Three views on Presburger arithmetic

Dmitry Chistikov

University of Warwick, United Kingdom

One FLAT World Seminar

29 April 2026

1. Introduction to Presburger arithmetic

2. Three views on Presburger arithmetic

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

3. Computational complexity of decision problems

Presburger arithmetic (Linear integer arithmetic):

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

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

Presburger arithmetic (Linear integer 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, ...)

Presburger arithmetic (Linear integer 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

Presburger arithmetic (Linear integer 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
- form Boolean combinations ($\wedge$, $\vee$, $\neg$) of assertions

Presburger arithmetic (Linear integer arithmetic):

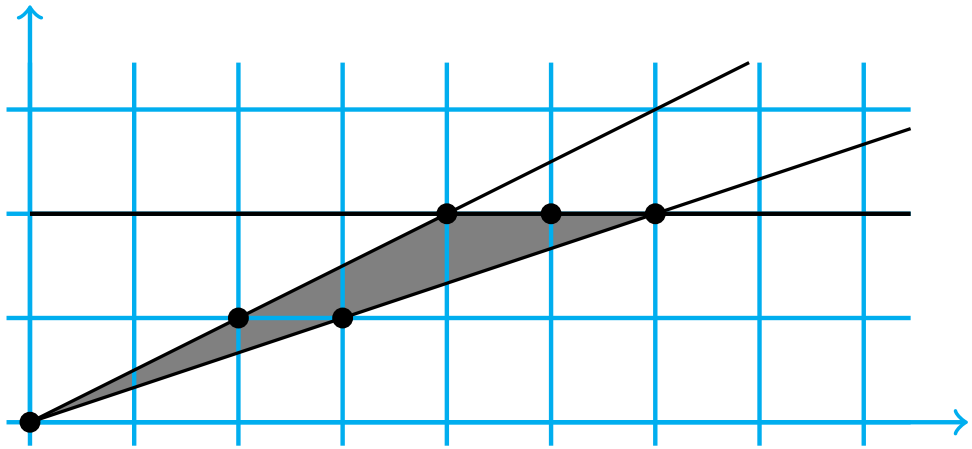
the first-order theory of natural numbers with addition and order.

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

It is a language in which we can:

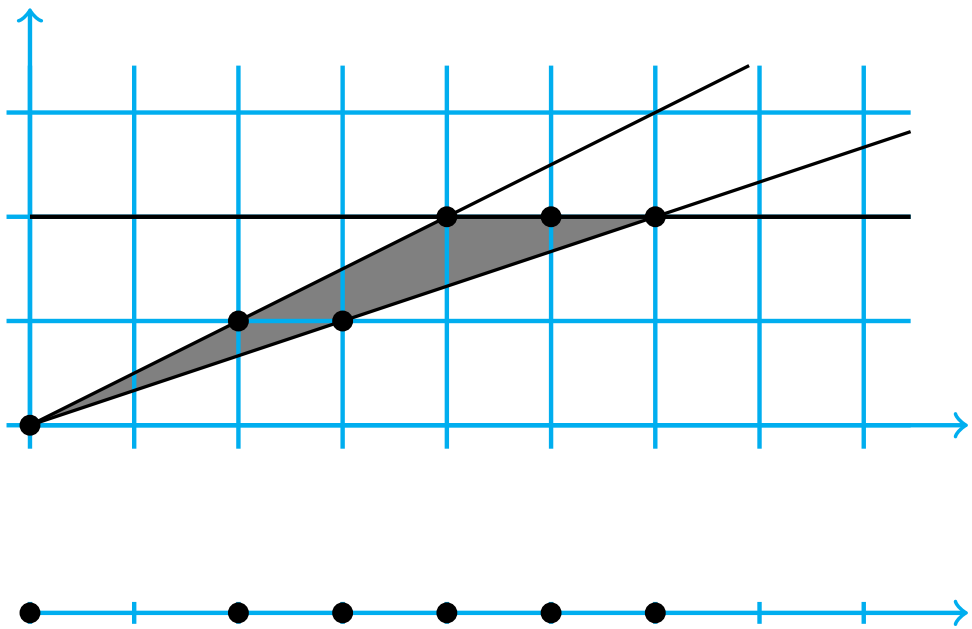
- talk about natural numbers (referred to as variables x, y, ...)
- assert linear inequalities involving these numbers
- form Boolean combinations ($\wedge$, $\vee$, $\neg$) of assertions
- quantify over (all) natural numbers ($\exists$, $\forall$)

Presburger arithmetic (Linear integer arithmetic):



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

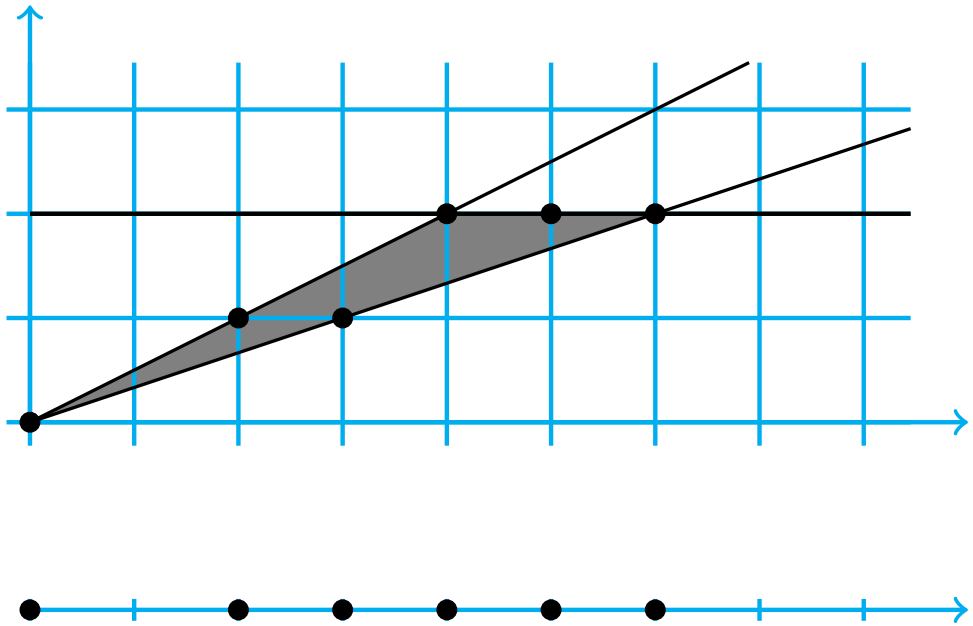
Presburger arithmetic (Linear integer arithmetic):

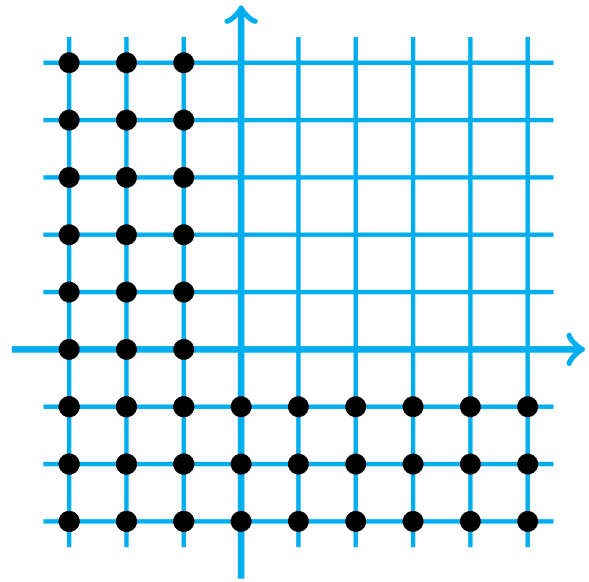


$$\exists y \ (x \leqslant 3y) \wedge (2y \leqslant x) \wedge (y \leqslant 2)$$

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

Presburger arithmetic (Linear integer arithmetic):





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

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

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

Presburger arithmetic (Linear integer 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
- form Boolean combinations ($\wedge$, $\vee$, $\neg$) of assertions
- quantify over (all) natural numbers ($\exists$, $\forall$)

Presburger arithmetic (Linear integer arithmetic):

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

Mojżesz Presburger

Example: 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?

