Integer Hulls, $\mathbb{Z}$-Polyhedra and Presburger Arithmetic in Action

Rui-Juan Jing [1], Yuzhuo Lei [2], Christopher F. S. Maligec [2], Marc Moreno Maza [2], Chirantan Mukherjee [2]

[2]Ontario Research Center for Computer Algebra (ORCCA), UWO, London, Ontario
[1]School of Mathematical Science, Jiangsu University, Zhenjiang, China

July 30, 2025

Acknowledgements

- This software demo is based on research projects in which **many former and current ORCCA students** have played an essential role. By alphabetic order: Xiaohui Chen, Rui-Juan Jing, Yuzhuo Lei, Christopher Maligec, Chirantan Mukherjee, Delaram Talaashrafi, Linxiao Wang, Ning Xie, Rong Xiao and Haoze Yuan.

Acknowledgements

- This software demo is based on research projects in which **many former and current ORCCA students** have played an essential role. By alphabetic order: Xiaohui Chen, Rui-Juan Jing, Yuzhuo Lei, Christopher Maligec, Chirantan Mukherjee, Delaram Talaashrafi, Linxiao Wang, Ning Xie, Rong Xiao and Haoze Yuan.
- This software demo is based on **academic and industry collaborations** with Maplesoft, MIT/CSAIL, NVIDIA, Intel and IBM Canada, with funding support from Maplesoft, MITACS, IBM and NSERC of Canada, together with hardware/software from NVIDIA and Intel.

Acknowledgements

- This software demo is based on research projects in which **many former and current ORCCA students** have played an essential role. By alphabetic order: Xiaohui Chen, Rui-Juan Jing, Yuzhuo Lei, Christopher Maligec, Chirantan Mukherjee, Delaram Talaashrafi, Linxiao Wang, Ning Xie, Rong Xiao and Haoze Yuan.
- This software demo is based on **academic and industry collaborations** with Maplesoft, MIT/CSAIL, NVIDIA, Intel and IBM Canada, with funding support from Maplesoft, MITACS, IBM and NSERC of Canada, together with hardware/software from NVIDIA and Intel.
- Most of the algorithms presented in this software demo are **available in MAPLE's `PolyhedralSets`** library.

Motivations and objectives

- Studying the integer solutions of linear systems of equations and inequalities is of practical importance in various areas of scientific computing:

Motivations and objectives

- Studying the integer solutions of linear systems of equations and inequalities is of practical importance in various areas of scientific computing:
  - **combinatorial optimization**, in particular, integer linear programming,

Motivations and objectives

- Studying the integer solutions of linear systems of equations and inequalities is of practical importance in various areas of scientific computing:
  - **combinatorial optimization**, in particular, integer linear programming,
  - **compiler optimization** in particular, the analysis, transformation, and scheduling of nested loops in computer programs.

Motivations and objectives

- Studying the integer solutions of linear systems of equations and inequalities is of practical importance in various areas of scientific computing:
  - **combinatorial optimization**, in particular, integer linear programming,
  - **compiler optimization** in particular, the analysis, transformation, and scheduling of nested loops in computer programs.
- One important objective for us is to support **Presburger arithmetic**, that is, quantifier elimination over the integers, see our ISSAC 2025 paper.

Software architecture and history

Component	Release year	Key functionalities
`PolyhedralSets`	2015	Basic routines for polyhedra over $\mathbb{Q}$

Software architecture and history

Component	Release year	Key functionalities
`PolyhedralSets`	2015	Basic routines for polyhedra over $\mathbb{Q}$
`IntegerHull`	2022	fast computations of integer hulls

Software architecture and history

Component	Release year	Key functionalities
`PolyhedralSets`	2015	Basic routines for polyhedra over $\mathbb{Q}$
`IntegerHull`	2022	fast computations of integer hulls
`ZPolyhedralSets`	2023	$\mathbb{Z}$-polyhedra

Software architecture and history

Component	Release year	Key functionalities
`PolyhedralSets`	2015	Basic routines for polyhedra over $\mathbb{Q}$
`IntegerHull`	2022	fast computations of integer hulls
`ZPolyhedralSets`	2023	$\mathbb{Z}$-polyhedra
`FME_SatMat`	2025	FME, double-description method

