Upcoming p-adic functionality in FLINT

Sebastian Pancratz

Sage-FLINT Days, Warwick, 17–22 July 2011

Motivation

Motivation for the implementation.

- I need p-adic arithmetic for my own research code, which is largely based on FLINT.

Purpose of the talk.

- Present the already implemented functionality;
- Raise awareness for Sage Days 19–23 Feb 2012, San Diego;
- Ask for feedback.

Overview

- Comparison with Laurent series over $\mathbf{F}_p$
- Elements of $\mathbf{Q}_p$
- Functions on $\mathbf{Q}_p$
- Addition
- Multiplication
- Inversion
- Teichmüller lift
- Exponential
- Logarithm
- Polynomials over $\mathbf{Q}_p$

Comparison with Laurent series over $\mathbf{F}_p$

A Laurent series consists of the data $(m, n, (a_m, \ldots, a_n))$ giving

$$\sum_{i=m}^{n} a_i X^i$$

that is,

$$X^m \sum_{i=0}^{n-m} a_{i+m} X^i.$$

Given $f(X)$ and $g(X)$, we can compute their sum modulo X^N as

$$f(X) + g(X) = \sum_{i=\min\{m_f, m_g\}}^{\min\{\max\{n_f, n_g\}, N-1\}} (a_i + b_i) X^i$$

As coefficients are readily available, it is reasonable for operations to treat inputs as exact and require only the output precision N.

Elements of $\mathbf{Q}_p$

Consider,

$$x = 3 + 2 \times 5 + 1 \times 5^2 + 4 \times 5^3$$
$$y = 1 + 1 \times 5 + 4 \times 5^2 + 2 \times 5^3 + 3 \times 5^4$$

Computing their sum modulo 5^2,

$$x + y = (3 + 1) + (2 + 1)5.$$

But this is *not* what is happening in practical implementations. The p-adic digits are not readily available, and for $p \ll 2^{64}$ this is certainly not desirable anyway.

Elements of $\mathbf{Q}_p$

Instead, an element $x \neq 0$ is typically stored as $x = p^v u$ with $v = \mathrm{ord}_p(x) \in \mathbf{Z}$ and $u \in \mathbf{Z}$ with $p \nmid u$. In FLINT, we choose

```
typedef struct {
    fmpz u;
    long v;
} padic_struct;
```

Remarks.

- Improved maintainability by having *one* data type; no special case depending on the size of p or p^N;
- Eventually, $p = 2$ should have a special case.

Functions on $\mathbf{Q}_p$

Philosophy.

- Treat input arguments as exact elements in $\mathbf{Q}_p$ and return the ouput reduced modulo p^N.

For example, for the p-adic inversion function,

$$\begin{tikzcd} \mathbf{Q} \arrow[r, hook, "\iota"] \arrow[d, "(-)^{-1}"'] & \mathbf{Q}_p \arrow[d, "(-)^{-1}"] \\ \mathbf{Q} \arrow[r, hook, "\iota"'] & \mathbf{Q}_p \end{tikzcd}$$

For $x \in \mathbf{Q}_p^\times$, we want

$$\iota\big((\iota^{-1}x)^{-1}\big) \equiv x^{-1} \pmod{p^N}.$$

Benchmarks

We present some timings for arithmetic in $\mathbf{Q}_p \bmod p^N$ where $p = 17$, $N = 2^i$, $i = 0, \ldots, 10$, comparing the three systems Magma (V2.17-13), Sage (4.8) and FLINT (2.3) on a machine with Intel Xeon CPUs running at 2.93GHz.

To avoid worrying about taking the same random sequences of elements, we instead fix elements $x = 3^{3N}$ and $y = 5^{2N}$ modulo p^N.

