Quantifier Elimination Over the Integers

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 29, 2025

Plan

1. Overview
2. Basic concepts
3. Quantifier elimination over the integers
4. Integer projection
5. Experimentation
6. Concluding remarks

Quantifier elimination

Input

Consider a formula in prenex normal form,

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

where:

1. $Q_1, \ldots Q_m$ is a sequence of quantifiers (existential $\exists$ or universal $\forall$),
2. $x_1, \ldots, x_m$ are bound variables,
3. $y_1, \ldots, y_n$ are free variables and,
4. $\phi(x_1, \ldots, x_m, y_1, \ldots, y_n)$ is a quantifier-free formula.

Quantifier elimination

Input

Consider a formula in prenex normal form,

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

where:

1. $Q_1, \ldots Q_m$ is a sequence of quantifiers (existential $\exists$ or universal $\forall$),
2. $x_1, \ldots, x_m$ are bound variables,
3. $y_1, \ldots, y_n$ are free variables and,
4. $\phi(x_1, \ldots, x_m, y_1, \ldots, y_n)$ is a quantifier-free formula.

Possible output

1. **A sample point:** a tuple of values for $(y_1, \ldots, y_n)$ making F true, if such a tuple exists, `false` otherwise,

Quantifier elimination

Input

Consider a formula in prenex normal form,

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

where:

1. $Q_1, \ldots Q_m$ is a sequence of quantifiers (existential $\exists$ or universal $\forall$),
2. $x_1, \ldots, x_m$ are bound variables,
3. $y_1, \ldots, y_n$ are free variables and,
4. $\phi(x_1, \ldots, x_m, y_1, \ldots, y_n)$ is a quantifier-free formula.

Possible output

1. **A sample point:** a tuple of values for $(y_1, \ldots, y_n)$ making F true, if such a tuple exists, `false` otherwise,
2. **An equivalent formula:** describing the set $D(y_1, \ldots, y_n)$ consisting of all tuples of values for $(y_1, \ldots, y_n)$ making F true.

QE over the integers

1. **Linear integer arithmetic** (LIA), started with the works of Mojżesz Presbuger [26] and David Cooper [6].

QE over the integers

1. **Linear integer arithmetic** (LIA), started with the works of Mojżesz Presbuger [26] and David Cooper [6].
2. Effective approaches for LIA, developed by William Pugh [24] and Sven Verdoolaege et al. [28]:
   a. eliminate a sequence of existential quantifiers in the sense of **computing a formula**;
   b. however, no complexity estimates are known for these approaches.

QE over the integers

1. **Linear integer arithmetic** (LIA), started with the works of Mojżesz Presbuger [26] and David Cooper [6].
2. Effective approaches for LIA, developed by William Pugh [24] and Sven Verdoolaege et al. [28]:
   a. eliminate a sequence of existential quantifiers in the sense of **computing a formula**;
   b. however, no complexity estimates are known for these approaches.
3. Our algorithm `IntegerPointDecomposition` [12], which is based on William Pugh's projection,
   a. eliminates a sequence of existential quantifiers for LIA in the sense of **computing a formula**;
   b. under some technical assumptions, it runs in single exponential time w.r.t. the dimension of the ambient space.

QE over the integers

1. **Linear integer arithmetic** (LIA), started with the works of Mojżesz Presbuger [26] and David Cooper [6].
2. Effective approaches for LIA, developed by William Pugh [24] and Sven Verdoolaege et al. [28]:
   a. eliminate a sequence of existential quantifiers in the sense of **computing a formula**;
   b. however, no complexity estimates are known for these approaches.
3. Our algorithm `IntegerPointDecomposition` [12], which is based on William Pugh's projection,
   a. eliminates a sequence of existential quantifiers for LIA in the sense of **computing a formula**;
   b. under some technical assumptions, it runs in single exponential time w.r.t. the dimension of the ambient space.
4. Still for a sequence of existential quantifiers in LIA, the algorithm [10] by Christoph Haase et al. **computes sample points in singly exponential time**.

QE over the integers

1. **Linear integer arithmetic** (LIA), started with the works of Mojżesz Presbuger [26] and David Cooper [6].
2. Effective approaches for LIA, developed by William Pugh [24] and Sven Verdoolaege et al. [28]:
   a. eliminate a sequence of existential quantifiers in the sense of **computing a formula**;
   b. however, no complexity estimates are known for these approaches.
3. Our algorithm `IntegerPointDecomposition` [12], which is based on William Pugh's projection,
   a. eliminates a sequence of existential quantifiers for LIA in the sense of **computing a formula**;
   b. under some technical assumptions, it runs in single exponential time w.r.t. the dimension of the ambient space.
4. Still for a sequence of existential quantifiers in LIA, the algorithm [10] by Christoph Haase et al. **computes sample points in singly exponential time**.
5. Can we eliminate a sequence of existential quantifiers for LIA in the sense of computing a formula in single exponential time?

Applications

- Optimizing Compilers:
  - Array Dependence Analysis
  - Polyhedral frameworks (GCC's Graphite [22], LLVM's Polly [8]).
- Program Verification:
  - Stanford Pascal Verifier [18]
  - CompCert [16]
- Theorem Proving:
  - SAT/SMT Solvers (Z3 [20], CVC5 [1])
  - Proof Assistants (Coq [4], Isabelle [21], HOL Light [11], Lean [19]).

Software Implementations

- ISL (Integer Set Library) [27]
- TaPAS (Talence Presburger Arithmetic Suite) [15]
- Yices [7]
- Princess (Scala Theorem Prover) [25]
- Our Software, see the ISSAC 2025 software demo.

Our contributions

1. We study **integer projection**, that is, the elimination of a sequence of existential quantifiers in a Presburger formula

Our contributions

1. We study **integer projection**, that is, the elimination of a sequence of existential quantifiers in a Presburger formula
2. We compare three well-know projections in a **unified framework**: Cooper's algorithm [6], Williams' projection [29] and the Omega test [24]

Our contributions

1. We study **integer projection**, that is, the elimination of a sequence of existential quantifiers in a Presburger formula
2. We compare three well-know projections in a **unified framework**: Cooper's algorithm [6], Williams' projection [29] and the Omega test [24]
3. We show that the former two are equivalent while the latter is a substantial optimization of these.

Our contributions

1. We study **integer projection**, that is, the elimination of a sequence of existential quantifiers in a Presburger formula
2. We compare three well-know projections in a **unified framework**: Cooper's algorithm [6], Williams' projection [29] and the Omega test [24]
3. We show that the former two are equivalent while the latter is a substantial optimization of these.
4. We have implemented a QE solver where integer projection can be done either

Our contributions

1. We study **integer projection**, that is, the elimination of a sequence of existential quantifiers in a Presburger formula
2. We compare three well-know projections in a **unified framework**: Cooper's algorithm [6], Williams' projection [29] and the Omega test [24]
3. We show that the former two are equivalent while the latter is a substantial optimization of these.
4. We have implemented a QE solver where integer projection can be done either
   a. via `IntegerPointDecomposition` [12],

Our contributions

1. We study **integer projection**, that is, the elimination of a sequence of existential quantifiers in a Presburger formula
2. We compare three well-know projections in a **unified framework**: Cooper's algorithm [6], Williams' projection [29] and the Omega test [24]
3. We show that the former two are equivalent while the latter is a substantial optimization of these.
4. We have implemented a QE solver where integer projection can be done either
   - a. via `IntegerPointDecomposition` [12],
   - b. or a parametric adaptation [13] of **Barvinok's algorithm** for counting integer points, thus computing Ehrhart polynomials.

Our contributions

1. We study **integer projection**, that is, the elimination of a sequence of existential quantifiers in a Presburger formula
2. We compare three well-know projections in a **unified framework**: Cooper's algorithm [6], Williams' projection [29] and the Omega test [24]
3. We show that the former two are equivalent while the latter is a substantial optimization of these.
4. We have implemented a QE solver where integer projection can be done either
   a. via `IntegerPointDecomposition` [12],
   b. or a parametric adaptation [13] of **Barvinok's algorithm** for counting integer points, thus computing Ehrhart polynomials.
5. Experimental results are provided.

Plan

1. Overview
2. Basic concepts
3. Quantifier elimination over the integers
4. Integer projection
5. Experimentation
6. Concluding remarks

Plan

Polyhedra

1. A subset $P \subseteq \mathbb{R}^d$ is a **convex polyhedron** (or simply a polyhedron) if
$$P = \{\mathbf{x} \in \mathbb{R}^d \mid A\mathbf{x} \leq \mathbf{b}\}$$
holds, for a matrix $A \in \mathbb{R}^{m \times d}$ and a vector $\mathbf{b} \in \mathbb{R}^m$, where $m \in \mathbb{N}_{>0}$;

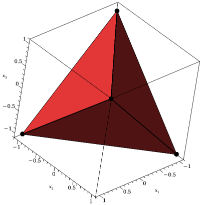
Polyhedra

1. A subset $P \subseteq \mathbb{R}^d$ is a **convex polyhedron** (or simply a polyhedron) if
$$P = \{\mathbf{x} \in \mathbb{R}^d \mid A\mathbf{x} \leq \mathbf{b}\}$$
holds, for a matrix $A \in \mathbb{R}^{m \times d}$ and a vector $\mathbf{b} \in \mathbb{R}^m$, where $m \in \mathbb{N}_{>0}$;
2. we call the linear system $\{A\mathbf{x} \leq \mathbf{b}\}$ an **H-representation** of P and denote by Polyhedron$(A, \mathbf{b})$ the polyhedron P, that is, the solution set of the system of linear inequalities $A\mathbf{x} \leq \mathbf{b}$.

