MATHIC, SINGULAR & XMALLOC

Christian Eder

POLSYS Team, UPMC, Paris, France

June 11, 2013
SINGULAR
Signature-based Gröbner Basis algorithms
Restructuring SINGULAR

XMALLOC

MATHIC
Overall structure
Matrix reduction part
Methods of Parallelization
Future steps
Signature-based Gröbner Basis algorithms

Implementation of different variants of F5:

- **Kernel implementation**
  - Up to 10 times faster than SINGULAR’s std implementation
  - Not using any linear algebra, plain polynomial reduction
  - Officially available in SINGULAR 4.0
Implementation of different variants of F5:

- **Kernel implementation**
  - Up to 10 times faster than SINGULAR’s std implementation
  - Not using any linear algebra, plain polynomial reduction
  - Officially available in SINGULAR 4.0

- **Library implementation**
  - together with John Perry
  - Teaching purpose only
Restructuring SINGULAR

Since May 2011 the team is restructuring the kernel of SINGULAR:

- Making kernel readable again
- Documentation of the code
- Enabling other systems to use only parts of the kernel:
  - Memory management
  - Poly structures and arithmetic
  - Gröbner layer
  . . .
Since May 2011 the team is restructuring the kernel of SINGULAR:

- Making kernel readable again
- Documentation of the code
- Enabling other systems to use only parts of the kernel:
  - Memory management
  - Poly structures and arithmetic
  - Gröbner layer
  - ...
Since May 2011 the team is restructuring the kernel of SINGULAR:

- Making kernel readable again
- Documentation of the code
- Enabling other systems to use only parts of the kernel:
  - Memory management
  - Poly structures and arithmetic
  - Gröbner layer
  - ...

First official release: End of 2013
SINGULAR
Signature-based Gröbner Basis algorithms
Restructuring SINGULAR

XMALLOC

MATHIC
Overall structure
Matrix reduction part
Methods of Parallelization
Future steps
SINGULAR’s memory manager

SINGULAR depends on special purpose memory manager called OMALLOC.

Problems:
1. OMALLOC deeply integrated in SINGULAR’s kernel
2. OMALLOC not thread-safe

⇒ XMALLOC
  ▶ standalone library with interface to SINGULAR
  ▶ step by step making it thread-safe
  ▶ keeping OMALLOC’s speed and memory footprint when used in SINGULAR
SINGULAR depends on special purpose memory manager called OMALLOC.

2 Problems:
1. OMALLOC deeply integrated in SINGULAR’s kernel
2. OMALLOC not thread-safe
SINGULAR’s memory manager

SINGULAR depends on a special purpose memory manager called OMALLOC.

2 Problems:

1. OMALLOC deeply integrated in SINGULAR’s kernel
2. OMALLOC not thread-safe

⇒ XMALLOC

- standalone library with interface to SINGULAR
- step by step making it thread-safe
- keeping OMALLOC’s speed and memory footprint when used in SINGULAR
SINGULAR
Signature-based Gröbner Basis algorithms
Restructuring SINGULAR

XMALLOC

MATHIC
Overall structure
Matrix reduction part
Methods of Parallelization
Future steps
Main structure of the library

Consists of 3 big parts:

- MathicGB: Gröbner basis structures and algorithms
- Mathic: Data types, structures, hashing
- Memtailor: Small arena memory manager
Mathic (Roune, Stillman)

- C++ library of data structures designed for Gröbner basis computation, like S-pairs, performing divisor queries, ordering polynomial terms during reduction, ...
Mathic (Roune, Stillman)

- C++ library of data structures designed for Gröbner basis computation, like S-pairs, performing divisor queries, ordering polynomial terms during reduction, . . .

- Highly templated, thus applicable with a wide range of monomial/term resp. coefficient representations.

Note: The paper "Practical Groebner Basis Computation" by Roune and Stillman describes the data structures from a high level. The paper was presented at ISSAC12 and is available (in an extended version) at http://arxiv.org/abs/1206.6940.
Mathic (Roune, Stillman)

- C++ library of data structures designed for Gröbner basis computation, like S-pairs, performing divisor queries, ordering polynomial terms during reduction, ...  

- Highly templated, thus applicable with a wide range of monomial/term resp. coefficient representations.

- Only suitable for dense representations of monomials/terms, e.g. Mathic regularly asks for “What is the exponent of variable number \( k \) in this monomial/term?”

Note: The paper “Practical Groebner Basis Computation” by Roune and Stillman describes the data structures from a high level. The paper was presented at ISSAC12 and is available (in an extended version) at http://arxiv.org/abs/1206.6940.
Mathic (Roune, Stillman)

- C++ library of data structures designed for Gröbner basis computation, like S-pairs, performing divisor queries, ordering polynomial terms during reduction, . . .

- Highly templated, thus applicable with a wide range of monomial/term resp. coefficient representations.

- Only suitable for dense representations of monomials/terms, e.g. Mathic regularly asks for “What is the exponent of variable number \( k \) in this monomial/term?”

Note

The paper “Practical Groebner Basis Computation” by Roune and Stillman describes the data structures from a high level. The paper was presented at ISSAC12 and is available (in an extended version) at http://arxiv.org/abs/1206.6940.
MathicGB (Roune, Stillman, E.)

- C++ library for computing Gröbner bases.
C++ library for computing Gröbner bases.

Initially started to implement Buchberger-like algorithms; those depend highly on Mathic.
MathicGB (Roune, Stillman, E.)