Example: 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?

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

Example: 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?

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

Example: 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?

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

Example: 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?

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

Example: 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.

Detailed example: a proof of Parikh's theorem

Commutative/Parikh mapping:

Given $\Sigma = \{a_1, \ldots, a_r\}$ and $\mathcal{L} \subseteq \Sigma^*$,

$$\psi(\mathcal{L}) = \Big\{ (m_1, \ldots, m_r) : \exists\, w \in \mathcal{L} \text{ with exactly } m_i \text{ occurrences of } a_i \Big\} \subseteq \mathbb{N}^r.$$

For instance:

- $\psi\big(\{\, a\,a\,b\,b\,b\,b\,a \,\}\big) = \{(3,4)\}$
- $\psi\big(\{\, a^m b^m : m \geqslant 0 \,\}\big) = \psi\big((ab)^*\big) = \{(m,m) : m \geqslant 0\}$

Theorem (Parikh, 1961).

For every context-free language $\mathcal{L}$ there is a regular language $\mathcal{R}$ such that $\psi(\mathcal{L}) = \psi(\mathcal{R})$.

Presburger description via balance and connectivity

[Verma, Seidl, Schwentick 2005]

Goal:

Write a formula in Presburger arithmetic that defines $\psi(\mathcal{L})$.

Idea:

Look at dependencies in the grammar. Every nonterminal sends flow to other nonterminals when productions are fired.

Presburger description via balance and connectivity

[Verma, Seidl, Schwentick 2005]

Goal:

Write a formula in Presburger arithmetic that defines $\psi(\mathcal{L})$.

Idea:

Look at dependencies in the grammar. Every nonterminal sends flow to other nonterminals when productions are fired.

Kirchhoff's first law for electrical circuits (1845):

> *The current entering any junction*
> *is equal to the current leaving that junction.*

Conservation law for nonterminals in a context-free derivation (communication-free Petri nets: [Esparza (1997)]):

> *Any non-axiom nonterminal A appears on lhs of fired productions as many times as it appears on the rhs.*

Balance equation

Adjust the conservation law for context-free grammars
to account (1) for the axiom and (2) for terminal symbols.

The obtained balance equation (a system of linear equalities) is then a necessary condition on the vector of:

- $y_{A \to BC}$: the number of times each production is applied in a derivation, and
- x_A and x_a: the number of times each A and a appears on rhs of fired productions.

Balance equation is also sufficient

Lemma (Esparza 1997, reformulated).
Suppose the sets $\mathcal{M} = \{A \in \mathcal{N} : x_A > 0\}$ and
$\mathcal{R} = \{(A \to BC) \in \mathcal{P} : y_{A \to BC} > 0\}$ form a connected graph.

Then the balance equation is sufficient: If it is satisfied, then there exists a derivation that respects the values of x_A and $y_{A \to BC}$.

Corollary: A small Presburger formula (special case)

Let $\mathcal{L}$ be generated with a reduced context-free grammar. Suppose

$$\begin{aligned}\boldsymbol{x}' &= (x_{A_1}, \ldots, x_{A_n}),\\ \boldsymbol{x}'' &= (x_{a_1}, \ldots, x_{a_k}),\\ \boldsymbol{y} &= (y_{A\to BC}, \ldots).\end{aligned}$$

The existential Presburger formula

$$\Phi^+(\boldsymbol{x}'') := \exists \boldsymbol{x}', \boldsymbol{x}'', \boldsymbol{y}\text{: balance equation holds for } (\boldsymbol{x}', \boldsymbol{x}'', \boldsymbol{y}) \wedge \bigwedge_{A\in\mathcal{N}} x_A > 0 \wedge \bigwedge_{(A\to BC)\in\mathcal{P}} y_{A\to BC} > 0$$

defines the set $\psi(\mathcal{L}^{(+)})$ where $\mathcal{L}^{(+)}$ consists of all words from $\mathcal{L}$ that have derivations in which all nonterminals appear and all productions are applied.

Corollary: A big Presburger formula

For $\mathcal{M} \subseteq \mathcal{N}$ and $\mathcal{R} \subseteq \mathcal{P}$, let $\Phi^{+}_{\mathcal{M},\mathcal{R}}(\boldsymbol{x}'')$ be the formula $\Phi^{+}(\boldsymbol{x}'')$ written for the grammar where nonterminals $\mathcal{N} \setminus \mathcal{M}$ and productions $\mathcal{P} \setminus \mathcal{R}$ are removed.

Then the existential Presburger formula

$$\Phi(\boldsymbol{x}'') := \bigvee_{\substack{\mathcal{M}\subseteq\mathcal{N},\\ \mathcal{R}\subseteq\mathcal{P}\\ \text{connected}}} \Phi^{+}_{\mathcal{M},\mathcal{R}}(\boldsymbol{x}'') \wedge \bigwedge_{A\in\mathcal{N}\setminus\mathcal{M}} x_A = 0 \wedge \bigwedge_{(A\to BC)\in\mathcal{P}\setminus\mathcal{R}} y_{A\to BC} = 0$$

defines the set $\psi(\mathcal{L})$.

Ensuring connectivity

[Verma, Seidl, Schwentick 2005]
[Seidl, Schwentick, Muscholl, Habermehl 2004]

Claim

The sets $\mathcal{M} = \{A \in \mathcal{N} : x_A > 0\}$ and $\mathcal{R} = \{(A \to BC) \in \mathcal{P} : y_{A \to BC} > 0\}$ form a connected graph if and only if the following formula holds:

$$\exists\, z_{A_1}, \ldots, z_{A_n} : z_S = 1 \wedge \bigwedge_{A \in \mathcal{N}} (z_A > 0 \leftrightarrow x_A > 0) \wedge \bigwedge_{\substack{A \in \mathcal{N} \\ A \neq S}} \big(z_A > 0 \to \bigvee_{\substack{B \to AC \text{ or} \\ B \to CA}} z_A = z_B + 1 \wedge y_{B \to AC \text{ or } B \to CA} > 0 \wedge z_B > 0\big)$$

Proof:

Connectivity = Existence of spanning tree.

Wrap-up: Parikh's theorem via Presburger arithmetic

[Verma, Seidl, Schwentick 2005]

Theorem

For every context-free language $\mathcal{L}$ there exists
a small existential Presburger formula that defines the set $\psi(\mathcal{L})$.

Applications:

- Analysis of parametric one-counter automata
  [Haase et al. 2009]
- Verification of multi-threaded programs
  [Esparza and Ganty 2011]
  [Hague and Lin 2011]
- Integer vector addition systems with states and resets
  [Haase et al. 2016]

Decision problem for Presburger arithmetic

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

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

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?

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

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

$$\forall x\ \exists y\ ((y > x) \wedge P(y) \wedge P(y + 2)) \qquad \text{(not in P.a.)}$$