Addition

Signature.

```
void padic_add(z, x, y, ctx)
```

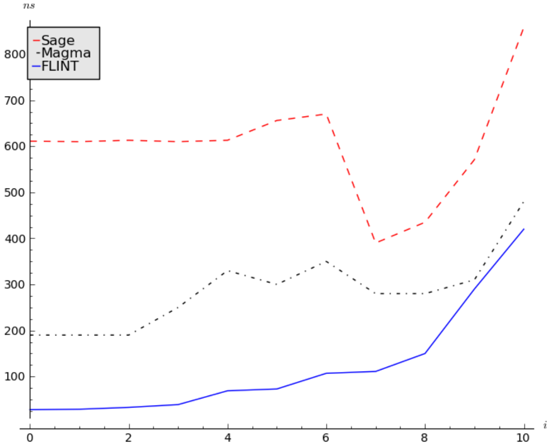
Contract.

Assumes that x and y are reduced modulo p^N and returns z in reduced form, too.

Algorithm.

Avoids expensive modulo operation.

Addition

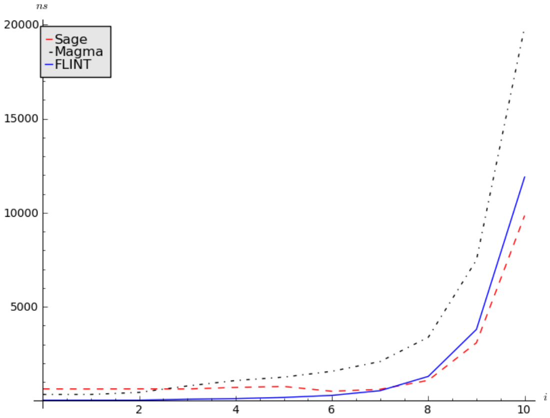


Multiplication

Signature.
```
void padic_mul(z, x, y, ctx)
```

Contract.
Makes no assumptions on x, y but returns z reduced modulo p^N.

Multiplication



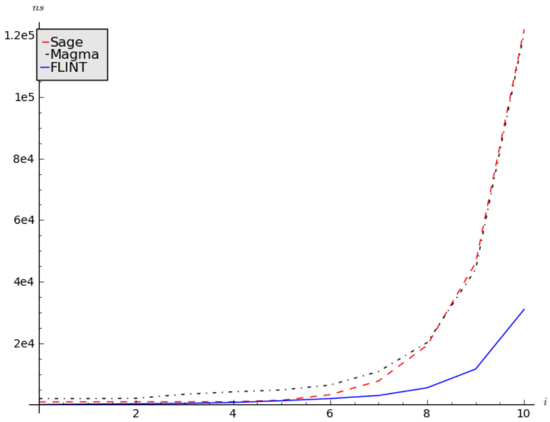
Inversion

Signature.
`void padic_inv(z, x, ctx)`

Contract.
Makes no assumptions on $x \neq 0$, returns z reduced modulo p^N.

Algorithm.
Hensel lifting.

Inversion



Teichmüller lift

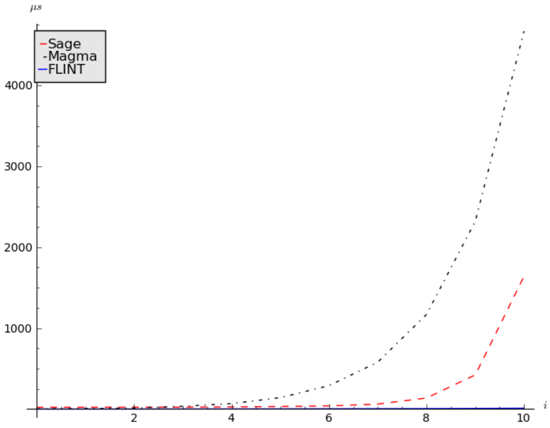
Signature.
```
void padic_teichmuller(z, x, ctx)
```

Contract.
Assumes only that $\mathrm{ord}_p(x) = 1$, returns z reduced modulo p^N.

Algorithm.
Hensel lifting.

Teichmüller lift

Computes the Teichmüller lift of $x \bmod p^N$ to the required precision N.