Polyhedra

1. A subset $P \subseteq \mathbb{R}^d$ is a **convex polyhedron** (or simply a polyhedron) if
$$P = \{\mathbf{x} \in \mathbb{R}^d \mid A\mathbf{x} \leq \mathbf{b}\}$$
holds, for a matrix $A \in \mathbb{R}^{m \times d}$ and a vector $\mathbf{b} \in \mathbb{R}^m$, where $m \in \mathbb{N}_{>0}$;

2. we call the linear system $\{A\mathbf{x} \leq \mathbf{b}\}$ an **H-representation** of P and denote by Polyhedron$(A, \mathbf{b})$ the polyhedron P, that is, the solution set of the system of linear inequalities $A\mathbf{x} \leq \mathbf{b}$.



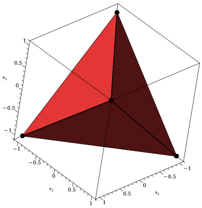
Polyhedra

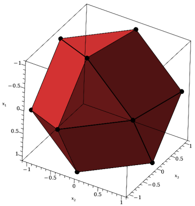
1. A subset $P \subseteq \mathbb{R}^d$ is a **convex polyhedron** (or simply a polyhedron) if

$$P = \{\mathbf{x} \in \mathbb{R}^d \mid A\mathbf{x} \leq \mathbf{b}\}$$

holds, for a matrix $A \in \mathbb{R}^{m \times d}$ and a vector $\mathbf{b} \in \mathbb{R}^m$, where $m \in \mathbb{N}_{>0}$;

2. we call the linear system $\{A\mathbf{x} \leq \mathbf{b}\}$ an **H-representation** of P and denote by Polyhedron$(A, \mathbf{b})$ the polyhedron P, that is, the solution set of the system of linear inequalities $A\mathbf{x} \leq \mathbf{b}$.



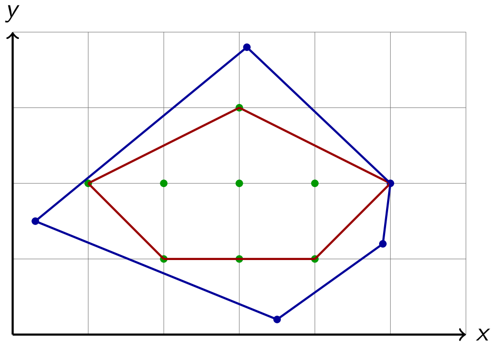


Integer hulls and lattices

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 $P \cap \mathbb{Z}^d$.

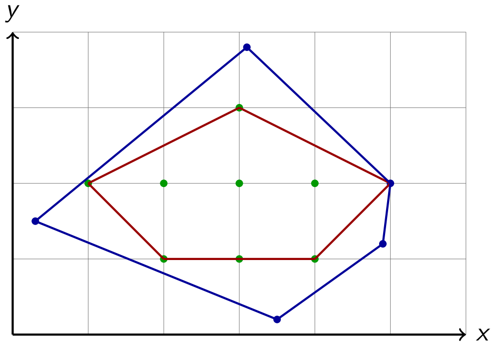
Integer hulls and lattices

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

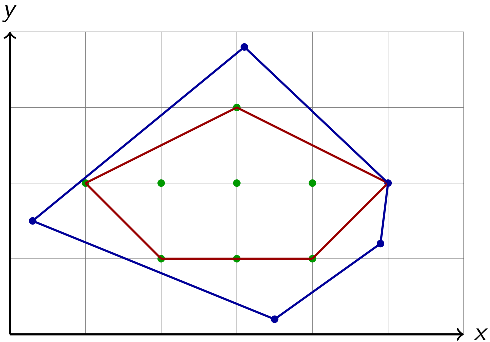
1. The **integer hull** of a polyhedron $P \subseteq \mathbb{Q}^d$, is the smallest convex polyhedron containing all the integer points of P. Thus, this is is the intersection of all convex polyhedra containing $P \cap \mathbb{Z}^d$.



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 $\mathbb{Z}\text{Polyhedron}(P, L)$.

skip slide

$\mathbb{Z}$-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

skip slide

$\mathbb{Z}$-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

skip slide

$\mathbb{Z}$-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
   $$\begin{cases} a_0 & \le x_1 \le & b_0 \\ a_1 & \le x_2 \le & b_1 \\ \vdots & \vdots & \vdots \\ a_{n-1} & \le x_d \le & b_{n-1}, \end{cases} \tag{2.1}$$
   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 $\mathbb{Z}\mathrm{Polyhedron}(P, L)$.
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

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

   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

skip slide

$\mathbb{Z}$-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 [14] decomposes any $\mathbb{Z}$-polyhedron into **normalized** $\mathbb{Z}$-polyhedra.

Solving parametric systems of linear congruences

1. Let $r, n \in \mathbb{Z}_{>0}$,

Solving parametric systems of linear congruences

1. Let $r, n \in \mathbb{Z}_{>0}$,
2. let $N \in \mathbb{Z}^{r \times n}$ be an integer matrix,

Solving parametric systems of linear congruences

1. Let $r, n \in \mathbb{Z}_{>0}$,
2. let $N \in \mathbb{Z}^{r \times n}$ be an integer matrix,
3. let $\mathbf{z}$ be an n-dimensional column vector whose coordinates are n independent integral variables $z_1, \ldots, z_n$,

Solving parametric systems of linear congruences

1. Let $r, n \in \mathbb{Z}_{>0}$,
2. let $N \in \mathbb{Z}^{r \times n}$ be an integer matrix,
3. let $\mathbf{z}$ be an n-dimensional column vector whose coordinates are n independent integral variables $z_1, \ldots, z_n$,
4. let $\mathbf{q}$ be an r-dimensional column vector whose coordinates are linear polynomials $q_1, \ldots, q_r \in \mathbb{Z}[w_1, \ldots, w_\nu]$,

Solving parametric systems of linear congruences

1. Let $r, n \in \mathbb{Z}_{>0}$,
2. let $N \in \mathbb{Z}^{r \times n}$ be an integer matrix,
3. let $\mathbf{z}$ be an n-dimensional column vector whose coordinates are n independent integral variables $z_1, \ldots, z_n$,
4. let $\mathbf{q}$ be an r-dimensional column vector whose coordinates are linear polynomials $q_1, \ldots, q_r \in \mathbb{Z}[w_1, \ldots, w_\nu]$,
5. we regard the variables $\mathbf{w} = w_1, \ldots, w_\nu$ as parameters,

Solving parametric systems of linear congruences

1. Let $r, n \in \mathbb{Z}_{>0}$,
2. let $N \in \mathbb{Z}^{r \times n}$ be an integer matrix,
3. let $\mathbf{z}$ be an n-dimensional column vector whose coordinates are n independent integral variables $z_1, \ldots, z_n$,
4. let $\mathbf{q}$ be an r-dimensional column vector whose coordinates are linear polynomials $q_1, \ldots, q_r \in \mathbb{Z}[w_1, \ldots, w_\nu]$,
5. we regard the variables $\mathbf{w} = w_1, \ldots, w_\nu$ as parameters,
6. let $\mathbf{m} \in {\mathbb{Z}_{>0}}^r$.

Solving parametric systems of linear congruences

1. Let $r, n \in \mathbb{Z}_{>0}$,
2. let $N \in \mathbb{Z}^{r \times n}$ be an integer matrix,
3. let $\mathbf{z}$ be an n-dimensional column vector whose coordinates are n independent integral variables $z_1, \ldots, z_n$,
4. let $\mathbf{q}$ be an r-dimensional column vector whose coordinates are linear polynomials $q_1, \ldots, q_r \in \mathbb{Z}[w_1, \ldots, w_\nu]$,
5. we regard the variables $\mathbf{w} = w_1, \ldots, w_\nu$ as parameters,
6. let $\mathbf{m} \in {\mathbb{Z}_{>0}}^r$.
7. Consider the system

$$N\ \mathbf{z} \equiv \mathbf{q} \mod \mathbf{m}. \tag{2.2}$$

Solving parametric systems of linear congruences

1. Let $r, n \in \mathbb{Z}_{>0}$,
2. let $N \in \mathbb{Z}^{r \times n}$ be an integer matrix,
3. let $\mathbf{z}$ be an n-dimensional column vector whose coordinates are n independent integral variables $z_1, \ldots, z_n$,
4. let $\mathbf{q}$ be an r-dimensional column vector whose coordinates are linear polynomials $q_1, \ldots, q_r \in \mathbb{Z}[w_1, \ldots, w_\nu]$,
5. we regard the variables $\mathbf{w} = w_1, \ldots, w_\nu$ as parameters,
6. let $\mathbf{m} \in {\mathbb{Z}_{>0}}^r$.
7. Consider the system

$$N \ \mathbf{z} \ \equiv \ \mathbf{q} \mod \mathbf{m}. \tag{2.2}$$

Theorem 1 (parametric multivariate CRT)

The values of $(w_1, \ldots, w_\nu)$ for which the above system has solutions form a lattice of $\mathbb{Z}^\nu$. Moreover, for each value of $(w_1, \ldots, w_\nu)$, the $\mathbf{z}$-solutions form a lattice of $\mathbb{Z}^n$.