Decision problem for Presburger arithmetic

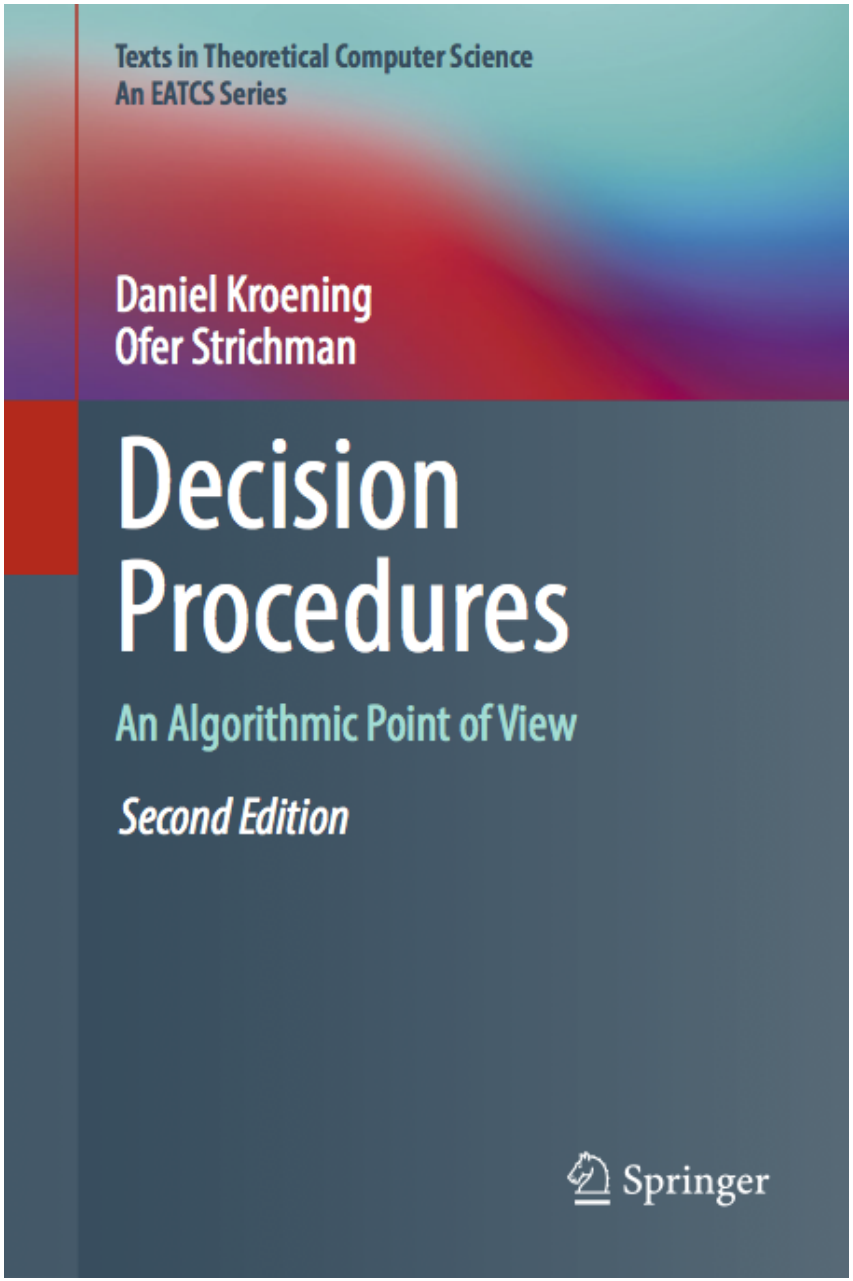
Input: sentence φ in Presburger arithmetic
Output: is φ true or false?

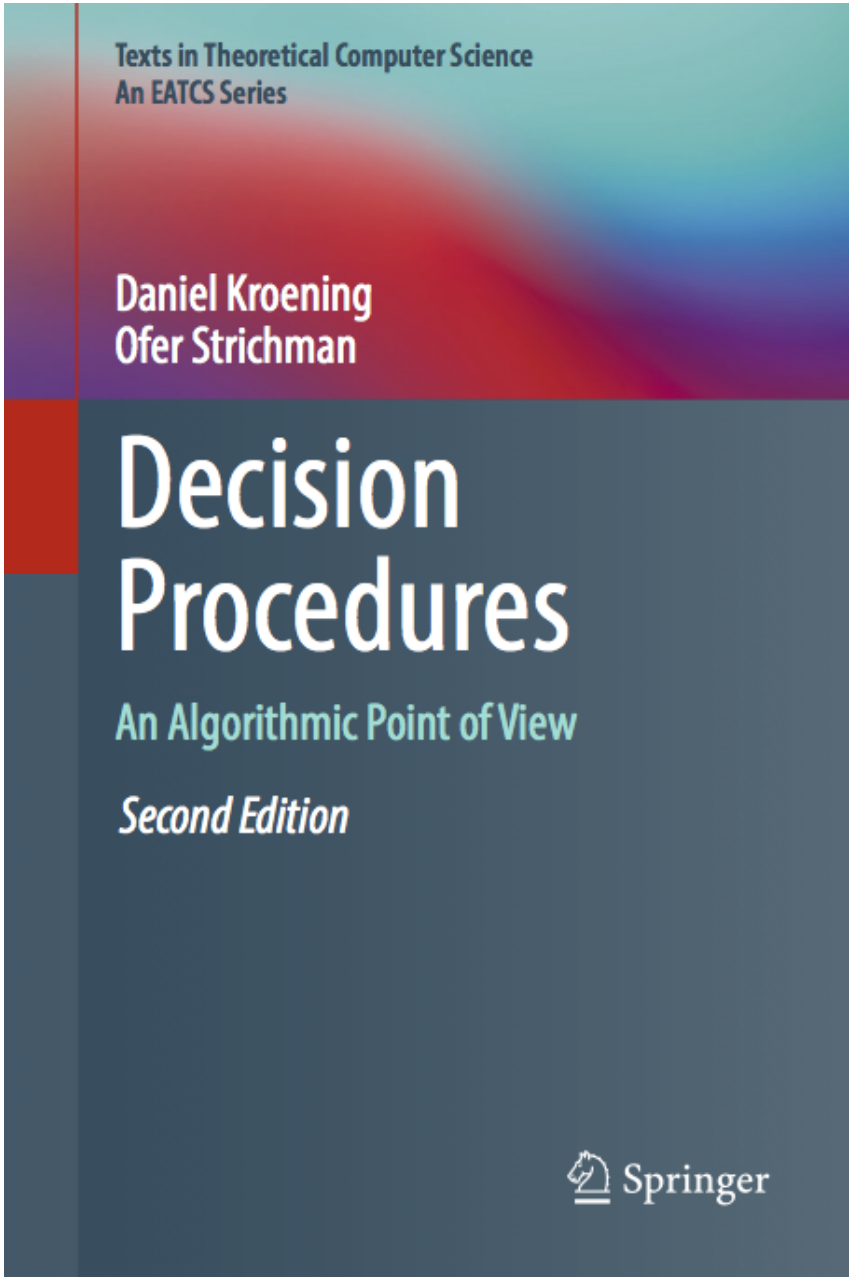
$$\forall x\ \exists y\ \exists z\ ((x = 2y) \vee (x = 2z + 1)) \qquad \text{(in P.a.)}$$

