Linear integer arithmetic and Gaussian elimination

Dmitry Chistikov

University of Warwick, United Kingdom

CAALM 2025

06 June 2025

Linear Integer Arithmetic (Presburger arithmetic):

the first-order theory of natural numbers with addition and order.

$$\forall x \; \exists y \; \exists z \; ((x = 2y) \vee (x = 2z + 1))$$

Linear Integer Arithmetic (Presburger arithmetic):

the first-order theory of natural numbers with addition and order.

$$\forall x \; \exists y \; \exists z \; ((x = 2y) \vee (x = 2z + 1))$$

It is a language in which we can:

- talk about natural numbers (referred to as variables x, y, ...)

Linear Integer Arithmetic (Presburger arithmetic):

the first-order theory of natural numbers with addition and order.

$$\forall x \; \exists y \; \exists z \; ((x = 2y) \vee (x = 2z + 1))$$

It is a language in which we can:

- talk about natural numbers (referred to as variables x, y, ...)
- assert linear inequalities involving these numbers

Linear Integer Arithmetic (Presburger arithmetic):

the first-order theory of natural numbers with addition and order.

$$\forall x \; \exists y \; \exists z \; ((x = 2y) \vee (x = 2z + 1))$$

It is a language in which we can:

- talk about natural numbers (referred to as variables x, y, $\ldots$)
- assert linear inequalities involving these numbers
- form Boolean combinations ($\wedge$, $\vee$, $\neg$) of assertions

Linear Integer Arithmetic (Presburger arithmetic):

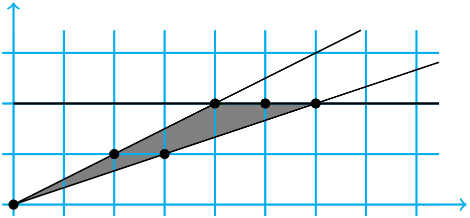
the first-order theory of natural numbers with addition and order.

$$\forall x \; \exists y \; \exists z \; ((x = 2y) \vee (x = 2z + 1))$$

It is a language in which we can:

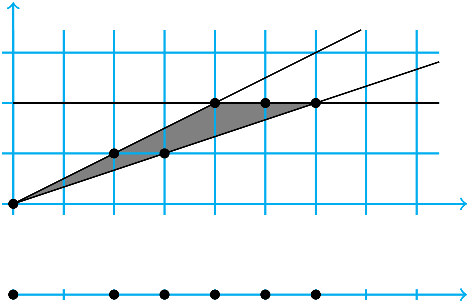
- talk about natural numbers (referred to as variables x, y, $\ldots$)
- assert linear inequalities involving these numbers
- form Boolean combinations ($\wedge$, $\vee$, $\neg$) of assertions
- quantify over (all) natural numbers ($\exists$, $\forall$)

Linear Integer Arithmetic (Presburger arithmetic):



$$(x \leqslant 3y) \wedge (2y \leqslant x) \wedge (y \leqslant 2)$$

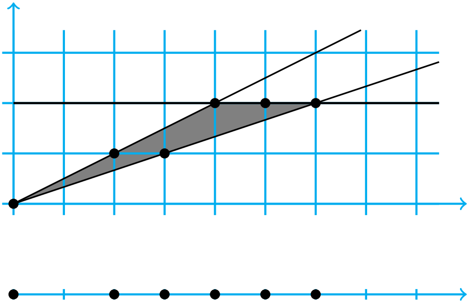
Linear Integer Arithmetic (Presburger arithmetic):

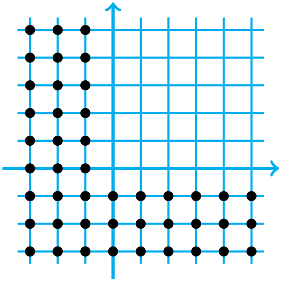


$\exists y \; (x \leqslant 3y) \wedge (2y \leqslant x) \wedge (y \leqslant 2)$

$\{0, 2, 3, 4, 5, 6\}$

Linear Integer Arithmetic (Presburger arithmetic):





$\exists y \ (x \leqslant 3y) \wedge (2y \leqslant x) \wedge (y \leqslant 2)$

$(x < 0) \vee (y < 0)$

$\{0, 2, 3, 4, 5, 6\}$

Linear Integer Arithmetic (Presburger arithmetic):

the first-order theory of natural numbers with addition and order.

$$\forall x \; \exists y \; \exists z \; ((x = 2y) \lor (x = 2z + 1))$$

It is a language in which we can:

- talk about natural numbers (referred to as variables x, y, ...)
- assert linear inequalities involving these numbers
- form Boolean combinations ($\land$, $\lor$, $\neg$) of assertions
- quantify over (all) natural numbers ($\exists$, $\forall$)

Linear Integer Arithmetic (Presburger arithmetic):

the first-order theory of natural numbers with addition and order.

Mojżesz Presburger

The Frobenius coin problem

Given a big supply of coins in denominations $a_1, \ldots, a_k \in \mathbb{N}$,
what is the largest amount f that cannot be generated?
Does such an f exist?

The Frobenius coin problem

Given a big supply of coins in denominations $a_1, \ldots, a_k \in \mathbb{N}$, what is the largest amount f that cannot be generated?
Does such an f exist?

$$(n = a_1 \cdot x_1 + \cdots + a_k \cdot x_k)$$

The Frobenius coin problem

Given a big supply of coins in denominations $a_1, \ldots, a_k \in \mathbb{N}$, what is the largest amount f that cannot be generated? Does such an f exist?

$$\exists x_1 \, \exists x_2 \ldots \exists x_k \; (n = a_1 \cdot x_1 + \cdots + a_k \cdot x_k)$$

The Frobenius coin problem

Given a big supply of coins in denominations $a_1, \ldots, a_k \in \mathbb{N}$,
what is the largest amount f that cannot be generated?
Does such an f exist?

$$\forall n \left(n \leqslant f \vee \exists x_1 \, \exists x_2 \ldots \exists x_k \, (n = a_1 \cdot x_1 + \cdots + a_k \cdot x_k \right)$$

The Frobenius coin problem

Given a big supply of coins in denominations $a_1, \ldots, a_k \in \mathbb{N}$, what is the largest amount f that cannot be generated?
Does such an f exist?

$$\Phi(f): \quad \forall n \left(n \leqslant f \vee \exists x_1\, \exists x_2 \ldots \exists x_k\, (n = a_1 \cdot x_1 + \cdots + a_k \cdot x_k)\right)$$

The Frobenius coin problem

Given a big supply of coins in denominations $a_1, \ldots, a_k \in \mathbb{N}$,
what is the largest amount f that cannot be generated?
Does such an f exist?

$$\Phi(f): \quad \forall n \left(n \leqslant f \vee \exists x_1 \, \exists x_2 \ldots \exists x_k \, (n = a_1 \cdot x_1 + \cdots + a_k \cdot x_k) \right)$$

Now $\Phi(f) \wedge \neg\Phi(f-1)$ expresses the property in question.