Proof.

Compute the Hermite normal forms of the appropriate matrices. □

Plan

1. Overview
2. Basic concepts
3. Quantifier elimination over the integers
4. Integer projection
5. Experimentation
6. Concluding remarks

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 (1/2)

Theorem 2

Presburger arithmetic admits quantifier elimination.

Proof.

1. See the thesis of Mojżesz Presburger [26], the paper of David Cooper [6], and Christoph Haase's **Survival Guide to Presburger Arithmetic** [9].
2. See also our own proof in a few slides.

□

Quantifier elimination (1/2)

Theorem 2

Presburger arithmetic admits quantifier elimination.

Proof.

1. See the thesis of Mojżesz Presburger [26], the paper of David Cooper [6], and Christoph Haase's **Survival Guide to Presburger Arithmetic** [9].
2. See also our own proof in a few slides.

□

Recall our Presburger formula

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

Quantifier elimination (1/2)

Theorem 2

Presburger arithmetic admits quantifier elimination.

Proof.

1. See the thesis of Mojżesz Presburger [26], the paper of David Cooper [6], and Christoph Haase's **Survival Guide to Presburger Arithmetic** [9].
2. See also our own proof in a few slides.

□

Recall our Presburger formula

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

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.

Quantifier elimination (2/2)

Remark 1

1. *Recall*

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

Quantifier elimination (2/2)

Remark 1

1. *Recall*
$$F = Q_1 x_1 \cdots Q_m x_m \ \phi(x_1, \ldots, x_m, y_1, \ldots, y_n),$$
2. *If $m = 0$, then it "suffices" to determine the tuples of integer values $(y_1, \ldots, y_n)$ for which $\phi(x_1, \ldots, x_m, y_1, \ldots, y_n)$ is true.*

Quantifier elimination (2/2)

Remark 1

1. *Recall*

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

2. *If $m = 0$, then it "suffices" to determine the tuples of integer values $(y_1, \ldots, y_n)$ for which $\phi(x_1, \ldots, x_m, y_1, \ldots, y_n)$ is true.*
3. *Suppose $m > 0$. By induction, assume also $F = Q x_1 F'$, where F' is quantifier-free.*

Quantifier elimination (2/2)

Remark 1

1. *Recall*

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

2. *If $m = 0$, then it "suffices" to determine the tuples of integer values $(y_1, \ldots, y_n)$ for which $\phi(x_1, \ldots, x_m, y_1, \ldots, y_n)$ is true.*
3. *Suppose $m > 0$. By induction, assume also $F = Q x_1 F'$, where F' is quantifier-free.*
   a. *If $Q = \exists$, then we are now dealing with* **integer projection**, *see next section.*

Quantifier elimination (2/2)

Remark 1

1. *Recall*
$$F = Q_1 x_1 \cdots Q_m x_m \ \phi(x_1, \ldots, x_m, y_1, \ldots, y_n),$$
2. *If $m = 0$, then it "suffices" to determine the tuples of integer values $(y_1, \ldots, y_n)$ for which $\phi(x_1, \ldots, x_m, y_1, \ldots, y_n)$ is true.*
3. *Suppose $m > 0$. By induction, assume also $F = Q x_1 F'$, where F' is quantifier-free.*
   a. *If $Q = \exists$, then we are now dealing with* **integer projection***, see next section.*
   b. *If $Q = \forall$, then we can replace $\forall x_1 \ F'$ with $\neg(\exists x_1 \ \neg(F'))$,*

Quantifier elimination (2/2)

Remark 1

1. *Recall*

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

2. *If $m = 0$, then it "suffices" to determine the tuples of integer values $(y_1, \ldots, y_n)$ for which $\phi(x_1, \ldots, x_m, y_1, \ldots, y_n)$ is true.*
3. *Suppose $m > 0$. By induction, assume also $F = Q x_1 F'$, where F' is quantifier-free.*
   a. *If $Q = \exists$, then we are now dealing with* **integer projection***, see next section.*
   b. *If $Q = \forall$, then we can replace $\forall x_1 \ F'$ with $\neg(\exists x_1 \ \neg(F'))$,*
   c. *Whenever possible, we should make use of rules like:*

$$\forall x_1 \cdots \forall x_m \ C \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = \mathbf{q} \ \Rightarrow \ C = \mathbf{0} \ \bigwedge \ \mathbf{q} = \mathbf{0}, \qquad (3.1)$$

where

1. *$C \in \mathbb{Z}^{r \times m}$ is a matrix, and*

Quantifier elimination (2/2)

Remark 1

1. *Recall*

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

2. *If $m = 0$, then it "suffices" to determine the tuples of integer values $(y_1, \ldots, y_n)$ for which $\phi(x_1, \ldots, x_m, y_1, \ldots, y_n)$ is true.*
3. *Suppose $m > 0$. By induction, assume also $F = Q x_1 F'$, where F' is quantifier-free.*
   a. *If $Q = \exists$, then we are now dealing with* **integer projection**, *see next section.*
   b. *If $Q = \forall$, then we can replace $\forall x_1 \ F'$ with $\neg(\exists x_1 \ \neg(F'))$,*
   c. *Whenever possible, we should make use of rules like:*

$$\forall x_1 \cdots \forall x_m \ C \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = \mathbf{q} \ \Rightarrow \ C = \mathbf{0} \bigwedge \mathbf{q} = \mathbf{0}, \qquad (3.1)$$

   where
   1. *$C \in \mathbb{Z}^{r \times m}$ is a matrix, and*
   2. *$\mathbf{q} \in (\mathbb{Z}[y_1, \ldots, y_m])^r$ is a vector of linear polynomials.*

Coarsening the atoms

Remark 2

In Cooper's algorithm [6], when processing $\exists x_1\ F'$*, the formula* F' *uses the following four types of atoms:*

$$A_y < ax_1,\ ax_1 < A_y,\ k \mid (ax_1 + A_y), \text{ and } \neg(k \mid (ax_1 + A_y)), \qquad (3.2)$$

where $a \in \mathbb{Z}$ *and* $A_y \in \mathbb{Z}[y_1, \ldots, y_n]$ *is a linear polynomial.*

Coarsening the atoms

Remark 2

In Cooper's algorithm [6], when processing $\exists x_1\ F'$, the formula F' uses the following four types of atoms:

$$A_y < ax_1,\ ax_1 < A_y,\ k \mid (ax_1 + A_y), \text{ and } \neg\left(k \mid (ax_1 + A_y)\right), \tag{3.2}$$

where $a \in \mathbb{Z}$ and $A_y \in \mathbb{Z}[y_1, \ldots, y_n]$ is a linear polynomial.

Remark 3

We can rearrange our quantifier-free formula to:

$$\phi(x_1, \ldots, x_m, y_1, \ldots, y_n) = \bigvee_i Z_i(x_1, \ldots, x_m, y_1, \ldots, y_n), \tag{3.3}$$

where each Z_i is a predicate of the form

$$\begin{pmatrix} x_1 \\ \vdots \\ x_m \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{Z}\text{Polyhedron}(P_i, L_i), \tag{3.4}$$

for some polyhedron P_i and some integer lattice L_i.
We call such a predicate a **$\mathbb{Z}$-polyhedron predicate**.

Plan

Integer projection: $n = 1$

Remark 4

1. *From the above section, we consider the formula*

$$\exists x \ \phi(x, y_1, \ldots, y_n), \ \text{where} \ \phi(x, y_1, \ldots, y_n) = \bigvee_i \phi_i(x, y_1, \ldots, y_n), \tag{4.1}$$

   where $\phi_i(x, y_1, \ldots, y_n)$ is a conjunction of congruence relations and non-strict inequalities.

Integer projection: $n = 1$

Remark 4

1. *From the above section, we consider the formula*
$$\exists x \; \phi(x, y_1, \ldots, y_n), \quad \text{where} \quad \phi(x, y_1, \ldots, y_n) = \bigvee_i \; \phi_i(x, y_1, \ldots, y_n), \tag{4.1}$$
where $\phi_i(x, y_1, \ldots, y_n)$ is a conjunction of congruence relations and non-strict inequalities.
2. *We want the values of $\mathbf{y} = y_1, \ldots y_n$ so that $\exists x \; \phi(x, y_1, \ldots, y_n)$ holds.*

Integer projection: $n = 1$

Remark 4

1. *From the above section, we consider the formula*
$$\exists x\ \phi(x, y_1, \ldots, y_n), \quad \text{where} \quad \phi(x, y_1, \ldots, y_n) = \bigvee_i \phi_i(x, y_1, \ldots, y_n), \tag{4.1}$$
where $\phi_i(x, y_1, \ldots, y_n)$ *is a conjunction of congruence relations and non-strict inequalities.*
2. *We want the values of* $\mathbf{y} = y_1, \ldots y_n$ *so that* $\exists x\ \phi(x, y_1, \ldots, y_n)$ *holds.*
3. *We can further reduce the problem as follows.*