Software architecture and history

Component	Release year	Key functionalities
`PolyhedralSets`	2015	Basic routines for polyhedra over $\mathbb{Q}$
`IntegerHull`	2022	fast computations of integer hulls
`ZPolyhedralSets`	2023	$\mathbb{Z}$-polyhedra
`FME_SatMat`	2025	FME, double-description method
`NumberOfIntegerPoints`	2025	Ehrhart polynomials

Software architecture and history

Component	Release year	Key functionalities
`PolyhedralSets`	2015	Basic routines for polyhedra over $\mathbb{Q}$
`IntegerHull`	2022	fast computations of integer hulls
`ZPolyhedralSets`	2023	$\mathbb{Z}$-polyhedra
`FME_SatMat`	2025	FME, double-description method
`NumberOfIntegerPoints`	2025	Ehrhart polynomials
`ValuesUnderConstrainsts`	2025	computations with piece-wise functions

Software architecture and history

Component	Release year	Key functionalities
`PolyhedralSets`	2015	Basic routines for polyhedra over $\mathbb{Q}$
`IntegerHull`	2022	fast computations of integer hulls
`ZPolyhedralSets`	2023	$\mathbb{Z}$-polyhedra
`FME_SatMat`	2025	FME, double-description method
`NumberOfIntegerPoints`	2025	Ehrhart polynomials
`ValuesUnderConstrainsts`	2025	computations with piece-wise functions
`QuasiPolynomials`	2025	computations with quasi-polynomials

Software architecture and history

Component	Release year	Key functionalities
PolyhedralSets	2015	Basic routines for polyhedra over $\mathbb{Q}$
IntegerHull	2022	fast computations of integer hulls
ZPolyhedralSets	2023	$\mathbb{Z}$-polyhedra
FME_SatMat	2025	FME, double-description method
NumberOfIntegerPoints	2025	Ehrhart polynomials
ValuesUnderConstrainsts	2025	computations with piece-wise functions
QuasiPolynomials	2025	computations with quasi-polynomials
QuantifierEliminationOverZ	new	QE over $\mathbb{Z}$ (back-engine)

Software architecture and history

Component	Release year	Key functionalities
`PolyhedralSets`	2015	Basic routines for polyhedra over $\mathbb{Q}$
`IntegerHull`	2022	fast computations of integer hulls
`ZPolyhedralSets`	2023	$\mathbb{Z}$-polyhedra
`FME_SatMat`	2025	FME, double-description method
`NumberOfIntegerPoints`	2025	Ehrhart polynomials
`ValuesUnderConstrainsts`	2025	computations with piece-wise functions
`QuasiPolynomials`	2025	computations with quasi-polynomials
`QuantifierEliminationOverZ`	new	QE over $\mathbb{Z}$ (back-engine)
`PresburgerFormulas`	new	QE over $\mathbb{Z}$ (front-engine)

Software architecture and history

Component	Release year	Key functionalities
PolyhedralSets	2015	Basic routines for polyhedra over $\mathbb{Q}$
IntegerHull	2022	fast computations of integer hulls
ZPolyhedralSets	2023	$\mathbb{Z}$-polyhedra
FME_SatMat	2025	FME, double-description method
NumberOfIntegerPoints	2025	Ehrhart polynomials
ValuesUnderConstrainsts	2025	computations with piece-wise functions
QuasiPolynomials	2025	computations with quasi-polynomials
QuantifierEliminationOverZ	new	QE over $\mathbb{Z}$ (back-engine)
PresburgerFormulas	new	QE over $\mathbb{Z}$ (front-engine)

- All these components are libraries, except IntegerHull and NumberOfIntegerPoints which are commands.