Decision problem for Presburger arithmetic

Input: sentence φ in Presburger arithmetic
Output: is φ true or false?

$\forall x\ \exists y\ \exists z\ ((x = 2y) \vee (x = 2z + 1))$ (in P.a.)

Decision problem for Presburger arithmetic

Input: sentence φ in Presburger arithmetic
Output: is φ true or false?

$$\forall x\ \exists y\ \exists z\ ((x = 2y) \vee (x = 2z + 1)) \qquad \text{(in P.a.)}$$

$$\forall s_1\ \forall s_2\ \exists x_1\ \exists x_2\ \ s_1 + ax_1 = s_2 + bx_2 \qquad \text{(in P.a. for fixed } a, b \in \mathbb{N}\text{)}$$

Decision problem for Presburger arithmetic

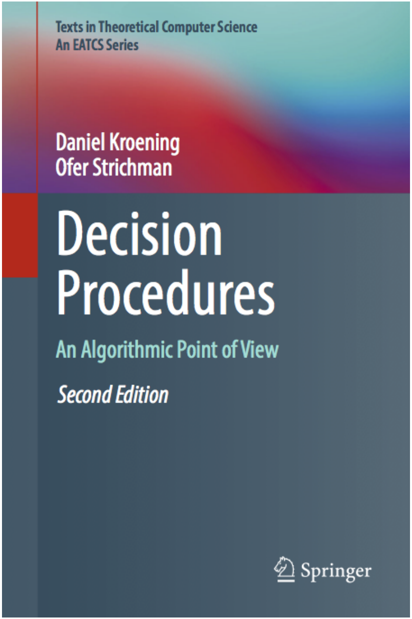
Input: sentence φ in Presburger arithmetic
Output: is φ true or false?

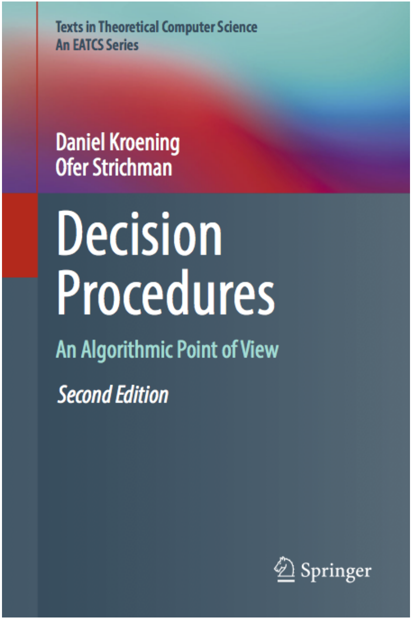
$\forall x\ \exists y\ \exists z\ ((x = 2y) \vee (x = 2z + 1))$	(in P.a.)
$\forall s_1\ \forall s_2\ \exists x_1\ \exists x_2\ \ s_1 + ax_1 = s_2 + bx_2$	(in P.a. for fixed $a, b \in \mathbb{N}$)
$\forall x\ \exists y\ ((y > x) \wedge P(y) \wedge P(y + 2))$	(not in P.a.)

Decision problem for Presburger arithmetic

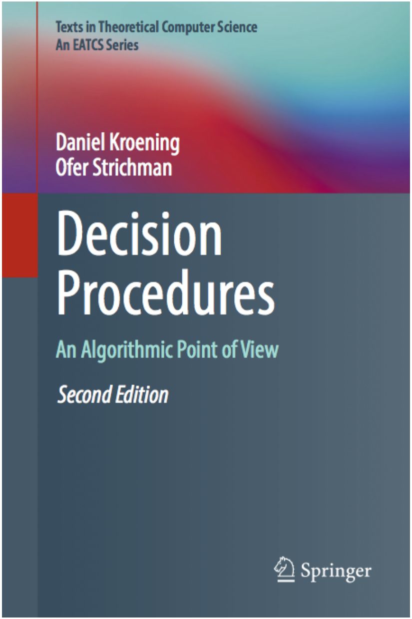
Input: sentence φ in Presburger arithmetic
Output: is φ true or false?

$\forall x\ \exists y\ \exists z\ ((x = 2y) \vee (x = 2z + 1))$	(in P.a.)
$\forall s_1\ \forall s_2\ \exists x_1\ \exists x_2\ \ s_1 + ax_1 = s_2 + bx_2$	(in P.a. for fixed $a, b \in \mathbb{N}$)
$\forall x\ \exists y\ ((y > x) \wedge P(y) \wedge P(y + 2))$	(not in P.a.)
$\forall x\ \forall y\ [(y \mid x) \wedge (y \mid x + 1)] \rightarrow y \leqslant 1$	(not in P.a.)


Texts in Theoretical Computer Science
An EATCS Series
Daniel Kroening
Ofer Strichman
Decision Procedures
An Algorithmic Point of View
Second Edition
Springer



Decision problems are solved by decision procedures, implemented in **satisfiability modulo theories (SMT)** solvers.



Decision problems are solved by decision procedures, implemented in **satisfiability modulo theories (SMT)** solvers.

- Common framework/toolbox for problems from various domains
- Growing software support

Z3 CVC5

Redlog

Three views: three types of decision procedures

View	Geometry	Automata theory	Symbolic computation (quantifier elimination)
Repr.	Semi-linear sets	Finite automata	Logical formulas

1. Three views on linear integer arithmetic (LIA)

   — from geometry: Semi-linear sets
   — from automata theory: k-automatic sets
   — from symbolic computation: Quantifier elimination

2. Integer programming in NP by quantifier elimination

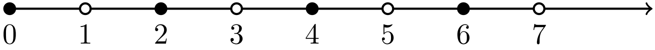
   — Gaussian elimination

View from geometry: **Semi-linear sets**

Periodic and ultimately periodic sets of natural numbers

Suppose $S \subseteq \mathbb{N}$.

