Software Development Within the SPP1489

Wolfram Decker

TU Kaiserslautern

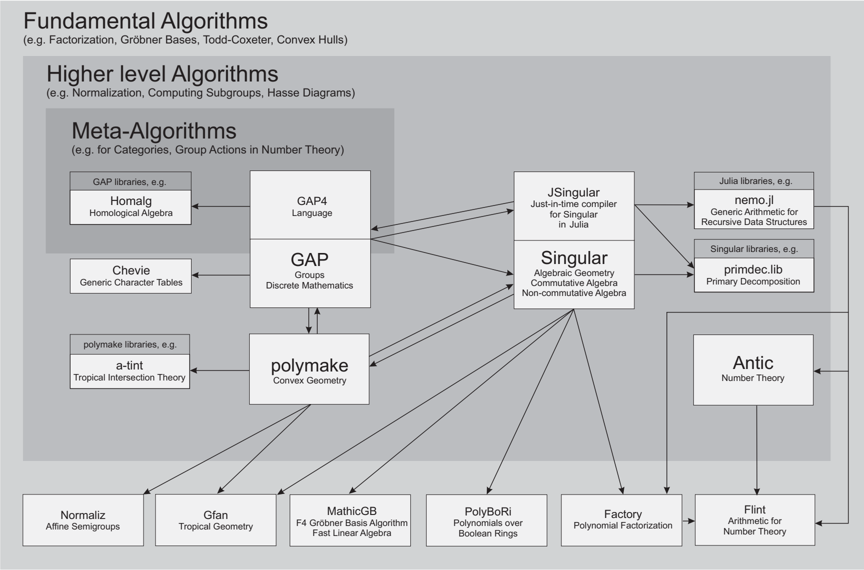
Osnabrück, September 29, 2015

Algorithmic and Experimental Methods
in Algebra, Geometry, and Number Theory
DFG Priority Project SPP 1489

Some basic statements

- Computer algebra algorithms may be implemented in specialized libraries or packages, but it is their incorporation into computer algebra systems - with convenient languages for direct user interaction, comfortable help functions and comprehensive manuals - which make the resulting tools accessible to the interested researcher.

Our Vision


Fundamental Algorithms
(e.g. Factorization, Gröbner Bases, Todd-Coxeter, Convex Hulls)
Higher level Algorithms
(e.g. Normalization, Computing Subgroups, Hasse Diagrams)
Meta-Algorithms
(e.g. for Categories, Group Actions in Number Theory)
GAP libraries, e.g.
Homalg
Homological Algebra
GAP4
Language
GAP
Groups
Discrete Mathematics
Chevie
Generic Character Tables
polymake libraries, e.g.
a-tint
Tropical Intersection Theory
polymake
Convex Geometry
JSingular
Just-in-time compiler for Singular in Julia
Singular
Algebraic Geometry
Commutative Algebra
Non-commutative Algebra
Julia libraries, e.g.
nemo.jl
Generic Arithmetic for Recursive Data Structures
Singular libraries, e.g.
primdec.lib
Primary Decomposition
Antic
Number Theory
Normaliz
Affine Semigroups
Gfan
Tropical Geometry
MathicGB
F4 Gröbner Basis Algorithm
Fast Linear Algebra
PolyBoRi
Polynomials over Boolean Rings
Factory
Polynomial Factorization
Flint
Arithmetic for Number Theory

Some basic statements

- Through computer algebra software, a large treasure of mathematical knowledge becomes accessible to and can also be applied by non-experts.

Citation Record (data from https://zbmath.org)

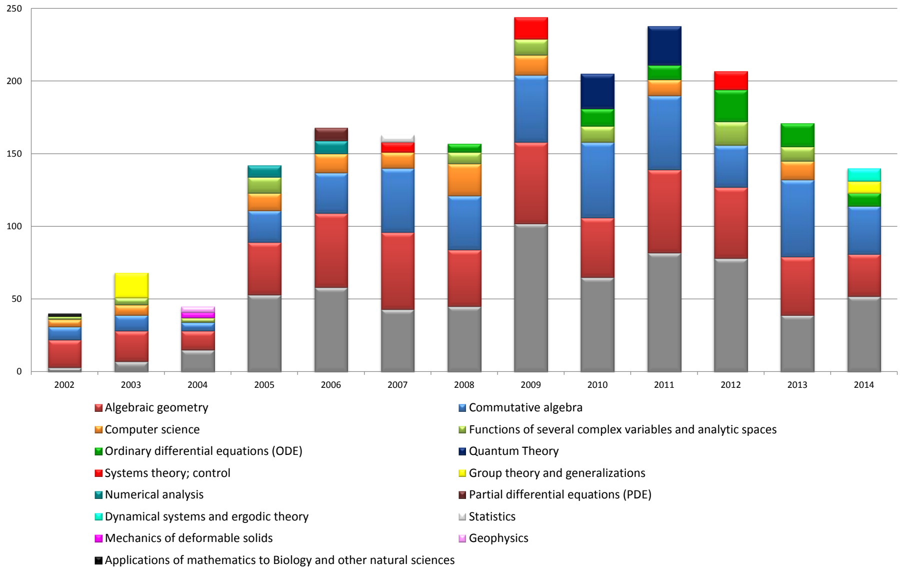
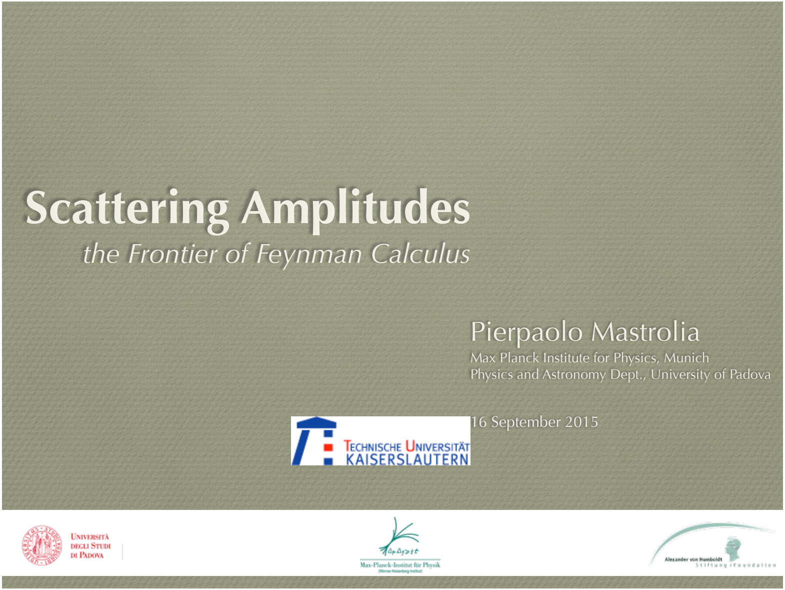


Figure: Top five MSC areas for SINGULAR per year, remaining areas in grey

Some basic statements

- The design and further development of successful computer algebra software is always driven by intended applications, irrespective of whether these applications lie within or outside of mathematics.

Potential Applications: High Energy Physics



Potential Applications: High Energy Physics

High Energy Particle Physics

- Discovery of the Higgs boson: **Standard Model** completed!