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

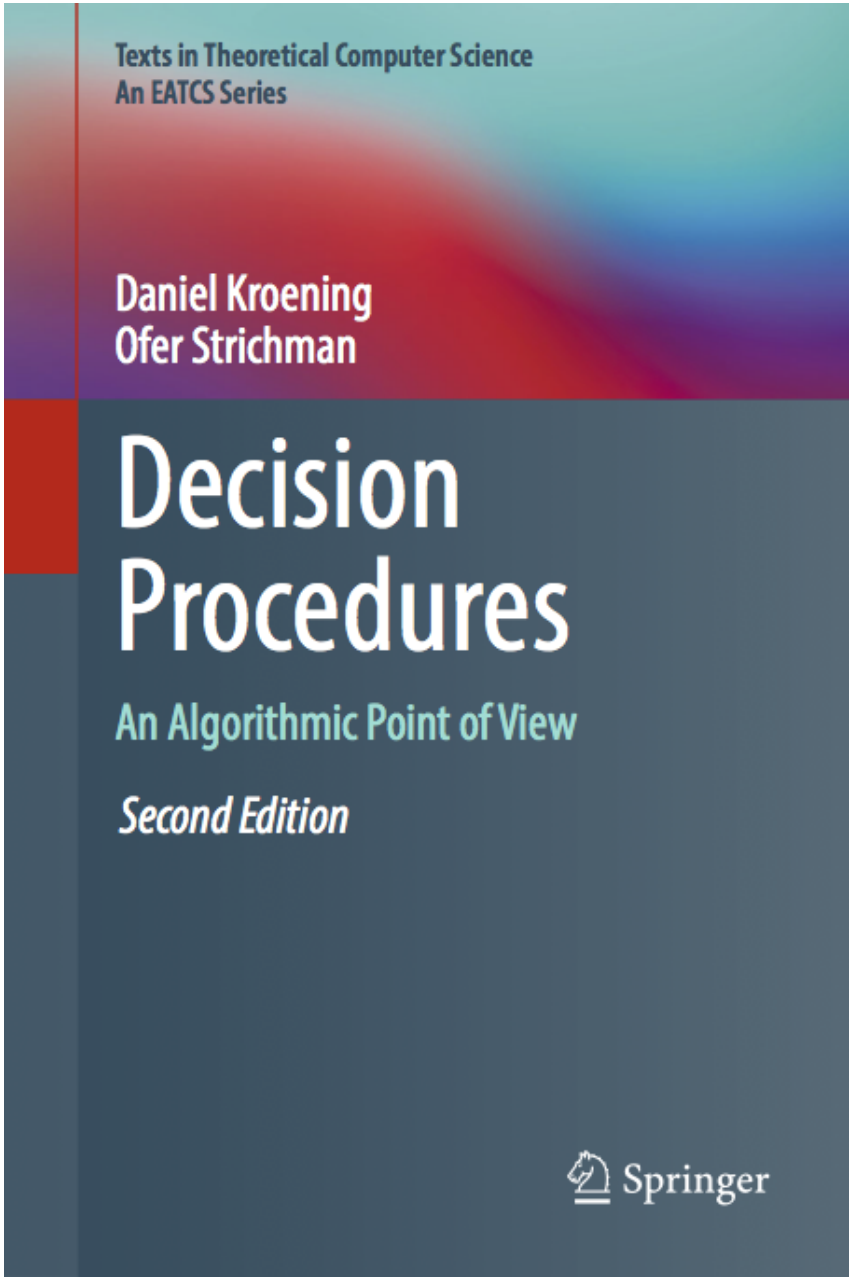
$$\forall x\ \exists y\ ((y > x) \wedge P(y) \wedge P(y + 2)) \qquad \text{(not in P.a.)}$$

$$\forall x\ \forall y\ [(y \mid x) \wedge (y \mid x + 1)] \to y \leqslant 1 \qquad \text{(not in P.a.)}$$


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. Introduction to Presburger arithmetic

2. Three views on Presburger arithmetic

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

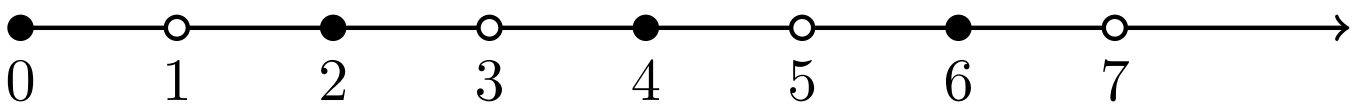
3. Computational complexity of decision problems

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

Periodic and ultimately periodic sets of natural numbers

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

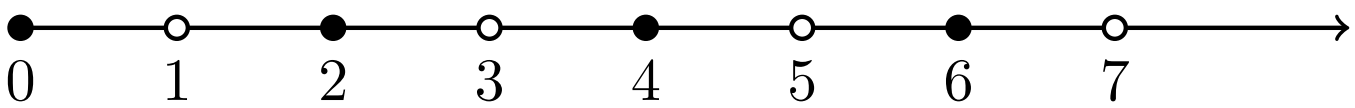
S is periodic if there exists a $p > 0$ such that,
for all $x \in \mathbb{N}$: $\quad x \in S$ iff $x + p \in S$.



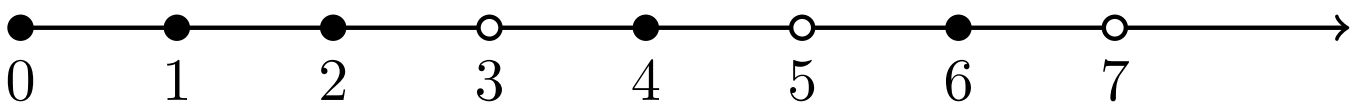
Periodic and ultimately periodic sets of natural numbers

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 $x + p \in S$.



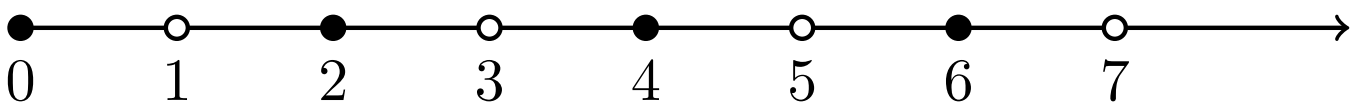
S is ultimately periodic if there exist N and $p > 0$ such that,
for all $x \geqslant N$: $\quad x \in S$ iff $x + p \in S$.



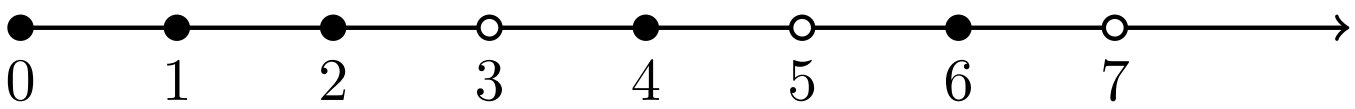
Periodic and ultimately periodic sets of natural numbers

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 $x + p \in S$.

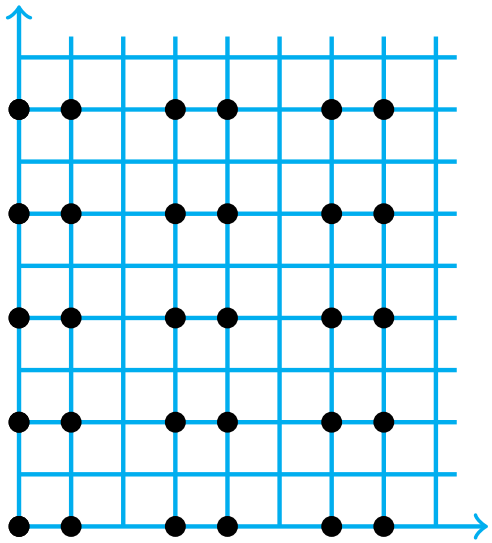


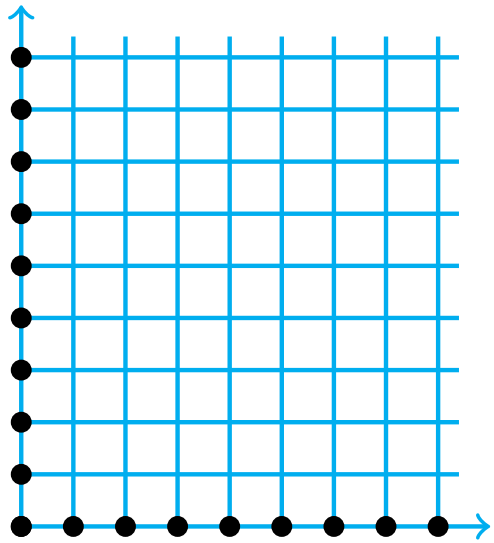
S is ultimately periodic if there exist N and $p > 0$ such that,
for all $x \geqslant N$: $\quad x \in S$ iff $x + p \in S$.