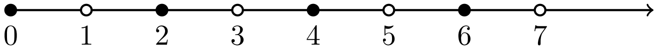
S is periodic if there exists a $p > 0$ such that,
for all $x \in \mathbb{N}$: $\quad x \in S$ iff $x + p \in S$.



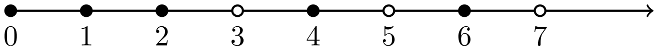
Periodic and ultimately periodic sets of natural numbers

Suppose $S \subseteq \mathbb{N}$.

S is periodic if there exists a $p > 0$ such that,
for all $x \in \mathbb{N}$: $\quad x \in S$ iff $x + p \in S$.



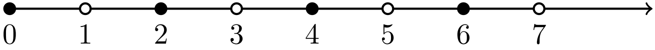
S is ultimately periodic if there exist N and $p > 0$ such that,
for all $x \geqslant N$: $\quad x \in S$ iff $x + p \in S$.



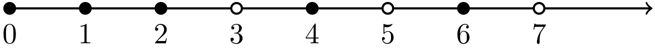
Periodic and ultimately periodic sets of natural numbers

Suppose $S \subseteq \mathbb{N}$.

S is periodic if there exists a $p > 0$ such that,
for all $x \in \mathbb{N}$: $\quad x \in S$ iff $x + p \in S$.

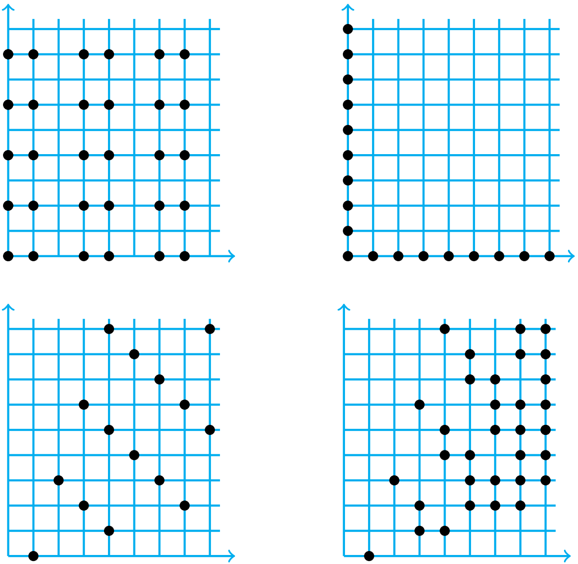


S is ultimately periodic if there exist N and $p > 0$ such that,
for all $x \geqslant N$: $\quad x \in S$ iff $x + p \in S$.



Ultimately periodic $=$ finite union of arithmetic progressions.

Ultimately periodic sets in higher dimension



Linear and semi-linear sets

[Parikh (1961)]

Vector $\boldsymbol{b}$, set of vectors $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_s\}$

Linear set (integer cone): $|P| < \infty$

$$L(\boldsymbol{b}, P) = \{\boldsymbol{b} + \lambda_1 \boldsymbol{p}_1 + \ldots + \lambda_s \boldsymbol{p}_s : \lambda_1, \ldots, \lambda_s \in \mathbb{N}\}$$

Rohit J. Parikh

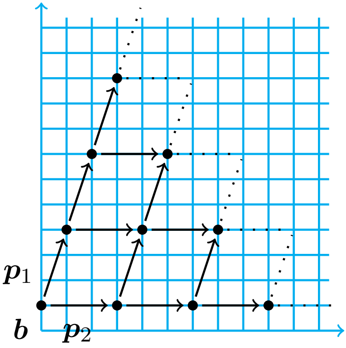
Linear and semi-linear sets

[Parikh (1961)]

Vector $\boldsymbol{b}$, set of vectors $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_s\}$

Linear set (integer cone): $|P| < \infty$

$$L(\boldsymbol{b}, P) = \{\boldsymbol{b} + \lambda_1 \boldsymbol{p}_1 + \ldots + \lambda_s \boldsymbol{p}_s : \lambda_1, \ldots, \lambda_s \in \mathbb{N}\}$$



Linear and semi-linear sets

[Parikh (1961)]

Vector $\boldsymbol{b}$, set of vectors $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_s\}$

Linear set (integer cone): $|P| < \infty$

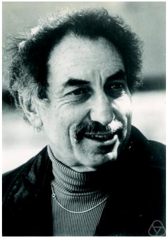
$$L(\boldsymbol{b}, P) = \{\boldsymbol{b} + \lambda_1 \boldsymbol{p}_1 + \ldots + \lambda_s \boldsymbol{p}_s : \lambda_1, \ldots, \lambda_s \in \mathbb{N}\}$$

Semi-linear set: $|I|, |P_i| < \infty$

$$M = \bigcup_{i \in I} L(\boldsymbol{b}_i, P_i)$$

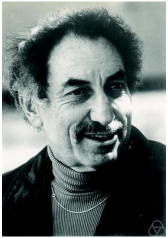
Theorem 1 (Ginsburg and Spanier, 1964).
Semi-linear sets = sets definable in Presburger arithmetic.


Seymour Ginsburg


Edwin H. Spanier

Theorem 1 (Ginsburg and Spanier, 1964).
Semi-linear sets = sets definable in Presburger arithmetic.


Seymour Ginsburg


Edwin H. Spanier

Corollary (Presburger, 1929): Presburger arithmetic is decidable.

More from the geometric view:

$\Rightarrow$ generating functions [Barvinok 1994]

$\Rightarrow$ syntactic sugar: Presburger with star
[Piskac and Kuncak 2008] [Haase and Zetzsche 2019]

$\Rightarrow$ nonlinear generalisations: almost semilinear sets
[Leroux 2011] [Esparza, Guttenberg, Raskin 2023]

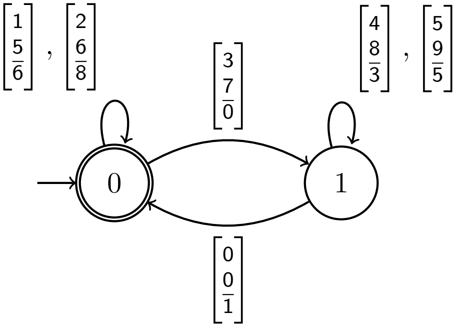
View from automata theory: **k-automatic sets**

