Massively Parallel Methods and Other Trends in the Design of Computer Algebra Systems - With a Focus on Feynman Integral Reduction

Janko Boehm

Technische Universität Kaiserslautern

March 20, 2019

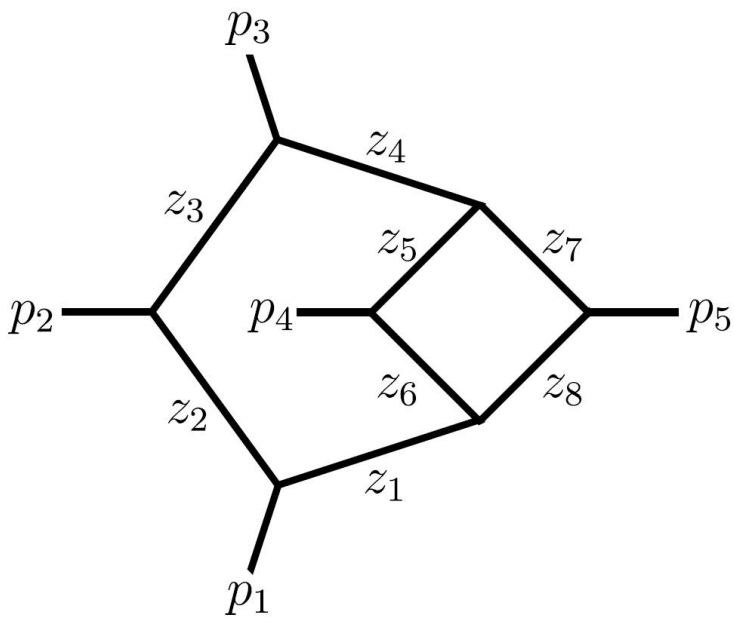
supported by

TR 195
SYMBOLIC TOOLS

Algorithmic Setup

Motivating example

Determine the IBP relations for the non-planar hexgon box



J. Boehm, A. Georgoudis, K. J. Larsen, H. Schönemann, Y. Zhang. *Complete integration-by-parts reductions of the non-planar hexagon-box via module intersections.* J. High Energ. Phys. (2018).

Integration-by-parts identities

Baikov representation with no doubled propagators for Feynman graph of genus L and E independent external legs, and propagators D_j

$$\int d^D l_1 \cdots \int d^D l_L \prod_{j=1}^{k} \frac{1}{D_j^{a_j}} \qquad a_1, \ldots, a_m \leq 1,\ a_{m+1}, \ldots, a_k \leq 0$$

where m is the number of internal edges and $k = LE + \frac{L(L+1)}{2}$.

Integration-by-parts identities

Baikov representation with no doubled propagators for Feynman graph of genus L and E independent external legs, and propagators D_j

$$\int d^D l_1 \cdots \int d^D l_L \prod_{j=1}^{k} \frac{1}{D_j^{a_j}} \qquad a_1, \ldots, a_m \leq 1,\ a_{m+1}, \ldots, a_k \leq 0$$

where m is the number of internal edges and $k = LE + \frac{L(L+1)}{2}$.
Using the balancing condition, write the Gram matrix

$$S = (v_i \cdot v_j)_{i,j=1,\ldots,L+E}$$

with external and internal momenta $v_1, \ldots, v_{L+E}$ in terms of scalar products $v_i \cdot v_j$ and change coordinate system to internal propagators z_i and ISP, and let

$$F = \det S$$

IBP Relations

Integration-by-parts identities arise from total differentials

$$0 = \int d\left(\sum_i \frac{a_i(z_1,\ldots,z_m)}{z_1\cdots z_m} F(z)^{\frac{D-L-E-1}{2}} dz_1\cdots\widehat{dz_i}\cdots dz_m\right)$$

which translate into a relations

$$\sum_{i=1}^{m} a_i(z)\frac{\partial F(z)}{\partial z_i} + b(z)F(z) = 0. \qquad (*)$$

If one knows a full set of such relations up to a sufficiently high degree d in z and with $z_i \mid a_i(z)$, then any Feynman integal can be reduced to master integrals.

IBP Relations

Integration-by-parts identities arise from total differentials

$$0 = \int d\left(\sum_i \frac{a_i(z_1,\ldots,z_m)}{z_1\cdots z_m} F(z)^{\frac{D-L-E-1}{2}} dz_1\cdots\widehat{dz_i}\cdots dz_m\right)$$

which translate into a relations

$$\sum_{i=1}^{m} a_i(z)\frac{\partial F(z)}{\partial z_i} + b(z)F(z) = 0. \qquad (*)$$

If one knows a full set of such relations up to a sufficiently high degree d in z and with $z_i \mid a_i(z)$, then any Feynman integal can be reduced to master integrals. So if

$$M_1 = \langle a(z) \text{ with } (*)\rangle \qquad M_2 = \langle z_i e_i \mid i \leq m\rangle + \langle e_i \mid i > m\rangle$$

we have to calculate $(M_1 \cap M_2)_{\leq d}$.

Computer Algebra

Singular

- Open Source computer algebra system for polynomial computations, over 30 development teams worldwide, over 140 libraries.

`https://www.singular.uni-kl.de/`

Singular

- Open Source computer algebra system for polynomial computations, over 30 development teams worldwide, over 140 libraries.

`https://www.singular.uni-kl.de/`

- Founded by Gert-Martin Greuel, Gerhard Pfister, Hans Schönemann. Head of team: Wolfram Decker.

Features

Commutative Algebra:

- Gröbner bases over fields and integers, free resolutions
- Local computations
- Primary decomposition
- Normalization
- Polynomial factorization: Factory
- Non-commutative features: Plural, Letterplace

Features

Commutative Algebra:

- Gröbner bases over fields and integers, free resolutions
- Local computations
- Primary decomposition
- Normalization
- Polynomial factorization: Factory
- Non-commutative features: Plural, Letterplace

Singularities:

- Classification
- Deformation theory

Features

Commutative Algebra:

- Gröbner bases over fields and integers, free resolutions
- Local computations
- Primary decomposition
- Normalization
- Polynomial factorization: Factory
- Non-commutative features: Plural, Letterplace

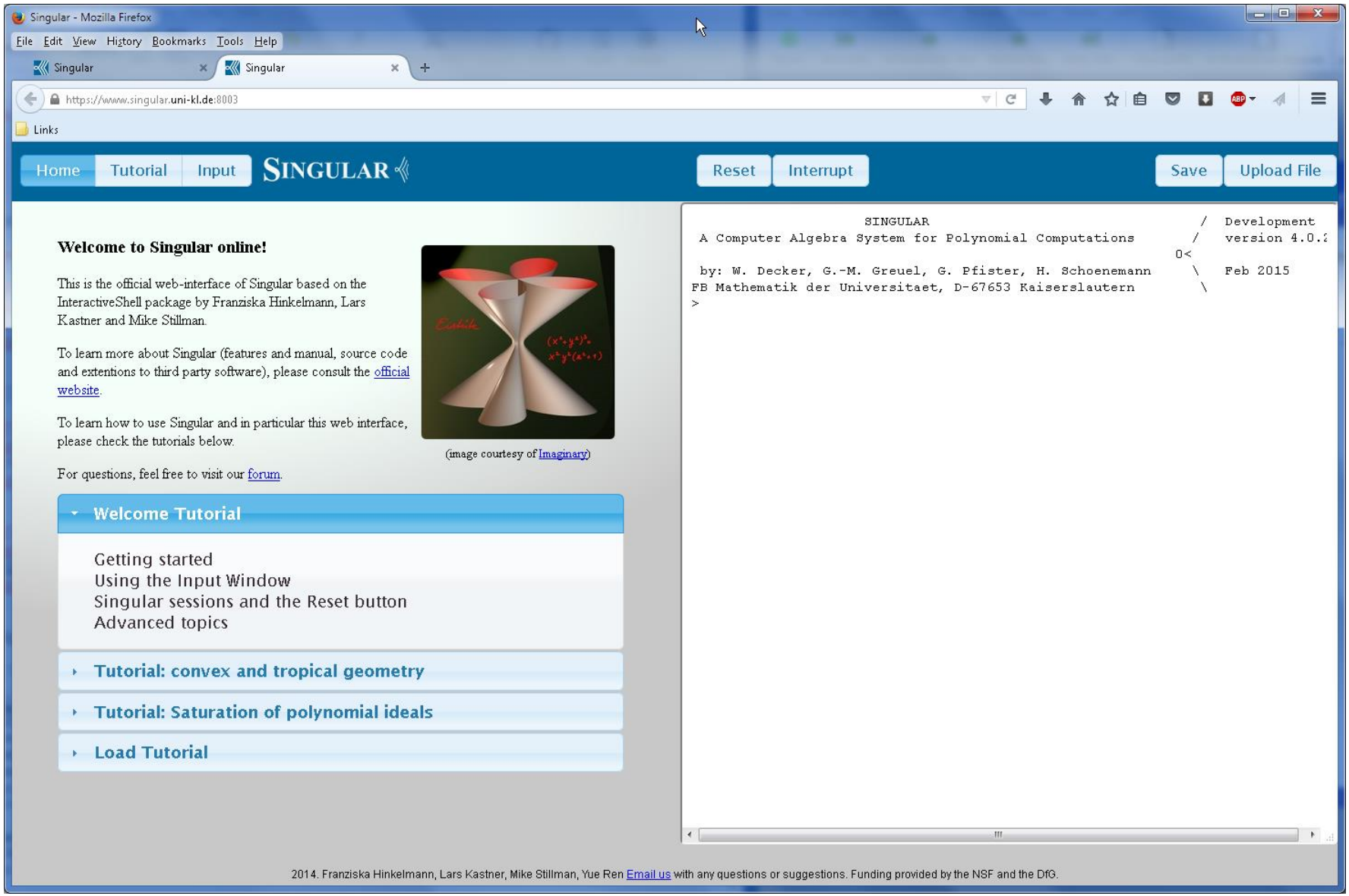
Singularities:

- Classification
- Deformation theory

Algebraic Geometry:

- Hironaka resolution of singularities
- Sheaf cohomology
- Rational parametrizations
- tropicalizations
- GIT fans

Singular Online



The Main Computational Tool: Gröbner Bases

Division with remainder in one variable successively eliminates the highest power.