We've got a beautiful theory for describing (many, if not all) phenomena at colliders and elsewhere

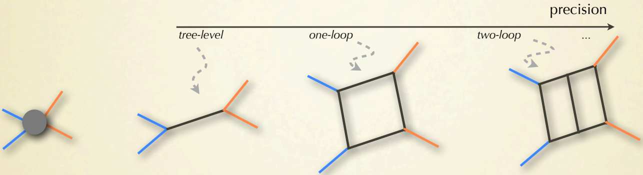
? Higgs properties

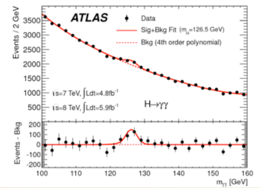
- Higgs couplings extracted from normalization of cross-sections that are sensitive to radiative corrections
- Further verification of Higgs mechanism will require detailed theoretical predictions for production cross-sections and decay rates

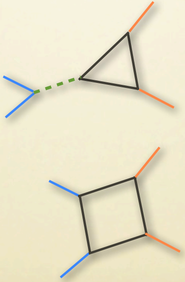
? The Standard Model cannot be the ultimate theory:
(ex. neutrino masses and dark matter not accounted for)
searching for ***New Physics***

- ***New Physics*** effect carried by **massive particles**
- Current major focus (*main stream*):
improving perturbative prediction for partonic cross sections:
a very important and active field of research

Scattering ~ Feynman Diagrams





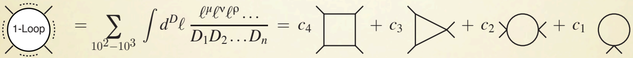


- same signature in the experiment
- only one of them contributes to the little bump
- precision calculation to distinguish them:

  signal vs background analysis

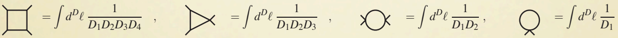
One-Loop Scattering Amplitudes

- n-particle Scattering: $1+2 \to 3+4+\ldots+n$
- Reduction to a Scalar-Integral Basis   Passarino-Veltman



$$= \sum_{10^2-10^3} \int d^D\ell \, \frac{\ell^\mu \ell^\nu \ell^\rho \ldots}{D_1 D_2 \ldots D_n} = c_4 \,\square + c_3 \,\triangleright + c_2 \,\bigcirc + c_1 \,\bigcirc$$

- Known: Master Integrals



$$\square = \int d^D\ell \, \frac{1}{D_1 D_2 D_3 D_4}, \quad \triangleright = \int d^D\ell \, \frac{1}{D_1 D_2 D_3}, \quad \bigcirc = \int d^D\ell \, \frac{1}{D_1 D_2}, \quad \bigcirc = \int d^D\ell \, \frac{1}{D_1}$$

- Unknowns: c_i are rational functions of external kinematic invariants

Potential Applications: High Energy Physics

Example

```
> ring R = (0,p11,p12,p22,e34,m1,m2,m3), (x1,x2,x3,x4), dp;
> poly D1 = 2*x3*x4*e34+x1*(p11*x1+p12*x2)
.              +(x1*p12+p22*x2)*x2-m1;
> poly D2 = -m2+2*x3*x4*e34-2*p11*x1+p11+x1*(p11*x1+p12*x2)
.              +(x1*p12+p22*x2)*x2-2*p12*x2;
> poly D3 = 2*x3*x4*e34+2*x1*p12-m3+x1*(p11*x1+p12*x2)+p22
.              +(x1*p12+p22*x2)*x2+2*p22*x2;
> ideal I = D1, D2, D3;
> ideal GI = groebner(I);
```

Joint project with Pierpaolo Mastrolia, Tiziano Peraro, and Janko Böhm.

Potential Applications: Surfaces of General Type With $p_g = 0$

"Can a computer classify all surfaces of general type with $p_g = 0$?"

David Mumford, Montreal, 1980

Potential Applications: Surfaces of General Type With $p_g = 0$

"Can a computer classify all surfaces of general type with $p_g = 0$?"

David Mumford, Montreal, 1980

The problem

A minimal surface of general type with $p_g = 0$ (hence $q = 0$) satisfies $1 \leq K^2 \leq 9$ (Bogomolov-Miyaoka-Yau inequality).

Potential Applications: Surfaces of General Type With $p_g = 0$

"Can a computer classify all surfaces of general type with $p_g = 0$?"

David Mumford, Montreal, 1980

The problem

A minimal surface of general type with $p_g = 0$ (hence $q = 0$) satisfies $1 \leq K^2 \leq 9$ (Bogomolov-Miyaoka-Yau inequality). For each $1 \leq n \leq 9$, there is the Gieseker moduli space parametrising the isomorphism classes of surfaces with $K^2 = n$.

Potential Applications: Surfaces of General Type With $p_g = 0$

"Can a computer classify all surfaces of general type with $p_g = 0$?"

David Mumford, Montreal, 1980

The problem

A minimal surface of general type with $p_g = 0$ (hence $q = 0$) satisfies $1 \leq K^2 \leq 9$ (Bogomolov-Miyaoka-Yau inequality). For each $1 \leq n \leq 9$, there is the Gieseker moduli space parametrising the isomorphism classes of surfaces with $K^2 = n$. At current state, the complete description of these moduli spaces is wide open.

Potential Applications: Surfaces of General Type With $p_g = 0$

"Can a computer classify all surfaces of general type with $p_g = 0$?"

David Mumford, Montreal, 1980

The problem

A minimal surface of general type with $p_g = 0$ (hence $q = 0$) satisfies $1 \leq K^2 \leq 9$ (Bogomolov-Miyaoka-Yau inequality). For each $1 \leq n \leq 9$, there is the Gieseker moduli space parametrising the isomorphism classes of surfaces with $K^2 = n$. At current state, the complete description of these moduli spaces is wide open.

Work by Castelnuovo, Enriques, Godeaux, Campedelli, Miyaoka, Reid and students, Beauville, Bauer-Catanese and students, and many more.

Potential Applications: Surfaces of General Type With $p_g = 0$

Example (The case $K^2 = 1$: Numerical Godeaux surfaces)

For such a surface X, it is known that $H_1(X, \mathbb{Z})$ is cyclic of order at most 5, and constructions have been given for each choice of order.

Potential Applications: Surfaces of General Type With $p_g = 0$

Example (The case $K^2 = 1$: Numerical Godeaux surfaces)

For such a surface X, it is known that $H_1(X, \mathbb{Z})$ is cyclic of order at most 5, and constructions have been given for each choice of order. It is conjectured that there is precisely one irreducible family of surfaces for each order, and that in each case $\pi_1(X) \cong H_1(X, \mathbb{Z})$.

Potential Applications: Surfaces of General Type With $p_g = 0$

Idea of a construction in case $H_1(X, \mathbb{Z}) = 0$

Let $\{x_0, x_1\}$ be a basis of $|2K_S|$ and $\{y_0, y_1, y_2, y_3\}$ a basis of $|3K_S|$.

Potential Applications: Surfaces of General Type With $p_g = 0$

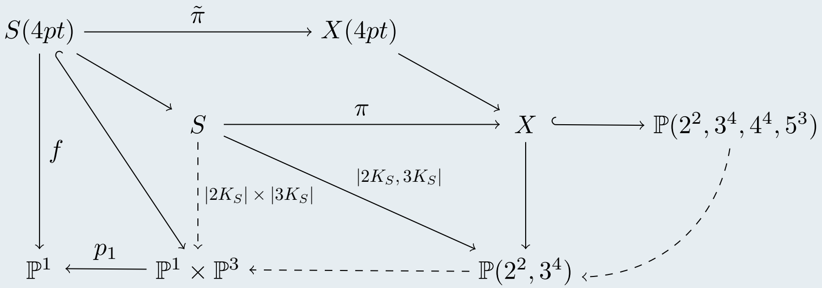
Idea of a construction in case $H_1(X, \mathbb{Z}) = 0$

Let $\{x_0, x_1\}$ be a basis of $|2K_S|$ and $\{y_0, y_1, y_2, y_3\}$ a basis of $|3K_S|$. We consider the canonical ring $R(S) = \bigoplus_{n \geq 0} \mathrm{H}^0(S, \mathcal{O}_S(nK_S))$ as a module over the weighted polynomial ring $\mathbb{C}[x_0, x_1, y_0, y_1, y_2, y_3]$ and study the image of $X = \mathbf{Proj}(R(S))$ in $\mathbb{P}(2^2, 3^4)$ under the map induced by $|2K_S, 3K_S|$.

Potential Applications: Surfaces of General Type With $p_g = 0$

Idea of a construction in case $H_1(X, \mathbb{Z}) = 0$

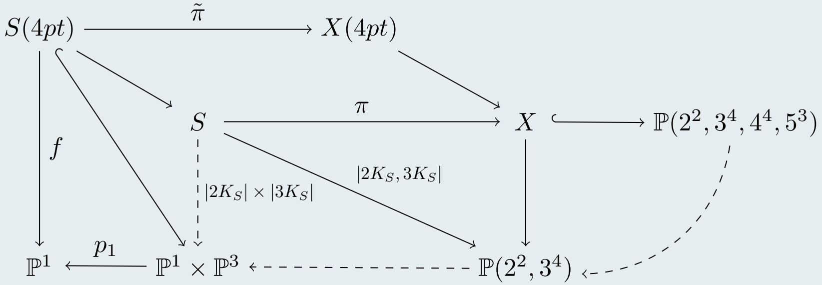
The bicanonical system $|2K_S|$ has no fixed part and 4 distinct base points.