Software architecture and history

Component	Release year	Key functionalities
`PolyhedralSets`	2015	Basic routines for polyhedra over $\mathbb{Q}$
`IntegerHull`	2022	fast computations of integer hulls
`ZPolyhedralSets`	2023	$\mathbb{Z}$-polyhedra
`FME_SatMat`	2025	FME, double-description method
`NumberOfIntegerPoints`	2025	Ehrhart polynomials
`ValuesUnderConstrainsts`	2025	computations with piece-wise functions
`QuasiPolynomials`	2025	computations with quasi-polynomials
`QuantifierEliminationOverZ`	new	QE over $\mathbb{Z}$ (back-engine)
`PresburgerFormulas`	new	QE over $\mathbb{Z}$ (front-engine)

- All these components are libraries, except `IntegerHull` and `NumberOfIntegerPoints` which are commands.
- All libraries and commands are MAPLE code, except `FME_SatMat` which is is written in C/C++ in support of efficiency critical routines.

Plan

1. Overview
2. Integer hulls of polyhedra
3. Integer point counting for parametric polyhedra
4. Quantifier elimination over the integers
5. Concluding remarks

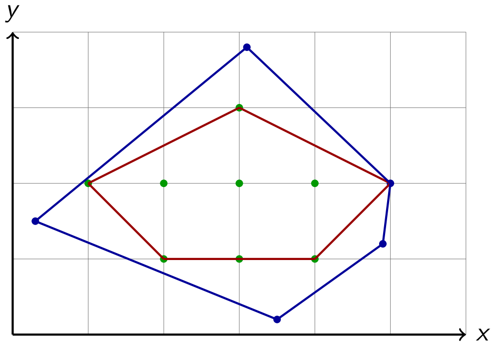
Plan

Integer hulls and lattices

1. The **integer hull** of a polyhedron $P \subseteq \mathbb{Q}^d$, is the smallest convex polyhedron containing all the integer points of P. Thus, this is is the intersection of all convex polyhedra containing $P \cap \mathbb{Z}^d$.

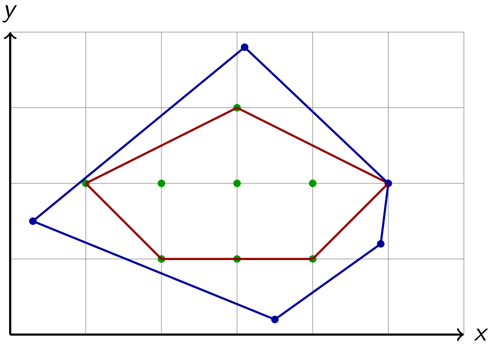
Integer hulls and lattices

1. The **integer hull** of a polyhedron $P \subseteq \mathbb{Q}^d$, is the smallest convex polyhedron containing all the integer points of P. Thus, this is is the intersection of all convex polyhedra containing $P \cap \mathbb{Z}^d$.



Integer hulls and lattices

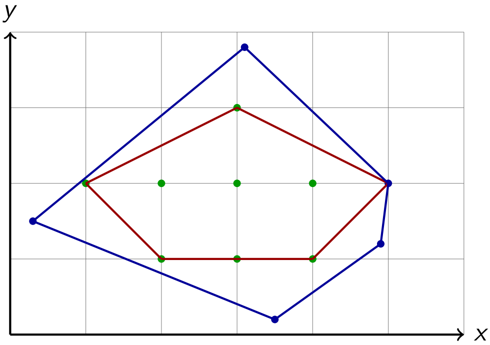
1. The **integer hull** of a polyhedron $P \subseteq \mathbb{Q}^d$, is the smallest convex polyhedron containing all the integer points of P. Thus, this is is the intersection of all convex polyhedra containing $P \cap \mathbb{Z}^d$.



2. A subset $L \subseteq \mathbb{Z}^d$ is called an **integer lattice** (or simply a lattice) if
$$L = \{\mathbf{x} \in \mathbb{Z}^d \mid (\exists \mathbf{t} \in \mathbb{Z}^c)\ \mathbf{x} = A\mathbf{t} + \mathbf{b}\}$$
holds, for a matrix $A \in \mathbb{Z}^{d \times c}$ and a vector $\mathbf{b} \in \mathbb{Z}^d$, where c is a positive integer.

Integer hulls and lattices

1. The **integer hull** of a polyhedron $P \subseteq \mathbb{Q}^d$, is the smallest convex polyhedron containing all the integer points of P. Thus, this is is the intersection of all convex polyhedra containing $P \cap \mathbb{Z}^d$.