The Main Computational Tool: Gröbner Bases

Division with remainder in one variable successively eliminates the highest power. In more than one variable we have to fix a **monomial ordering** (a total ordering compatible with multiplication).

The Main Computational Tool: Gröbner Bases

Division with remainder in one variable successively eliminates the highest power. In more than one variable we have to fix a **monomial ordering** (a total ordering compatible with multiplication).
Divide $x^2 - y^2$ durch $x^2 + y$ und $xy + x$ with respect to lexicographic ordering.

$$\begin{array}{rl} x^2 - y^2 = & 1 \cdot (x^2 + y) + (-y^2 - y) \\ \hline x^2 + y & \\ \hline -y^2 - y & \end{array}$$

The Main Computational Tool: Gröbner Bases

Division with remainder in one variable successively eliminates the highest power. In more than one variable we have to fix a **monomial ordering** (a total ordering compatible with multiplication).
Divide $x^2 - y^2$ durch $x^2 + y$ und $xy + x$ with respect to lexicographic ordering.

$$\begin{array}{rl} x^2 - y^2 = & 1 \cdot (x^2 + y) + (-y^2 - y) \\ \hline x^2 + y & \\ \hline -y^2 - y & \end{array}$$

so remainder $\neq 0$, but

$$x^2 - y^2 = -y\,(x^2 + y) + x\,(xy + x) \in I := \langle x^2 + y, xy + x \rangle$$

The Main Computational Tool: Gröbner Bases

Division with remainder in one variable successively eliminates the highest power. In more than one variable we have to fix a **monomial ordering** (a total ordering compatible with multiplication).
Divide $x^2 - y^2$ durch $x^2 + y$ und $xy + x$ with respect to lexicographic ordering.

$$\begin{array}{rl} x^2 - y^2 = & 1 \cdot (x^2 + y) + (-y^2 - y) \\ \hline x^2 + y & \\ -y^2 - y & \end{array}$$

so remainder $\neq 0$, but

$$x^2 - y^2 = -y\,(x^2 + y) + x\,(xy + x) \in I := \langle x^2 + y, xy + x \rangle$$

Problem: Lead terms cancel, division algorithm can't do that.

The Main Computational Tool: Gröbner Bases

Division with remainder in one variable successively eliminates the highest power. In more than one variable we have to fix a **monomial ordering** (a total ordering compatible with multiplication).
Divide $x^2 - y^2$ durch $x^2 + y$ und $xy + x$ with respect to lexicographic ordering.

$$\begin{array}{l} x^2 - y^2 = \quad 1 \cdot (x^2 + y) + (-y^2 - y) \\ \underline{x^2 + y} \\ -y^2 - y \end{array}$$

so remainder $\neq 0$, but

$$x^2 - y^2 = -y\,(x^2 + y) + x\,(xy + x) \in I := \langle x^2 + y, xy + x \rangle$$

Problem: Lead terms cancel, division algorithm can't do that.
Solution: Add $y^2 + y$ to the divisor set. The result is a **Gröbner basis** G of I. Then

The Main Computational Tool: Gröbner Bases

Division with remainder in one variable successively eliminates the highest power. In more than one variable we have to fix a **monomial ordering** (a total ordering compatible with multiplication).
Divide $x^2 - y^2$ durch $x^2 + y$ und $xy + x$ with respect to lexicographic ordering.

$$\begin{array}{ll} x^2 - y^2 = & 1 \cdot (x^2 + y) + (-y^2 - y) \\ \hline x^2 + y & \\ -y^2 - y & \end{array}$$

so remainder $\neq 0$, but

$$x^2 - y^2 = -y\,(x^2 + y) + x\,(xy + x) \in I := \langle x^2 + y, xy + x \rangle$$

Problem: Lead terms cancel, division algorithm can't do that.
Solution: Add $y^2 + y$ to the divisor set. The result is a **Gröbner basis** G of I. Then

$$f \in I \iff NF(f, G) = 0$$

The Main Computational Tool: Gröbner Bases

Gröbner Bases can be used for fundamental computations with ideals and modules:

Example

- eliminate variables ($\rightarrow$ birational geometry),
- determine intersections,
- compute ideal quotients and saturations,
- compute syzygies.

Greuel, G.-M., Pfister, G.: *A Singular Introduction to Commutative Algebra.* Springer.

Algorithmic building blocks

Definition

Let $R = K[x_1, \ldots, x_n]$. For a matrix $G \in R^{t \times s}$ we define the **syzygy module** of G as the kernel

$$\mathrm{Syz}(G) = \ker(G) \subset R^s,$$

and the elements are called **syzygies** of G. Any matrix $S \in R^{s \times g}$ with

$$\mathrm{im}(S) = \mathrm{Syz}(G),$$

we call a **syzygy matrix** of G.

Computing module intersections

Lemma

Let $R = K[x_1, \ldots, x_n]$. *Given modules* $M_1 = \langle f_1, \ldots, f_a \rangle \subset R^t$ *and* $M_2 = \langle g_1, \ldots, g_b \rangle \subset R^t$, *consider the matrix*

$$G = \begin{pmatrix} f_1 & \ldots & f_a & 0 & \ldots & 0 & E_t \\ 0 & \ldots & 0 & g_1 & \ldots & g_b & E_t \end{pmatrix},$$

with unit matrix E_t

Computing module intersections

Lemma

Let $R = K[x_1, \ldots, x_n]$. *Given modules* $M_1 = \langle f_1, \ldots, f_a \rangle \subset R^t$ *and* $M_2 = \langle g_1, \ldots, g_b \rangle \subset R^t$, *consider the matrix*

$$G = \begin{pmatrix} f_1 & \ldots & f_a & 0 & \ldots & 0 & E_t \\ 0 & \ldots & 0 & g_1 & \ldots & g_b & E_t \end{pmatrix},$$

with unit matrix E_t *and let* $S = \mathrm{syz}(G) \in R^{(a+b+t)\times c}$ *be a syzygy matrix of* G.

Computing module intersections

Lemma

Let $R = K[x_1, \ldots, x_n]$. Given modules $M_1 = \langle f_1, \ldots, f_a \rangle \subset R^t$ and $M_2 = \langle g_1, \ldots, g_b \rangle \subset R^t$, consider the matrix

$$G = \begin{pmatrix} f_1 & \ldots & f_a & 0 & \ldots & 0 & E_t \\ 0 & \ldots & 0 & g_1 & \ldots & g_b & E_t \end{pmatrix},$$

with unit matrix E_t and let $S = \mathrm{syz}(G) \in R^{(a+b+t)\times c}$ be a syzygy matrix of G. Then the columns of the last t rows of S generate $M_1 \cap M_2$.

Computing syzygies

Given a global monomial ordering $>$ and $G = (g_1, \ldots, g_s)$ every successful Buchberger test

$$\mathrm{NF}(S(g_i, g_j), G) = 0$$

gives an expression

Computing syzygies

Given a global monomial ordering $>$ and $G = (g_1, \ldots, g_s)$ every successful Buchberger test

$$\mathrm{NF}(S(g_i, g_j), G) = 0$$

gives an expression

$$\underbrace{\frac{\mathrm{lcm}(L(g_i), L(g_j))}{LT(g_i)}}_{m} \cdot g_i \quad - \quad \underbrace{\frac{\mathrm{lcm}(L(g_i), L(g_j))}{LT(g_j)}}_{w} \cdot g_j \quad - \quad \sum_{k=1}^{s} a_k g_k \quad = \quad 0$$

and, hence, a syzygy

Computing syzygies

Given a global monomial ordering $>$ and $G = (g_1, \ldots, g_s)$ every successful Buchberger test

$$\mathrm{NF}(S(g_i, g_j), G) = 0$$

gives an expression

$$\underbrace{\frac{\mathrm{lcm}(L(g_i), L(g_j))}{LT(g_i)}}_{m} \cdot g_i \quad - \quad \underbrace{\frac{\mathrm{lcm}(L(g_i), L(g_j))}{LT(g_j)}}_{w} \cdot g_j \quad - \quad \sum_{k=1}^{s} a_k g_k \quad = \quad 0$$

and, hence, a syzygy $\mathrm{s}(g_i, g_j) := m \cdot e_i - w \cdot e_j - \sum_{k=1}^{s} a_k e_k$

$$\in \mathrm{Syz}(G) \subset R^s$$

Computing syzygies

Given a global monomial ordering $>$ and $G = (g_1, \ldots, g_s)$ every successful Buchberger test

$$\mathrm{NF}(S(g_i, g_j), G) = 0$$

gives an expression

$$\underbrace{\frac{\mathrm{lcm}(L(g_i), L(g_j))}{LT(g_i)}}_{m} \cdot g_i \quad - \quad \underbrace{\frac{\mathrm{lcm}(L(g_i), L(g_j))}{LT(g_j)}}_{w} \cdot g_j \quad - \quad \sum_{k=1}^{s} a_k g_k \quad = \quad 0$$

and, hence, a syzygy $\mathrm{s}(g_i, g_j) := m \cdot e_i - w \cdot e_j - \sum_{k=1}^{s} a_k e_k \in \mathrm{Syz}(G) \subset R^s$

Theorem (Schreyer)

With regard to a clever choice of monomial odering $>_G$ on R^s the $\mathrm{s}(g_i, g_j)$ form a Gröbner basis and, hence, a generating system of $\mathrm{Syz}(G)$.