Finite automaton can read triplets of digits and check the equality $x + y = z$:

$$\begin{array}{ll} x: & \;\;54\,321 \\ & + \\ y: & \;\;98\,765 \\ \hline z: & \;\;153\,086 \end{array}$$

Finite automaton can read triplets of digits and check the equality $x + y = z$:

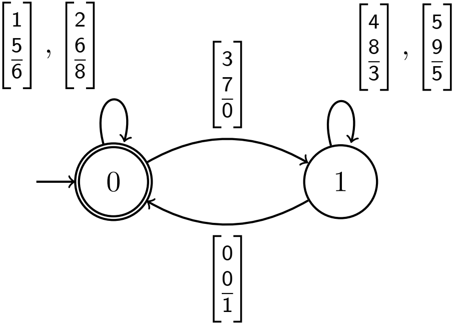
$$\begin{array}{lr} x: & 54\,321 \\ y: & + \; 98\,765 \\ \hline z: & 153\,086 \end{array}$$



Finite automaton can read triplets of digits and check the equality $x + y = z$:

$$\begin{array}{lr} x: & 54\,321 \\ y: & +\;98\,765 \\ \hline z: & 153\,086 \end{array}$$

J. Richard Büchi



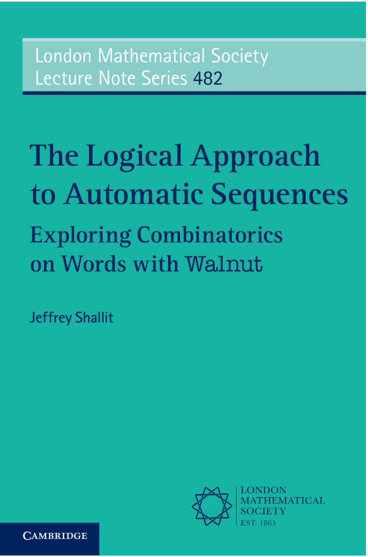
[1960]

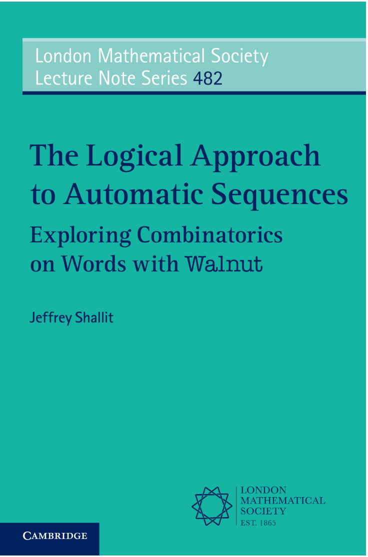
For $d \geqslant 1$, a set $S \subseteq \mathbb{N}^d$ is **2-automatic** (or: **2-recognizable**) if there is a deterministic finite automaton (DFA) that accepts the language

$$\{(w_1, \ldots, w_d) \in \left(\{0,1\}^d\right)^* : \text{for some } (n_1, \ldots, n_d) \in S, \text{ each } w_i \text{ is a binary expansion of } n_i\}.$$

Theorem 2 (Büchi 1960 + Bruyère 1985, corollary).

1. Every set definable in Presburger arithmetic is (effectively) 2-automatic.
2. There exists a 2-automatic set $S \subseteq \mathbb{N}$ that is **not** definable in Presburger arithmetic.


London Mathematical Society
Lecture Note Series 482
The Logical Approach to Automatic Sequences
Exploring Combinatorics on Words with Walnut
Jeffrey Shallit
LONDON MATHEMATICAL SOCIETY
EST. 1865
CAMBRIDGE



Theorems about Sturmian Words

We can use Pecan to prove many interesting properties of Sturmian words: one fundamental result is that Sturmian words are not *eventually periodic*.

Definition. *A word is* eventually periodic *if it is of the form* $abbbbb\ldots$ *for some subwords* a *and* b *(e.g.,* $0.10245454545\ldots$ *where the repeating part is* 45*).*

Theorem. *Sturmian words are not eventually periodic.*

Proof. In Pecan, prove the statement by writing the definition of "eventually periodic" and stating the theorem. Running the Pecan program below proves the theorem.

```
eventually_periodic(a, p) :=
    p > 0 ∧ ∃n. ∀i. if i > n then C[i] = C[i+p]

Theorem ("Sturmian words are not eventually periodic", {
    ∀a,p. if p > 0 then ¬eventually_periodic(a,p)
}).
```

We omit the pictures of the intermediate automata, as they have hundreds (or even thousands) of states, and so it is nearly impossible to understand them by looking at pictures of them. □

In this example, we state and prove a theorem about **all** Sturmian words.

- Previous theorem provers (e.g., Walnut [2]) in the same area could only prove theorems about a single Sturmian word, or small subsets of Sturmian words.

Using Pecan, we proved many other theorems about Sturmian words, including many classical results, some recent results, and notably, some **new** results.

[Lin, Ma, Oei, Teng, Vuksanovic, Schulz, Tursi, Hieronymi]

More from the automata-theoretic view:

$\Rightarrow$ links with numeration systems
[Michaux, Point, Rigo, Villemaire]

$\Rightarrow$ automatic structures [Hodgson 1976]
[Khoussainov, Nerode 1995] [Blumensath, Grädel 2000] etc.

View from symbolic computation: **Quantifier elimination**

Example (**not** in Presburger arithmetic):

$$\begin{aligned} \exists x \in \mathbb{R} :\ & x^2 + px + q = 0 \\ \Leftrightarrow \quad & p^2 - 4q \geqslant 0 \end{aligned}$$

Quantifier elimination for Presburger arithmetic

Example:

Quantifier elimination for Presburger arithmetic

Example:

$$\exists y.\ [(2x + z + 3 \leqslant y) \wedge (y \leqslant 6x - 11)] \ \leftrightarrow\ 2x + z + 3 \leqslant 6x - 11$$

Quantifier elimination for Presburger arithmetic

Example:

$$\exists y.\ [(2x + z + 3 \leqslant y) \land (y \leqslant 6x - 11)] \ \leftrightarrow \ 2x + z + 3 \leqslant 6x - 11$$

$$\exists y.\ [(2x + z + 3 \leqslant 2y) \land (2y \leqslant 6x - 11)] \ \leftrightarrow$$