Integer projection: $n = 1$

Remark 4

1. *From the above section, we consider the formula*
$$\exists x\ \phi(x, y_1, \ldots, y_n), \ \text{where} \ \phi(x, y_1, \ldots, y_n) = \bigvee_i \ \phi_i(x, y_1, \ldots, y_n), \tag{4.1}$$
where $\phi_i(x, y_1, \ldots, y_n)$ is a conjunction of congruence relations and non-strict inequalities.
2. *We want the values of $\mathbf{y} = y_1, \ldots y_n$ so that $\exists x\ \phi(x, y_1, \ldots, y_n)$ holds.*
3. *We can further reduce the problem as follows.*

Remark 5

1. *Let $f_1, \ldots, f_s$, $g_1, \ldots, g_r \in \mathbb{Z}[x, \mathbf{y}]$ be linear and let $k_1, \ldots, k_r \in \mathbb{Z}_{>0}$.*

Integer projection: $n = 1$

Remark 4

1. *From the above section, we consider the formula*
   $$\exists x \; \phi(x, y_1, \dots, y_n), \quad \text{where} \quad \phi(x, y_1, \dots, y_n) = \bigvee_i \phi_i(x, y_1, \dots, y_n), \tag{4.1}$$
   where $\phi_i(x, y_1, \dots, y_n)$ *is a conjunction of congruence relations and non-strict inequalities.*
2. *We want the values of* $\mathbf{y} = y_1, \dots y_n$ *so that* $\exists x \; \phi(x, y_1, \dots, y_n)$ *holds.*
3. *We can further reduce the problem as follows.*

Remark 5

1. *Let* $f_1, \dots, f_s, \; g_1, \dots, g_r \in \mathbb{Z}[x, \mathbf{y}]$ *be linear and let* $k_1, \dots, k_r \in \mathbb{Z}_{>0}$.
2. *Consider the formula:*
   $$F(\mathbf{y}) : (\exists x \in \mathbb{Z}) \left\{ \begin{array}{ccc} f_1 & \leq & 0 \\ \vdots & \vdots & \vdots \\ f_s & \leq & 0 \end{array} \right. \wedge \begin{array}{cccc} g_1 & \equiv & 0 & \mod k_1 \\ \vdots & \vdots & \vdots & \\ g_r & \equiv & 0 & \mod k_r \end{array} \tag{4.2}$$

Integer projection: $n = 1$

Remark 4

1. *From the above section, we consider the formula*
$$\exists x\ \phi(x, y_1, \dots, y_n), \quad \text{where} \quad \phi(x, y_1, \dots, y_n) = \bigvee_i \phi_i(x, y_1, \dots, y_n), \tag{4.1}$$
where $\phi_i(x, y_1, \dots, y_n)$ is a conjunction of congruence relations and non-strict inequalities.
2. *We want the values of $\mathbf{y} = y_1, \dots y_n$ so that $\exists x\ \phi(x, y_1, \dots, y_n)$ holds.*
3. *We can further reduce the problem as follows.*

Remark 5

1. *Let $f_1, \dots, f_s, g_1, \dots, g_r \in \mathbb{Z}[x, \mathbf{y}]$ be linear and let $k_1, \dots, k_r \in \mathbb{Z}_{>0}$.*
2. *Consider the formula:*
$$F(\mathbf{y}) : (\exists x \in \mathbb{Z}) \left\{ \begin{array}{ccc} f_1 & \leq & 0 \\ \vdots & \vdots & \vdots \\ f_s & \leq & 0 \end{array} \right. \wedge \begin{array}{cccl} g_1 & \equiv & 0 & \bmod k_1 \\ \vdots & \vdots & \vdots & \\ g_r & \equiv & 0 & \bmod k_r \end{array} \tag{4.2}$$
3. *We shall determine the set $D(\mathbf{y})$ of integer tuples $(y_1, \dots y_n)$ for which $F(\mathbf{y})$ holds. We call $D(\mathbf{y})$ the* **integer projection** *of $F(\mathbf{y})$.*

Integer projection: $n = 1$, removing congruences

Remark 6

1. *Suppose that $r > 0$ holds, that is, we do have congruences.*

Integer projection: $n = 1$, removing congruences

Remark 6

1. *Suppose that $r > 0$ holds, that is, we do have congruences.*
2. *We should also check for implicit equations.*

Integer projection: $n = 1$, removing congruences

Remark 6

1. *Suppose that $r > 0$ holds, that is, we do have congruences.*
2. *We should also check for implicit equations.*
3. *We apply Hermite Normal Form, see Section 3 of the paper.*

Integer projection: $n = 1$, removing congruences

Remark 6

1. *Suppose that $r > 0$ holds, that is, we do have congruences.*
2. *We should also check for implicit equations.*
3. *We apply Hermite Normal Form, see Section 3 of the paper.*
4. *Now some of $x, y_1, \ldots, y_n$ are given by a lattice.*

Integer projection: $n = 1$, removing congruences

Remark 6

1. *Suppose that $r > 0$ holds, that is, we do have congruences.*
2. *We should also check for implicit equations.*
3. *We apply Hermite Normal Form, see Section 3 of the paper.*
4. *Now some of $x, y_1, \ldots, y_n$ are given by a lattice.*
5. *This process*

Integer projection: $n = 1$, removing congruences

Remark 6

1. *Suppose that $r > 0$ holds, that is, we do have congruences.*
2. *We should also check for implicit equations.*
3. *We apply Hermite Normal Form, see Section 3 of the paper.*
4. *Now some of $x, y_1, \ldots, y_n$ are given by a lattice.*
5. *This process*
   a. *introduces new variables (in order to define the lattice),*

Integer projection: $n = 1$, removing congruences

Remark 6

1. *Suppose that $r > 0$ holds, that is, we do have congruences.*
2. *We should also check for implicit equations.*
3. *We apply Hermite Normal Form, see Section 3 of the paper.*
4. *Now some of $x, y_1, \ldots, y_n$ are given by a lattice.*
5. *This process*
   a. *introduces new variables (in order to define the lattice),*
   b. *but eliminates at least* 1+ *the same number of variables from our system of linear inequalities*

Integer projection: $n = 1$, removing congruences

Remark 6

1. *Suppose that $r > 0$ holds, that is, we do have congruences.*
2. *We should also check for implicit equations.*
3. *We apply Hermite Normal Form, see Section 3 of the paper.*
4. *Now some of $x, y_1, \ldots, y_n$ are given by a lattice.*
5. *This process*
   a. *introduces new variables (in order to define the lattice),*
   b. *but eliminates at least $1+$ the same number of variables from our system of linear inequalities*
6. *As a result, we now have a $\mathbb{Z}$-polyhedron predicate.*

Integer projection: $n = 1$, removing congruences

Remark 6

1. *Suppose that $r > 0$ holds, that is, we do have congruences.*
2. *We should also check for implicit equations.*
3. *We apply Hermite Normal Form, see Section 3 of the paper.*
4. *Now some of $x, y_1, \ldots, y_n$ are given by a lattice.*
5. *This process*
   a. *introduces new variables (in order to define the lattice),*
   b. *but eliminates at least 1+ the same number of variables from our system of linear inequalities*
6. *As a result, we now have a $\mathbb{Z}$-polyhedron predicate.*
7. *If that process* **solves for** *x, then our problem becomes that of describing the points of a $\mathbb{Z}$-polyhedron, which can be done by our* IntegerPointDecomposition.

Integer projection: $n = 1$, removing congruences

Remark 6

1. *Suppose that $r > 0$ holds, that is, we do have congruences.*
2. *We should also check for implicit equations.*
3. *We apply Hermite Normal Form, see Section 3 of the paper.*
4. *Now some of $x, y_1, \ldots, y_n$ are given by a lattice.*
5. *This process*
   a. *introduces new variables (in order to define the lattice),*
   b. *but eliminates at least* 1+ *the same number of variables from our system of linear inequalities*
6. *As a result, we now have a $\mathbb{Z}$-polyhedron predicate.*
7. *If that process* **solves for** x*, then our problem becomes that of describing the points of a $\mathbb{Z}$-polyhedron, which can be done by our* IntegerPointDecomposition*.*
8. *If that process* **does not solve for** x*, then go to next slide.*

Integer projection: $n = 1$, no congruences, 2 inequalities

Remark 7

1. *We have used the congruences and we can focus on the inequalities.*

Integer projection: $n = 1$, no congruences, 2 inequalities

Remark 7

1. *We have used the congruences and we can focus on the inequalities.*
2. *We start with the case $s = 2$ and rename f_1, f_2 to A, B.*

Integer projection: $n = 1$, no congruences, 2 inequalities

Remark 7

1. *We have used the congruences and we can focus on the inequalities.*
2. *We start with the case $s = 2$ and rename f_1, f_2 to A, B.*
3. *We also write:*
$$A = A_{\mathbf{y}} - a\,x, \quad \text{and} \quad B = -B_{\mathbf{y}} + b\,x, \tag{4.3}$$
where $a, b \in \mathbb{Z}$ are non-zero and where $A_{\mathbf{y}}, B_{\mathbf{y}} \in \mathbb{Z}[\mathbf{y}]$ are linear