Computing syzygies

Given a global monomial ordering $>$ and $G = (g_1, \ldots, g_s)$ every successful Buchberger test

$$\mathrm{NF}(S(g_i, g_j), G) = 0$$

gives an expression

$$\underbrace{\frac{\mathrm{lcm}(L(g_i), L(g_j))}{LT(g_i)}}_{m} \cdot g_i \quad - \quad \underbrace{\frac{\mathrm{lcm}(L(g_i), L(g_j))}{LT(g_j)}}_{w} \cdot g_j \quad - \quad \sum_{k=1}^{s} a_k g_k \quad = \quad 0$$

and, hence, a syzygy $\mathrm{s}(g_i, g_j) := m \cdot e_i - w \cdot e_j - \sum_{k=1}^{s} a_k e_k \in \mathrm{Syz}(G) \subset R^s$

Theorem (Schreyer)

With regard to a clever choice of monomial odering $>_G$ on R^s the $\mathrm{s}(g_i, g_j)$ form a Gröbner basis and, hence, a generating system of $\mathrm{Syz}(G)$.

$$x^\alpha e_i >_G x^\beta e_j :\Longleftrightarrow L(x^\alpha g_i) > L(x^\beta g_j) \text{ or } \left(L(x^\alpha g_i) = L(x^\beta g_j) \text{ and } i < j\right)$$

Syzygies from Laplace expansion

Generators of M_1 can be computed via Gröbner bases and Schreyer's algorithm.

Syzygies from Laplace expansion

Generators of M_1 can be computed via Gröbner bases and Schreyer's algorithm.
However, a shorter, more symmetric generating system of M_1 can be obtained via a theoretical result using the Józefiak complex, which describes a resolution of the ideal of submaximal minors of a generic symmetric matrix by Laplace expansion.

J. Boehm, A. Georgoudis, K. J. Larsen, M. Schulze, Y. Zhang. *Complete sets of logarithmic vector fields for integration-by-parts identities of Feynman integrals.* Phys. Rev. D (2018).

This speeds up the subsequent module intersection computation.

Localization

Definition

Given monomial orderings $>_1$ and $>_2$ on the monomials in $z_1, \ldots, z_n$ and $c_1, \ldots, c_r$, respectively, a monomial ordering $>$ is given by

$$z^\alpha c^\beta > z^{\alpha'} c^{\beta'} \quad :\Longleftrightarrow \quad z^\alpha >_1 z^{\alpha'} \text{ or } (z^\alpha = z^{\alpha'} \text{ and } c^\beta >_2 c^{\beta'})$$

We call $>$ the *block ordering* $(>_1, >_2)$ associated to $>_1$and $>_2$.

Localization

Definition

Given monomial orderings $>_1$ and $>_2$ on the monomials in $z_1, \ldots, z_n$ and $c_1, \ldots, c_r$, respectively, a monomial ordering $>$ is given by

$$z^\alpha c^\beta > z^{\alpha'} c^{\beta'} \quad :\Longleftrightarrow \quad z^\alpha >_1 z^{\alpha'} \text{ or } (z^\alpha = z^{\alpha'} \text{ and } c^\beta >_2 c^{\beta'})$$

We call $>$ the *block ordering* $(>_1, >_2)$ associated to $>_1$and $>_2$.

Lemma (Localization)

Let $A = \mathbb{Q}(c_1, \ldots, c_r)[z_1, \ldots, z_n]$, *let* $B = \mathbb{Q}[z_1, \ldots, z_n, c_1, \ldots, c_r]$, *let* $v_1, \ldots, v_l$ *be vectors with entries in* B, *and define*

$$U = \langle v_1, \ldots, v_l \rangle \subset A^t \qquad U' = \langle v_1, \ldots, v_l \rangle \subset B^t$$

Let $G \subset B^t$ *be a Gröbner basis of* U' *with respect to a global block ordering* $(>_1, >_2)$ *with blocks* $z_1, \ldots, z_n > c_1, \ldots, c_r$. *Then* G *is also a Gröbner basis of* U *with respect to* $>_1$.

Trimming of generating systems

Algorithm (Trimming)

Given a generating system $g_1, \ldots, g_l$ of a submodule $U \subset A^t$ with polynomial coefficients in $\mathbb{Z}[c_1, \ldots, c_r]$, apply iteratively:

Trimming of generating systems

Algorithm (Trimming)

Given a generating system $g_1, \ldots, g_l$ of a submodule $U \subset A^t$ with polynomial coefficients in $\mathbb{Z}[c_1, \ldots, c_r]$, apply iteratively:

1. *Substitute the c_i in the g_j by pairwise different large prime numbers p_i obtaining polynomials $h_j \in \mathbb{Z}[z_1, \ldots, z_n]$.*

Trimming of generating systems

Algorithm (Trimming)

Given a generating system $g_1, \ldots, g_l$ of a submodule $U \subset A^t$ with polynomial coefficients in $\mathbb{Z}[c_1, \ldots, c_r]$, apply iteratively:

1. *Substitute the c_i in the g_j by pairwise different large prime numbers p_i obtaining polynomials $h_j \in \mathbb{Z}[z_1, \ldots, z_n]$.*
2. *Choose a large prime p different to the primes p_i. Apply the canonical map $\mathbb{Z}[z_1, \ldots, z_n] \to \mathbb{F}_p[z_1, \ldots, z_n]$ to the h_j obtaining $\overline{h_1}, \ldots, \overline{h_l}$.*

Trimming of generating systems

Algorithm (Trimming)

Given a generating system $g_1, \ldots, g_l$ of a submodule $U \subset A^t$ with polynomial coefficients in $\mathbb{Z}[c_1, \ldots, c_r]$, apply iteratively:

1. *Substitute the c_i in the g_j by pairwise different large prime numbers p_i obtaining polynomials $h_j \in \mathbb{Z}[z_1, \ldots, z_n]$.*
2. *Choose a large prime p different to the primes p_i. Apply the canonical map $\mathbb{Z}[z_1, \ldots, z_n] \to \mathbb{F}_p[z_1, \ldots, z_n]$ to the h_j obtaining $\overline{h_1}, \ldots, \overline{h_l}$.*
3. *Choose an integer $j_0 \in \{1, \ldots, l\}$, such that h_{j_0} has large height.*

Trimming of generating systems

Algorithm (Trimming)

Given a generating system $g_1, \ldots, g_l$ of a submodule $U \subset A^t$ with polynomial coefficients in $\mathbb{Z}[c_1, \ldots, c_r]$, apply iteratively:

1. *Substitute the c_i in the g_j by pairwise different large prime numbers p_i obtaining polynomials $h_j \in \mathbb{Z}[z_1, \ldots, z_n]$.*
2. *Choose a large prime p different to the primes p_i. Apply the canonical map $\mathbb{Z}[z_1, \ldots, z_n] \to \mathbb{F}_p[z_1, \ldots, z_n]$ to the h_j obtaining $\overline{h_1}, \ldots, \overline{h_l}$.*
3. *Choose an integer $j_0 \in \{1, \ldots, l\}$, such that h_{j_0} has large height.*
4. *Compute the reduced Gröbner bases G_1 of*
$$\left\langle \overline{h_j} \mid j = 1, \ldots, l \right\rangle$$
and G_2 of
$$\left\langle \overline{h_j} \mid j = 1, \ldots, l \text{ with } j \neq j_0 \right\rangle.$$
5. *If $G_1 = G_2$ return $\{g_j \mid j = 1, \ldots, l \text{ and } j \neq j_0\}$.*

General structure of IBP Algorithm

- Pass to cuts required for deriving complete IBP identities

General structure of IBP Algorithm

- Pass to cuts required for deriving complete IBP identities
- On each cut, compute generating system of $M_1 \cap M_2$ using Gröbner bases over the field of rational functions $\mathbb{K}$ in the external momenta and D using localization method.

General structure of IBP Algorithm

- Pass to cuts required for deriving complete IBP identities
- On each cut, compute generating system of $M_1 \cap M_2$ using Gröbner bases over the field of rational functions $\mathbb{K}$ in the external momenta and D using localization method.
- Trim generating system.

General structure of IBP Algorithm

- Pass to cuts required for deriving complete IBP identities
- On each cut, compute generating system of $M_1 \cap M_2$ using Gröbner bases over the field of rational functions $\mathbb{K}$ in the external momenta and D using localization method.
- Trim generating system.
- Generate a linear system of relations between integrals occuring in IBPs in $(M_1 \cap M_2)_{\leq d}$.

General structure of IBP Algorithm

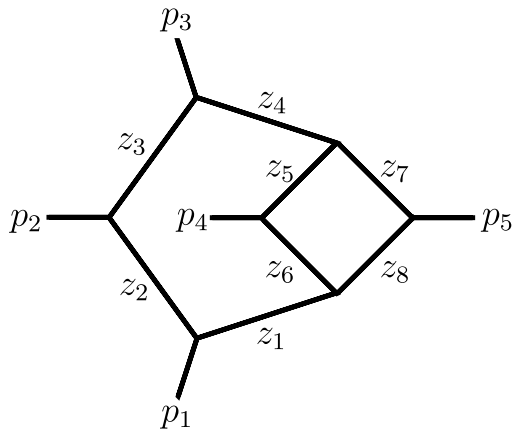
- Pass to cuts required for deriving complete IBP identities
- On each cut, compute generating system of $M_1 \cap M_2$ using Gröbner bases over the field of rational functions $\mathbb{K}$ in the external momenta and D using localization method.
- Trim generating system.
- Generate a linear system of relations between integrals occuring in IBPs in $(M_1 \cap M_2)_{\leq d}$.
- Compute reduced row echelon form (RREF) over $\mathbb{K}$.

