A SAGE Introduction

http://www.sagemath.org

Martin Albrecht (M.R.Albrecht@rhul.ac.uk)

Samos, 4. Mai 2007

Table of Contents

1. Why Open-Source Matters

2. SAGE Overview

3. SAGE for Cryptographers

Outline

1 Why Open-Source Matters

2 SAGE Overview

3 SAGE for Cryptographers

“Neubüsser’s Law” I

“You can read [some] Theorem and its proof in [some] book in the library [. . .] then you can use [that] Theorem for the rest of your life free of charge, but for many computer algebra systems license fees have to be paid regularly [...]. You press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

“Neubüsser’s Law” II

With this situation **two of the most basic rules of conduct in mathematics are violated**: In mathematics **information is passed on free of charge** and **everything is laid open for checking**. Not applying these rules to computer algebra systems that are made for mathematical research [. . .] means **moving in a most undesirable direction**. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?”

– J. Neubüser (1993) (he started GAP in 1986).

Neubüser's Law and SAGE

"Both the SAGE development model and the technology in SAGE itself is distinguished by an extremely strong emphasis on openness, community, cooperation, and collaboration":

- SAGE is released under the GNU General Public License (GPL) with no restrictions (e.g. academic use only) and free of charge.
- SAGE only includes software under GPL-compatible licenses.
- SAGE is developed by a worldwide community of developers.
- The $HOME directories of every developer on our main development machine are world-readable at `http://sage.math.washington.edu/home`.
- public bugtracker, developer's mailinglists, support mailinglists, and IRC channel.

Outline

1 Why Open-Source Matters

2 SAGE Overview

3 SAGE for Cryptographers

What is SAGE?

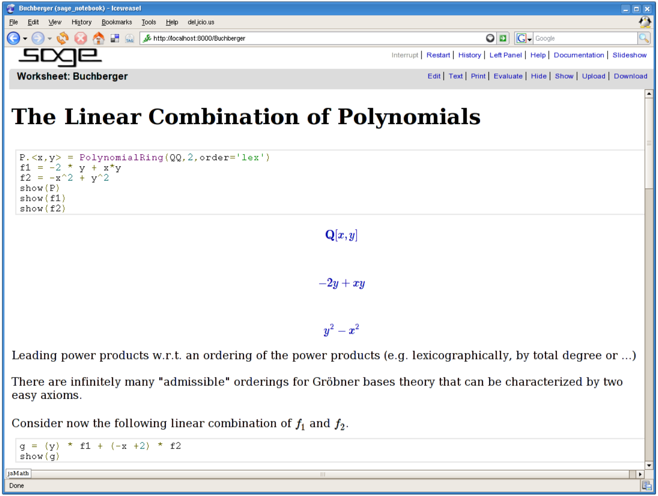
SAGE has **three** complementary aspects:

1. **A Free Distribution** of open source math software. 90MB source tarball that builds easily on OS X, Linux, MS Windows, and soon Solaris.
2. **New Functionality** that fill in gaps in what is available elsewhere.
3. **A Unified Interface** to most math software: to Magma, Macaulay2, Singular, Maple, MATLAB, Mathematica, Axiom, etc.

1. A Free Distribution

Arithmetic	**GMP, MPFR, Givaro**
Commutative algebra	**SINGULAR** (libSINGULAR)
Linear algebra	**LinBox, M4RI, IML**
Cryptography	**OpenSSL, PyOpenSSL, PyCrypto**
Factoring	**FlintQS, ECM**, distributed SAGE
Group theory and combinatorics	**GAP**
Graph theory	**NetworkX**
Number Theory	**PARI, NTL**
Numerical computation	**GSL, Numpy**
Calculus, Symbolic comp	**Maxima**
Specialized math	many C/C++ programs...
Interface	Notebook, **jsmath, Moin wiki, IPython**
Plotting	**Matplotlib, Tachyon, libgd**
Networking	**Twisted**
Database	**ZODB, SQLite**, Python Pickles
Programming language	**Python**, SageX (compiled)

The Notebook


Buchberger (sage_notebook) - Iceweasel
File Edit View History Bookmarks Tools Help del.icio.us
http://localhost:8000/Buchberger
Google
sage
Interrupt | Restart | History | Left Panel | Help | Documentation | Slideshow
Edit | Text | Print | Evaluate | Hide | Show | Upload | Download
jsMath
Done

Worksheet: Buchberger

The Linear Combination of Polynomials

```
P.<x,y> = PolynomialRing(QQ,2,order='lex')
f1 = -2 * y + x*y
f2 = -x^2 + y^2
show(P)
show(f1)
show(f2)
```

$$\mathbf{Q}[x,y]$$

$$-2y + xy$$

$$y^2 - x^2$$

Leading power products w.r.t. an ordering of the power products (e.g. lexicographically, by total degree or ...)

There are infinitely many "admissible" orderings for Gröbner bases theory that can be characterized by two easy axioms.