- C++ library for computing Gröbner bases.
- Initially started to implement Buchberger-like algorithms; those depend highly on Mathic.
- The signature-based algorithm SB is implemented in MathicGB (presented at ISSAC12).
**MathicGB (Roune, Stillman, E.)**

- C++ library for computing Gröbner bases.
- Initially started to implement Buchberger-like algorithms; those depend highly on Mathic.
- The signature-based algorithm SB is implemented in MathicGB (presented at ISSAC12).
- Summer 2012: Bjarke came to Kaiserslautern, started thinking about F4 and matrix reduction.
MathicGB (Roune, Stillman, E.)

▷ C++ library for computing Gröbner bases.

▷ Initially started to implement Buchberger-like algorithms; those depend highly on Mathic.

▷ The signature-based algorithm SB is implemented in MathicGB (presented at ISSAC12).

▷ Summer 2012: Bjarke came to Kaiserslautern, started thinking about F4 and matrix reduction

Implementation of matrix reduction

- Implementing idea of Faugère and Lachartre:

```
\begin{array}{cccc}
A & B & C & D \\
\end{array}
```

- Not reducing $A$, but directly eliminating $C$.
- Focusing on fields of prime characteristic $< 2^{16}$.
- Delayed modulus when reducing $D$ to $D'$ by the surrounding parts.
Implementing idea of Faugère and Lachartre: \textbf{Quadmatrix}, already in ABCD shape.

\begin{center}
\begin{tikzpicture}
\fill[red] (0,0) rectangle (2,1);
\fill[green] (0,1) rectangle (1,2);
\fill[blue] (1,0) rectangle (2,1);
\fill[yellow] (0,0) rectangle (1,1);
\node at (0.5,1.5) {A};
\node at (1.5,1.5) {B};
\node at (1.5,0.5) {C};
\node at (0.5,0.5) {D};
\end{tikzpicture}
\end{center}

Not reducing $A$, but directly eliminating $C$.

Focussing on fields of prime characteristic $< 2^{16}$.

$\Rightarrow$ Delayed modulus when reducing $D$ to $D'$ by the surrounding parts.
Implementing idea of Faugère and Lachartre: Quadmatrix, already in ABCD shape.

Not reducing $A$, but directly eliminating $C$. 
Implementing idea of Faugère and Lachartre: 
**Quadmatrix**, already in ABCD shape.

▶ Not reducing $A$, but directly eliminating $C$.

▶ Focussing on fields of prime characteristic $< 2^{16}$.

$\Rightarrow$ Delayed modulus when reducing $D$ to $D'$ by the surrounding parts.
How our matrices look like
Implementation of matrix reduction

▶ Sparse and dense matrix representations
Implementation of matrix reduction

- Sparse and dense matrix representations
- No blocking at the moment.
Implementation of matrix reduction

- Sparse and dense matrix representations
- No blocking at the moment.
- Straightforward parallelization over the rows.

Working on more general input in order to compare with Martani's implementation directly.

Ongoing tasks: Matrix format, (parallel) matrix construction.
Implementation of matrix reduction

- Sparse and dense matrix representations
- No blocking at the moment.
- Straightforward parallelization over the rows.
- Working on more general input in order to compare with Martani’s implementation directly.
Implementation of matrix reduction

- Sparse and dense matrix representations
- No blocking at the moment.
- Straightforward parallelization over the rows.
- Working on more general input in order to compare with Martani’s implementation directly.
- Ongoing tasks: Matrix format, (parallel) matrix construction
Methods of Parallelization

- Memtailor uses *pthreads* at some crucial points.

- The parallelization of the matrix reduction in MathicGB is implemented using *Intel Threading Building Blocks*:
Methods of Parallelization

- Memtailor uses **pthreads** at some crucial points.

- The parallelization of the matrix reduction in MathicGB is implemented using **Intel Threading Building Blocks**:  
  - Very C++ish, templates galore, etc.
Methods of Parallelization

- Memtailor uses ** pthreads ** at some crucial points.

- The parallelization of the matrix reduction in MathicGB is implemented using ** Intel Threading Building Blocks **:
  - Very C++ish, templates galore, etc.
  - Until now only basic PARALLEL_FOR / BLOCKED_RANGE implementations.
Methods of Parallelization

- Memtailor uses pthreads at some crucial points.
- The parallelization of the matrix reduction in MathicGB is implemented using Intel Threading Building Blocks:
  - Very C++ish, templates galore, etc.
  - Until now only basic PARALLEL_FOR / BLOCKED_RANGE implementations.
  - Needs more abstraction on the task level.
Near future

- **Do not double memory while preparing matrix**
  Permuting rows and columns currently copies.

- **Break matrices into smaller stripes of columns**
  Enabling 16-bit column indices for each stripe; trying to put several rows of such a stripe into L1-cache.

- **Finer graining in parallelization**
  Until now only Intel Threading Blocks is used.
On the longer run

- **Rewrite of the matrix reduction**
  Right now it is a rather straightforward implementation; needs to become an own layer.

- **F5F4**
  Until now this is a plain F4 implementation. Signature-based computations are only available without linear algebra at the moment.

- **Syzygy computations**

- **Investigating GPUs**
  Whole new business when it comes to efficient implementation due to different architecture.
LELA
https://github.com/martani/LELA

SINGULAR
https://github.com/Singular/Sources

XMALLOC
https://github.com/ederc/xmalloc

MATHIC
https://github.com/broune/memtailor
https://github.com/broune/mathic
https://github.com/broune/mathicgb