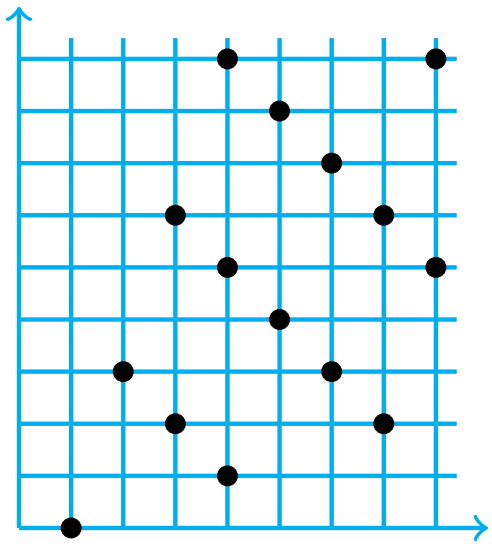


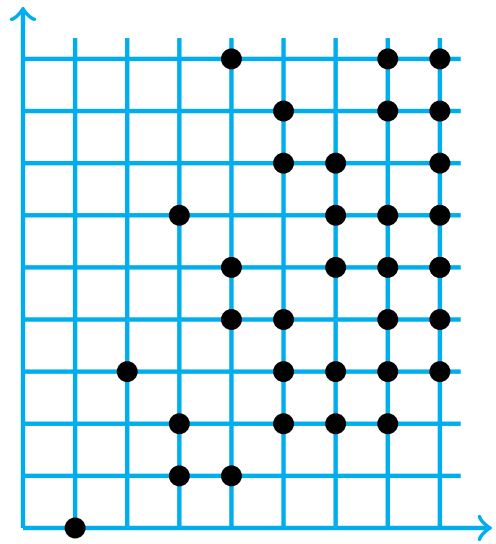
Ultimately periodic = finite union of arithmetic progressions.

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

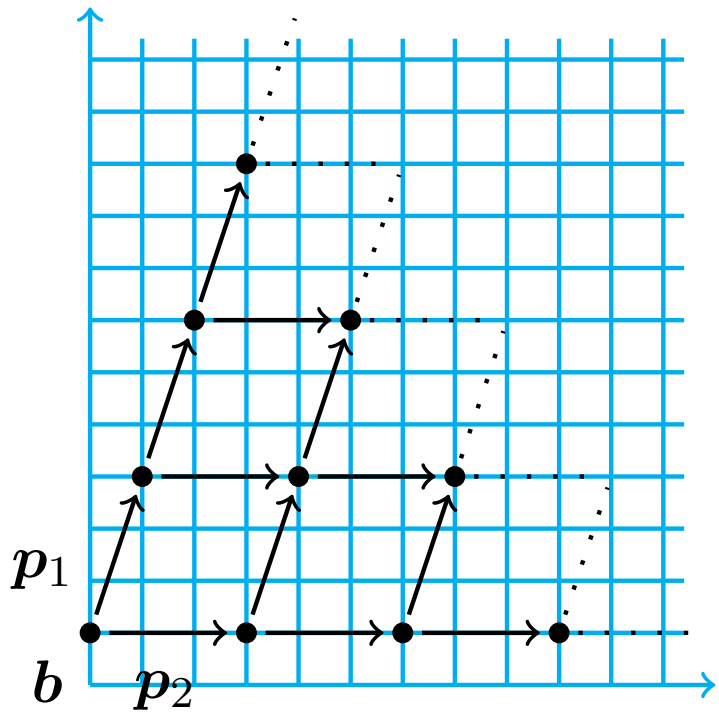
Linear and semi-linear sets

[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): $|P| < \infty$

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

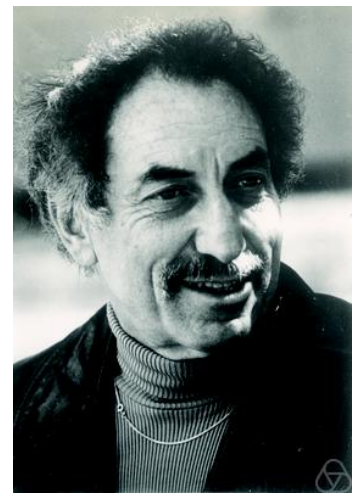
Semi-linear set: $|I|, |P_i| < \infty$

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

Theorem 1 (Ginsburg and Spanier, 1964).

Semi-linear sets = sets definable in Presburger arithmetic.

Seymour Ginsburg

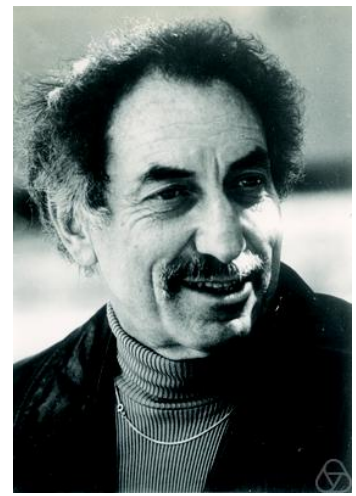


Edwin H. Spanier

Left: ACM SIGMOD Record, 34(1), 03/2005. Right: Mathematisches Forschungsinstitut Oberwolfach gGmbH / CC-BY-SA 2.0 DE

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**

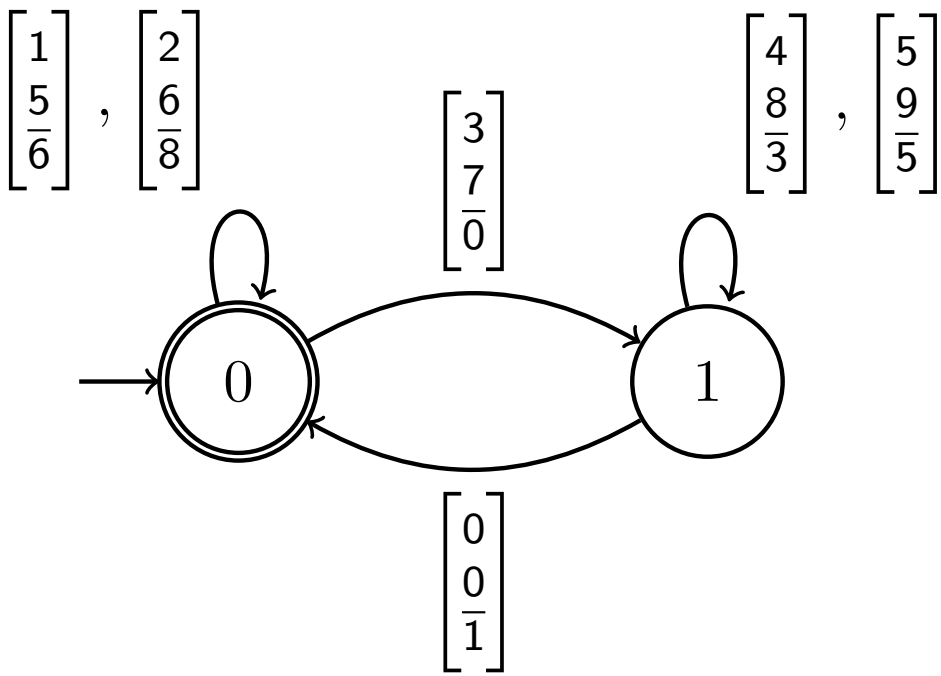
Finite automaton can read triplets of digits and check the equality $x + y = z$:

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

Image source: http://static.classora.com/files/uploads/images/entries/581092/main.jpg (J.R. Büchi) CC BY-SA 4.0, Wikipedia

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

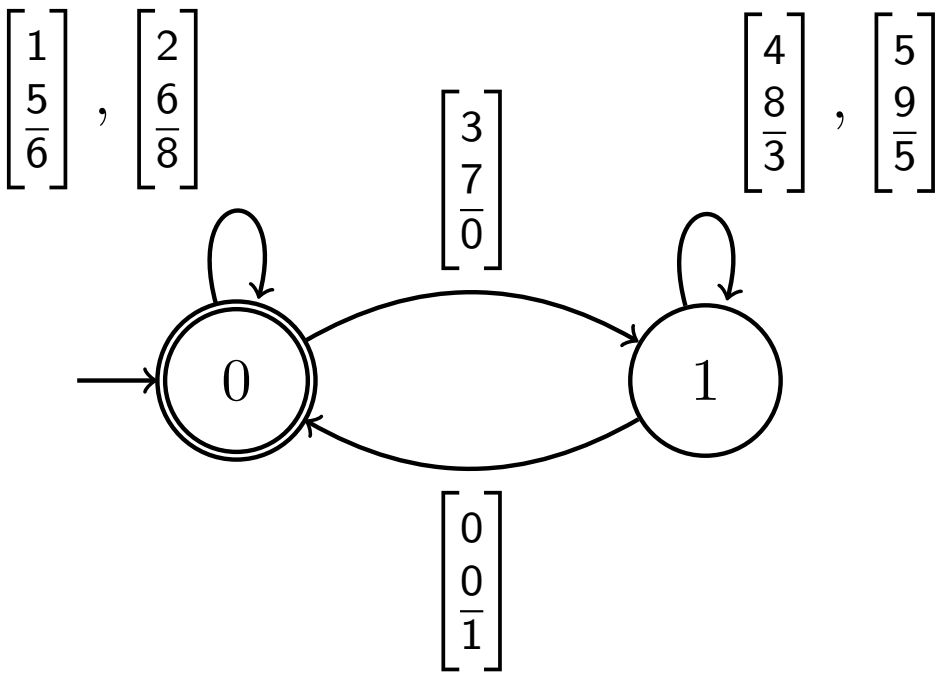
$$\begin{array}{rr} x: & 54\,321 \\ + & \\ y: & 98\,765 \\ \hline z: & 153\,086 \end{array}$$