Potential Applications: Surfaces of General Type With $p_g = 0$

Idea of a construction in case $H_1(X, \mathbb{Z}) = 0$

The bicanonical system $|2K_S|$ has no fixed part and 4 distinct base points.



Joint project with Frank-Olaf Schreyer and Isabel Stenger.

Potential Applications: Surfaces of General Type With $p_g = 0$

Why is this computationally hard?

```
> Rextension;
//   characteristic : 0
//   1 parameter    : a
//   minpoly        :
(248738794738328172995583944749904330252605378584297O0*a^8
+412197480758832021377448558823165698794277118212212070*a^7
+625366891611244986389942014312773193649951168354090190*a^6
-436561073546512334083477547357856090524552855592558795*a^5
-914947642504230095779800456657440020138074539186145912*a^4
-2227325279423247966617649640155997715235288113299887954*a^3
+2312070077580715288467637707530192772778088469836344950*a^2
+1366053134215201364075122803745127996518986576734818612*a
-1156759915557562158859054495379551857229358735237021536)
//   number of vars : 12
```

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

Fundamental stuff

- Gröbner and standard Bases;

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

Fundamental stuff

- Gröbner and standard Bases;
- syzygies and free resolutions;

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

Fundamental stuff

- Gröbner and standard Bases;
- syzygies and free resolutions;
- polynomial factorization.

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

Fundamental stuff

- Gröbner and standard Bases;
- syzygies and free resolutions;
- polynomial factorization.

Higher level stuff

- Primary decomposition; algorithms of Gianni-Trager-Zacharias, Shimoyama-Yokoyama, Eisenbud-Huneke-Vasconcelos: `primdec.lib`.;

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

Fundamental stuff

- Gröbner and standard Bases;
- syzygies and free resolutions;
- polynomial factorization.

Higher level stuff

- Primary decomposition; algorithms of Gianni-Trager-Zacharias, Shimoyama-Yokoyama, Eisenbud-Huneke-Vasconcelos: `primdec.lib`.;
- normalization: algorithms of de Jong, Greuel-Laplagne-Seelisch.

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

Fundamental stuff

- Gröbner and standard Bases;
- syzygies and free resolutions;
- polynomial factorization.

Higher level stuff

- Primary decomposition; algorithms of Gianni-Trager-Zacharias, Shimoyama-Yokoyama, Eisenbud-Huneke-Vasconcelos: `primdec.lib`.;
- normalization: algorithms of de Jong, Greuel-Laplagne-Seelisch.
- means to analyse singularities: Hamburger-Noether expansions, blow-ups, resolution of singularities, and more.

Example

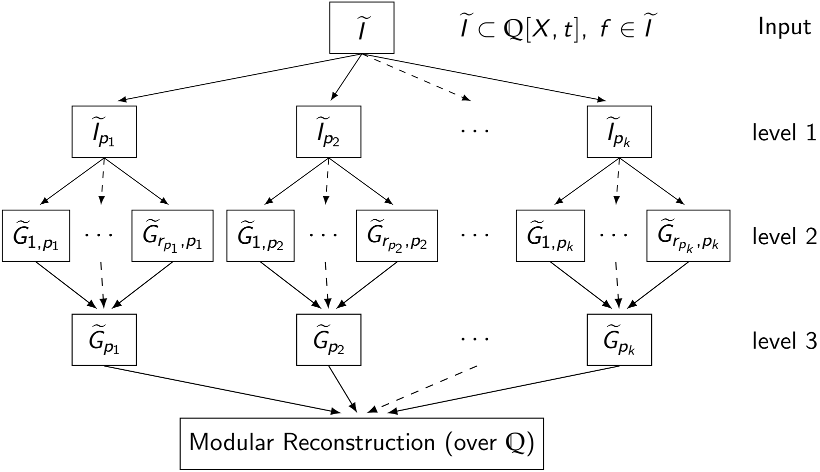
- Dereje Kifle Boku, Wolfram Decker, Claus Fieker, Andreas Steenpass: *Gröbner Bases over Algebraic Number Fields.* Accepted paper for PASCO 2015

Example

- Dereje Kifle Boku, Wolfram Decker, Claus Fieker, Andreas Steenpass:
  Gröbner Bases over Algebraic Number Fields.
  Accepted paper for PASCO 2015
- Burcin Erocal, Oleksandr Motsak, Frank-Olaf Schreyer, Andreas Steenpass:
  Refined Algorithms to Compute Syzygies.
  To appear in J. Symb. Comp.

Ongoing work: Gröbner bases over rational function fields, various approaches to computing syzygies.

Example: Gröbner Bases over Number Fields



See talk by Andreas Steenpass.

Second Challenge: Parallelization

Parallelization is a fundamental challenge to all CAS from both a computer science and a mathematical point of view.

Second Challenge: Parallelization

Parallelization is a fundamental challenge to all CAS from both a computer science and a mathematical point of view.

Computer science point of view

In principle, there are two types of parallelization:

- **Coarse-grained parallelisation** works by starting different processes not sharing memory space, and elaborate but infrequent ways of exchanging global data.

Second Challenge: Parallelization

Parallelization is a fundamental challenge to all CAS from both a computer science and a mathematical point of view.

Computer science point of view

In principle, there are two types of parallelization:

- **Coarse-grained parallelisation** works by starting different processes not sharing memory space, and elaborate but infrequent ways of exchanging global data.
- **Fine-grained parallelisation** works with multiple threads in a single process sharing both memory space and global data, and typically with frequent but efficient communications.

Second Challenge: Parallelization

Parallelization is a fundamental challenge to all CAS from both a computer science and a mathematical point of view.

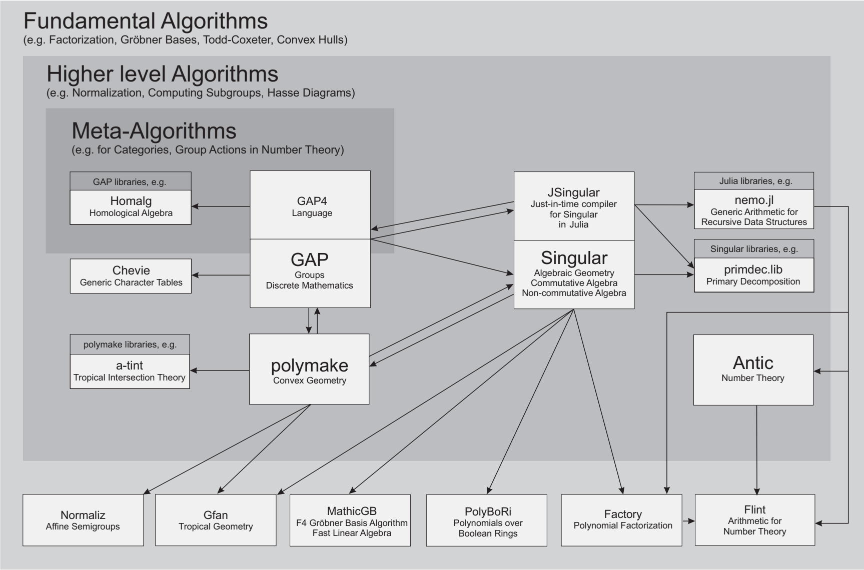
Computer science point of view

In principle, there are two types of parallelization:

- **Coarse-grained parallelisation** works by starting different processes not sharing memory space, and elaborate but infrequent ways of exchanging global data.
- **Fine-grained parallelisation** works with multiple threads in a single process sharing both memory space and global data, and typically with frequent but efficient communications.

The latter requires, for example, to make the memory managment of the CAS thread-safe in an effective way.