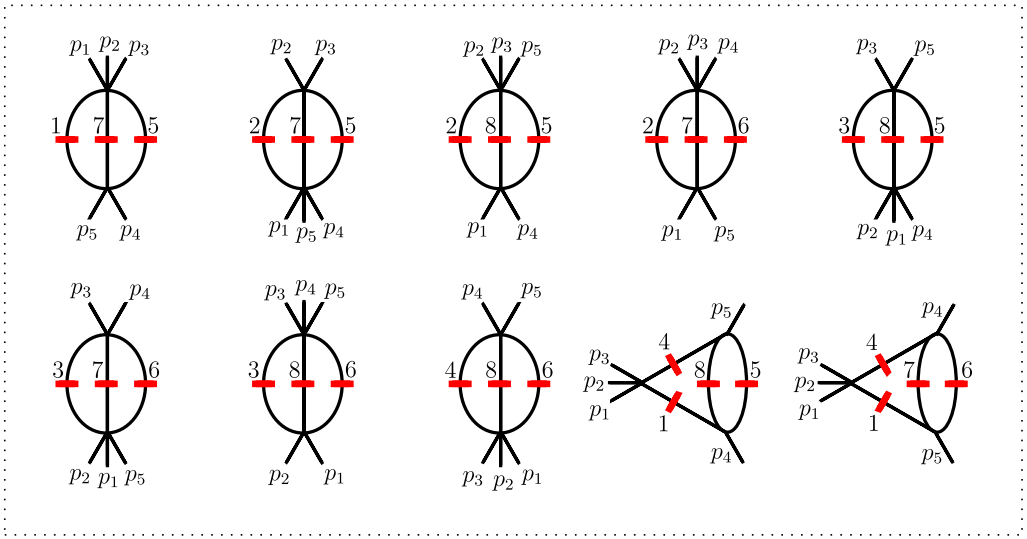
General structure of IBP Algorithm

- Pass to cuts required for deriving complete IBP identities
- On each cut, compute generating system of $M_1 \cap M_2$ using Gröbner bases over the field of rational functions $\mathbb{K}$ in the external momenta and D using localization method.
- Trim generating system.
- Generate a linear system of relations between integrals occuring in IBPs in $(M_1 \cap M_2)_{\leq d}$.
- Compute reduced row echelon form (RREF) over $\mathbb{K}$.
- Recombine cuts

Pass to cuts

Cuts required for deriving complete IBP identities for the non-planar hexagon-box diagram:





Timings for module intersection for hexagon box

Runtimes for cuts of hexagon box diagrams:

cuts	157	257	258	267	358	367	368	468	1458	1467
time/s	220	40	300	740	400	700	24	800	50	200
size/MB	68	25	49	100	97	80	10	21	5	10
size/MB (simplified)	10	1.4	3.1	2.8	3.7	3.6	1.6	1.6	3.6	4.1

Row reduction: Dynamic reordering of integrals

- Over a function field with a small number of variables, determine a trace to a good row echelon form (REF) via a pivoting strategy aimed at small size and sparseness. Do row and column pivoting, i.e., **dynamic reordering of the Feynman integrals at runtime**.

Row reduction: Dynamic reordering of integrals

- Over a function field with a small number of variables, determine a trace to a good row echelon form (REF) via a pivoting strategy aimed at small size and sparseness. Do row and column pivoting, i.e., **dynamic reordering of the Feynman integrals at runtime**.
- Compute the reduced row echelon form (RREF).

Row reduction: Dynamic reordering of integrals

- Over a function field with a small number of variables, determine a trace to a good row echelon form (REF) via a pivoting strategy aimed at small size and sparseness. Do row and column pivoting, i.e., **dynamic reordering of the Feynman integrals at runtime**.
- Compute the reduced row echelon form (RREF).
- For the expressions for the target variables, change basis of free variables to the master integrals without column pivoting.

Row reduction: Dynamic reordering of integrals

- Over a function field with a small number of variables, determine a trace to a good row echelon form (REF) via a pivoting strategy aimed at small size and sparseness. Do row and column pivoting, i.e., **dynamic reordering of the Feynman integrals at runtime**.
- Compute the reduced row echelon form (RREF).
- For the expressions for the target variables, change basis of free variables to the master integrals without column pivoting.
- From the results over univariate function fields with regard to the trace, find the degrees of the rational function coefficients in the respective variables.

Row reduction: Dynamic reordering of integrals

- Over a function field with a small number of variables, determine a trace to a good row echelon form (REF) via a pivoting strategy aimed at small size and sparseness. Do row and column pivoting, i.e., **dynamic reordering of the Feynman integrals at runtime**.
- Compute the reduced row echelon form (RREF).
- For the expressions for the target variables, change basis of free variables to the master integrals without column pivoting.
- From the results over univariate function fields with regard to the trace, find the degrees of the rational function coefficients in the respective variables.
- Compute the coefficients via interpolation.

$\rightarrow$ reduction of non-planar hexagon-box integrals, with degree-four numerators, to a basis of 73 master integrals.

Computer algebra: New tools relevant to Feynman integral reduction

OSCAR

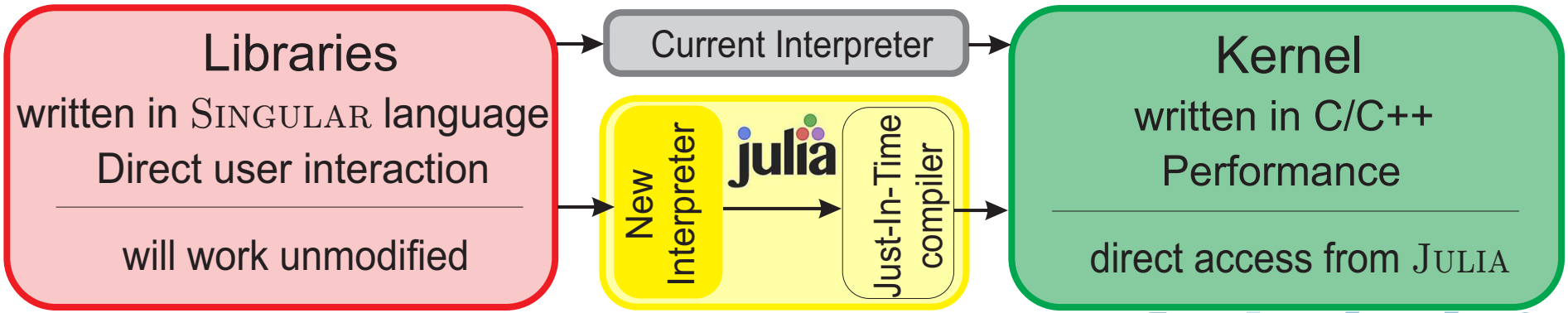
Next generation Open Source Computeralgebrasystem OSCAR developed in SFB TRR 195 "Symbolic Tools in Mathematics and Their Application":

OSCAR

Next generation Open Source Computeralgebrasystem OSCAR developed in SFB TRR 195 "Symbolic Tools in Mathematics and Their Application":

OSCAR

Next generation Open Source Computeralgebrasystem OSCAR developed in SFB TRR 195 "Symbolic Tools in Mathematics and Their Application":



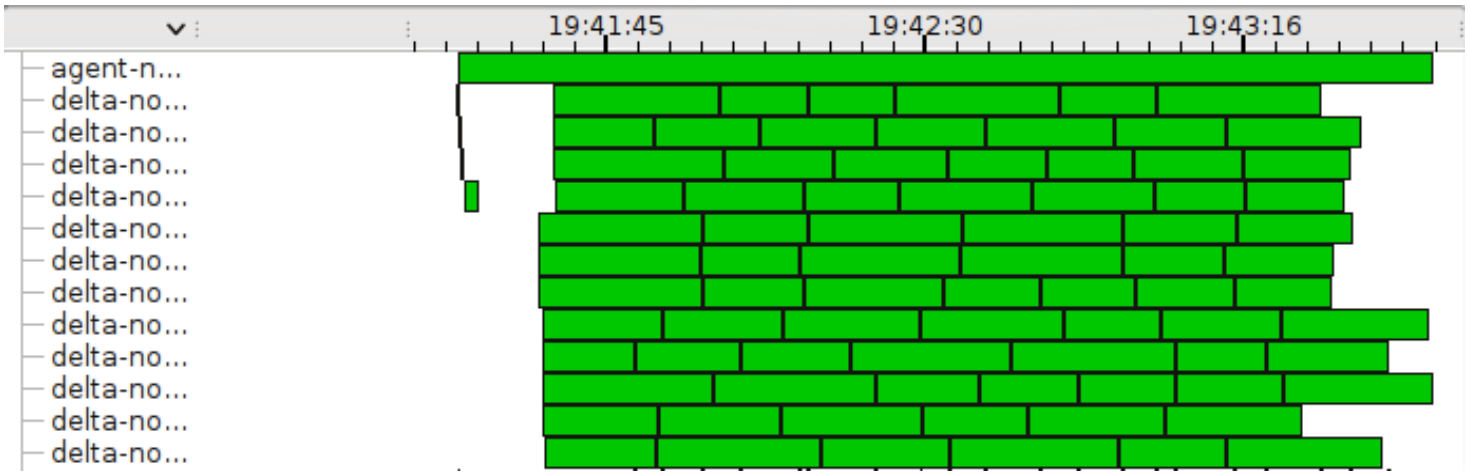
GPI-Space

- developed by Fraunhofer for Industrial Mathematics ITWM

GPI-Space

- developed by Fraunhofer for Industrial Mathematics ITWM

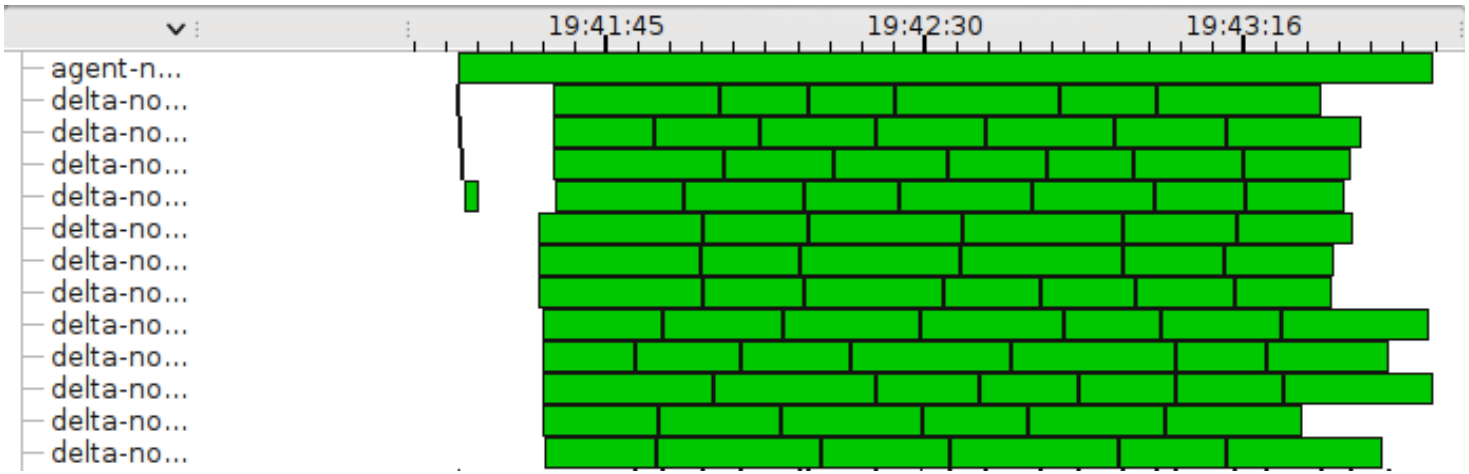
- task based workflow management system for massively parallel computations



GPI-Space

- developed by Fraunhofer for Industrial Mathematics ITWM

- task based workflow management system for massively parallel computations



- based on idea of separating coordination and computation.