Integer projection: $n = 1$, no congruences, 2 inequalities

Remark 7

1. *We have used the congruences and we can focus on the inequalities.*
2. *We start with the case* $s = 2$ *and rename* f_1, f_2 *to* A, B.
3. *We also write:*
$$A = A_{\mathbf{y}} - a\,x, \quad \text{and} \quad B = -B_{\mathbf{y}} + b\,x, \tag{4.3}$$
where $a, b \in \mathbb{Z}$ *are non-zero and where* $A_{\mathbf{y}}, B_{\mathbf{y}} \in \mathbb{Z}[\mathbf{y}]$ *are linear*
4. *We further assume that* $a > 0$ *and* $b > 0$ *both hold.*

Integer projection: $n = 1$, no congruences, 2 inequalities

Remark 7

1. *We have used the congruences and we can focus on the inequalities.*
2. *We start with the case $s = 2$ and rename f_1, f_2 to A, B.*
3. *We also write:*
$$A = A_{\mathbf{y}} - a\,x, \quad \text{and} \quad B = -B_{\mathbf{y}} + b\,x, \tag{4.3}$$
where $a, b \in \mathbb{Z}$ are non-zero and where $A_{\mathbf{y}}, B_{\mathbf{y}} \in \mathbb{Z}[\mathbf{y}]$ are linear
4. *We further assume that $a > 0$ and $b > 0$ both hold.*
5. *With these assumptions, we call the inequalities $A \leq 0$ and $B \leq 0$, respectively a* **lower bound** *and an* **upper bound** *for x.*

Integer projection: $n = 1$, no congruences, 2 inequalities

Remark 7

1. *We have used the congruences and we can focus on the inequalities.*
2. *We start with the case $s = 2$ and rename f_1, f_2 to A, B.*
3. *We also write:*
$$A = A_{\mathbf{y}} - a\,x, \quad \text{and} \quad B = -B_{\mathbf{y}} + b\,x, \tag{4.3}$$
where $a, b \in \mathbb{Z}$ are non-zero and where $A_{\mathbf{y}}, B_{\mathbf{y}} \in \mathbb{Z}[\mathbf{y}]$ are linear
4. *We further assume that $a > 0$ and $b > 0$ both hold.*
5. *With these assumptions, we call the inequalities $A \leq 0$ and $B \leq 0$, respectively a* **lower bound** *and an* **upper bound** *for x.*
6. *Observe that Formula (4.2* null*) simplifies to:*
$$F(\mathbf{y}) : (\exists x \in \mathbb{Z})\ (A_{\mathbf{y}} \leq a\,x)\ \wedge\ (b\,x \leq B_{\mathbf{y}}), \tag{4.4}$$

Integer projection: $n = 1$, no congruences, 2 inequalities

Remark 7

1. *We have used the congruences and we can focus on the inequalities.*
2. *We start with the case $s = 2$ and rename f_1, f_2 to A, B.*
3. *We also write:*
$$A = A_{\mathbf{y}} - a\,x, \quad \text{and} \quad B = -B_{\mathbf{y}} + b\,x, \tag{4.3}$$
where $a, b \in \mathbb{Z}$ are non-zero and where $A_{\mathbf{y}}, B_{\mathbf{y}} \in \mathbb{Z}[\mathbf{y}]$ are linear
4. *We further assume that $a > 0$ and $b > 0$ both hold.*
5. *With these assumptions, we call the inequalities $A \leq 0$ and $B \leq 0$, respectively a* **lower bound** *and an* **upper bound** *for x.*
6. *Observe that Formula (4.2 null) simplifies to:*
$$F(\mathbf{y}) : (\exists x \in \mathbb{Z})\ (A_{\mathbf{y}} \leq a\,x)\ \wedge\ (b\,x \leq B_{\mathbf{y}}), \tag{4.4}$$
7. *We present a first formula for $D(\mathbf{y})$ based on Harris Williams [29, 30]*

Integer projection: $n = 1$, no congruences, 2 inequalities

Remark 7

1. *We have used the congruences and we can focus on the inequalities.*
2. *We start with the case* $s = 2$ *and rename* f_1, f_2 *to* A, B.
3. *We also write:*
$$A = A_{\mathbf{y}} - a\,x, \quad \text{and} \quad B = -B_{\mathbf{y}} + b\,x, \tag{4.3}$$
where $a, b \in \mathbb{Z}$ *are non-zero and where* $A_{\mathbf{y}}, B_{\mathbf{y}} \in \mathbb{Z}[\mathbf{y}]$ *are linear*
4. *We further assume that* $a > 0$ *and* $b > 0$ *both hold.*
5. *With these assumptions, we call the inequalities* $A \leq 0$ *and* $B \leq 0$, *respectively a* **lower bound** *and an* **upper bound** *for* x.
6. *Observe that Formula (4.2* null*) simplifies to:*
$$F(\mathbf{y}) : (\exists x \in \mathbb{Z})\ (A_{\mathbf{y}} \leq a\,x)\ \wedge\ (b\,x \leq B_{\mathbf{y}}), \tag{4.4}$$
7. *We present a first formula for* $D(\mathbf{y})$ *based on Harris Williams [29, 30]*
8. *Then, we present a second one based on William Pugh's Omega test [23, 24].*

Williams-style Projection

Theorem 3

Let $\ell = \mathrm{lcm}(a, b)$*,* $b' = \ell/a$ *and* $a' = \ell/b$*. For* $0 \le k < b$*, define*

$$E_k := \{\mathbf{y} \mid \mathrm{rem}(B_{\mathbf{y}}, b) = k\}.$$

Then, the following two formulas are equivalent:

1. $F(\mathbf{y})$: $(\exists x \in \mathbb{Z})\ (A_{\mathbf{y}} \le a\,x) \ \wedge\ (b\,x \le B_{\mathbf{y}})$,
2. $\bigvee_{k=0}^{k=b-1} (\mathbf{y} \in E_k) \wedge (a' k \ \le\ a' B_{\mathbf{y}} - b' A_{\mathbf{y}})$.

Williams-style Projection

Theorem 3

Let $\ell = \mathrm{lcm}(a, b)$*,* $b' = \ell/a$ *and* $a' = \ell/b$*. For* $0 \le k < b$*, define*

$$E_k := \{\mathbf{y} \mid \mathrm{rem}(B_{\mathbf{y}}, b) = k\}.$$

Then, the following two formulas are equivalent:

1. $F(\mathbf{y})$: $(\exists x \in \mathbb{Z})\ (A_{\mathbf{y}} \le a\,x) \ \wedge\ (b\,x \le B_{\mathbf{y}})$,
2. $\bigvee_{k=0}^{k=b-1} (\mathbf{y} \in E_k) \wedge (a' k \ \le\ a' B_{\mathbf{y}} - b' A_{\mathbf{y}})$.

Proof.

1. If $a = 1$ or $b = 1$ holds, then: $F(\mathbf{y}) \iff b'\,A_{\mathbf{y}} \le a'\,B_{\mathbf{y}}$.

□

Williams-style Projection

Theorem 3

Let $\ell = \mathrm{lcm}(a, b)$, $b' = \ell/a$ *and* $a' = \ell/b$. *For* $0 \leq k < b$, *define*

$$E_k := \{\mathbf{y} \mid \mathrm{rem}(B_{\mathbf{y}}, b) = k\}.$$

Then, the following two formulas are equivalent:

1. $F(\mathbf{y})$: $(\exists x \in \mathbb{Z})\ (A_{\mathbf{y}} \leq a\,x)\ \wedge\ (b\,x \leq B_{\mathbf{y}})$,
2. $\bigvee_{k=0}^{k=b-1} (\mathbf{y} \in E_k) \wedge (a'k \ \leq\ a'B_{\mathbf{y}} - b'A_{\mathbf{y}})$.

Proof.

1. If $a = 1$ or $b = 1$ holds, then: $F(\mathbf{y}) \iff b'\,A_{\mathbf{y}} \leq a'\,B_{\mathbf{y}}$.
2. From now on, assume $a > 1$ and $b > 1$ both hold. Observe that

$$F(\mathbf{y}) \iff (\exists x \in \mathbb{Z})\ (b'A_{\mathbf{y}} \leq \ell\,x)\ \wedge\ (\ell\,x \leq a'B_{\mathbf{y}}).$$

□

Williams-style Projection

Theorem 3

Let $\ell = \mathrm{lcm}(a, b)$, $b' = \ell/a$ *and* $a' = \ell/b$. *For* $0 \le k < b$, *define*

$$E_k := \{\mathbf{y} \mid \mathrm{rem}(B_{\mathbf{y}}, b) = k\}.$$

Then, the following two formulas are equivalent:

1. $F(\mathbf{y})$: $(\exists x \in \mathbb{Z})\ (A_{\mathbf{y}} \le a\,x) \wedge (b\,x \le B_{\mathbf{y}})$,
2. $\bigvee_{k=0}^{k=b-1} (\mathbf{y} \in E_k) \wedge (a' k \le a' B_{\mathbf{y}} - b' A_{\mathbf{y}})$.

Proof.