2. A subset $L \subseteq \mathbb{Z}^d$ is called an **integer lattice** (or simply a lattice) if
$$L = \{\mathbf{x} \in \mathbb{Z}^d \mid (\exists \mathbf{t} \in \mathbb{Z}^c)\ \mathbf{x} = A\mathbf{t} + \mathbf{b}\}$$
holds, for a matrix $A \in \mathbb{Z}^{d \times c}$ and a vector $\mathbf{b} \in \mathbb{Z}^d$, where c is a positive integer.
3. It is convenient to see this lattice as the solution set of the systems of congruence relations $\mathbf{x} \equiv \mathbf{b} \mod A$.

$\mathbb{Z}$-polyhedra

1. A **$\mathbb{Z}$-polyhedron** of $\mathbb{Z}^d$ is the intersection (in $\mathbb{Z}^d$) of a polyhedron $P \subseteq \mathbb{Q}^d$ and a lattice $L \subseteq \mathbb{Z}^d$; we denote it by $\mathbb{Z}\text{Polyhedron}(P, L)$.

$\mathbb{Z}$-polyhedra

1. A **$\mathbb{Z}$-polyhedron** of $\mathbb{Z}^d$ is the intersection (in $\mathbb{Z}^d$) of a polyhedron $P \subseteq \mathbb{Q}^d$ and a lattice $L \subseteq \mathbb{Z}^d$; we denote it by $\mathbb{Z}\text{Polyhedron}(P, L)$.
2. Denote by $x_1 < x_2 < \cdots < x_d$ the coordinates of $\mathbb{Z}^d$. We say that $\mathbb{Z}\text{Polyhedron}(P, L)$ is **normalized** if

$\mathbb{Z}$-polyhedra

1. A **$\mathbb{Z}$-polyhedron** of $\mathbb{Z}^d$ is the intersection (in $\mathbb{Z}^d$) of a polyhedron $P \subseteq \mathbb{Q}^d$ and a lattice $L \subseteq \mathbb{Z}^d$; we denote it by $\mathbb{Z}\mathrm{Polyhedron}(P, L)$.
2. Denote by $x_1 < x_2 < \cdots < x_d$ the coordinates of $\mathbb{Z}^d$. We say that $\mathbb{Z}\mathrm{Polyhedron}(P, L)$ is **normalized** if
   a. it is non-empty, and P is given by a system of linear inequalities of the form

$$\left\{ \begin{array}{ccc} a_0 & \leq x_1 \leq & b_0 \\ a_1 & \leq x_2 \leq & b_1 \\ \vdots & \vdots & \vdots \\ a_{n-1} & \leq x_d \leq & b_{n-1}, \end{array} \right. \tag{2.1}$$

   where

$\mathbb{Z}$-polyhedra

1. A **$\mathbb{Z}$-polyhedron** of $\mathbb{Z}^d$ is the intersection (in $\mathbb{Z}^d$) of a polyhedron $P \subseteq \mathbb{Q}^d$ and a lattice $L \subseteq \mathbb{Z}^d$; we denote it by $\mathbb{Z}\mathrm{Polyhedron}(P, L)$.
2. Denote by $x_1 < x_2 < \cdots < x_d$ the coordinates of $\mathbb{Z}^d$. We say that $\mathbb{Z}\mathrm{Polyhedron}(P, L)$ is **normalized** if
   a. it is non-empty, and P is given by a system of linear inequalities of the form

$$\begin{cases} a_0 & \leq x_1 \leq & b_0 \\ a_1 & \leq x_2 \leq & b_1 \\ \vdots & \vdots & \vdots \\ a_{n-1} & \leq x_d \leq & b_{n-1}, \end{cases} \tag{2.1}$$

   where
   b. a_i (resp. b_i) is either $-\infty$ (resp. $+\infty$) or an expression of the form $\max(\ell_{i,1} \ldots \ell_{i,e_i})$ (resp. $\min(\ell_{i,1} \ldots \ell_{i,e_i})$), and