Philosophy and Features of GPI-Space

- Separate, specialized language for coordination layer (Petri nets).

Philosophy and Features of GPI-Space

- Separate, specialized language for coordination layer (Petri nets).
- Both implementations of recurring patterns in the coordination layer, and computational core routines can be reused.

Philosophy and Features of GPI-Space

- Separate, specialized language for coordination layer (Petri nets).
- Both implementations of recurring patterns in the coordination layer, and computational core routines can be reused.
  %item Optimizations on either side can be done by the respective experts
- Complex coordination hidden from domain experts: automatic parallelization, cost optimized data transfers, latency hiding, adaptation to dynamic changes in the computing environment, resilience.

Philosophy and Features of GPI-Space

- Separate, specialized language for coordination layer (Petri nets).
- Both implementations of recurring patterns in the coordination layer, and computational core routines can be reused.
  %item Optimizations on either side can be done by the respective experts
- Complex coordination hidden from domain experts: automatic parallelization, cost optimized data transfers, latency hiding, adaptation to dynamic changes in the computing environment, resilience.

Philosophy and Features of GPI-Space

Sociable:

- Legacy applications (e.g. SINGULAR) can be used without changes as long as they can be called as a C-library.

Philosophy and Features of GPI-Space

Sociable:

- Legacy applications (e.g. SINGULAR) can be used without changes as long as they can be called as a C-library.
- Arbitrary computational tools written by domain experts can be mixed.

Philosophy and Features of GPI-Space

Sociable:

- Legacy applications (e.g. SINGULAR) can be used without changes as long as they can be called as a C-library.
- Arbitrary computational tools written by domain experts can be mixed.

Virtual memory layer allows to

Philosophy and Features of GPI-Space

Sociable:

- Legacy applications (e.g. SINGULAR) can be used without changes as long as they can be called as a C-library.
- Arbitrary computational tools written by domain experts can be mixed.

Virtual memory layer allows to

- scale computations beyond limitations of single machine.

Philosophy and Features of GPI-Space

Sociable:

- Legacy applications (e.g. SINGULAR) can be used without changes as long as they can be called as a C-library.
- Arbitrary computational tools written by domain experts can be mixed.

Virtual memory layer allows to

- scale computations beyond limitations of single machine.
- legacy applications to interoperate in an efficient way.

Philosophy and Features of GPI-Space

Sociable:

- Legacy applications (e.g. SINGULAR) can be used without changes as long as they can be called as a C-library.
- Arbitrary computational tools written by domain experts can be mixed.

Virtual memory layer allows to

- scale computations beyond limitations of single machine.
- legacy applications to interoperate in an efficient way.
- switch between low latency, low capacity memory and high latency, high capacity memory without changing the implementation.

Structure and Implementation of GPI-Space

Three main components:

- Distributed, scalable, resilient **runtime system** for huge dynamic environments: manages memory and computational resources. Scheduler assigns activities to resources w.r.t. both the needs of the current computations and the overall optimization goals.

GPI-Space: Scheduler

Structure and Implementation of GPI-Space

Three main components:

- Distributed, scalable, resilient **runtime system** for huge dynamic environments: manages memory and computational resources. Scheduler assigns activities to resources w.r.t. both the needs of the current computations and the overall optimization goals.

Structure and Implementation of GPI-Space

Three main components:

- Distributed, scalable, resilient **runtime system** for huge dynamic environments: manages memory and computational resources. Scheduler assigns activities to resources w.r.t. both the needs of the current computations and the overall optimization goals.
- Petri net based **workflow engine**: manages the full application state and is responsible for automatic parallelization and dependency tracking.

Structure and Implementation of GPI-Space

Three main components:

- Distributed, scalable, resilient **runtime system** for huge dynamic environments: manages memory and computational resources. Scheduler assigns activities to resources w.r.t. both the needs of the current computations and the overall optimization goals.
- Petri net based **workflow engine**: manages the full application state and is responsible for automatic parallelization and dependency tracking.
- **Virtual memory manager**: allows algorithmic building blocks to communicate, share partial results. Asynchronous data transfer managed by the runtime system rather than the domain applications, synchronisation is done in a way that tries to hide latency.

Petri nets

Introduced by Carl Adam Petri (1926–2010) in 1962 to describe concurrent asynchronous systems (this is how real world physics works!). In fact he invented them already much earlier to remember chemical reactions in school.

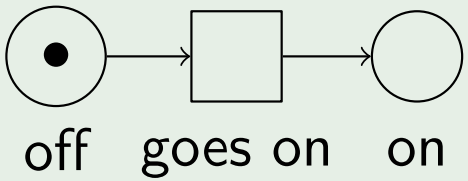
Petri nets

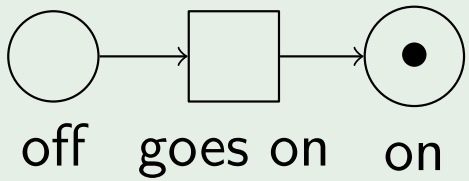
Petri nets are a graphical way to model algorithms: Consist out of **places** and **transitions**. By marking of places a state is described, if the conditions of a transition are satisfied, it changes the state.

Petri nets

Petri nets are a graphical way to model algorithms: Consist out of **places** and **transitions**. By marking of places a state is described, if the conditions of a transition are satisfied, it changes the state.

Example

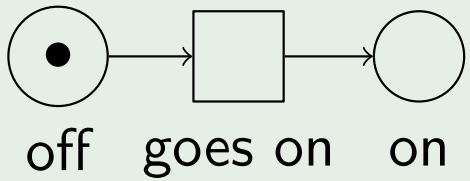


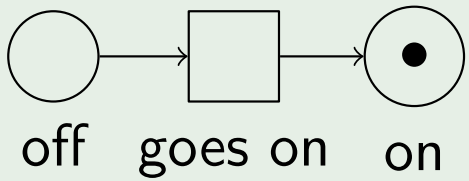


Petri nets

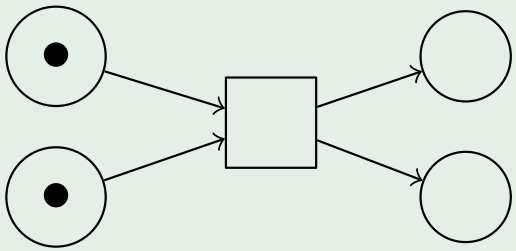
Petri nets are a graphical way to model algorithms: Consist out of **places** and **transitions**. By marking of places a state is described, if the conditions of a transition are satisfied, it changes the state.