1. If $a = 1$ or $b = 1$ holds, then: $F(\mathbf{y}) \iff b'\,A_{\mathbf{y}} \le a'\,B_{\mathbf{y}}$.
2. From now on, assume $a > 1$ and $b > 1$ both hold. Observe that

$$F(\mathbf{y}) \iff (\exists x \in \mathbb{Z})\ (b' A_{\mathbf{y}} \le \ell\,x) \wedge (\ell\,x \le a' B_{\mathbf{y}}).$$

3. Hence, $F(\mathbf{y})$ says that a multiple of ℓ lies between $b' A_{\mathbf{y}}$ and $a' B_{\mathbf{y}}$).

□

Williams-style Projection

Theorem 3

Let $\ell = \mathrm{lcm}(a, b)$*,* $b' = \ell/a$ *and* $a' = \ell/b$*. For* $0 \le k < b$*, define*

$$E_k := \{\mathbf{y} \mid \mathrm{rem}(B_{\mathbf{y}}, b) = k\}.$$

Then, the following two formulas are equivalent:

1. *F(**y**):* $(\exists x \in \mathbb{Z})\ (A_{\mathbf{y}} \le a\,x)\ \wedge\ (b\,x \le B_{\mathbf{y}})$*,*
2. $\bigvee_{k=0}^{k=b-1} (\mathbf{y} \in E_k) \wedge (a'k \ \le\ a' B_{\mathbf{y}} - b' A_{\mathbf{y}})$*.*

Proof.

1. If $a = 1$ or $b = 1$ holds, then: $F(\mathbf{y}) \iff b'\,A_{\mathbf{y}} \le a'\,B_{\mathbf{y}}$.
2. From now on, assume $a > 1$ and $b > 1$ both hold. Observe that
$$F(\mathbf{y}) \iff (\exists x \in \mathbb{Z})\ (b' A_{\mathbf{y}} \le \ell\,x)\ \wedge\ (\ell\,x \le a' B_{\mathbf{y}}).$$
3. Hence, $F(\mathbf{y})$ says that a multiple of ℓ lies between $b' A_{\mathbf{y}}$ and $a' B_{\mathbf{y}})$.
4. Thus, $F(\mathbf{y}) \iff b'\,A_{\mathbf{y}} \ \le\ a'\,B_{\mathbf{y}} - \mathrm{rem}(a'\,B_{\mathbf{y}}, \ell)$.
5. That is, $F(\mathbf{y}) \iff b' A_{\mathbf{y}} \ \le\ a'(B_{\mathbf{y}} - \mathrm{rem}(B_{\mathbf{y}}, b))$.

□

Pugh's omega test (1/2)

Lemma 4 (William Pugh)

If we have:

$$aB_{\mathbf{y}} - bA_{\mathbf{y}} \;\geq\; (a-1)(b-1) \tag{4.5}$$

then $F(\mathbf{y})$ holds.

Pugh's omega test (1/2)

Lemma 4 (William Pugh)

If we have:

$$aB_{\mathbf{y}} - bA_{\mathbf{y}} \ \geq \ (a-1)(b-1) \tag{4.5}$$

then $F(\mathbf{y})$ holds.

Proof.

1. Consider the closed interval: $I \ := \ \left(\frac{A_{\mathbf{y}}}{a}, \frac{B_{\mathbf{y}}}{b}\right)$.

Pugh's omega test (1/2)

Lemma 4 (William Pugh)

If we have:

$$aB_{\mathbf{y}} - bA_{\mathbf{y}} \;\geq\; (a-1)(b-1) \tag{4.5}$$

then $F(\mathbf{y})$ holds.

Proof.

1. Consider the closed interval: $I \;:=\; \left(\frac{A_{\mathbf{y}}}{a}, \frac{B_{\mathbf{y}}}{b}\right)$.
2. If I does not contain an integer, then we have:

$$i < \frac{A_{\mathbf{y}}}{a} \leq \frac{B_{\mathbf{y}}}{b} < i+1, \quad \text{where} \quad i \;=\; \left\lfloor \frac{A_{\mathbf{y}}}{a} \right\rfloor. \tag{4.6}$$

Pugh's omega test (1/2)

Lemma 4 (William Pugh)

If we have:

$$aB_{\mathbf{y}} - bA_{\mathbf{y}} \ \geq \ (a-1)(b-1) \tag{4.5}$$

then $F(\mathbf{y})$ *holds.*

Proof.

1. Consider the closed interval: $I \ := \ \left(\frac{A_{\mathbf{y}}}{a}, \frac{B_{\mathbf{y}}}{b}\right)$.
2. If I does not contain an integer, then we have:
$$i < \frac{A_{\mathbf{y}}}{a} \leq \frac{B_{\mathbf{y}}}{b} < i+1, \quad \text{where} \quad i \ = \ \left\lfloor \frac{A_{\mathbf{y}}}{a} \right\rfloor. \tag{4.6}$$
3. Let $\rho := \mathrm{rem}(A_{\mathbf{y}}, a)$. Since $i < \frac{A_{\mathbf{y}}}{a}$ holds, we have:
$$A_{\mathbf{y}} \ = \ i\,a + \rho \ \text{ and } 0 < \rho < a, \tag{4.7}$$
4. from which we deduce: $\frac{A_{\mathbf{y}}}{a} - i \ \geq \ \frac{1}{a}$.

Pugh's omega test (1/2)

Lemma 4 (William Pugh)

If we have:

$$aB_{\mathbf{y}} - bA_{\mathbf{y}} \ \geq \ (a-1)(b-1) \tag{4.5}$$

then $F(\mathbf{y})$ holds.

Proof.

1. Consider the closed interval: $I \ := \ \left(\frac{A_{\mathbf{y}}}{a}, \frac{B_{\mathbf{y}}}{b}\right)$.
2. If I does not contain an integer, then we have:
$$i < \frac{A_{\mathbf{y}}}{a} \leq \frac{B_{\mathbf{y}}}{b} < i+1, \quad \text{where} \quad i \ = \ \left\lfloor \frac{A_{\mathbf{y}}}{a} \right\rfloor. \tag{4.6}$$
3. Let $\rho := \mathrm{rem}(A_{\mathbf{y}}, a)$. Since $i < \frac{A_{\mathbf{y}}}{a}$ holds, we have:
$$A_{\mathbf{y}} \ = \ i\,a + \rho \ \ \text{and} \ 0 < \rho < a, \tag{4.7}$$
4. from which we deduce: $\frac{A_{\mathbf{y}}}{a} - i \ \geq \ \frac{1}{a}$.
5. Similarly, we obtain: $i + 1 - \frac{B_{\mathbf{y}}}{b} \ \geq \ \frac{1}{b}$.

Pugh's omega test (1/2)

Lemma 4 (William Pugh)

If we have:

$$aB_{\mathbf{y}} - bA_{\mathbf{y}} \;\geq\; (a-1)(b-1) \tag{4.5}$$

then $F(\mathbf{y})$ *holds.*

Proof.

1. Consider the closed interval: $I \;:=\; \left(\frac{A_{\mathbf{y}}}{a}, \frac{B_{\mathbf{y}}}{b}\right)$.
2. If I does not contain an integer, then we have:
$$i < \frac{A_{\mathbf{y}}}{a} \leq \frac{B_{\mathbf{y}}}{b} < i+1, \quad \text{where} \quad i \;=\; \left\lfloor \frac{A_{\mathbf{y}}}{a} \right\rfloor. \tag{4.6}$$
3. Let $\rho := \mathrm{rem}(A_{\mathbf{y}}, a)$. Since $i < \frac{A_{\mathbf{y}}}{a}$ holds, we have:
$$A_{\mathbf{y}} \;=\; i\,a + \rho \;\; \text{and} \; 0 < \rho < a, \tag{4.7}$$
4. from which we deduce: $\frac{A_{\mathbf{y}}}{a} - i \;\geq\; \frac{1}{a}$.
5. Similarly, we obtain: $i + 1 - \frac{B_{\mathbf{y}}}{b} \;\geq\; \frac{1}{b}$.
6. From the above two inequalities, elementary manipulations yield:
$$aB_{\mathbf{y}} - bA_{\mathbf{y}} \;\leq\; ab - a - b. \tag{4.8}$$

Pugh's omega test (1/2)

Lemma 4 (William Pugh)

If we have:

$$aB_{\mathbf{y}} - bA_{\mathbf{y}} \ \geq \ (a-1)(b-1) \tag{4.5}$$

then $F(\mathbf{y})$ *holds.*

Proof.

1. Consider the closed interval: $I \ := \ \left(\frac{A_{\mathbf{y}}}{a}, \frac{B_{\mathbf{y}}}{b}\right)$.
2. If I does not contain an integer, then we have:
$$i < \frac{A_{\mathbf{y}}}{a} \leq \frac{B_{\mathbf{y}}}{b} < i+1, \quad \text{where} \quad i \ = \ \left\lfloor \frac{A_{\mathbf{y}}}{a} \right\rfloor. \tag{4.6}$$
3. Let $\rho := \mathrm{rem}(A_{\mathbf{y}}, a)$. Since $i < \frac{A_{\mathbf{y}}}{a}$ holds, we have:
$$A_{\mathbf{y}} \ = \ i\,a + \rho \ \text{ and } 0 < \rho < a, \tag{4.7}$$
4. from which we deduce: $\frac{A_{\mathbf{y}}}{a} - i \ \geq \ \frac{1}{a}$.
5. Similarly, we obtain: $i + 1 - \frac{B_{\mathbf{y}}}{b} \ \geq \ \frac{1}{b}$.
6. From the above two inequalities, elementary manipulations yield:
$$aB_{\mathbf{y}} - bA_{\mathbf{y}} \ \leq \ ab - a - b. \tag{4.8}$$
7. Therefore, if the above inequality does not hold, that is, if $aB_{\mathbf{y}} - bA_{\mathbf{y}} \ \geq \ (a-1)(b-1)$ does hold, then I contains an integer.