Quantifier elimination for Presburger arithmetic

Example:

$$\exists y.\ [(2x + z + 3 \leqslant y) \wedge (y \leqslant 6x - 11)] \ \leftrightarrow\ 2x + z + 3 \leqslant 6x - 11$$

$$\exists y.\ [(2x + z + 3 \leqslant 2y) \wedge (2y \leqslant 6x - 11)] \ \leftrightarrow$$

$$(2x + z + 3 \leqslant 6x - 11) \wedge (6x - 11 \equiv 0 \mod 2) \vee$$
$$(2x + z + 3 \leqslant 6x - 12) \wedge (6x - 11 \equiv 1 \mod 2)$$

Quantifier elimination for Presburger arithmetic

Example:

$$\exists y.\ [(2x + z + 3 \leqslant y) \wedge (y \leqslant 6x - 11)] \ \leftrightarrow\ 2x + z + 3 \leqslant 6x - 11$$

$$\exists y.\ [(2x + z + 3 \leqslant 2y) \wedge (2y \leqslant 6x - 11)] \ \leftrightarrow$$

$$(2x + z + 3 \leqslant 6x - 11) \wedge (6x - 11 \equiv 0 \mod 2) \vee$$
$$(2x + z + 3 \leqslant 6x - 12) \wedge (6x - 11 \equiv 1 \mod 2)$$

Theorem 3 (Presburger 1929).
There exists an algorithm that,
given a quantifier-free formula φ and variable x,
outputs a quantifier-free formula φ' such that $(\exists x\ \varphi) \Leftrightarrow \varphi'$.

More from the symbolic computation view:

$\Rightarrow$ nonlinear extensions [Semenov 1980, 1984]

$\Rightarrow$ counting quantifiers $\exists^{\geqslant y} x \, \varphi$ [Schweikardt 2005] [Habermehl, Kuske 2015, 2023] [Ch., Haase, Mansutti 2022]

$\Rightarrow$ parametric Presburger arithmetic [Bogart, Goodrick, Woods 2017]

Theorem (Oppen 1973).
There is an algorithm that solves the decision problem for Presburger arithmetic in triply exponential time.

In fact, **all three views** provide 3-exp decision procedures.
[Klaedtke 2008, Durand-Gasselin and Habermehl 2010, 2012]
[Ch., Haase, Mansutti 2022]

Theorem (Oppen 1973).
There is an algorithm that solves the decision problem for Presburger arithmetic in triply exponential time.

In fact, **all three views** provide 3-exp decision procedures.
[Klaedtke 2008, Durand-Gasselin and Habermehl 2010, 2012]
[Ch., Haase, Mansutti 2022]

- Deciding Presburger arithmetic requires nondet. 2-exp time
[Fischer and Rabin, 1974]

Theorem (Oppen 1973).
There is an algorithm that solves the decision problem for Presburger arithmetic in triply exponential time.

In fact, **all three views** provide 3-exp decision procedures.
[Klaedtke 2008, Durand-Gasselin and Habermehl 2010, 2012]
[Ch., Haase, Mansutti 2022]

- Deciding Presburger arithmetic requires nondet. 2-exp time [Fischer and Rabin, 1974]
- . . . and is complete for STA$(*, 2^{2^{n^{O(1)}}}, n)$ [Berman, 1980]

Three views on integer programming

[Feasibility problem of] **integer (linear) programming:**

Input: matrix $A \in \mathbb{Z}^{m \times n}$ and vector $\boldsymbol{c} \in \mathbb{Z}^m$.
Output: does the system $A \cdot \boldsymbol{x} \leqslant \boldsymbol{c}$ have a solution in $\mathbb{Z}^n$?

Three views on integer programming

[Feasibility problem of] **integer (linear) programming:**

Input: matrix $A \in \mathbb{Z}^{m \times n}$ and vector $\boldsymbol{c} \in \mathbb{Z}^m$.
Output: does the system $A \cdot \boldsymbol{x} \leqslant \boldsymbol{c}$ have a solution in $\mathbb{Z}^n$?

Theorem (Borosh, Treybig 1976,
von zur Gathen, Sieveking 1978, Papadimitriou 1981).

Integer programming is NP-complete.

Three views on integer programming

[Feasibility problem of] **integer (linear) programming:**

Input: matrix $A \in \mathbb{Z}^{m \times n}$ and vector $\boldsymbol{c} \in \mathbb{Z}^m$.
Output: does the system $A \cdot \boldsymbol{x} \leqslant \boldsymbol{c}$ have a solution in $\mathbb{Z}^n$?

Theorem (Borosh, Treybig 1976,
von zur Gathen, Sieveking 1978, Papadimitriou 1981).

Integer programming is NP-complete.

Corollary (Oppen 1978).

Existential Presburger arithmetic is NP-complete.

Three views on integer programming

[Feasibility problem of] **integer (linear) programming:**

Input: matrix $A \in \mathbb{Z}^{m \times n}$ and vector $\boldsymbol{c} \in \mathbb{Z}^m$.
Output: does the system $A \cdot \boldsymbol{x} \leqslant \boldsymbol{c}$ have a solution in $\mathbb{Z}^n$?

Theorem (Borosh, Treybig 1976,
von zur Gathen, Sieveking 1978, Papadimitriou 1981).

Integer programming is NP-complete.

Corollary (Oppen 1978).

Existential Presburger arithmetic is NP-complete.

Actually: **membership in** NP **via each of three views.**

1. Three views on linear integer arithmetic (LIA)

— from geometry: Semi-linear sets
— from automata theory: k-automatic sets
— from symbolic computation: Quantifier elimination

2. Integer programming in NP by quantifier elimination

— Gaussian elimination

Why use quantifier elimination for integer programming?

Theorem (just seen).
Integer linear programming is in NP.

Why use quantifier elimination for integer programming?

Theorem (just seen).
Integer linear programming is in NP.

This talk: show this by quantifier elimination.
(Prior QE procedures produce 2-exp big formulas on ILP.)

Why use quantifier elimination for integer programming?

Theorem (just seen).
Integer linear programming is in NP.

This talk: show this by quantifier elimination.
(Prior QE procedures produce 2-exp big formulas on ILP.)

Theorem (Ch., Mansutti, Starchak 2024).
Integer linear-exponential programming is in NP.

(That is: additionally handling $y = 2^x$ and $y = (z \bmod 2^x)$.)