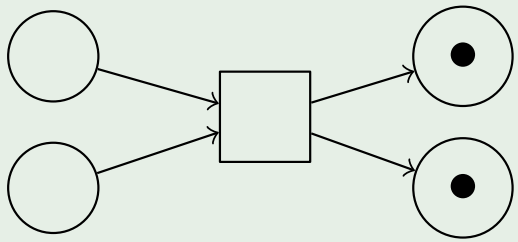
Example

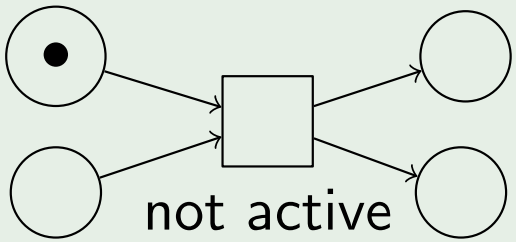




Example

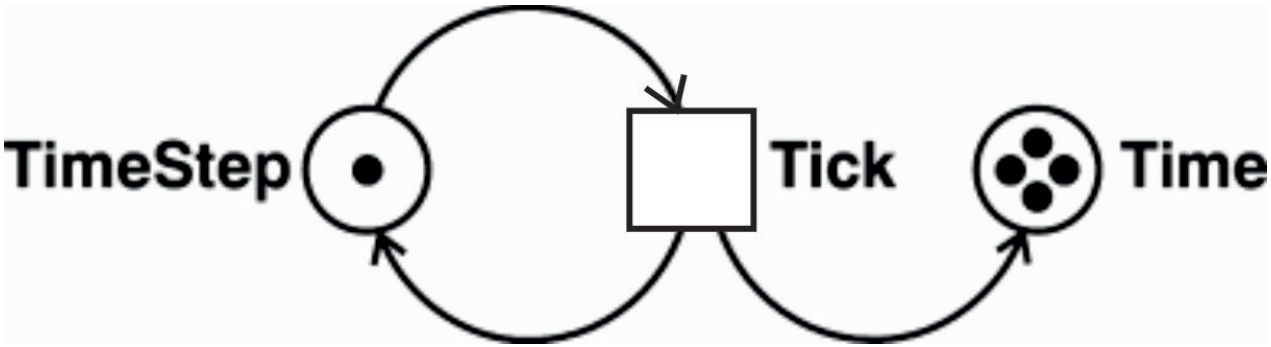




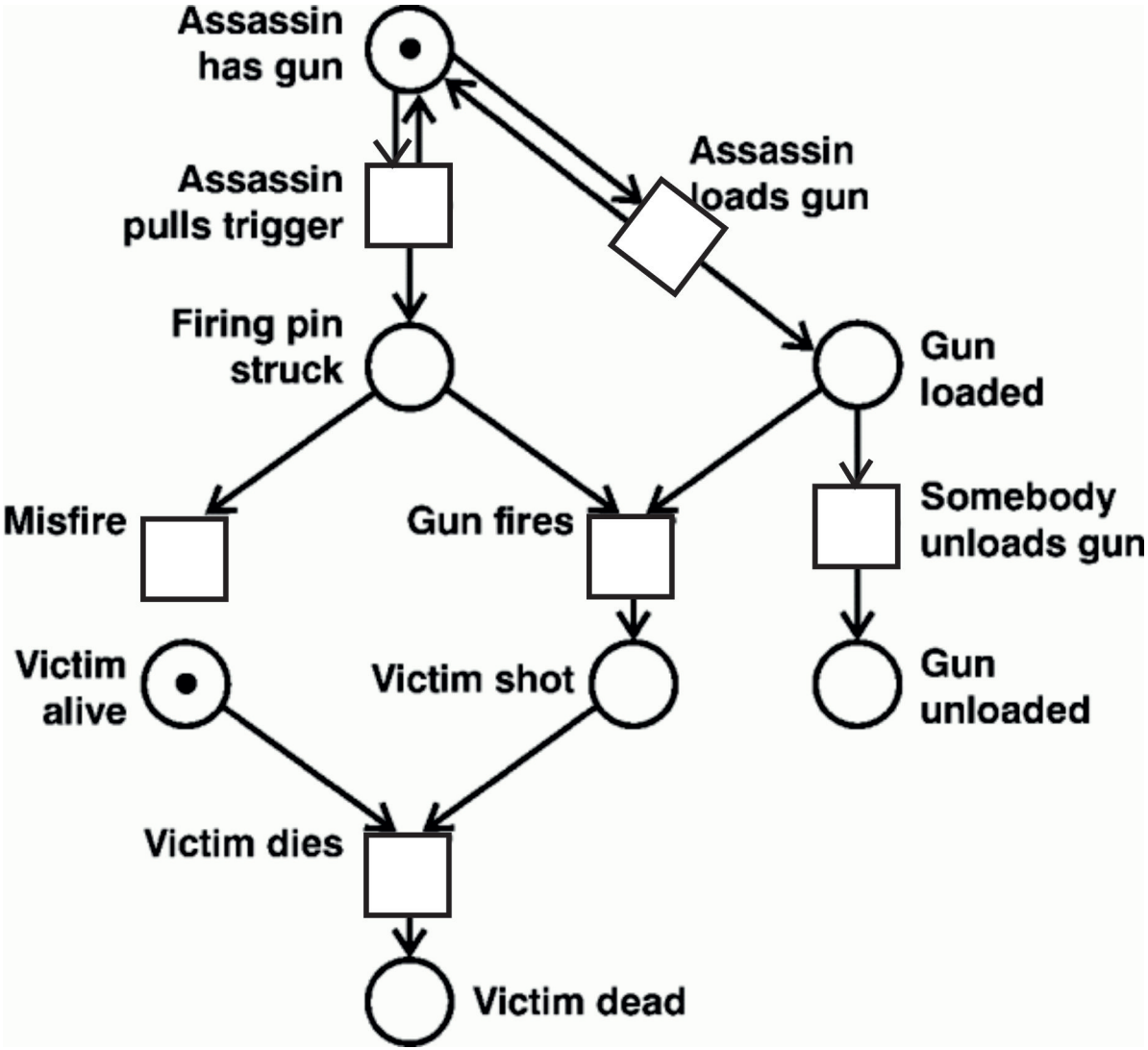


Real world examples

Clock at time $t = 4$:



Real world examples



Features of Petri nets

- Graphical, so accessible not only to experts in programming.

Features of Petri nets

- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.

Features of Petri nets

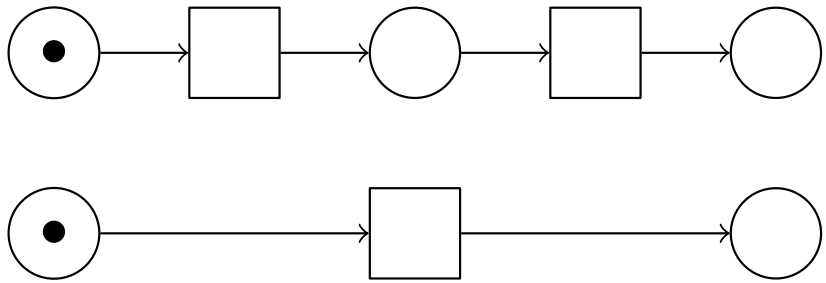
- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.
- Resembles mathematical thinking, functional/declarative programming, in contrast to imperative programming (*what* in contrast to *how*).
- Well suited for concurrent environments:

Features of Petri nets

- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.
- Resembles mathematical thinking, functional/declarative programming, in contrast to imperative programming (*what* in contrast to *how*).
- Well suited for concurrent environments:
  - Locality of dependencies, no global events.

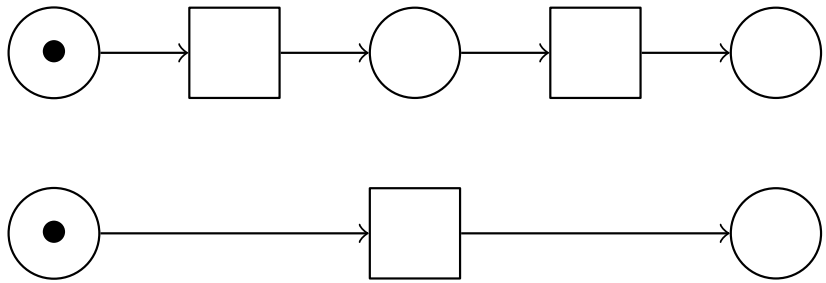
Features of Petri nets

- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.
- Resembles mathematical thinking, functional/declarative programming, in contrast to imperative programming (*what* in contrast to *how*).
- Well suited for concurrent environments:
  - Locality of dependencies, no global events.



Features of Petri nets

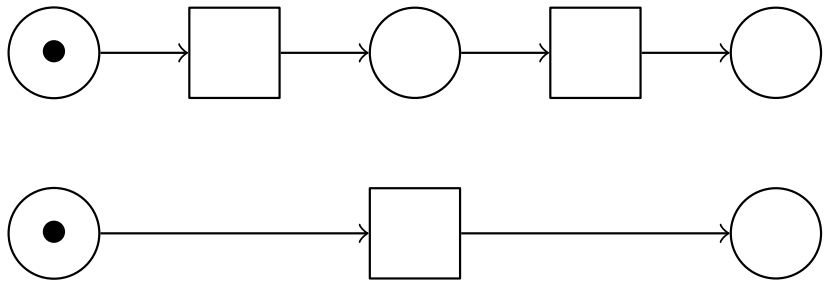
- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.
- Resembles mathematical thinking, functional/declarative programming, in contrast to imperative programming (*what* in contrast to *how*).
- Well suited for concurrent environments:
  - Locality of dependencies, no global events.



  - We can optimize locally to add parallelism (e.g. by term rewriting).

Features of Petri nets

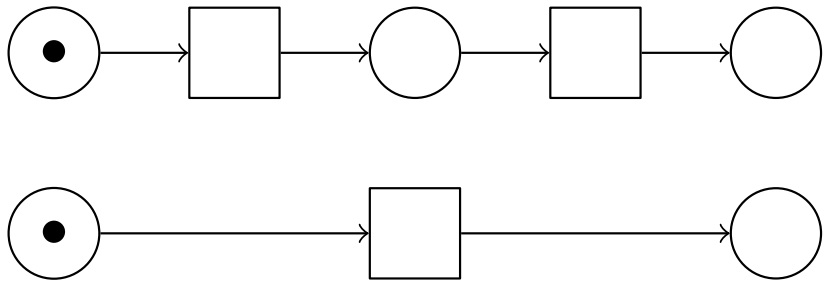
- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.
- Resembles mathematical thinking, functional/declarative programming, in contrast to imperative programming (*what* in contrast to *how*).
- Well suited for concurrent environments:
  - Locality of dependencies, no global events.



  - We can optimize locally to add parallelism (e.g. by term rewriting).
  - State based, describes at any time the full state of the application.

Features of Petri nets

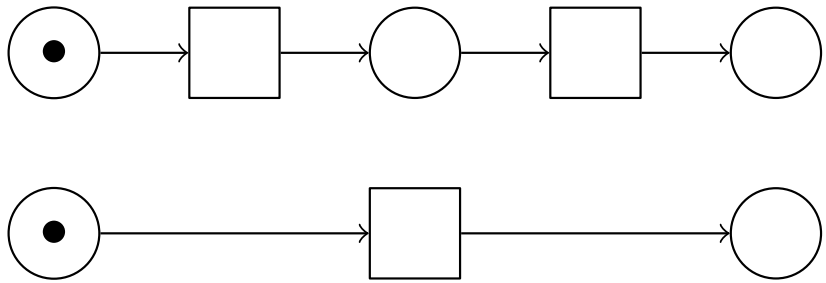
- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.
- Resembles mathematical thinking, functional/declarative programming, in contrast to imperative programming (*what* in contrast to *how*).
- Well suited for concurrent environments:
  - Locality of dependencies, no global events.