Pugh's omega test (2/2)

Theorem 5

Define $\kappa(a,b) := \lceil \frac{(a-1)(b-1)}{a'} \rceil$*. Then, Formula* $F(\mathbf{y})$ *is equivalent to:*

$$((a-1)(b-1) \leq aB_{\mathbf{y}} - bA_{\mathbf{y}}) \bigvee_{k=\kappa(a,b)}^{k=b-1} (\mathbf{y} \in E_k) \wedge (a'k \leq a'B_{\mathbf{y}} - b'A_{\mathbf{y}}) \,. \quad (4.9)$$

Proof.

This is a direct consequence of William Pugh's lemma and Harris Williams' projection formula □

Remark 8

1. *William Pugh's lemma reduces significantly the number of "cuts"*
2. *To take a concrete example, say with* $a = 7$ *and* $b = 11$*:*
   - a. *with Williams' projection alone* k *ranges from* 0 *to* 10*,*
   - b. *with William Pugh's lemma,* k *ranges from* 9 *to* 10*.*

Integer projection: $n = 1$, s inequalities

We now describe a procedure Projection$(f_1, \ldots, f_s; x)$ computing $D(\mathbf{y})$.

Integer projection: $n = 1$, s inequalities

We now describe a procedure Projection$(f_1, \ldots, f_s; x)$ computing $D(\mathbf{y})$.

1. If $f_1, \ldots, f_s$ only count lower (resp. upper) bounds for x, then return true.

Integer projection: $n = 1$, s inequalities

We now describe a procedure Projection$(f_1, \ldots, f_s; x)$ computing $D(\mathbf{y})$.

1. If $f_1, \ldots, f_s$ only count lower (resp. upper) bounds for x, then return true.
2. Initialize $D(\mathbf{y})$ to true.

Integer projection: $n = 1$, s inequalities

We now describe a procedure Projection($f_1, \ldots, f_s; x$) computing $D(\mathbf{y})$.

1. If $f_1, \ldots, f_s$ only count lower (resp. upper) bounds for x, then return true.
2. Initialize $D(\mathbf{y})$ to true.
3. For each pair (A, B) consisting of a lower bound and an upper bound of x, replace $D(\mathbf{y})$ with $D(\mathbf{y}) \wedge$ Projection(A, B), where Projection(A, B) is given by Pugh's omega test.

Integer projection: $n = 1$, s inequalities

We now describe a procedure Projection$(f_1, \ldots, f_s; x)$ computing $D(\mathbf{y})$.

1. If $f_1, \ldots, f_s$ only count lower (resp. upper) bounds for x, then return true.
2. Initialize $D(\mathbf{y})$ to true.
3. For each pair (A, B) consisting of a lower bound and an upper bound of x, replace $D(\mathbf{y})$ with $D(\mathbf{y}) \wedge$ Projection(A, B), where Projection(A, B) is given by Pugh's omega test.
4. Convert $D(\mathbf{y})$ to DNF yielding a formula of the form
$$S_0 \vee (C_1 \wedge S_1) \vee \cdots \vee (C_e \wedge S_e) \tag{4.10}$$
where

Integer projection: $n = 1$, s inequalities

We now describe a procedure Projection$(f_1, \ldots, f_s; x)$ computing $D(\mathbf{y})$.

1. If $f_1, \ldots, f_s$ only count lower (resp. upper) bounds for x, then return true.
2. Initialize $D(\mathbf{y})$ to true.
3. For each pair (A, B) consisting of a lower bound and an upper bound of x, replace $D(\mathbf{y})$ with $D(\mathbf{y}) \wedge$ Projection(A, B), where Projection(A, B) is given by Pugh's omega test.
4. Convert $D(\mathbf{y})$ to DNF yielding a formula of the form
$$S_0 \vee (C_1 \wedge S_1) \vee \cdots \vee (C_e \wedge S_e) \quad (4.10)$$
where
   - a. $S_0, S_1, \ldots, S_e$ are systems of non-strict linear inequalities in the variables $\mathbf{y}$,

Integer projection: $n = 1$, s inequalities

We now describe a procedure Projection$(f_1, \ldots, f_s; x)$ computing $D(\mathbf{y})$.

1. If $f_1, \ldots, f_s$ only count lower (resp. upper) bounds for x, then return true.
2. Initialize $D(\mathbf{y})$ to true.
3. For each pair (A, B) consisting of a lower bound and an upper bound of x, replace $D(\mathbf{y})$ with $D(\mathbf{y}) \wedge$ Projection(A, B), where Projection(A, B) is given by Pugh's omega test.
4. Convert $D(\mathbf{y})$ to DNF yielding a formula of the form
$$S_0 \vee (C_1 \wedge S_1) \vee \cdots \vee (C_e \wedge S_e) \tag{4.10}$$
where
   a. $S_0, S_1, \ldots, S_e$ are systems of non-strict linear inequalities in the variables $\mathbf{y}$,
   b. $C_1, \ldots, C_e$ are systems of congruences in the variables $\mathbf{y}$, and

Integer projection: $n = 1$, s inequalities

We now describe a procedure Projection$(f_1, \ldots, f_s; x)$ computing $D(\mathbf{y})$.

1. If $f_1, \ldots, f_s$ only count lower (resp. upper) bounds for x, then return true.
2. Initialize $D(\mathbf{y})$ to true.
3. For each pair (A, B) consisting of a lower bound and an upper bound of x, replace $D(\mathbf{y})$ with $D(\mathbf{y}) \wedge$ Projection(A, B), where Projection(A, B) is given by Pugh's omega test.
4. Convert $D(\mathbf{y})$ to DNF yielding a formula of the form
$$S_0 \vee (C_1 \wedge S_1) \vee \cdots \vee (C_e \wedge S_e) \quad (4.10)$$
where
   - a. $S_0, S_1, \ldots, S_e$ are systems of non-strict linear inequalities in the variables $\mathbf{y}$,
   - b. $C_1, \ldots, C_e$ are systems of congruences in the variables $\mathbf{y}$, and
   - c. S_0 is the conjunction of the $((a-1)(b-1) \leq aB_{\mathbf{y}} - bA_{\mathbf{y}})$, for all pairs (A, B) of lower and upper bounds of x.

Plan

1. Overview
2. Basic concepts
3. Quantifier elimination over the integers
4. Integer projection
5. Experimentation
6. Concluding remarks

Experimentation

Two strategies for integer projection

1. **IPD:** the one presented in this paper and based on the IntegerPointDecomposition algorithm [13],
2. **NIP:** one based on NumberOfIntegerPoints [13] and thus a parametric adaptation of **Barvinok's algorithm** [3, 17, 28].

Sources of test cases

1. examples from the literature (mainly from compiler theory)
2. examples from the SMT-LIA category of the SMT-LIB data-base [2],

Our code can be accessed here.

test	IPD MEMORY(MB)	IPD TIME(s)	NIP MEMORY(MB)	NIP TIME(s)
T1[**BoGoWo17**]	24.232	0.121	33.811	0.193
T2[**BoGoWo17**]	57.843	0.281	59.136	0.344
T3[**BoGoWo17**]	121.978	0.671	189.439	1.256
T4[**BoGoWo17**]	42.531	0.240	65.162	0.378
T5[**BoGoWo17**]	22.114	0.110	31.725	0.167
T6[**SeLoMe12**]	97.739	0.481	64.456	0.333
T7[**St23**]	671.154	3.506	1066.889	6.608
T8[**KVeWo08**]	69.087	0.338	58.668	0.328
T9 [**KVeWo08**]	245.156	1.235	979.964	6.462
T17[**BoGoWo17**]	5.315	0.043	12.771	0.060
T18[**CaLiZh22**]	39.055	0.200	48.237	0.205
T19[**Fe88**]	355.466	1.786	1715.958	10.941
T20[**Ve24**]	25.453	0.154	28.667	0.180
T32[**SeLoMe12**]	28216.613	156.989	$>$ 10 GB	$>$ 600
T33[**SeLoMe12**]	70.135	0.351	345.340	1.920
T34[**SeLoMe12**]	178.657	0.928	366.935	2.487
T35[**SeLoMe12**]	121.098	0.645	165.582	1.053
T36[**SeLoMe12**]	1243.682	6.209	798.004	4.822
T44[**Ve24**]	1.549	0.013	1.550	0.014
T45[**Ve24**]	1.549	0.014	1.551	0.013
T46[**Ve24**]	1.551	0.017	1.552	0.013
T47[**Ve15**]	49.726	0.236	45.779	0.219
T48[**Ve15**]	53.819	0.260	98.094	0.540
T49[**Ve15**]	32.190	0.197	27.997	0.153

Table: Maple 2024, Ubuntu 24.04.1 LTS, 16GB RAM and 12th Gen Intel(R) Core(TM) i5-1235U processor

1. **IPD = IntegerPointDecomposition**
2. **NIP = NumberOfIntegerPoints**

Using the SMT-LIB data-base

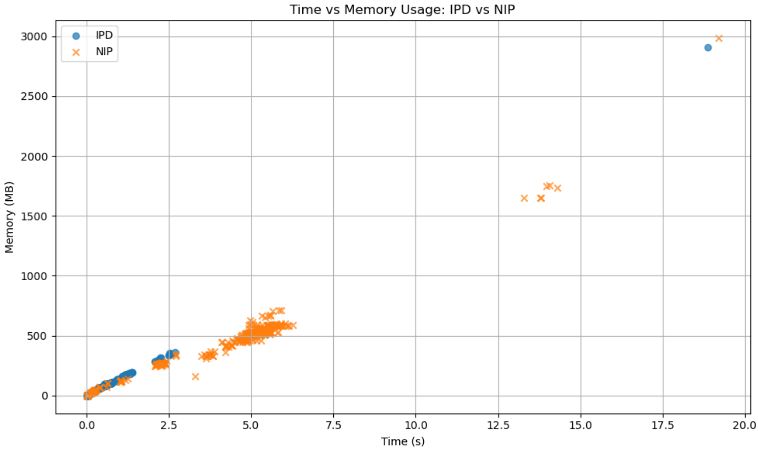


Figure: Time vs Memory for the SMT-LIA examples.