Our Vision


Fundamental Algorithms
(e.g. Factorization, Gröbner Bases, Todd-Coxeter, Convex Hulls)
Higher level Algorithms
(e.g. Normalization, Computing Subgroups, Hasse Diagrams)
Meta-Algorithms
(e.g. for Categories, Group Actions in Number Theory)
GAP libraries, e.g.
Homalg
Homological Algebra
GAP4
Language
GAP
Groups
Discrete Mathematics
Chevie
Generic Character Tables
polymake libraries, e.g.
a-tint
Tropical Intersection Theory
polymake
Convex Geometry
JSingular
Just-in-time compiler
for Singular
in Julia
Singular
Algebraic Geometry
Commutative Algebra
Non-commutative Algebra
Julia libraries, e.g.
nemo.jl
Generic Arithmetic for
Recursive Data Structures
Singular libraries, e.g.
primdec.lib
Primary Decomposition
Antic
Number Theory
Normaliz
Affine Semigroups
Gfan
Tropical Geometry
MathicGB
F4 Gröbner Basis Algorithm
Fast Linear Algebra
PolyBoRi
Polynomials over
Boolean Rings
Factory
Polynomial Factorization
Flint
Arithmetic for
Number Theory

Example

```
> LIB("parallel.lib","random.lib");
> ring R = 0,x(1..4),dp;
> ideal I = randomid(maxideal(3),3,100);
```

Coarse Grained Parallelism in SINGULAR by Example

Example

```
> LIB("parallel.lib","random.lib");
> ring R = 0,x(1..4),dp;
> ideal I = randomid(maxideal(3),3,100);
> proc sizeStd(ideal I, string monord){
      def R = basering; list RL = ringlist(R);
      RL[3][1][1] = monord; def S = ring(RL); setring(S);
      return(size(std(imap(R,I))));}
```

Example

```
> LIB("parallel.lib","random.lib");
> ring R = 0,x(1..4),dp;
> ideal I = randomid(maxideal(3),3,100);
> proc sizeStd(ideal I, string monord){
      def R = basering; list RL = ringlist(R);
      RL[3][1][1] = monord; def S = ring(RL); setring(S);
      return(size(std(imap(R,I))));}
> list commands = "sizeStd","sizeStd";
> list args = list(I,"lp"),list(I,"dp");
```

Example

```
> LIB("parallel.lib","random.lib");
> ring R = 0,x(1..4),dp;
> ideal I = randomid(maxideal(3),3,100);
> proc sizeStd(ideal I, string monord){
      def R = basering; list RL = ringlist(R);
      RL[3][1][1] = monord; def S = ring(RL); setring(S);
      return(size(std(imap(R,I))));}
> list commands = "sizeStd","sizeStd";
> list args = list(I,"lp"),list(I,"dp");
> parallelWaitFirst(commands, args);
  [1] empty list
  [2] 11
```

Coarse Grained Parallelism in Singular by Example

Example

```
> LIB("parallel.lib","random.lib");
> ring R = 0,x(1..4),dp;
> ideal I = randomid(maxideal(3),3,100);
> proc sizeStd(ideal I, string monord){
      def R = basering; list RL = ringlist(R);
      RL[3][1][1] = monord; def S = ring(RL); setring(S);
      return(size(std(imap(R,I))));}
> list commands = "sizeStd","sizeStd";
> list args = list(I,"lp"),list(I,"dp");
> parallelWaitFirst(commands, args);
  [1] empty list
  [2] 11
> parallelWaitAll(commands, args);
  [1] 55
  [2] 11
```

Mathematical point of view

There are algorithms whose basic strategy is inherently parallel, whereas others are sequential in nature.

Second Challenge: Parallelization

Mathematical point of view

There are algorithms whose basic strategy is inherently parallel, whereas others are sequential in nature. A prominent example of the former type is Villamayor's constructive version of Hironaka's desingularization theorem.

Example: Resolution of Singularities

Theorem (Hironaka, 1964)

For every algebraic variety over a field K with $\operatorname{char} K = 0$ *a desingularization can be obtained by a finite sequence of blow-ups along smooth centers.*

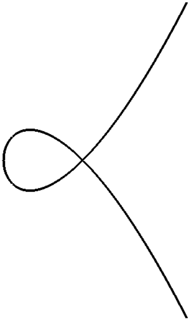
Example: Resolution of Singularities

Theorem (Hironaka, 1964)

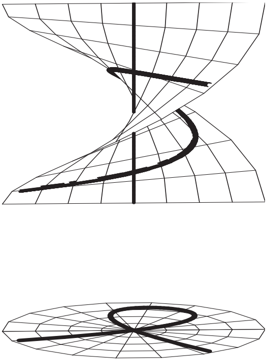
For every algebraic variety over a field K with $\operatorname{char} K = 0$ *a desingularization can be obtained by a finite sequence of blow-ups along smooth centers.*

For example, resolve the

node



by one blow-up



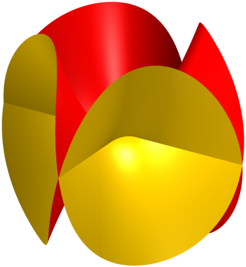
which replaces the singular point inside the plane by a line, hence

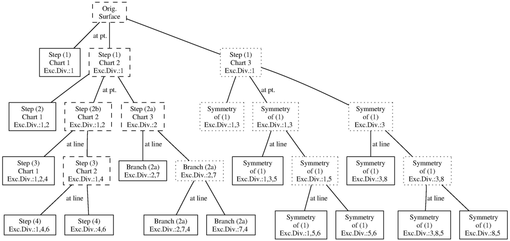
Example: Resolution of Singularities

Working with blow-ups means to work with different charts.

Example: Resolution of Singularities

Working with blow-ups means to work with different charts.
In this way, the resolution of singularities leads to a tree of charts. Here is the graph for resolving the singularities of $z^2 - x^2y^2 = 0$





Second Challenge: Parallelization

Mathematical point of view

The algebraic concept of normalization "improves" the singularities, typically without yielding their full resolution.

Second Challenge: Parallelization

Mathematical point of view

The algebraic concept of normalization "improves" the singularities, typically without yielding their full resolution. The classical normalization algorithm is a prominent example of an algorithm which is sequential in nature.

Second Challenge: Parallelization

Mathematical point of view

The algebraic concept of normalization "improves" the singularities, typically without yielding their full resolution. The classical normalization algorithm is a prominent example of an algorithm which is sequential in nature. It proceeds by successively enlarging the given ring until the Grauert and Remmert normalization criterion allows one to stop:

$$A = A^{(0)} \subset A^{(1)} \cdots \subset A^{(m)} = \overline{A}.$$

Second Challenge: Parallelization

Mathematical point of view

The algebraic concept of normalization "improves" the singularities, typically without yielding their full resolution. The classical normalization algorithm is a prominent example of an algorithm which is sequential in nature. It proceeds by successively enlarging the given ring until the Grauert and Remmert normalization criterion allows one to stop:

$$A = A^{(0)} \subset A^{(1)} \cdots \subset A^{(m)} = \overline{A}.$$

The systematic design of parallel algorithms in areas where no such algorithms exist is a tremendous task.

Second Challenge: Parallelization

Mathematical point of view

The algebraic concept of normalization "improves" the singularities, typically without yielding their full resolution. The classical normalization algorithm is a prominent example of an algorithm which is sequential in nature. It proceeds by successively enlarging the given ring until the Grauert and Remmert normalization criterion allows one to stop:

$$A = A^{(0)} \subset A^{(1)} \cdots \subset A^{(m)} = \overline{A}.$$

The systematic design of parallel algorithms in areas where no such algorithms exist is a tremendous task. For computations over the rationals, it is important to identify algorithms which allow parallelization via modular methods.

Second Challenge: Parallelization

Mathematical point of view

The algebraic concept of normalization "improves" the singularities, typically without yielding their full resolution. The classical normalization algorithm is a prominent example of an algorithm which is sequential in nature. It proceeds by successively enlarging the given ring until the Grauert and Remmert normalization criterion allows one to stop:

$$A = A^{(0)} \subset A^{(1)} \cdots \subset A^{(m)} = \overline{A}.$$

The systematic design of parallel algorithms in areas where no such algorithms exist is a tremendous task. For computations over the rationals, it is important to identify algorithms which allow parallelization via modular methods. Here, mathematical ideas are needed to design the final verification steps.