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only		
Inequalities		

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	
Inequalities		

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	in P
Inequalities		

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	in P
Inequalities	in P	

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	in P
Inequalities	in P	NP-complete

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	in P
Inequalities	in P	NP-complete

Gaussian elimination

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \times \boldsymbol{x} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	in P
Inequalities	in P	NP-complete

Gaussian elimination

$$\begin{bmatrix} * & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \times \boldsymbol{x} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	in P
Inequalities	in P	NP-complete

Gaussian elimination

$$\begin{bmatrix} * & . & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & . \end{bmatrix} \times \boldsymbol{x} = \begin{bmatrix} . \\ . \\ . \\ . \\ . \end{bmatrix}$$

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	in P
Inequalities	in P	NP-complete

Gaussian elimination

$$\begin{bmatrix} * & . & . & . & . & . & . & . \\ 0 & * & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & . \end{bmatrix} \times \boldsymbol{x} = \begin{bmatrix} . \\ . \\ . \\ . \\ . \end{bmatrix}$$

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	in P
Inequalities	in P	NP-complete

Gaussian elimination

$$\begin{bmatrix} * & . & . & . & . & . & . & . \\ 0 & * & . & . & . & . & . & . \\ 0 & 0 & . & . & . & . & . & . \\ 0 & 0 & . & . & . & . & . & . \\ 0 & 0 & . & . & . & . & . & . \end{bmatrix} \times \boldsymbol{x} = \begin{bmatrix} . \\ . \\ . \\ . \\ . \end{bmatrix}$$

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	in P
Inequalities	in P	NP-complete

Gaussian elimination (Gauss—Jordan 1888, Clasen 1888)

$$\begin{bmatrix} * & 0 & . & . & . & . & . & . \\ 0 & * & . & . & . & . & . & . \\ 0 & 0 & . & . & . & . & . & . \\ 0 & 0 & . & . & . & . & . & . \\ 0 & 0 & . & . & . & . & . & . \end{bmatrix} \times \boldsymbol{x} = \begin{bmatrix} . \\ . \\ . \\ . \\ . \end{bmatrix}$$

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	in P
Inequalities	in P	NP-complete

Gaussian elimination (Gauss—Jordan 1888, Clasen 1888)

$$\begin{bmatrix} * & 0 & . & . & . & . & . & . \\ 0 & * & . & . & . & . & . & . \\ 0 & 0 & * & . & . & . & . & . \\ 0 & 0 & . & . & . & . & . & . \\ 0 & 0 & . & . & . & . & . & . \end{bmatrix} \times \boldsymbol{x} = \begin{bmatrix} . \\ . \\ . \\ . \\ . \end{bmatrix}$$

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	in P
Inequalities	in P	NP-complete

Gaussian elimination (Gauss—Jordan 1888, Clasen 1888)

$$\begin{bmatrix} * & 0 & 0 & . & . & . & . & . \\ 0 & * & 0 & . & . & . & . & . \\ 0 & 0 & * & . & . & . & . & . \\ 0 & 0 & 0 & . & . & . & . & . \\ 0 & 0 & 0 & . & . & . & . & . \end{bmatrix} \times \boldsymbol{x} = \begin{bmatrix} . \\ . \\ . \\ . \\ . \end{bmatrix}$$

Solving systems of linear equations

	over $\mathbb{Q}$	over $\mathbb{Z}$
Equalities only	in P	in P
Inequalities	in P	NP-complete

Gaussian elimination (Gauss—Jordan 1888, Clasen 1888)

$$\begin{bmatrix} * & 0 & 0 & . & . & . & . & . \\ 0 & * & 0 & . & . & . & . & . \\ 0 & 0 & * & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \times \boldsymbol{x} = \begin{bmatrix} . \\ . \\ . \\ . \\ . \end{bmatrix}$$

Integer linear programming in NP by quantifier elimination

$\varphi(\boldsymbol{x}, \boldsymbol{y})$: system of linear equalities

replace each $\tau \leqslant 0$ with $\tau + z = 0$, where $z \in \mathbb{N}$ is a slack variable
$\ell \leftarrow 1; \quad s \leftarrow ()$
foreach x in $\boldsymbol{x}$ **do**
  if no equality contains x **then continue**
  $a \cdot x + \tau = 0$ (with $a \neq 0$) $\leftarrow$ an arbitrary equality that contains x
  $p \leftarrow \ell; \quad \ell \leftarrow a$
  if τ contains a slack variable z not assigned by s **then**
    $v \leftarrow$ **guess** an integer in $[0, |a| \cdot \mathit{mod}(\varphi) - 1]$
    append pair $\langle v, z \rangle$ to s
  multiply all constraints in φ by a and replace x by $-\tau/a$
  divide each constraint in φ by p
  $\varphi \leftarrow \varphi \wedge (a \mid \tau)$
apply substitutions of s

Integer linear programming in NP by quantifier elimination

$\varphi(\boldsymbol{x}, \boldsymbol{y})$: system of linear equalities

replace each $\tau \leqslant 0$ with $\tau + z = 0$, where $z \in \mathbb{N}$ is a slack variable
$\ell \leftarrow 1; \quad s \leftarrow ()$
foreach x in $\boldsymbol{x}$ **do** /* growth of coefficients is exponential */
  if no equality contains x **then continue**
  $a \cdot x + \tau = 0$ (with $a \neq 0$) $\leftarrow$ an arbitrary equality that contains x
  $p \leftarrow \ell; \quad \ell \leftarrow a$
  if τ contains a slack variable z not assigned by s **then**
    $v \leftarrow$ **guess** an integer in $[0, |a| \cdot \textit{mod}(\varphi) - 1]$
    append pair $\langle v, z \rangle$ to s
  multiply all constraints in φ by a and replace x by $-\tau/a$
  divide each constraint in φ by p
  $\varphi \leftarrow \varphi \wedge (a \mid \tau)$
apply substitutions of s

Integer linear programming in NP by quantifier elimination

$\varphi(\boldsymbol{x}, \boldsymbol{y})$: system of linear equalities

replace each $\tau \leqslant 0$ with $\tau + z = 0$, where $z \in \mathbb{N}$ is a slack variable
$\ell \leftarrow 1$; $s \leftarrow ()$
foreach x in $\boldsymbol{x}$ **do** /* growth of coefficients is now polynomial */
  if no equality contains x **then continue**
  $a \cdot x + \tau = 0$ (with $a \neq 0$) $\leftarrow$ an arbitrary equality that contains x
  $p \leftarrow \ell$; $\ell \leftarrow a$
  if τ contains a slack variable z not assigned by s **then**
    $v \leftarrow$ **guess** an integer in $[0, |a| \cdot \mathit{mod}(\varphi) - 1]$
    append pair $\langle v, z \rangle$ to s
  multiply all constraints in φ by a and replace x by $-\tau/a$
  divide each constraint in φ by p
  $\varphi \leftarrow \varphi \wedge (a \mid \tau)$
apply substitutions of s