Consider now the following linear combination of f_1 and f_2.

```
g = (y) * f1 + (-x +2) * f2
show(g)
```

2. New Functionality

```
# 131245 unique lines of code (including documentation)
$ cat */*.py */*/*.py */*/*/*.py */*.pyx \
    */*/*.pyx */*/*.pyx |sort |uniq | wc -l
131245

# 15690 lines of example inputs
$ cat */*.py */*/*.py */*/*/*.py */*.pyx \
      */*/*.pyx */*/*.pyx |sort|uniq|grep "sage:" | wc -l
15690
```

Examples: re-implementation of Nauty's graph isomorphism, arithmetic with *p*-adic numbers, heaps of elliptic curve code, sparse linear algebra (over $\mathbb{F}_p$), task farming distributed computing.

3. Unified Interfaces

- SAGE **interfaces to**: Axiom, GAP, **GP/PARI**, Kash, Macaulay2, **Magma**, Maple, Mathematica, MATLAB, Maxima, MuPad, Octave, **Singular**, etc.

- This gives SAGE a wide range of **functionality**.

- Unified **command completion and help**.

How the Interfaces Work

Use buffered psuedo-tty, files, and Python objects that wrap native objects. This makes it possible to wrap **all** math software that has a command line interface using similar code.

```
sage: x = gp('9+6')     # the GP/PARI math software
```

This fires up one copy of GP/PARI and sends the line 'sage[1] = 9+6' to it

```
sage: !ps ax |grep gp
16389  p5  Ss+    0:00.02 /opt/sage/local/bin/gp --fast ...
sage: type(x)
<class 'sage.interfaces.gp.GpElement'>
sage: x, x.name()
15, 'sage[1]'
sage: x.factor()
[3, 1; 5, 1]
```

Another Example

```
sage: I.groebner_basis?
...
Return a Groebner basis of this ideal.

INPUT:
    algorithm -- determines the algorithm to use, available are:
                 * None - autoselect (default)
                 * 'singular:groebner' - Singulars groebner command
                 * 'singular:std' - Singulars std command
                 * 'singular:stdhilb' - Singulars stdhib command
                 * 'singular:stdfglm' - Singulars stdfglm command
                 * 'singular:slimgb' - Singulars slimgb command
                 * 'macaulay2:gb' (if available) - Macaulay2s gb command
                 * 'magma:GroebnerBasis' (if available) - MAGMAs Groebnerbasis command

ALGORITHM: Uses Singular, MAGMA, or Macaulay2 (if available)
```

Outline

1. Why Open-Source Matters

2. SAGE Overview

3. SAGE for Cryptographers

by Field of Interest I

see http://www.sagemath.org:9001/cando

Commutative Algebra highlevel commutative algebra over $\mathbb{F}_q$ using SINGULAR, basic arithmetic over arbitrary rings.

Linear Algebra dense linear algebra over $\mathbb{F}_q$ using LinBox, M4RI, and custom code, and sparse linear algebra over $\mathbb{F}_q$ using custom code.

Group Theory permutations groups, abelian groups, matrix groups (in particular, classical groups over finite fields)

Statistics sorry, SAGE doesn't really come with a full-blown statistics package.

by Field of Interest II

see http://www.sagemath.org:9001/cando

Number Theory compute Mordell-Weil groups of (many) elliptic curves using both invariants and algebraic 2-descents, a wide range of number theoretic functions, e.g., euler_phi, primes enumeration, sigma, tau_qexp, etc. optimized modern quadratic sieve for factoring integers $n = p \cdot q$, optimized implementation of the elliptic curve factorization method, modular symbols for general weight, character, Gamma1, and GammaH, modular forms for general weight ≥ 2, character, Gamma1, and GammaH.

by Field of Interest III

see `http://www.sagemath.org:9001/cando`

Elliptic Curves all standard invariants of elliptic curves over $\mathbb{Q}$, division polynomials, etc. , compute the number of points on an elliptic curve modulo p for all primes p less than a million in seconds, optimized implementation of the Schoof-Elkies-Atkin point counting algorithm for counting points modulo p when p is large, complex and p-adic L-functions of elliptic curves. Can compute p-adic heights and regulators for $p < 100000$ in a reasonable amount of time.

by Objection

MAGMA/other CAS If you have MAGMA installed, you can use all your MAGMA code from within SAGE. Additionally you get a graphical user interface, better tab completion, and interaction with other computer algebra systems. Also, in a **few** areas SAGE already beats MAGMA speed- and feature-wise.

Custom C programs SAGE is easily extendible using SageX/Pyrex or the pseudo-tty interfaces. Consequently, custom C/C++ programs can be integrated with other computer algebra software, there is no need for writing a custom user interface, and existing libraries can be re-used.

Questions?

You can try SAGE directly from your webbrowser at
`http://www.sagenb.org/`