$\mathbb{Z}$-polyhedra

1. A **$\mathbb{Z}$-polyhedron** of $\mathbb{Z}^d$ is the intersection (in $\mathbb{Z}^d$) of a polyhedron $P \subseteq \mathbb{Q}^d$ and a lattice $L \subseteq \mathbb{Z}^d$; we denote it by $\mathbb{Z}\text{Polyhedron}(P, L)$.
2. Denote by $x_1 < x_2 < \cdots < x_d$ the coordinates of $\mathbb{Z}^d$. We say that $\mathbb{Z}\text{Polyhedron}(P, L)$ is **normalized** if
   a. it is non-empty, and P is given by a system of linear inequalities of the form
   $$\begin{cases} a_0 & \leq x_1 \leq & b_0 \\ a_1 & \leq x_2 \leq & b_1 \\ \vdots & \vdots & \vdots \\ a_{n-1} & \leq x_d \leq & b_{n-1}, \end{cases} \tag{2.1}$$
   where
   b. a_i (resp. b_i) is either $-\infty$ (resp. $+\infty$) or an expression of the form $\max(\ell_{i,1} \ldots \ell_{i,e_i})$ (resp. $\min(\ell_{i,1} \ldots \ell_{i,e_i})$), and
   c. each $\ell_{i,j} \in \mathbb{Q}[x_1, \ldots, x_{i-1}]$ with degree at most 1, so that

$\mathbb{Z}$-polyhedra

1. A **$\mathbb{Z}$-polyhedron** of $\mathbb{Z}^d$ is the intersection (in $\mathbb{Z}^d$) of a polyhedron $P \subseteq \mathbb{Q}^d$ and a lattice $L \subseteq \mathbb{Z}^d$; we denote it by $\mathbb{Z}\text{Polyhedron}(P, L)$.
2. Denote by $x_1 < x_2 < \cdots < x_d$ the coordinates of $\mathbb{Z}^d$. We say that $\mathbb{Z}\text{Polyhedron}(P, L)$ is **normalized** if
   - a. it is non-empty, and P is given by a system of linear inequalities of the form

$$\left\{ \begin{array}{ccc} a_0 & \leq x_1 \leq & b_0 \\ a_1 & \leq x_2 \leq & b_1 \\ \vdots & \vdots & \vdots \\ a_{n-1} & \leq x_d \leq & b_{n-1}, \end{array} \right. \tag{2.1}$$

     where
   - b. a_i (resp. b_i) is either $-\infty$ (resp. $+\infty$) or an expression of the form $\max(\ell_{i,1} \ldots \ell_{i,e_i})$ (resp. $\min(\ell_{i,1} \ldots \ell_{i,e_i})$), and
   - c. each $\ell_{i,j} \in \mathbb{Q}[x_1, \ldots, x_{i-1}]$ with degree at most 1, so that
   - d. all the integer points of P are obtained by **back substitution**, that is, by specializing x_1 to every integer value v_1 in the interval (a_0, b_0), then by specializing x_2 to every integer value v_2 in the interval $(a_1(v_1), b_1(v_1))$, and so on.

$\mathbb{Z}$-polyhedra

1. A **$\mathbb{Z}$-polyhedron** of $\mathbb{Z}^d$ is the intersection (in $\mathbb{Z}^d$) of a polyhedron $P \subseteq \mathbb{Q}^d$ and a lattice $L \subseteq \mathbb{Z}^d$; we denote it by $\mathbb{Z}\text{Polyhedron}(P, L)$.
2. Denote by $x_1 < x_2 < \cdots < x_d$ the coordinates of $\mathbb{Z}^d$. We say that $\mathbb{Z}\text{Polyhedron}(P, L)$ is **normalized** if
   - a. it is non-empty, and P is given by a system of linear inequalities of the form

$$\left\{ \begin{array}{ccc} a_0 & \leq x_1 \leq & b_0 \\ a_1 & \leq x_2 \leq & b_1 \\ \vdots & \vdots & \vdots \\ a_{n-1} & \leq x_d \leq & b_{n-1}, \end{array} \right. \tag{2.1}$$