  - We can optimize locally to add parallelism (e.g. by term rewriting).
  - State based, describes at any time the full state of the application.
  - Reversible, can compute backwards, can recompute in case of a loss of a result.

Features of Petri nets

- Graphical, so accessible not only to experts in programming.
- Hierarchial, building blocks can again be Petri nets.
- Resembles mathematical thinking, functional/declarative programming, in contrast to imperative programming (*what* in contrast to *how*).
- Well suited for concurrent environments:
  - Locality of dependencies, no global events.

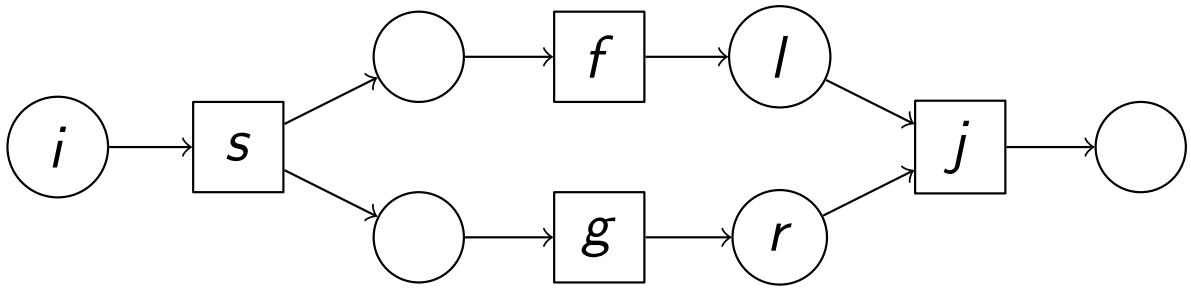


  - We can optimize locally to add parallelism (e.g. by term rewriting).
  - State based, describes at any time the full state of the application.
  - Reversible, can compute backwards, can recompute in case of a loss of a result.
  - Can add resources to running computations without any synchronization.

Features of Petri nets

Features of Petri nets

- Task parallelism:
  Transitions f and g can fire in parallel:



- Data parallelism:
  If the input holds multiple tokens, several instances of f can fire in parallel (similarly for g).

Extensions to Petri nets in GPI-Space

- Real world transitions take time:
  - Remove tokens from predecessor.
  - During computation transition holds tokens.
  - Put tokens on sucessor.

Extensions to Petri nets in GPI-Space

- Real world transitions take time:
  - Remove tokens from predecessor.
  - During computation transition holds tokens.
  - Put tokens on sucessor.

  Any state reached in this way can also be reached in the abstract Petri net.

Extensions to Petri nets in GPI-Space

- Real world transitions take time:
  - Remove tokens from predecessor.
  - During computation transition holds tokens.
  - Put tokens on sucessor.

  Any state reached in this way can also be reached in the abstract Petri net.
- Tokens can be complex data structures.

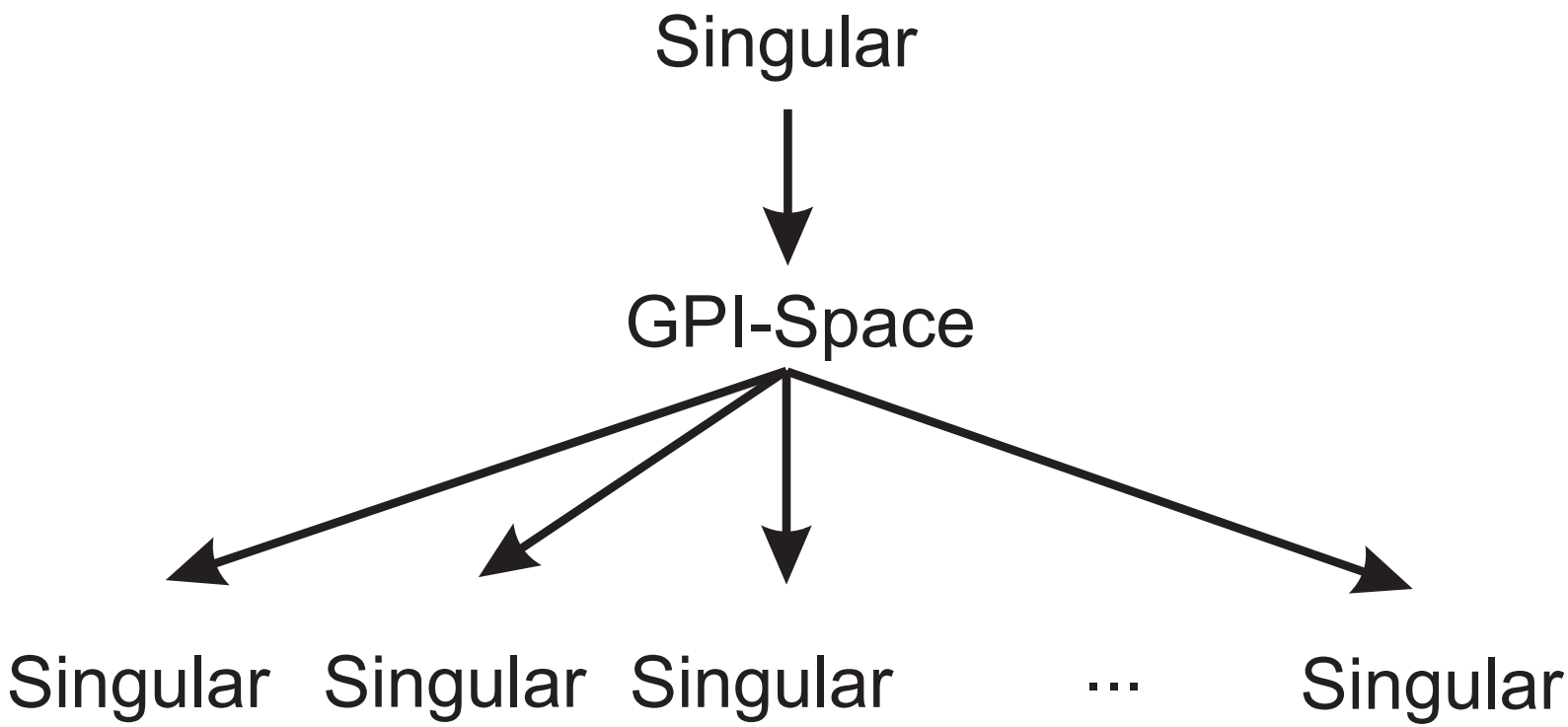
Extensions to Petri nets in GPI-Space

- Real world transitions take time:
  - Remove tokens from predecessor.
  - During computation transition holds tokens.
  - Put tokens on sucessor.

  Any state reached in this way can also be reached in the abstract Petri net.
- Tokens can be complex data structures.
- Expression language for small computations.

Singular and GPI-Space

- SINGULAR calls GPI-Space.
- GPI-Space uses LIBSINGULAR on the workers.



J. Boehm, A. Frühbis-Krüger, F.-J. Pfreundt, M. Rahn, L. Ristau. *Towards Massively Parallel Computations in Algebraic Geometry.* arXiv:1808.09727 (2018)

Massively parallel structures in algebraic geometry

- Key concept in modern algebraic geometry:
  - Description of schemes and sheaves in terms of coverings by **charts**.

Massively parallel structures in algebraic geometry

- Key concept in modern algebraic geometry:
  - Description of schemes and sheaves in terms of coverings by **charts**.
  - Global properties are related to local ones in the individual charts.

Massively parallel structures in algebraic geometry

- Key concept in modern algebraic geometry:
  - Description of schemes and sheaves in terms of coverings by **charts**.
  - Global properties are related to local ones in the individual charts.
- Use this natural parallel structure algorithmically?

Massively parallel structures in algebraic geometry

- Key concept in modern algebraic geometry:
  - Description of schemes and sheaves in terms of coverings by **charts**.
  - Global properties are related to local ones in the individual charts.
- Use this natural parallel structure algorithmically?
- Computational basis of most algorithms is Buchberger's Algorithm.

Massively parallel structures in algebraic geometry

- Key concept in modern algebraic geometry:
  - Description of schemes and sheaves in terms of coverings by **charts**.
  - Global properties are related to local ones in the individual charts.
- Use this natural parallel structure algorithmically?
- Computational basis of most algorithms is Buchberger's Algorithm.
- Buchberger's Algorithm has doubly exponential worst case complexity [Mayr, Meyer 1982], much faster in many practical examples of interest $\rightarrow$ unpredictable for parallelization (can parallelize *individual* computations via modular and linear algebra methods).
- $\rightarrow$ Single local computation may dominate the run-time.
- Solution: Model algorithm in a parallel way s.t. it automatically finds a good cover.

Modular methods and interpolation:

Error tolerant rational reconstruction and general reconstruction scheme for algebraic geometry:

J. Boehm, W. Decker, C. Fieker, G. Pfister. *The use of bad primes in rational reconstruction*, Math. Comp. 84 (2015).

Applications: Groebner bases over $\mathbb{Q}$, number fields, rational function fields, normalization, adjoint ideals, ...

Massively parallel structures in algebraic geometry

Modular methods and interpolation:

Error tolerant rational reconstruction and general reconstruction scheme for algebraic geometry:

J. Boehm, W. Decker, C. Fieker, G. Pfister. *The use of bad primes in rational reconstruction*, Math. Comp. 84 (2015).

Applications: Groebner bases over $\mathbb{Q}$, number fields, rational function fields, normalization, adjoint ideals, ...

Combinatorial structures in algebraic geometry:

Graph expansion in Gröbner fans, tropical geometry, ...

Applications: Singular and GPI-Space

- Determining smoothness for algebraic varieties.