Integer linear programming in NP by quantifier elimination

$\varphi(\boldsymbol{x}, \boldsymbol{y})$: system of linear equalities

---

replace each $\tau \leqslant 0$ with $\tau + z = 0$, where $z \in \mathbb{N}$ is a slack variable

$\ell \leftarrow 1$; $s \leftarrow ()$

foreach x in $\boldsymbol{x}$ **do** /* now over $\mathbb{Z}$ */

if no equality contains x **then continue**

$a \cdot x + \tau = 0$ (with $a \neq 0$) $\leftarrow$ an arbitrary equality that contains x

$p \leftarrow \ell$; $\ell \leftarrow a$

if τ contains a slack variable z not assigned by s **then**

$v \leftarrow$ **guess** an integer in $[0, |a| \cdot \mathit{mod}(\varphi) - 1]$

append pair $\langle v, z \rangle$ to s

multiply all constraints in φ by a and replace x by $-\tau/a$

divide each constraint in φ by p

$\varphi \leftarrow \varphi \land (a \mid \tau)$

apply substitutions of s

Integer linear programming in NP by quantifier elimination

$\varphi(\boldsymbol{x}, \boldsymbol{y})$: system of linear equalities and inequalities

---

replace each $\tau \leqslant 0$ with $\tau + z = 0$, where $z \in \mathbb{N}$ is a slack variable
$\ell \leftarrow 1$; $s \leftarrow ()$
foreach x in $\boldsymbol{x}$ **do** /* with inequalities... */
    if no equality contains x **then continue**
    $a \cdot x + \tau = 0$ (with $a \neq 0$) $\leftarrow$ an arbitrary equality that contains x
    $p \leftarrow \ell$; $\ell \leftarrow a$
    if τ contains a slack variable z not assigned by s **then**
        $v \leftarrow$ **guess** an integer in $[0, |a| \cdot \mathit{mod}(\varphi) - 1]$
        append pair $\langle v, z \rangle$ to s
    multiply all constraints in φ by a and replace x by $-\tau/a$
    divide each constraint in φ by p
    $\varphi \leftarrow \varphi \land (a \mid \tau)$
apply substitutions of s

Integer linear programming in NP by quantifier elimination

$\varphi(\boldsymbol{x}, \boldsymbol{y})$: system of linear equalities and inequalities

---

replace each $\tau \leqslant 0$ with $\tau + z = 0$, where $z \in \mathbb{N}$ is a slack variable
$\ell \leftarrow 1$; $s \leftarrow ()$
foreach x in $\boldsymbol{x}$ **do** /*... and shifts */
    if no equality contains x **then continue**
    $a \cdot x + \tau = 0$ (with $a \neq 0$) $\leftarrow$ **guess** an equality that contains x
    $p \leftarrow \ell$; $\ell \leftarrow a$
    if τ contains a slack variable z not assigned by s **then**
        $v \leftarrow$ **guess** an integer in $[0, |a| \cdot \mathit{mod}(\varphi) - 1]$
        append pair $\langle v, z \rangle$ to s /* substitution */
    multiply all constraints in φ by a and replace x by $-\tau/a$
    divide each constraint in φ by p
    $\varphi \leftarrow \varphi \wedge (a \mid \tau)$
apply substitutions of s

The magic of division by p (previous pivot)

$$ax + by + t' = 0$$
$$cx + dy + t'' = 0$$

The magic of division by p (previous pivot)

$$ax + by + t' = 0$$
$$cx + dy + t'' = 0$$
$$(ad - bc)\, y + (at'' - ct') = 0$$

Image source: John Tenniel. Book by Lewis Carroll, published by Macmillan, 1925.

The magic of division by p (previous pivot)

$$ax + by + t' = 0$$
$$cx + dy + t'' = 0$$
$$(ad - bc)\, y + (at'' - ct') = 0$$

The new coefficient is $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$

The magic of division by p (previous pivot)

$$ax + by + t' = 0$$
$$cx + dy + t'' = 0$$
$$(ad - bc)\, y + (at'' - ct') = 0$$

The new coefficient is $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ — the bit size of numbers doubles.

The magic of division by p (previous pivot)

$$ax + by + t' = 0$$
$$cx + dy + t'' = 0$$
$$(ad - bc)\, y + (at'' - ct') = 0$$

The new coefficient is $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ — the bit size of numbers doubles.
In fact, after 2 steps a common factor appears. [Weispfenning 1997]

Image source: John Tenniel. Book by Lewis Carroll, published by Macmillan, 1925.

The magic of division by p (previous pivot)

$$ax + by + t' = 0$$
$$cx + dy + t'' = 0$$
$$(ad - bc)\, y + (at'' - ct') = 0$$

The new coefficient is $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ — the bit size of numbers doubles.
In fact, after 2 steps a common factor appears. [Weispfenning 1997] [Bareiss 1968]

The magic of division by p (previous pivot)

$$ax + by + t' = 0$$
$$cx + dy + t'' = 0$$
$$(ad - bc)\, y + (at'' - ct') = 0$$

The new coefficient is $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ — the bit size of numbers doubles.
In fact, after 2 steps a common factor appears. [Weispfenning 1997] [Bareiss 1968] [Edmonds 1967]

The magic of division by p (previous pivot)

$$ax + by + t' = 0$$
$$cx + dy + t'' = 0$$
$$(ad - bc)\, y + (at'' - ct') = 0$$

The new coefficient is $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ — the bit size of numbers doubles.
In fact, after 2 steps a common factor appears. [Weispfenning 1997] [Bareiss 1968] [Edmonds 1967] [Clasen 1888]

Image source: John Tenniel. Book by Lewis Carroll, published by Macmillan, 1925.

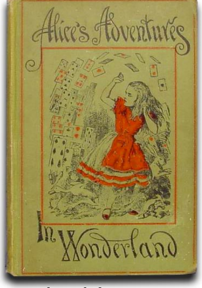
The magic of division by p (previous pivot)

$$ax + by + t' = 0$$
$$cx + dy + t'' = 0$$
$$(ad - bc)\, y + (at'' - ct') = 0$$

The new coefficient is $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ — the bit size of numbers doubles.
In fact, after 2 steps a common factor appears. [Weispfenning 1997] [Bareiss 1968] [Edmonds 1967] [Clasen 1888] [Dodgson 1867]

Image source: John Tenniel. Book by Lewis Carroll, published by Macmillan, 1925.