Example: Local-to-Global Approach to Normalization

New local-to-global approach to normalization

Example: Local-to-Global Approach to Normalization

New local-to-global approach to normalization

Janko Boehm, Wolfram Decker, Santiago Laplagne, Gerhard Pfister, Andreas Steenpass, Stefan Steidel:
Parallel algorithms for normalization.
J. Symb. Comp. 51 (2013), 99-114.

Example: Local-to-Global Approach to Normalization

New local-to-global approach to normalization

Janko Boehm, Wolfram Decker, Santiago Laplagne, Gerhard Pfister, Andreas Steenpass, Stefan Steidel:
Parallel algorithms for normalization.
J. Symb. Comp. 51 (2013), 99-114.

- Stratify the singular locus;

Example: Local-to-Global Approach to Normalization

New local-to-global approach to normalization

Janko Boehm, Wolfram Decker, Santiago Laplagne, Gerhard Pfister, Andreas Steenpass, Stefan Steidel:
Parallel algorithms for normalization.
J. Symb. Comp. 51 (2013), 99-114.

- Stratify the singular locus;
- compute a local contribution to the normalization at each stratum;

Example: Local-to-Global Approach to Normalization

New local-to-global approach to normalization

Janko Boehm, Wolfram Decker, Santiago Laplagne, Gerhard Pfister, Andreas Steenpass, Stefan Steidel:
Parallel algorithms for normalization.
J. Symb. Comp. 51 (2013), 99-114.

- Stratify the singular locus;
- compute a local contribution to the normalization at each stratum;
- put the local contributions together to get the normalization.

Example: Local-to-Global Approach to Normalization

New local-to-global approach to normalization

Janko Boehm, Wolfram Decker, Santiago Laplagne, Gerhard Pfister, Andreas Steenpass, Stefan Steidel:
Parallel algorithms for normalization.
J. Symb. Comp. 51 (2013), 99-114.

- Stratify the singular locus;
- compute a local contribution to the normalization at each stratum;
- put the local contributions together to get the normalization.

Approach is parallel in nature.

Example: Local-to-Global Approach to Normalization

New local-to-global approach to normalization

Janko Boehm, Wolfram Decker, Santiago Laplagne, Gerhard Pfister, Andreas Steenpass, Stefan Steidel:
Parallel algorithms for normalization.
J. Symb. Comp. 51 (2013), 99-114.

- Stratify the singular locus;
- compute a local contribution to the normalization at each stratum;
- put the local contributions together to get the normalization.

Approach is parallel in nature. In addition, there is a modular version.

New local-to-global approach to normalization

Janko Boehm, Wolfram Decker, Santiago Laplagne, Gerhard Pfister, Andreas Steenpass, Stefan Steidel:
Parallel algorithms for normalization.
J. Symb. Comp. 51 (2013), 99-114.

- Stratify the singular locus;
- compute a local contribution to the normalization at each stratum;
- put the local contributions together to get the normalization.

Approach is parallel in nature. In addition, there is a modular version.

Have also developed parallel algorithms for integral bases and adjoint curves.

Example: Local-to-Global Approach to Normalization

New local-to-global approach to normalization

Janko Boehm, Wolfram Decker, Santiago Laplagne, Gerhard Pfister, Andreas Steenpass, Stefan Steidel:
Parallel algorithms for normalization.
J. Symb. Comp. 51 (2013), 99-114.

- Stratify the singular locus;
- compute a local contribution to the normalization at each stratum;
- put the local contributions together to get the normalization.

Approach is parallel in nature. In addition, there is a modular version.

Have also developed parallel algorithms for integral bases and adjoint curves. See talk by Janko Böhm.

Third Challenge: Make More and More of the Abstract Concepts of Algebra, Geometry, and Number Theory Constructive

Example: Cohomology

Typical applications of Gröbner Bases and Syzygies in the non-commutative case:

Example: Cohomology

Typical applications of Gröbner Bases and Syzygies in the non-commutative case:

Example (De Rham cohomology)

Use the Weyl algebra to compute the de Rham cohomology of complements of affine varieties. Algorithm by Uli Walther, implemented by Cornelia Rottner in SINGULAR.

Example: Cohomology

Typical applications of Gröbner Bases and Syzygies in the non-commutative case:

Example (De Rham cohomology)

Use the Weyl algebra to compute the de Rham cohomology of complements of affine varieties. Algorithm by Uli Walther, implemented by Cornelia Rottner in SINGULAR.

Example (Sheaf cohomology)

Use the exterior algebra to compute the cohomology of coherent sheaves on projective space via the constructive version of the Bernstein-Gel'fand-Gel'fand (BGG) correspondence by Eisenbud-Fløystad-Schreyer.

Example: de Rham Cohomology

Example

```
> ring R = 0, (x,y,z), dp;
> list L = (xy,xz);
> deRhamCohomology(L);
 [1]:
    1
 [2]:
    1
 [3]:
    0
 [4]:
    1
 [5]:
    1
```

Notation

Let V be a vector space of dimension $n+1$ over a field K with dual space $W = V^*$, $S = \mathrm{Sym}_K(W)$ and $E = \bigwedge V$. We grade S and E by taking elements of W to have degree 1, and elements of V to have degree -1.
Let $\mathbb{P}^n = \mathbb{P}(V)$ be the projective space of lines in V.

Notation

Let V be a vector space of dimension $n+1$ over a field K with dual space $W = V^*$, $S = \mathrm{Sym}_K(W)$ and $E = \bigwedge V$. We grade S and E by taking elements of W to have degree 1, and elements of V to have degree -1. Let $\mathbb{P}^n = \mathbb{P}(V)$ be the projective space of lines in V.

The BGG correspondence relates bounded complexes of coherent sheaves on $\mathbb{P}^n$ and minimal doubly infinite free resolutions over E.

Notation

Let V be a vector space of dimension $n+1$ over a field K with dual space $W = V^*$, $S = \mathrm{Sym}_K(W)$ and $E = \bigwedge V$. We grade S and E by taking elements of W to have degree 1, and elements of V to have degree -1. Let $\mathbb{P}^n = \mathbb{P}(V)$ be the projective space of lines in V.

The BGG correspondence relates bounded complexes of coherent sheaves on $\mathbb{P}^n$ and minimal doubly infinite free resolutions over E. In particular, it associates to each finitely generated graded S-module a so-called *Tate resolution* which only depends on the sheafification $\widetilde{M}$ and which "reflects" the cohomology of $\widetilde{M}$ and all its twists.

Potential Applications: Hartshorne's Conjecture

"The question of the existence of non-trivial rank 2 vector bundles on $\mathbb{P}^n$, $n \geq 5$, is the most interesting unsolved problem in projective geometry that I know of."

David Mumford, GIT, second edition, 1982

Potential Applications: Hartshorne's Conjecture

"The question of the existence of non-trivial rank 2 vector bundles on $\mathbb{P}^n$, $n \geq 5$, is the most interesting unsolved problem in projective geometry that I know of."

David Mumford, GIT, second edition, 1982

Conjecture (Hartshorne, 1974)

If $n \geq 7$, there are no indecomposable vector bundles of rank 2 on $\mathbb{P}^n$.

Potential Applications: Hartshorne's Conjecture

"The question of the existence of non-trivial rank 2 vector bundles on $\mathbb{P}^n$, $n \geq 5$, is the most interesting unsolved problem in projective geometry that I know of."

David Mumford, GIT, second edition, 1982

Conjecture (Hartshorne, 1974)

If $n \geq 7$, there are no indecomposable vector bundles of rank 2 on $\mathbb{P}^n$.

Via Serre correspondence, and by Barth's Lefschetz type theorem, this conjecture is equivalent to the codimension 2 case of the following more general conjecture:

Potential Applications: Hartshorne's Conjecture

"The question of the existence of non-trivial rank 2 vector bundles on $\mathbb{P}^n$, $n \geq 5$, is the most interesting unsolved problem in projective geometry that I know of."