     where
   - b. a_i (resp. b_i) is either $-\infty$ (resp. $+\infty$) or an expression of the form $\max(\ell_{i,1} \ldots \ell_{i,e_i})$ (resp. $\min(\ell_{i,1} \ldots \ell_{i,e_i})$), and
   - c. each $\ell_{i,j} \in \mathbb{Q}[x_1, \ldots, x_{i-1}]$ with degree at most 1, so that
   - d. all the integer points of P are obtained by **back substitution**, that is, by specializing x_1 to every integer value v_1 in the interval (a_0, b_0), then by specializing x_2 to every integer value v_2 in the interval $(a_1(v_1), b_1(v_1))$, and so on.
3. The algorithm IntegerPointDecomposition [3] decomposes any $\mathbb{Z}$-polyhedron into **normalized** $\mathbb{Z}$-polyhedra.

Plan

Generating function of a polyhedral set

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$

Generating function of a polyhedral set

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:

$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$

Generating function of a polyhedral set

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$
- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.

Generating function of a polyhedral set

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$
- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.

Generating function of a polyhedral set

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$
- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray given by $y = 0$ and $x \geq 0$, then:

$$G(P, \mathbf{x}) =$$

Generating function of a polyhedral set

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$
- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray given by $y = 0$ and $x \geq 0$, then:
$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} =$$

Generating function of a polyhedral set

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$
- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray given by $y = 0$ and $x \geq 0$, then:
$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 =$$

Generating function of a polyhedral set

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$
- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray given by $y = 0$ and $x \geq 0$, then:
$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 = \sum_{n=0}^{n=\infty} x^n =$$

Generating function of a polyhedral set

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:

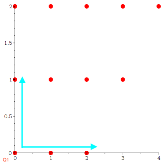
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$

- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray given by $y = 0$ and $x \geq 0$, then:

$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 = \sum_{n=0}^{n=\infty} x^n = \frac{1}{1 - x}.$$

Generating function of a polyhedral set

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$
- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray given by $y = 0$ and $x \geq 0$, then:
$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 = \sum_{n=0}^{n=\infty} x^n = \frac{1}{1-x}.$$
- Still for $d = 2$, $G(P, \mathbf{x})$ is computed as the sum of the generating functions of its vertex cones, thanks to Brion's theorem (1988) [1].



$$G(P, \mathbf{x}) \quad = \quad G(Q_1, \mathbf{x}) + \qquad + \qquad +$$

Generating function of a polyhedral set

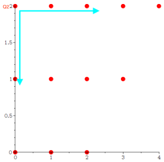
- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:

$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$

- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray given by $y = 0$ and $x \geq 0$, then:

$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 = \sum_{n=0}^{n=\infty} x^n = \frac{1}{1 - x}.$$

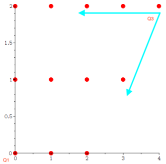
- Still for $d = 2$, $G(P, \mathbf{x})$ is computed as the sum of the generating functions of its vertex cones, thanks to Brion's theorem (1988) [1].



$$G(P, \mathbf{x}) \quad = \quad G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + \qquad\qquad +$$

Generating function of a polyhedral set

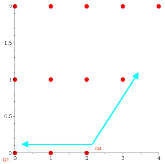
- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$
- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray given by $y = 0$ and $x \geq 0$, then:
$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 = \sum_{n=0}^{n=\infty} x^n = \frac{1}{1-x}.$$
- Still for $d = 2$, $G(P, \mathbf{x})$ is computed as the sum of the generating functions of its vertex cones, thanks to Brion's theorem (1988) [1].



$$G(P, \mathbf{x}) \quad = \quad G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x}) +$$

Generating function of a polyhedral set

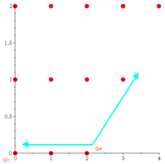
- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \dots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$
- If P is bounded, then $G(P, (1, \dots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray given by $y = 0$ and $x \geq 0$, then:
$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 = \sum_{n=0}^{n=\infty} x^n = \frac{1}{1 - x}.$$
- Still for $d = 2$, $G(P, \mathbf{x})$ is computed as the sum of the generating functions of its vertex cones, thanks to Brion's theorem (1988) [1].



$$G(P, \mathbf{x}) \quad = \quad G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x}) + G(Q_4, \mathbf{x})$$