Exponential

Signature.
`int padic_exp(z, x, ctx)`

Contract.
Assumes that $\exp_p(x)$ converges, that is, $\mathrm{ord}_p(x) \geq 2$ or $\mathrm{ord}_p(x) \geq 1$ as $p = 2$ or $p > 2$, respectively, and returns z reduced modulo p^N.

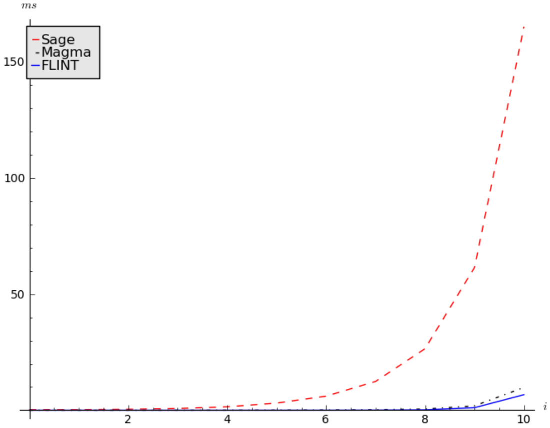
Algorithm.
Evaluates the truncated series

$$\exp_p(x) = \sum_{i=0}^{m} \frac{x^i}{i!}$$

over $\mathbf{Z}_p$ by multiplying through by $m!$, hence requiring only one p-adic inversion.

Exponential

Computes the exponential of $17^2 \times y$ to the required precision N.



Logarithm

Signature.
`int padic_log(z, x, ctx)`

Contract.
Assumes that $\log_p(x)$ converges, that is, $\mathrm{ord}_p(x-1) \geq 2$ or $\mathrm{ord}_p(x-1) \geq 1$ as $p = 2$ or $p > 2$, respectively, and returns z reduced modulo p^N.

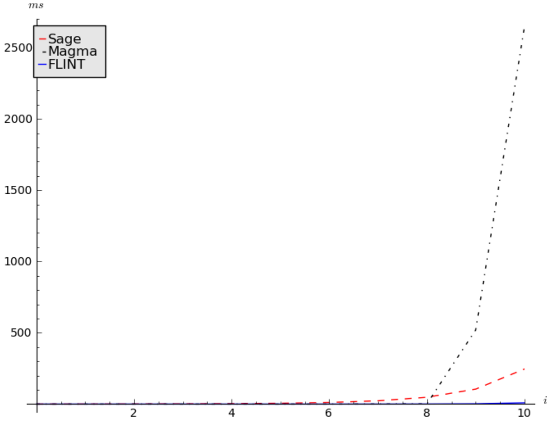
Algorithm.
Evaluates the truncated series

$$\log_p(x) = \sum_{i=1}^{m} (-1)^{i-1} \frac{(x-1)^i}{i}$$

over $\mathbf{Z}_p$ by inverting i at each step using a precomputed Hensel lifting structure.

Logarithm

Computes the logarithm of $1 - 17^2 y$ to the required precision N.



Other functions on $\mathbf{Q}_p$

Other functions include:

- Subtraction
- Negation
- Powers
- Inversion (with precomputed lifting structure)
- Division
- Square root

Polynomials over $\mathbf{Q}_p$

We represent a non-zero polynomial $f(X) \in \mathbf{Q}_p[X]$ as

$$f(X) = p^v\big(a_0 + a_1 X + \cdots + a_n X^n\big)$$

where $a_0, \dots, a_n \in \mathbf{Z}$ and, for at least one i, p does not divide a_i.

Remarks.

- Allows for transfer of many problems over $\mathbf{Q}_p$ to $\mathbf{Z}/(p^N)$, where fast implementations are available.
- Similar to the approach chosen over $\mathbf{Q}$ in FLINT (and Sage), see trac ticket #4000.

Polynomials over $\mathbf{Q}_p$

Functionality available.

- Conversions to polynomials over $\mathbf{Z}$ and $\mathbf{Q}$
- Coefficient manipulation
- Addition, subtraction, negation
- Scalar multiplication
- Multiplication
- Powers
- Series inversion
- Derivative
- Evaluation
- Composition