David Mumford, GIT, second edition, 1982

Conjecture (Hartshorne, 1974)

If $n \geq 7$, there are no indecomposable vector bundles of rank 2 on $\mathbb{P}^n$.

Via Serre correspondence, and by Barth's Lefschetz type theorem, this conjecture is equivalent to the codimension 2 case of the following more general conjecture:

Conjecture (Hartshorne, 1974)

If Y is a non-singular subvariety of dimension r of $\mathbb{P}^n$, and if $r > \frac{2}{3}n$, then Y is a complete intersection.

The border case

The only known non-trivial rank 2 vector bundles on $\mathbb{P}^4 = \mathbb{P}(V)$ are the Horrocks-Mumford bundle and its satellites.

The border case

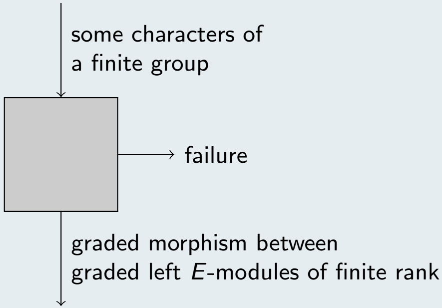
The only known non-trivial rank 2 vector bundles on $\mathbb{P}^4 = \mathbb{P}(V)$ are the Horrocks-Mumford bundle and its satellites. The construction of the Horrocks-Mumford bundle relies heavily on the representation theory of the Heisenberg group of level 5 and its normalizer in $\mathrm{SL}(V)$.

A homalg Example: From Groups to Vector Bundles

The border case

The only known non-trivial rank 2 vector bundles on $\mathbb{P}^4 = \mathbb{P}(V)$ are the Horrocks-Mumford bundle and its satellites. The construction of the Horrocks-Mumford bundle relies heavily on the representation theory of the Heisenberg group of level 5 and its normalizer in $\mathrm{SL}(V)$.

Mohamed Barakat's dream:



A homalg Example: From Groups to Vector Bundles

```
gap> LoadPackage( "repsn" );;
gap> LoadPackage( "GradedModules" );;
gap> G := SmallGroup( 1000, 93 );
<pc group of size 1000 with 6 generators>
gap> Display( StructureDescription( G ) );
((C5 x C5) : C5) : Q8

gap> V := Irr( G )[6];; Degree( V );
5
gap> T0 := Irr( G )[5];; Degree( T0 );
2
gap> T1 := Irr( G )[8];; Degree( T1 );
5
gap> mu0 := ConstructTateMap( V, T0, T1, 2 );
<A homomorphism of graded left modules>
```

A homalg Example:: From groups to Vector Bundles

```
gap> A := HomalgRing( mu0 );
Q{e0,e1,e2,e3,e4}
(weights: [ -1, -1, -1, -1, -1 ])
gap> M:=GuessModuleOfGlobalSectionsFromATateMap(2, mu0);;
gap> ByASmallerPresentation( M );
<A graded non-zero module presented by 92
  relations for 19 generators>

gap> S := HomalgRing( M );
Q[x0,x1,x2,x3,x4]
(weights: [ 1, 1, 1, 1, 1 ])
gap> ChernPolynomial( M );
( 2 | 1-h+4*h^2 ) -> P^4
gap> tate := TateResolution( M, -5, 5 );;
```

A homalg Example:: From Groups to Vector Bundles

```
gap> Display( BettiTable( tate ) );
total:  100  37  14  10   5   2   5  10  14  37 100   ?   ?   ?   ?
----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
    4:  100  35   4   .   .   .   .   .   .   .   .   0   0   0   0
    3:    *   .   2  10  10   5   .   .   .   .   .   .   0   0   0
    2:    *   *   .   .   .   .   .   2   .   .   .   .   .   0   0
    1:    *   *   *   .   .   .   .   .   .   5  10  10   2   .   0
    0:    *   *   *   *   .   .   .   .   .   .   .   .   4  35 100
----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---S
twist:   -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1   2   3   4   5
---------------------------------------------------------------------
Euler:  100  35   2 -10 -10  -5   0   2   0  -5 -10 -10   2  35 100
```

Third Challenge: Make More and More of the Abstract Concepts of Algebraic Geometry Constructive

Exploit derived equivalences in computer algebra

Capturing the intrinsic structure of geometric objects in terms of numbers or algebraic objects is a guiding theme in algebraic geometry. With the rapid increase of abstraction initiated by Grothendieck, this relies on more and more involved mathematical language and formalism.

Third Challenge: Make More and More of the Abstract Concepts of Algebraic Geometry Constructive

Exploit derived equivalences in computer algebra

Capturing the intrinsic structure of geometric objects in terms of numbers or algebraic objects is a guiding theme in algebraic geometry. With the rapid increase of abstraction initiated by Grothendieck, this relies on more and more involved mathematical language and formalism. Most prominently, the abstract language of derived categories provides a unifying and refining framework for constructions of homological algebra, duality, and cohomology theories.

Third Challenge: Make More and More of the Abstract Concepts of Algebraic Geometry Constructive

Exploit derived equivalences in computer algebra

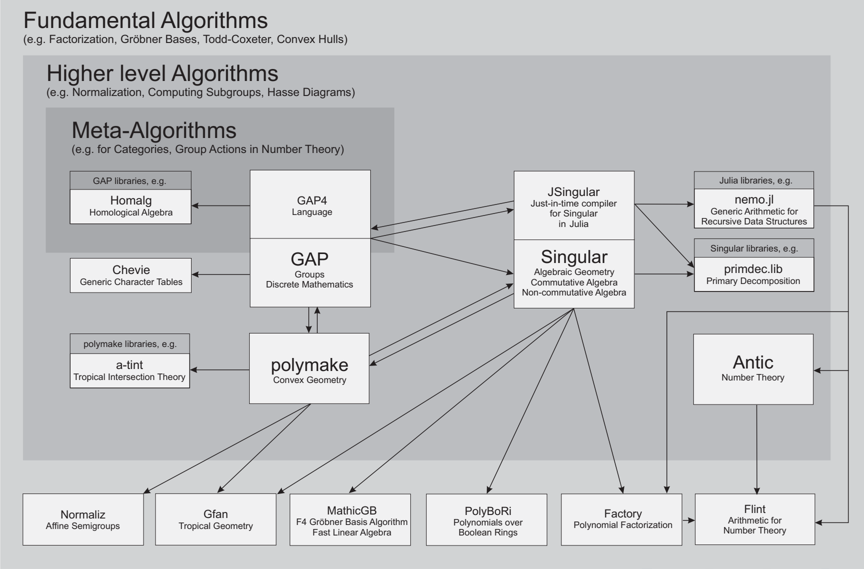
Capturing the intrinsic structure of geometric objects in terms of numbers or algebraic objects is a guiding theme in algebraic geometry. With the rapid increase of abstraction initiated by Grothendieck, this relies on more and more involved mathematical language and formalism. Most prominently, the abstract language of derived categories provides a unifying and refining framework for constructions of homological algebra, duality, and cohomology theories. Modeling such concepts in computer algebra is a fundamental task for the years to come.

Third Challenge: Make More and More of the Abstract Concepts of Algebraic Geometry Constructive

Exploit derived equivalences in computer algebra

Capturing the intrinsic structure of geometric objects in terms of numbers or algebraic objects is a guiding theme in algebraic geometry. With the rapid increase of abstraction initiated by Grothendieck, this relies on more and more involved mathematical language and formalism. Most prominently, the abstract language of derived categories provides a unifying and refining framework for constructions of homological algebra, duality, and cohomology theories. Modeling such concepts in computer algebra is a fundamental task for the years to come. In fact, the relationship between computer algebra and higher mathematical structures is of mutual benefit. Derived equivalences can, for example, be utilised to translate problems into an entirely different context with more efficient data structures and reduced complexity.

Fourth Challenge: Integration and Interaction of the Computer Algebra Systems and Libraries Involved



Computing the GIT-fan

$I \subset K[x_1, \dots, x_n]$ homogeneous w.r.t. $Q = (q_{ij}) \in \mathbb{Q}^{r \times n}$ and $X = V(I)$.