Applications: Singular and GPI-Space

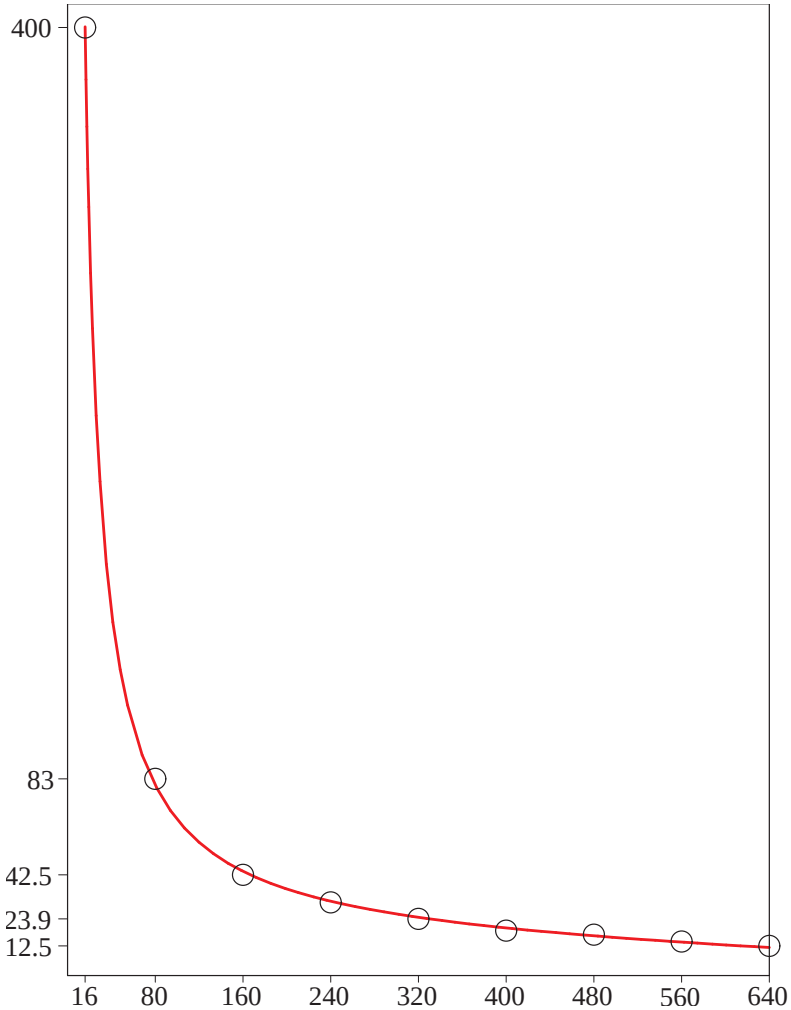
- Determining smoothness for algebraic varieties.
- Computing GIT-fans in geometric invariant theory.

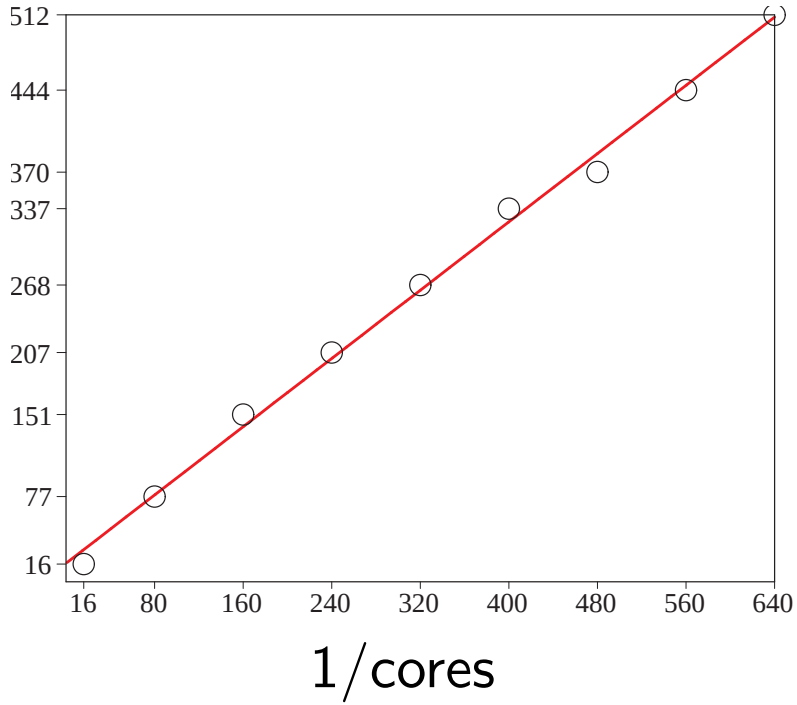
Applications: Singular and GPI-Space

- Determining smoothness for algebraic varieties.
- Computing GIT-fans in geometric invariant theory.
- Computing tropical varieties.

Timings and Scaling in GPI-Space

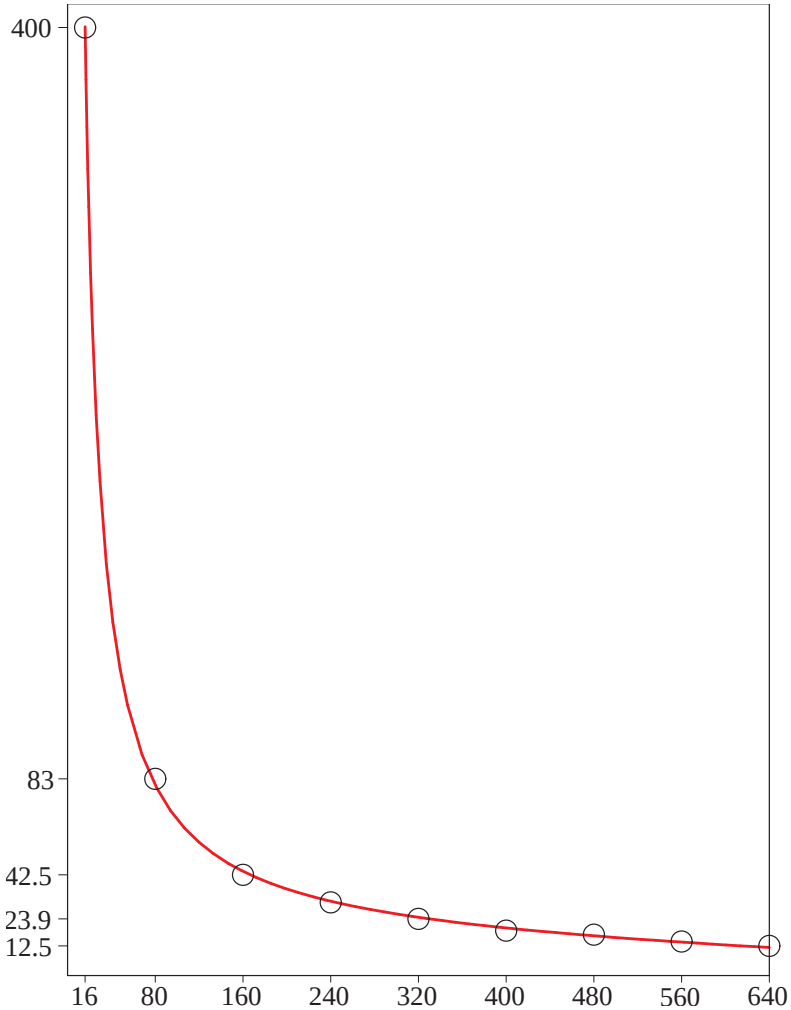
Symmetric GIT-fan algorithm relevant in geometric invariant theory:



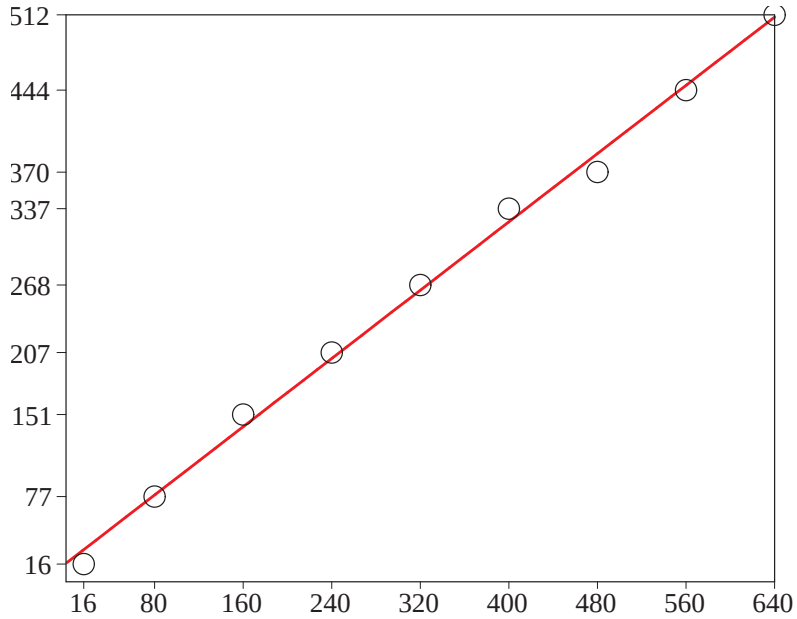


Timings and Scaling in GPI-Space

Symmetric GIT-fan algorithm relevant in geometric invariant theory:



min/cores



1/cores

Using the SINGULAR task model with 1 core 16 days, 16 cores 1 day.

References

J. Boehm, A. Georgoudis, K. J. Larsen, H. Schönemann, Y. Zhang. *Complete integration-by-parts reductions of the non-planar hexagon-box via module intersections.* J. High Energ. Phys. (2018).

J. Boehm, A. Georgoudis, K. J. Larsen, M. Schulze, Y. Zhang. *Complete sets of logarithmic vector fields for integration-by-parts identities of Feynman integrals.* Phys. Rev. D (2018).

J. Boehm, A. Frühbis-Krüger, F.-J. Pfreundt, M. Rahn, L. Ristau. *Towards Massively Parallel Computations in Algebraic Geometry.* arXiv:1808.09727 (2018)

J. Boehm, W. Decker, C. Fieker, G. Pfister. *The use of bad primes in rational reconstruction*, Math. Comp. 84 (2015).