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

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

J. Richard Büchi



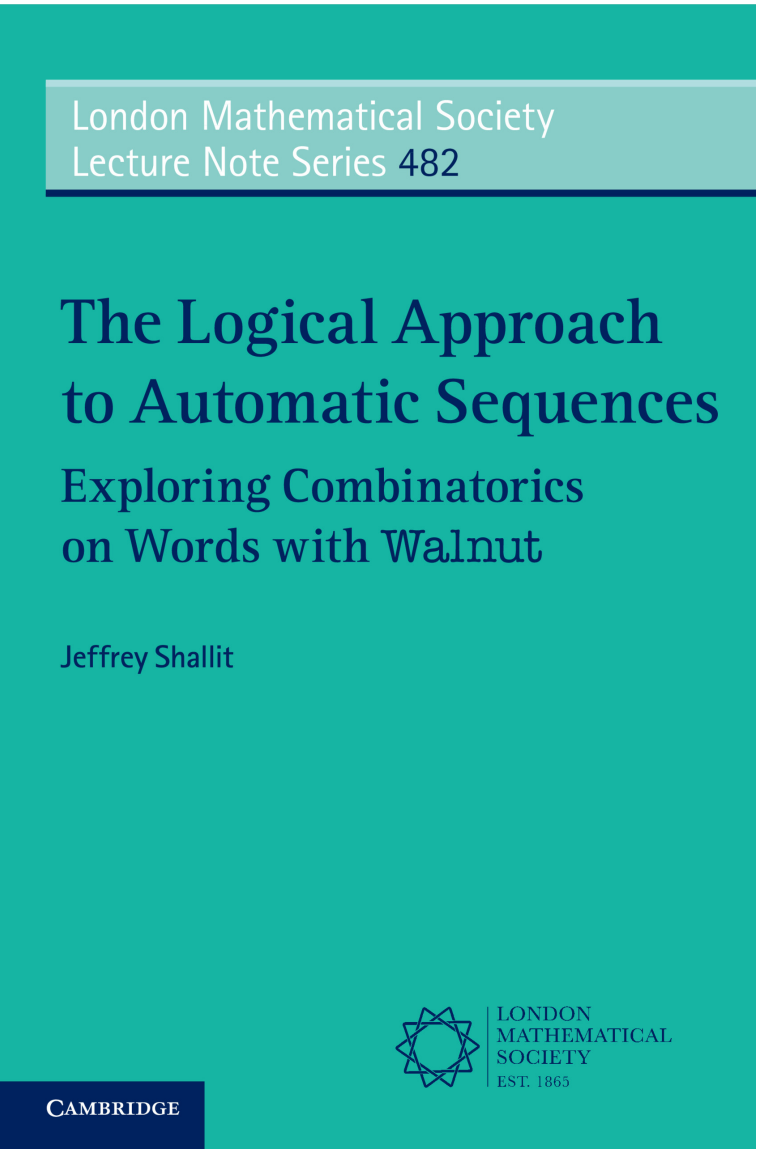
[1960]

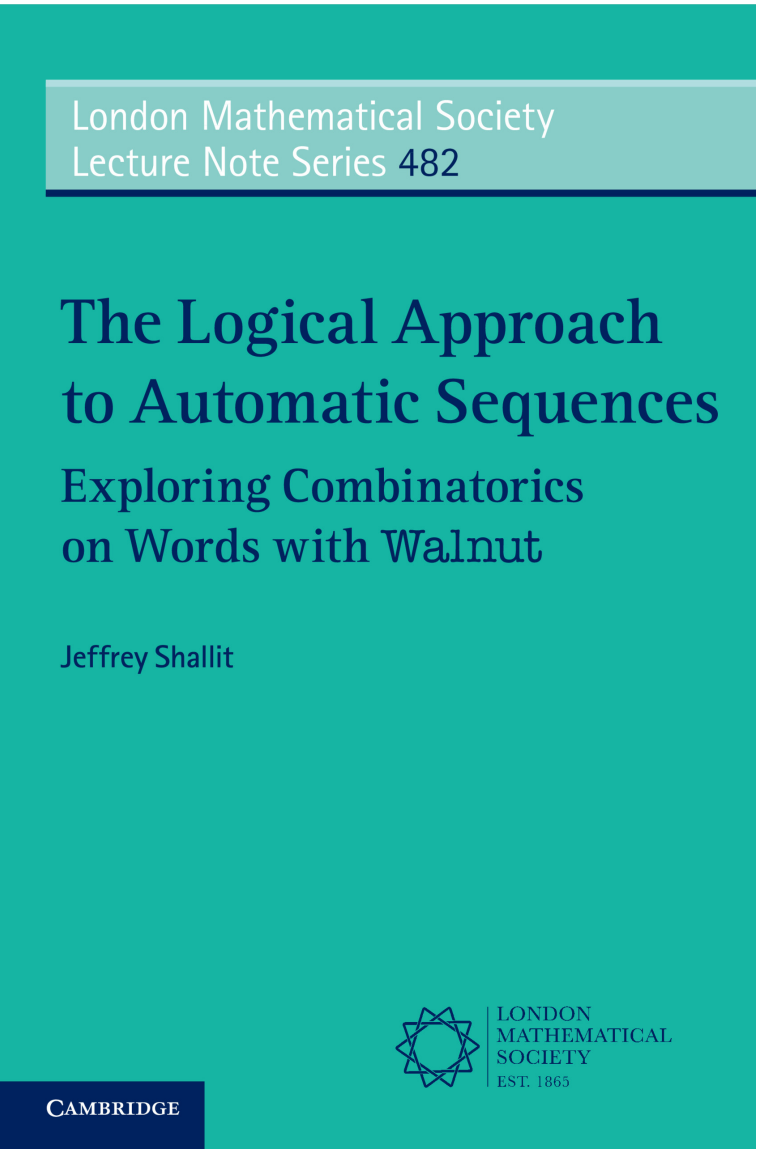
For $d \geqslant 1$, a set $S \subseteq \mathbb{N}^d$ is **2-automatic** (or: **2-recognizable**) if there is a deterministic finite automaton (DFA) that accepts the language

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

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.,* $0.1024545454545\ldots$ *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):

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

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:

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

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

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:

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

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

$$\begin{aligned}&(2x + z + 3 \leqslant 6x - 11) \wedge (6x - 11 \equiv 0 \mod 2) \vee \\ &(2x + z + 3 \leqslant 6x - 12) \wedge (6x - 11 \equiv 1 \mod 2)\end{aligned}$$

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]

1. Introduction to Presburger arithmetic

2. Three views on Presburger arithmetic

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

3. Computational complexity of decision problems

Decision procedures based on the three views

View	Geometry	Automata theory	Symbolic computation (quantifier elimination)
Repr.	Semi-linear sets	Finite automata	Logical formulas
$\neg$	expensive	NFA $\rightsquigarrow$ DFA	CNF $\rightsquigarrow$ DNF
$\vee$	trivial	trivial	trivial
$\wedge$	OK	easy	easy
$\exists$	trivial	easy	expensive

Decision procedures based on the three views

View	Geometry	Automata theory	Symbolic computation (quantifier elimination)
Repr.	Semi-linear sets	Finite automata	Logical formulas
$\neg$	expensive	NFA $\rightsquigarrow$ DFA	CNF $\rightsquigarrow$ DNF
$\vee$	trivial	trivial	trivial
$\wedge$	OK	easy	easy
$\exists$	trivial	easy	expensive

Decision procedures based on the three views

View	Geometry	Automata theory	Symbolic computation (quantifier elimination)
Repr.	Semi-linear sets	Finite automata	Logical formulas
$\neg$	expensive	NFA $\rightsquigarrow$ DFA	CNF $\rightsquigarrow$ DNF
$\vee$	trivial	trivial	trivial
$\wedge$	OK	easy	easy
$\exists$	trivial	easy	expensive

n-fold exponential: tower $2^{\cdot^{\cdot^{\cdot^{2}}}}$, where the height of the tower grows as n (the size of the input formula)

Elementary function $f : \mathbb{N} \to \mathbb{N}$: $f(n) \leqslant 2^{\cdot^{\cdot^{\cdot^{2^n}}}}$, with tower of any fixed height.