Another Example: Using POLYMAKE in SINGULAR

Computing the GIT-fan

$I \subset K[x_1, \ldots, x_n]$ homogeneous w.r.t. $Q = (q_{ij}) \in \mathbb{Q}^{r \times n}$ and $X = V(I)$.

$$Q \text{ defines a torus action} \qquad \mathbb{T}^r \times X \to X$$

- GIT-fan describes the variation of good quotients $U /\!\!/ \mathbb{T}^r$ with $U \subseteq X$.

Another Example: Using POLYMAKE in SINGULAR

Computing the GIT-fan

$I \subset K[x_1, \ldots, x_n]$ homogeneous w.r.t. $Q = (q_{ij}) \in \mathbb{Q}^{r \times n}$ and $X = V(I)$.

$$Q \text{ defines a torus action} \qquad \mathbb{T}^r \times X \to X$$

- GIT-fan describes the variation of good quotients $U /\!\!/ \mathbb{T}^r$ with $U \subseteq X$.
- Algorithm of [Keicher, 2012] computes the GIT-fan.

Another Example: Using POLYMAKE in SINGULAR

Computing the GIT-fan

$I \subset K[x_1, \dots, x_n]$ homogeneous w.r.t. $Q = (q_{ij}) \in \mathbb{Q}^{r \times n}$ and $X = V(I)$.

$$Q \text{ defines a torus action } \quad \mathbb{T}^r \times X \to X$$

- GIT-fan describes the variation of good quotients $U /\!\!/ \mathbb{T}^r$ with $U \subseteq X$.
- Algorithm of [Keicher, 2012] computes the GIT-fan.
- Uses **polyhedral geometry** (faces, intersection of cones).

Another Example: Using POLYMAKE in SINGULAR

Computing the GIT-fan

$I \subset K[x_1, \ldots, x_n]$ homogeneous w.r.t. $Q = (q_{ij}) \in \mathbb{Q}^{r \times n}$ and $X = V(I)$.

$$Q \text{ defines a torus action} \qquad \mathbb{T}^r \times X \to X$$

- GIT-fan describes the variation of good quotients $U /\!\!/ \mathbb{T}^r$ with $U \subseteq X$.
- Algorithm of [Keicher, 2012] computes the GIT-fan.
- Uses **polyhedral geometry** (faces, intersection of cones).
- Uses **Gröbner bases** (monomial containment test).

Another Example: Using POLYMAKE in SINGULAR

Computing the GIT-fan

$I \subset K[x_1, \ldots, x_n]$ homogeneous w.r.t. $Q = (q_{ij}) \in \mathbb{Q}^{r \times n}$ and $X = V(I)$.

Q defines a torus action $\quad \mathbb{T}^r \times X \to X$

- GIT-fan describes the variation of good quotients $U /\!\!/ \mathbb{T}^r$ with $U \subseteq X$.
- Algorithm of [Keicher, 2012] computes the GIT-fan.
- Uses **polyhedral geometry** (faces, intersection of cones).
- Uses **Gröbner bases** (monomial containment test).
- Action of a **finite symmetry group** makes complicated computations possible.
- Implemented in SINGULAR by Janko Böhm, Simon Keicher, and Yue Ren.

Application: Computing the GIT-fan

We compute the GIT-fan of $\mathbb{G}(2,4)$ in SINGULAR:

Example

```
> LIB "gitfan.lib";
> ring R = 0,x(1..6),dp;
> ideal I = x(1)*x(6) - x(2)*x(5) + x(3)*x(4);
```

Application: Computing the GIT-fan

We compute the GIT-fan of $\mathbb{G}(2,4)$ in SINGULAR:

Example

```
> LIB "gitfan.lib";
> ring R = 0,x(1..6),dp;
> ideal I = x(1)*x(6) - x(2)*x(5) + x(3)*x(4);
> intmat Q[3][6] = 1,0,0,1,1,0,
                   0,1,0,1,0,1,
                   0,0,1,0,1,1;
```

Application: Computing the GIT-fan

We compute the GIT-fan of $\mathbb{G}(2,4)$ in SINGULAR:

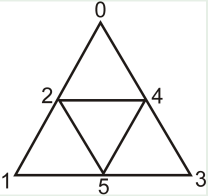
Example

```
> LIB "gitfan.lib";
> ring R = 0,x(1..6),dp;
> ideal I = x(1)*x(6) - x(2)*x(5) + x(3)*x(4);
> intmat Q[3][6] = 1,0,0,1,1,0,
                   0,1,0,1,0,1,
                   0,0,1,0,1,1;
> fan F = gitFan(I, Q);
> rays(F);
  0 0 1
  0 1 0
  0 1 1
  1 0 0
  1 0 1
  1 1 0
```

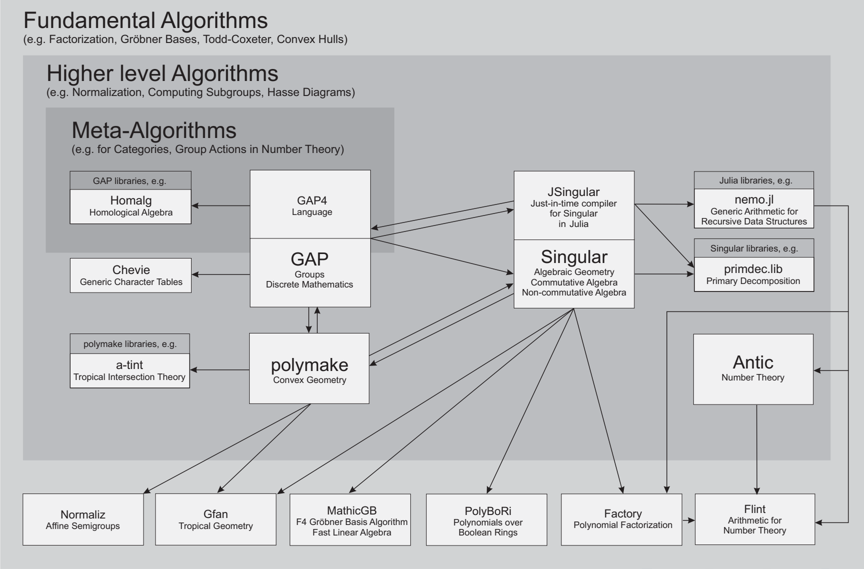
We compute the GIT-fan of $\mathbb{G}(2,4)$ in SINGULAR:

Example

```
> LIB "gitfan.lib";
> ring R = 0,x(1..6),dp;
> ideal I = x(1)*x(6) - x(2)*x(5) + x(3)*x(4);
> intmat Q[3][6] = 1,0,0,1,1,0,
                   0,1,0,1,0,1,
                   0,0,1,0,1,1;
> fan F = gitFan(I, Q);
> rays(F);
  0 0 1
  0 1 0
  0 1 1
  1 0 0
  1 0 1
  1 1 0
```



Fourth Challenge: Integration and Interaction of the Computer Algebra Systems and Libraries Involved



Software for Number Theory

FLINT (Bill Hart, Fredrik Johansson, et. al.) provides specific *highly optimized* implementations in C of various rings for computer algebra:

Software for Number Theory

FLINT (Bill Hart, Fredrik Johansson, et. al.) provides specific *highly optimized* implementations in C of various rings for computer algebra:

- Polynomials, power series, matrices, including linear algebra over a variety of specific rings.

Software for Number Theory

FLINT (Bill Hart, Fredrik Johansson, et. al.) provides specific *highly optimized* implementations in C of various rings for computer algebra:

- Polynomials, power series, matrices, including linear algebra over a variety of specific rings.
- $\mathbb{Z}/n\mathbb{Z}$, p-adics and unramified extensions, $\mathbb{Q}$, $\mathbb{Z}$, finite fields

Software for Number Theory

FLINT (Bill Hart, Fredrik Johansson, et. al.) provides specific *highly optimized* implementations in C of various rings for computer algebra:

- Polynomials, power series, matrices, including linear algebra over a variety of specific rings.
- $\mathbb{Z}/n\mathbb{Z}$, p-adics and unramified extensions, $\mathbb{Q}$, $\mathbb{Z}$, finite fields
- Polynomial factorisation over numerous rings.

