was a UCSD undergrad), Tom Boothby (UW undergrad), Dorian Raymer (UCSD physics), Bobby Moretti (UW undergrad)

- **Automated testing** (Alex C.) – he just implemented a system that records all input to the notebook, can play it back, check that results agree.
- **Security** (Tom B.) – plan to move to SSL and/or Twisted.
- **More Wiki-like functionality** (Tom, Alex, Dorian) – easier editing of pages, markup between compute cells, tracking of all versions of a worksheet.
- **Special purposes apps** – online quiz system for college teachers, specialized web sites that run SAGE behind the scenes.

SAGE Foundation: What is the purpose of SAGE?

- Be a comprehensive mainstream high quality open source free mathematics software system.
- Unify free open source mathematics software.
- To provide everyone (students, computer scientists, professional, ...), with stimulating, educational, high quality, open source, mathematical software for learning about and producing research in mathematics, at no cost.

Why does the SAGE Mathematics Foundation exist?

- Be a **not-for-profit** tax-exempt organization under **Section 501(c)(3)** of the IRS tax code. Can receive donations, license fees, payment for technical support, etc. Resulting money can then be used to support students, visitors, purchase of equipment, workshops, and give grants to applicants for SAGE development.
- Provide an **advisory board** to help people applying for grants (e.g., from NSF), for conferences, and deciding how to use funds they have. (First board: David Joyner, William Stein, A grad student (to be determined), Tom Boothby (UW undergrad).)
- Protect SAGE developer's **intellectual property rights**. **Copyright will stay with authors**; all coded submitted must be under the **GPL** (or compatible) license.
- **Trademark** and protect the SAGE name.
- To improve the **accessibility of mathematics** for everyone with a computer.
- To constantly improve the **interactive exploratory experience** available for anyone to learn about or perform research in mathematics using the SAGE program.

What is the Foundation going to do to achieve this purpose?

- Run **workshops**.
- Create an **advisory board** of directors.
- **Make available** on the internet, at no cost to the user, the SAGE program and extensive documentation.
- **Strongly encourage** SAGE developers, **funding** and/or training them if fiscally possible.
- Support SAGE end users by **hiring user support staff** which fixes reported bugs as soon as possible or offering work-arounds, offers programming advice, provides requested functionality, when possible.

What are our guiding principles?

- **All software** included in the SAGE core distribution must be **free and open source**, and arbitrary modifications and redistribution of every single line must be allowed.
- We should provide a **model for the mathematical community** of software development with a strong emphasis on **openness, community, cooperation, and collaboration**.
- We should always strive to create **professional quality software** and documentation that is available to everyone. That software must be **high quality, accessible, open source, and free** for everyone to download and use at no cost.
- We strive to provide an **encouraging, stable, productive, programming environment** for developing future mathematical programming projects.