The magic of division by p (previous pivot)

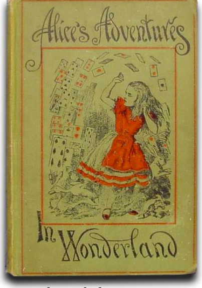


$$ax + by + t' = 0$$
$$cx + dy + t'' = 0$$
$$(ad - bc)\,y + (at'' - ct') = 0$$

The new coefficient is $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ — the bit size of numbers doubles.
In fact, after 2 steps a common factor appears. [Weispfenning 1997] [Bareiss 1968] [Edmonds 1967] [Clasen 1888] [Dodgson 1867]

Image source: John Tenniel. Book by Lewis Carroll, published by Macmillan, 1925.

The magic of division by p (previous pivot)

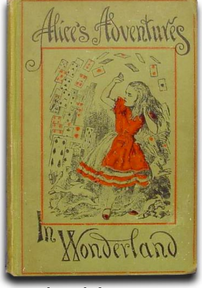


$$ax + by + t' = 0$$
$$cx + dy + t'' = 0$$
$$(ad - bc)\,y + (at'' - ct') = 0$$

The new coefficient is $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ — the bit size of numbers doubles.
In fact, after 2 steps a common factor appears. [Weispfenning 1997] [Bareiss 1968] [Edmonds 1967] [Clasen 1888] [Dodgson 1867]
(Sylvester's determinant identity / Desnanot—Jacobi identity 1851)

Image source: John Tenniel. Book by Lewis Carroll, published by Macmillan, 1925.

The magic of division by p (previous pivot)



$$ax + by + t' = 0$$
$$cx + dy + t'' = 0$$
$$(ad - bc)\, y + (at'' - ct') = 0$$

The new coefficient is $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ — the bit size of numbers doubles.
In fact, after 2 steps a common factor appears. [Weispfenning 1997] [Bareiss 1968] [Edmonds 1967] [Clasen 1888] [Dodgson 1867]
(Sylvester's determinant identity / Desnanot—Jacobi identity 1851)

$$\begin{vmatrix} \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} \\ \begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} \end{vmatrix} = a_{11} \cdot \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

Image source: John Tenniel. Book by Lewis Carroll, published by Macmillan, 1925.

Gaussian elimination

Gaussian elimination + slack variables

$$\frac{\text{Gaussian elimination} + \text{slack variables}}{\text{Bareiss factors}}$$

$$\frac{\text{Gaussian elimination} + \text{slack variables}}{\text{Bareiss factors}} \cdot \text{non-determinism}$$

$$\frac{\text{Gaussian elimination} + \text{slack variables}}{\text{Bareiss factors}} \cdot \text{non-determinism} \Longrightarrow$$

$$\Longrightarrow \text{IP in NP by QE}$$

Theorem (Ch., Mansutti, Starchak 2024).

The algorithm runs in non-deterministic polynomial time.
Given $\varphi\colon A \cdot \boldsymbol{x} + B \cdot \boldsymbol{y} \leqslant \boldsymbol{c}$, each non-deterministic branch β outputs $\psi_\beta\colon F_\beta \cdot \boldsymbol{y} \leqslant \boldsymbol{g}_\beta$ such that $(\exists \boldsymbol{x}\ \varphi) \Leftrightarrow \bigvee_\beta \psi_\beta$.

$$\frac{\text{Gaussian elimination} + \text{slack variables}}{\text{Bareiss factors}} \cdot \text{non-determinism} \implies$$

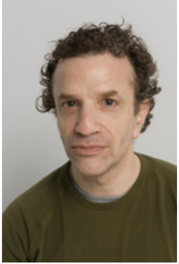
$$\implies \text{IP in NP by QE}$$

Theorem (Ch., Mansutti, Starchak 2024).
The algorithm runs in non-deterministic polynomial time.
Given $\varphi\colon A \cdot \boldsymbol{x} + B \cdot \boldsymbol{y} \leqslant \boldsymbol{c}$, each non-deterministic branch β outputs $\psi_\beta\colon F_\beta \cdot \boldsymbol{y} \leqslant \boldsymbol{g}_\beta$ such that $(\exists \boldsymbol{x}\ \varphi) \Leftrightarrow \bigvee_\beta \psi_\beta$.

$$\frac{\text{Gaussian elimination} + \text{slack variables}}{\text{Bareiss factors}} \cdot \text{non-determinism} \Longrightarrow$$

$$\Longrightarrow \text{IP in NP by QE}$$

Concurrently: a different QE procedure
[Haase, Krishna, Madnani, Mishra, Zetzsche 2024]



Michael Benedikt
(Oxford)

Alessio Mansutti
(IMDEA Software Institute)

Christoph Haase
(Oxford)

Mikhail Starchak
(St Petersburg)

Summary

1. Three views on linear integer arithmetic (LIA)

   — from geometry: Semi-linear sets
   — from automata theory: k-automatic sets
   — from symbolic computation: Quantifier elimination

2. Integer programming in NP by quantifier elimination

   — Gaussian elimination

Further directions

- Computational complexity of extensions
  (e.g.: counting quantifiers $\exists^{\geqslant y} x\ \varphi$; $\exists$ divisibility $x \mid y$)

- Decidability of nonlinear extensions
  (e.g., $\exists$ with several power predicates)

- Applications; computational complexity in special cases

Further directions

- Computational complexity of extensions
  (e.g.: counting quantifiers $\exists^{\geqslant y} x\ \varphi$; $\exists$ divisibility $x \mid y$)
- Decidability of nonlinear extensions
  (e.g., $\exists$ with several power predicates)
- Applications; computational complexity in special cases

Thank you!

https://warwick.ac.uk/chdir

Learn more:

1. A.R. Bradley, Z. Manna. The calculus of computation: decision procedures with applications to verification. Springer (2007).
2. S. Demri. Rudiments of Presburger arithmetic. Lecture notes (MPRI, M2, 2016). hal-03188114
3. C. Haase. A survival guide to Presburger arithmetic. SIGLOG News (2018).
4. D. Chistikov. An introduction to the theory of linear integer arithmetic. FSTTCS 2024.