Software for Number Theory

FLINT (Bill Hart, Fredrik Johansson, et. al.) provides specific *highly optimized* implementations in C of various rings for computer algebra:

- Polynomials, power series, matrices, including linear algebra over a variety of specific rings.
- $\mathbb{Z}/n\mathbb{Z}$, p-adics and unramified extensions, $\mathbb{Q}$, $\mathbb{Z}$, finite fields
- Polynomial factorisation over numerous rings.
- Real/complex arithmetic with guaranteed precision via FLINT/ARB extension library.

- ANTIC (Claus Fieker, Bill Hart, Tommy Hofmann): Fastest known library for number field arithmetic;

- ANTIC (Claus Fieker, Bill Hart, Tommy Hofmann): Fastest known library for number field arithmetic;
- NEMO (Bill Hart, Tommy Hofmann, Fredrik Johansson, Oleksandr Motsak): Implementation of recursive, generic rings in the Julia programming language.
- HECKE (Claus Fieker, Tommy Hofmann): Class groups and much more.

- ANTIC (Claus Fieker, Bill Hart, Tommy Hofmann): Fastest known library for number field arithmetic;
- NEMO (Bill Hart, Tommy Hofmann, Fredrik Johansson, Oleksandr Motsak): Implementation of recursive, generic rings in the Julia programming language.
- HECKE (Claus Fieker, Tommy Hofmann): Class groups and much more.

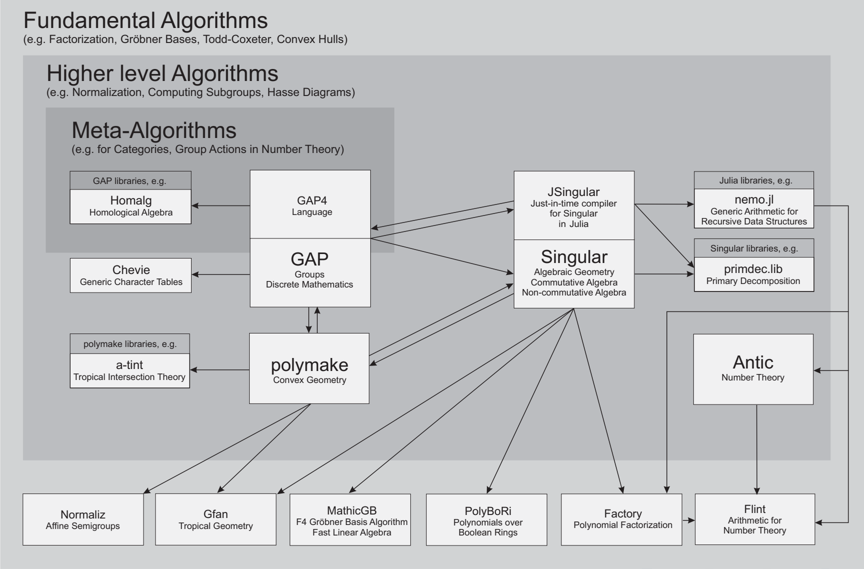
Resultant benchmark: Nemo

```
R = GF(17^11)
S = R[y]
T = S/(y^3 + 3x*y + 1)
U = T[z]
f = T(3y^2 + y + x)*z^2 + T((x + 2)*y^2 + x + 1)*z + T(4x*y + 3)
g = T(7y^2 - y + 2x + 7)*z^2 + T(3y^2 + 4x + 1)*z + T((2x + 1)*y + 1)
s = f^12
t = (s + g)^12
time r = resultant(s, t)
```

This benchmark is designed to test generics and computation of the resultant.

SageMath 6.8	Magma V2.21-4	Nemo-0.3
179907s	82s	2s

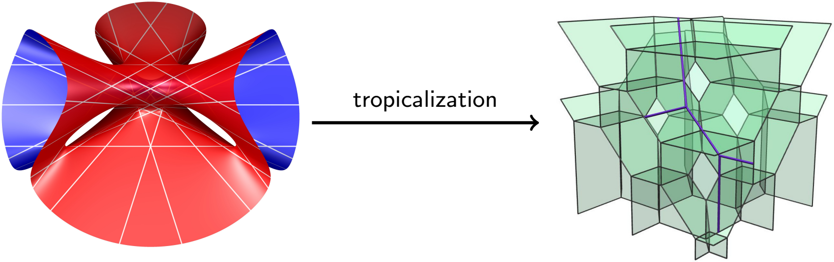
Fourth Challenge: Integration and Interaction of the Computer Algebra Systems and Libraries Involved



Software for Tropical Geometry

What is tropical geometry?

- Tropical geometry is a piece-vise linear version of algebraic geometry.
- Algebraic objects become discrete / polyhedral objects. Tropical varieties are polyhedral complexes.



GFAN (Anders Jensen) and SINGULAR

1. GFAN
   - first system capable of computing tropical varieties, based on
     - the work of Fukuda-Jensen-Thomas on computing Gröbner fans;
     - the work of Bogart-Jensen-Speyer-Sturmfels-Thomas on computing tropical varieties;
   - algorithms for $\mathbb{Q}$ with trivial valuation ($\rightarrow$ polyhedral fans).
2. SINGULAR
   - algorithms for $\mathbb{Q}$ with p-adic valuation ($\rightarrow$ polyhedral complexes);
   - use commutative algebra, based on the work of Thomas Markwig-Yue Ren, to lift to the trivial valuation case.

See talk by Thomas Markwig.

A-TINT (Simon Hampe)

- There is a notion of tropical intersection theory (Mikhalkin, Allermann-Rau, Francois-Rau, Shaw) on *smooth* tropical varieties.

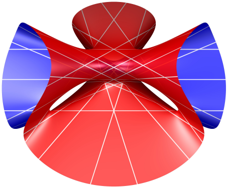
A-TINT (Simon Hampe)

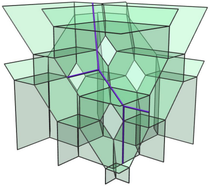
- There is a notion of tropical intersection theory (Mikhalkin, Allermann-Rau, Francois-Rau, Shaw) on *smooth* tropical varieties.

What is A-TINT?

- It is an extension of POLYMAKE (soon to be bundled with POLYMAKE!) for tropical intersection theory.
- Features include: Intersection products for the tropical torus and for smooth surfaces (the latter based on algorithms by Dennis Diefenbach and Kristin Shaw), divisors of rational functions, matroidal fans, moduli spaces of rational curves (including Hurwitz cycles),...
- Webpage: `https://github.com/simonhampe/atint`

Example: Classical Intersection Theory

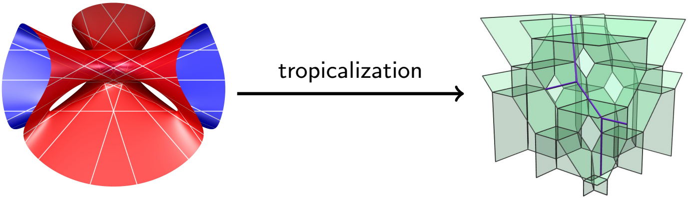




Example

```
> LIB"schubert.lib";
> variety G = Grassmannian(2,4);
> def R = G.baseRing; setring R;
> sheaf S = makeSheaf(G,subBundle);
> sheaf B = dualSheaf(S)^3;
> integral(G,topChernClass(B));
```

Example: Classical Intersection Theory



Example

```
> LIB"schubert.lib";
> variety G = Grassmannian(2,4);
> def R = G.baseRing; setring R;
> sheaf S = makeSheaf(G,subBundle);
> sheaf B = dualSheaf(S)^3;
> integral(G,topChernClass(B));
  27
```

Fifth Challenge: Easy Access

As there are more and more people applying constructive methods from algebraic geometry to other fields, we should considerably ease the access to the systems which offer implementations of the methods.

Fifth Challenge: Easy Access

As there are more and more people applying constructive methods from algebraic geometry to other fields, we should considerably ease the access to the systems which offer implementations of the methods.

New EU project in this this direction: OpenDreamKit.

Fifth Challenge: Easy Access