Generating function of a polyhedral set

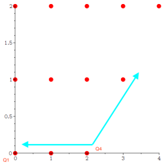
- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$
- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray given by $y = 0$ and $x \geq 0$, then:
$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 = \sum_{n=0}^{n=\infty} x^n = \frac{1}{1 - x}.$$
- Still for $d = 2$, $G(P, \mathbf{x})$ is computed as the sum of the generating functions of its vertex cones, thanks to Brion's theorem (1988) [1].



$$G(P, \mathbf{x}) \quad = \quad G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x}) + G(Q_4, \mathbf{x})$$

Generating function of a polyhedral set

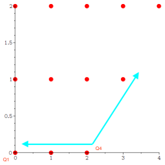
- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$
- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray given by $y = 0$ and $x \geq 0$, then:
$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 = \sum_{n=0}^{n=\infty} x^n = \frac{1}{1-x}.$$
- Still for $d = 2$, $G(P, \mathbf{x})$ is computed as the sum of the generating functions of its vertex cones, thanks to Brion's theorem (1988) [1].



$$\begin{array}{rcl} G(P, \mathbf{x}) & = & G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x}) + G(Q_4, \mathbf{x}) \\ & = & \frac{1}{1-x} \frac{1}{1-y} + \frac{1}{1-x} \frac{y^2}{1-y^{-1}} + \frac{x^4 y^2}{(1-x^{-1})(1-x^{-1}y^{-1})} + \frac{x^2 y^0}{(1-xy)(1-x^{-1})} \end{array}$$

Generating function of a polyhedral set

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$ and map each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- The **generating function** of P is the formal Laurent series:
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$
- If P is bounded, then $G(P, (1, \ldots, 1))$ counts its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal Laurent series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray given by $y = 0$ and $x \geq 0$, then:
$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 = \sum_{n=0}^{n=\infty} x^n = \frac{1}{1-x}.$$
- Still for $d = 2$, $G(P, \mathbf{x})$ is computed as the sum of the generating functions of its vertex cones, thanks to Brion's theorem (1988) [1].



$$\begin{aligned}
G(P, \mathbf{x}) &= G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x}) + G(Q_4, \mathbf{x}) \\
&= \frac{1}{1-x}\frac{1}{1-y} + \frac{1}{1-x}\frac{y^2}{1-y^{-1}} + \frac{x^4y^2}{(1-x^{-1})(1-x^{-1}y^{-1})} + \frac{x^2y^0}{(1-xy)(1-x^{-1})} \\
&= y^2 + xy^2 + x^2y^2 + x^3y^2 + x^4y^2 + y + xy + x^2y + \\
&\quad + x^3y + 1 + x + x^2.
\end{aligned}$$

Plan

Presburger arithmetic

The language of **Presburger arithmetic** is:

1. the first-order theory of the integers with addition, equality and order
2. extended by the divisibility predicates $D_k : x \longmapsto k \mid x$, for all $k \in \mathbb{Z}_{>0}$.

Presburger arithmetic

The language of **Presburger arithmetic** is:

1. the first-order theory of the integers with addition, equality and order
2. extended by the divisibility predicates $D_k : x \longmapsto k \mid x$, for all $k \in \mathbb{Z}_{>0}$.

A **Presburger formula F in prenex normal form** has the form:

$$F = Q_1 x_1 \cdots Q_m x_m \; \phi(x_1, \ldots, x_m, y_1, \ldots, y_n),$$

where:

Presburger arithmetic

The language of **Presburger arithmetic** is:

1. the first-order theory of the integers with addition, equality and order
2. extended by the divisibility predicates $D_k : x \longmapsto k \mid x$, for all $k \in \mathbb{Z}_{>0}$.

A **Presburger formula** F **in prenex normal form** has the form:

$$F = Q_1 x_1 \cdots Q_m x_m \ \phi(x_1, \ldots, x_m, y_1, \ldots, y_n),$$

where:

1. $Q_1 x_1 \cdots Q_m x_m$ is a sequence of quantifiers (existential or universal) and bound variables,

Presburger arithmetic

The language of **Presburger arithmetic** is:

1. the first-order theory of the integers with addition, equality and order
2. extended by the divisibility predicates $D_k : x \longmapsto k \mid x$, for all $k \in \mathbb{Z}_{>0}$.