Complexity class ELEMENTARY: elementary running time.

Elementary function $f : \mathbb{N} \to \mathbb{N}$: $f(n) \leqslant 2^{\cdot^{\cdot^{\cdot^{2^n}}}}$, with tower of any fixed height.

Complexity class ELEMENTARY: elementary running time.

> *"The name was coined by László Kalmár, in the context of recursive functions and undecidability; most problems in it are far from elementary."*
>
> — Wikipedia

Elementary function $f : \mathbb{N} \to \mathbb{N}$: $f(n) \leqslant 2^{\cdot^{\cdot^{\cdot^{2^n}}}}$, with tower of any fixed height.

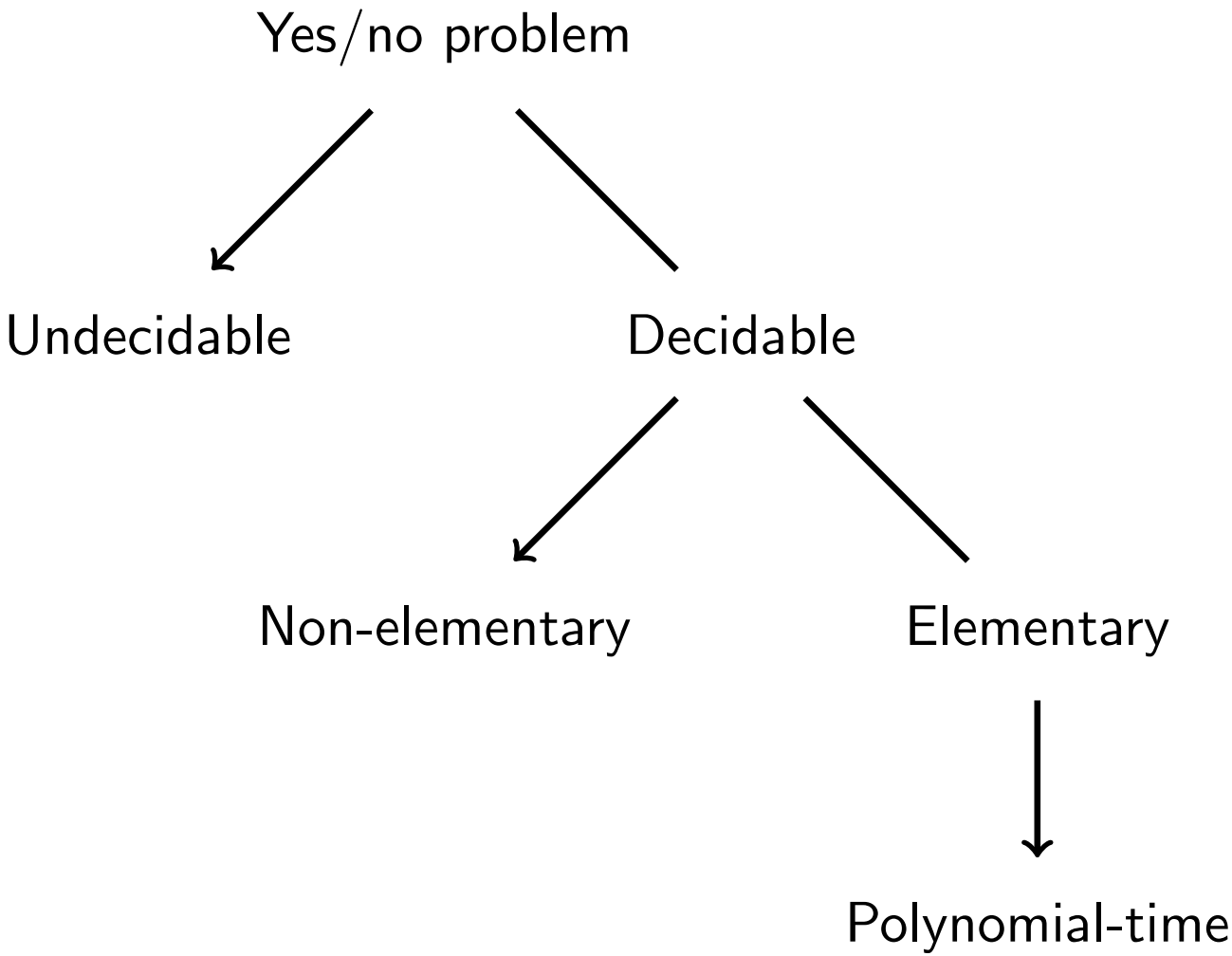
Complexity class ELEMENTARY: elementary running time.

> *"The name was coined by László Kalmár, in the context of recursive functions and undecidability; most problems in it are far from elementary."*
>
> — Wikipedia

The class $\mathcal{E}_2' = \mathcal{L}_2$ is often referred to as the class of *elementary functions*. We claim that all real problems are in fact elementary, that is, computable by loop(2) programs. To see this, note that by Theorem 10.6 a function is

— Walter S. Brainerd, Lawrence H. Landweber, *Theory of Computation* (1974)

A view from Voyager spacecraft



Elementary Bounds for Presburger Arithmetic

Derek C. Oppen
University of Toronto
Toronto, Canada

We consider the first-order theory whose language has as nonlogical symbols the constant symbols 0 and 1 , the binary relation symbols = and < , the unary function symbol - and the binary function symbol + .

This theory of integers under addition is commonly called the 'Presburger Arithmetic' and is known to be decidable for truth [Presburger (1929), Hilbert and Bernays (1968)]. We prove here that there exists a decision procedure for this theory, involving quantifier elimination, for which there is a superexponential upper bound on the size of formula produced when all variables have been eliminated. Thus there is a superexponential bound on the storage and time required to decide the truth of any formula, and the decision problem is elementary-recursive in the sense of Kalmar. This contrasts with Meyer's result [Meyer (1972)] that there is no elementary-recursive decision procedure for the weak monadic second-order theory of successor.

The bound on storage and deterministic time required for a sentence of length n is found to be $2^{2^{2^{pn\log n}}}$ for some constant $p > 1$. This upper theory whose language includes all the numerals rather than just the constants 0 and 1 .

The Decision Procedure for Truth

In the following, the terms 1+...+1 and t+...+t (1 and t each repeated k times) will be written as k and kt respectively, and the term $t_1 + (-t_2)$ as $t_1 - t_2$. We admit the other usual arithmetic binary relation symbols $\neq$, $\leq$, $>$ and $\geq$, and also the relation for divisibility by a positive integer, $k|t$, where k is a positive integer and t a term. Similarly we admit the relation $\nmid$ for indivisibility by a positive integer. Obviously, all these relations are definable in terms of + and < alone.

Consider a formula of the form $\exists xF(x)$ where $F(x)$ is quantifier-free but need not otherwise be in any special form. Assume like terms have been collected. We wish to construct an equivalent formula not containing x . The following algorithm for doing so is due to Cooper (1972).