We have tested all the examples (about 400) from SMT-LIA that are Presburger formulas.

Plan

1. Overview
2. Basic concepts
3. Quantifier elimination over the integers
4. Integer projection
5. Experimentation
6. Concluding remarks

Summary and notes

We have discussed algebraic issues for the problem of quantifier elimination in Presbuger arithmetic. Our findings are:

1. Cooper's algorithm [6] is equivalent to Williams' projection [29].
2. The Omega test [24] is a substantial optimization of the latter two projections.
3. The algorithm IntegerPointDecomposition [12], which is based on the Omega test, seems experimentally superior to algorithms based on parametric versions of Barvinok's algorithm.

Work in progress

1. We shall continue investigating **heuristics to bypass double negation** when dealing with universal quantifiers.
2. In the presence of free variables, QE tend to split computations more than necessary and we are developing algorithms dealing with this **expression swell** issue.
3. We are extending our implementation of Presburger arithmetic to support certain classes of **non-linear expressions** that are of practical interest in compiler theory [5].

References

[1] H. Barbosa, C. W. Barrett, M. Brain, G. Kremer, H. Lachnitt, M. Mann, A. Mohamed, M. Mohamed, A. Niemetz, A. Nötzli, A. Ozdemir, M. Preiner, A. Reynolds, Y. Sheng, C. Tinelli, and Y. Zohar. "cvc5: A Versatile and Industrial-Strength SMT Solver". In: Tools and Algorithms for the Construction and Analysis of Systems - 2 Ed. by D. Fisman and G. Rosu. Vol. 13243. Lecture Notes in Computer Science. Springer, 2022, pp. 415–442. URL: https://doi.org/10.1007/978-3-030-99524-9%5C_24.

[2] C. Barrett, P. Fontaine, and C. Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). www.SMT-LIB.org. 2016.

[3] A. I. Barvinok. "A Polynomial Time Algorithm for Counting Integral Points in Polyhedra When the Dimension is Fixed". In: Math. Oper. Res. 19.4 (1994), pp. 769–779.

[4] Y. Bertot and P. Castéran.
Interactive Theorem Proving and Program Development - Coq'Art: Th
Texts in Theoretical Computer Science. An EATCS Series.
Springer, 2004. URL:
https://doi.org/10.1007/978-3-662-07964-5.

[5] 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 Sci
Ed. by J. Gould, M. Litoiu, and H. Lutfiyya. IBM / ACM, 2015,
pp. 70–79. URL:
http://dl.acm.org/citation.cfm?id=2886456.

[6] D. C. Cooper. "Theorem proving in arithmetic without
multiplication". In: Machine intelligence 7.91-99 (1972), p. 300.

[7] B. Dutertre. "Yices 2.2". In:
Computer Aided Verification - 26th International Conference, CAV 201
Ed. by A. Biere and R. Bloem. Vol. 8559. Lecture Notes in
Computer Science. Springer, 2014, pp. 737–744. URL:
https://doi.org/10.1007/978-3-319-08867-9%5C_49.

[8] T. Grosser, A. Größlinger, and C. Lengauer. "Polly - Performing Polyhedral Optimizations on a Low-Level Intermediate Representation". In: Parallel Process. Lett. 22.4 (2012).

[9] C. Haase. "A survival guide to Presburger arithmetic". In: ACM SIGLOG News 5.3 (2018), pp. 67–82.

[10] C. Haase, S. N. Krishna, K. Madnani, O. S. Mishra, and G. Zetzsche. "An Efficient Quantifier Elimination Procedure for Presburger Arithmetic". In: 51st International Colloquium on Automata, Languages, and Programm
Ed. by K. Bringmann, M. Grohe, G. Puppis, and O. Svensson. Vol. 297. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 142:1–142:17.

[11] J. Harrison. "HOL Light: A tutorial introduction". In: International Conference on Formal Methods in Computer-Aided Desig
Springer. 1996, pp. 265–269.

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

[13] R. Jing, Y. Lei, C. F. S. Maligec, and M. Moreno Maza. "Counting the Integer Points of Parametric Polytopes: A Maple Implementation". In: Computer Algebra in Scientific Computing - 26th International Worksh Ed. by F. Boulier, C. Mou, T. M. Sadykov, and E. V. Vorozhtsov. Vol. 14938. Lecture Notes in Computer Science. Springer, 2024, pp. 140–160. URL: https://doi.org/10.1007/978-3-031-69070-9%5C_9.

[14] 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.

[15] J. Leroux and G. Point. "TaPAS: The Talence Presburger Arithmetic Suite". In: Tools and Algorithms for the Construction and Analysis of Systems, 15 Ed. by S. Kowalewski and A. Philippou. Vol. 5505. Lecture Notes in Computer Science. Springer, 2009, pp. 182–185. URL: https://doi.org/10.1007/978-3-642-00768-2%5C_18.

[16] X. Leroy, S. Blazy, D. Kästner, B. Schommer, M. Pister, and C. Ferdinand. "CompCert-a formally verified optimizing compiler". In: ERTS 2016: Embedded Real Time Software and Systems, 8th Europea 2016.

[17] J. A. D. Loera, R. Hemmecke, J. Tauzer, and R. Yoshida. "Effective lattice point counting in rational convex polytopes". In: J. Symb. Comput. 38.4 (2004), pp. 1273–1302.

[18] D. C. Luckham, S. M. German, F. W. von Henke, R. A. Karp, P. Milne, D. C. Oppen, W. Polak, and W. L. Scherlis. Stanford pascal verifier user manual. Stanford University, 1979.

[19] L. de Moura. "Formalizing Mathematics using the Lean Theorem Prover". In: International Symposium on Artificial Intelligence and Mathematics, IS 2016. URL: https://isaim2016.cs.ou.edu/papers/ISAIM2016%5C_Proofs%5C_DeMoura.pdf.

[20] L. M. de Moura and N. S. Bjørner. "Z3: An Efficient SMT Solver". In: Tools and Algorithms for the Construction and Analysis of Systems, 14 Ed. by C. R. Ramakrishnan and J. Rehof. Vol. 4963. Lecture Notes in Computer Science. Springer, 2008, pp. 337–340. URL: https://doi.org/10.1007/978-3-540-78800-3%5C_24.

[21] T. Nipkow, L. C. Paulson, and M. Wenzel. Isabelle/HOL - A Proof Assistant for Higher-Order Logic. Vol. 2283. Lecture Notes in Computer Science. Springer, 2002. URL: https://doi.org/10.1007/3-540-45949-9.

[22] S. Pop, A. Cohen, C. Bastoul, S. Girbal, G.-A. Silber, and N. Vasilache. "GRAPHITE: Polyhedral analyses and optimizations for GCC". In: proceedings of the 2006 GCC developers summit. Vol. 6. 2006, pp. 90–91.

[23] W. Pugh. "A Practical Algorithm for Exact Array Dependence Analysis". In: Commun. ACM 35.8 (1992), pp. 102–114.

[24] W. W. Pugh. “The Omega test: a fast and practical integer programming algorithm for dependence analysis”. In: Proceedings Supercomputing '91, Albuquerque, NM, USA, November 1 ACM, 1991, pp. 4–13.

[25] P. Rümmer. “A Constraint Sequent Calculus for First-Order Logic with Linear Integer Arithmetic”. In: Proc. LPAR 2008. Vol. 5330. LNCS. Springer, pp. 274–289.

[26] R. Stansifer. Presburger's article on integer arithmetic: Remarks and translation. Tech. rep. Cornell University, 1984.

[27] S. Verdoolaege. “isl: An Integer Set Library for the Polyhedral Model”. In: Mathematical Software - ICMS 2010, Third International Congress on M Ed. by K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama. Vol. 6327. Lecture Notes in Computer Science. Springer, 2010, pp. 299–302.

[28] S. Verdoolaege, R. Seghir, K. Beyls, V. Loechner, and M. Bruynooghe. “Counting Integer Points in Parametric Polytopes Using Barvinok's Rational Functions”. In: Algorithmica 48.1 (2007), pp. 37–66.

[29] H. P. Williams. "Fourier-Motzkin elimination extension to integer programming problems". In: Journal of combinatorial theory, series A 21.1 (1976), pp. 118–123.

[30] H. Williams and J. Hooker. "Integer programming as projection". In: Discrete Optimization 22 (2016), pp. 291–311. ISSN: 1572-5286.