A **Presburger formula F in prenex normal form** has the form:

$$F = Q_1 x_1 \cdots Q_m x_m \ \phi(x_1, \ldots, x_m, y_1, \ldots, y_n),$$

where:

1. $Q_1 x_1 \cdots Q_m x_m$ is a sequence of quantifiers (existential or universal) and bound variables,
2. $y_1, \ldots, y_n$ are free (or unbounded) variables,

Presburger arithmetic

The language of **Presburger arithmetic** is:

1. the first-order theory of the integers with addition, equality and order
2. extended by the divisibility predicates $D_k : x \longmapsto k \mid x$, for all $k \in \mathbb{Z}_{>0}$.

A **Presburger formula F in prenex normal form** has the form:

$$F = Q_1 x_1 \cdots Q_m x_m \ \phi(x_1, \ldots, x_m, y_1, \ldots, y_n),$$

where:

1. $Q_1 x_1 \cdots Q_m x_m$ is a sequence of quantifiers (existential or universal) and bound variables,
2. $y_1, \ldots, y_n$ are free (or unbounded) variables,
3. $\phi(x_1, \ldots, x_m, y_1, \ldots, y_n)$ is a quantifier-free formula, where each **atom** (= formula free of quantifiers and connectives) is either
   a. a non-strict inequality $\ell(x_1, \ldots, x_m, y_1, \ldots, y_n) \leq 0$,
   b. or a divisibility relation $k \mid \ell(x_1, \ldots, x_m, y_1, \ldots, y_n)$,

   where:

   a. $k \in \mathbb{Z}_{>0}$ is a constant, and
   b. $\ell(x_1, \ldots, x_m, y_1, \ldots, y_n)$ is a **linear integer** polynomial, thus with total degree at most 1.

Quantifier elimination

Theorem 1

Presburger arithmetic admits quantifier elimination.

Quantifier elimination

Theorem 1

Presburger arithmetic admits quantifier elimination.

Remark 1

Recall

$$F = Q_1 x_1 \cdots Q_m x_m \ \phi(x_1, \ldots, x_m, y_1, \ldots, y_n),$$

Our goal is to determine the set $D(y_1, \ldots, y_n) \subseteq \mathbb{Z}^n$ *of* **ALL** *integer tuples of* $(y_1, \ldots, y_n)$ *for which the formula* $F(x_1, \ldots, x_m, y_1, \ldots, y_n)$ *is true.*

Concluding remarks

Summary and notes

1. We have presented software libraries for computing with integer hulls and $\mathbb{Z}$-polyhedra.
2. They support integer point counting of parametric polyhedra (= computing Ehrhart polynomials) as well as Presburger arithmetic (= quantifier elimination over the integers).
3. The algorithm IntegerPointDecomposition plays an essential role.
4. Most of these functionalities are available in Maple 2025, except QE.

Work in progress

1. In the presence of free variables, QE tend to split computations more than necessary and we are developing algorithms dealing with this issue.
2. We are extending our implementation of Presburger arithmetic to support certain class of non-linear expressions that are of practical interest in compiler theory [2].

References

[1] M. Brion. "Points entiers dans les polyedres convexes". In: Annales scientifiques de l'École normale supérieure. Vol. 21. 4. 1988, pp. 653–663.

[2] C. Chen, X. Chen, A. Keita, M. Moreno Maza, and N. Xie. "MetaFork: a compilation framework for concurrency models targeting hardware accelerators and its application to the generation of parametric CUDA kernels". In: Proceedings of 25th Annual International Conference on Computer Scier Ed. by J. Gould, M. Litoiu, and H. Lutfiyya. IBM / ACM, 2015, pp. 70–79. URL: http://dl.acm.org/citation.cfm?id=2886456.

[3] R. Jing and M. Moreno Maza. "Computing the Integer Points of a Polyhedron, I: Algorithm". In: CASC 2017, Proceedings. Vol. 10490. LNCS. Springer, 2017, pp. 225–241.