Step 1. Eliminate logical negations by first driving them in as far as possible

Decision procedures based on the three views

View	Geometry	Automata theory	Symbolic computation (quantifier elimination)
Repr.	Semi-linear sets	Finite automata	Logical formulas
$\neg$	expensive	NFA $\rightsquigarrow$ DFA	CNF $\rightsquigarrow$ DNF
$\vee$	trivial	trivial	trivial
$\wedge$	OK	easy	easy
$\exists$	trivial	easy	expensive

n-fold exponential: tower $2^{\cdot^{\cdot^{\cdot^{2}}}}$, where the height of the tower grows as n (the size of the input formula)

Decision procedures based on the three views

View	Geometry	Automata theory	Symbolic computation (quantifier elimination)
Repr.	Semi-linear sets	Finite automata	Logical formulas
$\neg$	expensive	NFA $\rightsquigarrow$ DFA	CNF $\rightsquigarrow$ DNF
$\vee$	trivial	trivial	trivial
$\wedge$	OK	easy	easy
$\exists$	trivial	easy	expensive

n-fold exponential: tower $2^{\cdot^{\cdot^{\cdot^{2}}}}$, where the height of the tower grows as n (the size of the input formula)

All three approaches provide elementary decision procedures.

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 $\mathsf{STA}(*, 2^{2^{n^{O(1)}}}, n)$ [Berman 1980]

A clever use of quantifiers

Michael J. Fischer

Michael O. Rabin

Theorem (Fischer and Rabin, 1974).
There is a sequence of formulae $F_n(x, y)$ of size $O(n)$ such that

$$F_n(x, y) \text{ holds} \iff x = 2^{2^n} \cdot y.$$

Proof [Fischer and Rabin, 1974]

$F_0(x, y)$: $x = 2y$ $(2 = 2^{2^0})$

$F_{n+1}(x, y)$: $\exists z\ F_n(x, z) \wedge F_n(z, y)$ (for $x = 2^{2^n} \cdot y$)

Proof [Fischer and Rabin, 1974]

$F_0(x, y)$: $\quad x = 2y \qquad (2 = 2^{2^0})$

$F_{n+1}(x, y)$: $\quad \exists z\ F_n(x, z) \wedge F_n(z, y) \qquad (\text{for } x = 2^{2^n} \cdot y)$

$$\equiv \exists z\ \forall u\ \forall v$$
$$((u = x \wedge v = z) \vee (u = z \wedge v = y)) \rightarrow F_n(u, v) \quad \blacksquare$$

Proof [Fischer and Rabin, 1974]

$F_0(x, y)$: $\quad x = 2y \qquad (2 = 2^{2^0})$

$F_{n+1}(x, y)$: $\quad \exists z\ F_n(x, z) \wedge F_n(z, y) \qquad$ (for $x = 2^{2^n} \cdot y$)

$$\equiv \exists z\ \forall u\ \forall v$$
$$((u = x \wedge v = z) \vee (u = z \wedge v = y)) \rightarrow F_n(u, v) \quad \blacksquare$$

Going further:

- Exploit the use of integers (rather than reals): 3-exp numbers

Proof [Fischer and Rabin, 1974]

$F_0(x, y)$: $\quad x = 2y \qquad (2 = 2^{2^0})$

$F_{n+1}(x, y)$: $\quad \exists z\ F_n(x, z) \wedge F_n(z, y) \qquad$ (for $x = 2^{2^n} \cdot y$)

$$\equiv \exists z\ \forall u\ \forall v$$
$$((u = x \wedge v = z) \vee (u = z \wedge v = y)) \rightarrow F_n(u, v) \quad \blacksquare$$

Going further:

- Exploit the use of integers (rather than reals): 3-exp numbers
- Deciding Presburger arithmetic requires nondet. 2-exp time
  [Fischer and Rabin 1974]

Proof [Fischer and Rabin, 1974]

$F_0(x, y)$: $\quad x = 2y \qquad (2 = 2^{2^0})$

$F_{n+1}(x, y)$: $\quad \exists z\ F_n(x, z) \wedge F_n(z, y) \qquad$ (for $x = 2^{2^n} \cdot y$)

$$\equiv \exists z\ \forall u\ \forall v$$
$$((u = x \wedge v = z) \vee (u = z \wedge v = y)) \rightarrow F_n(u, v) \quad \blacksquare$$

Going further:

- Exploit the use of integers (rather than reals): 3-exp numbers
- 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]

Restrictions and extensions of Presburger arithmetic

A restriction: **existential** Presburger arithmetic

[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 $\mathbb{Z}^n$?

A restriction: **existential** Presburger arithmetic

[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.

A restriction: **existential** Presburger arithmetic

[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.

A restriction: **existential** Presburger arithmetic

[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.**

Extensions: Adding nonlinearity to Presburger arithmetic

Extensions: Adding nonlinearity to Presburger arithmetic

Gödel, Church, Turing (1930s):

multiplication function $x, y \mapsto x \cdot y$ — undecidable

Extensions: Adding nonlinearity to Presburger arithmetic

Gödel, Church, Turing (1930s):
multiplication function $x, y \mapsto x \cdot y$ — undecidable

Büchi (1960): "x is the largest power of 2 that divides y" — TOWER

Semenov (1980, 1984): power predicate $2^{\mathbb{N}}(\cdot)$ — decidable

Extensions: Adding nonlinearity to Presburger arithmetic

Gödel, Church, Turing (1930s):
multiplication function $x, y \mapsto x \cdot y$ — undecidable

Büchi (1960): "x is the largest power of 2 that divides y" — TOWER

Semenov (1980, 1984): power predicate $2^{\mathbb{N}}(\cdot)$ — decidable
function $x \mapsto 2^x$ — decidable

Cherlin and Point (1986, 2000): function $x \mapsto 2^x$ — TOWER

Cobham (1969), Hieronymi and Schulz (2022):

non-Presburger predicates definable in $2^{\mathbb{N}}(\cdot)$ and $3^{\mathbb{N}}(\cdot)$ — undecidable

Integer linear-exponential programming

System of

- Linear inequalities
- Constraints of the form $y = 2^x$ and $y = (z \bmod 2^x)$.

Integer linear-exponential programming

System of

- Linear inequalities
- Constraints of the form $y = 2^x$ and $y = (z \bmod 2^x)$.

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

Decidability, exponential space upper bound: [Draghici et al. 2024]

Integer linear-exponential programming

With $y = 2^x$ and $y = (z \bmod 2^x)$, one can express:

- bit string u has length v: $\quad 2^{v-1} \leqslant u < 2^v$
- tower of exponentials: $\quad a = 2,\ b = 2^a,\ c = 2^b,\ d = 2^c,\ \ldots$
- the ith rightmost bit of string u is 1:
  $\exists j : j = i + 1 \wedge \big((u \bmod 2^j) - (u \bmod 2^i) \geqslant 1\big)$

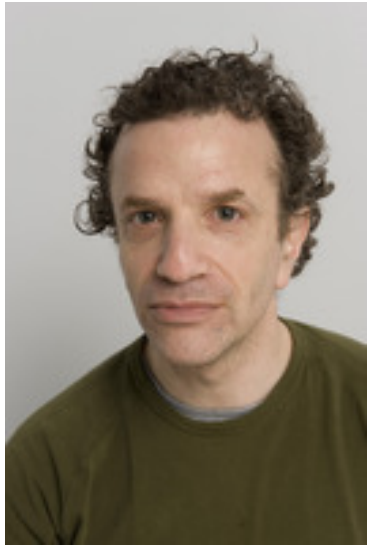
Integer linear-exponential programming

With $y = 2^x$ and $y = (z \bmod 2^x)$, one can express:

- bit string u has length v: $2^{v-1} \leqslant u < 2^v$
- tower of exponentials: $a = 2$, $b = 2^a$, $c = 2^b$, $d = 2^c$, ...
- the ith rightmost bit of string u is 1:
  $\exists j : j = i + 1 \wedge \big((u \bmod 2^j) - (u \bmod 2^i) \geqslant 1\big)$

More generally, one can express:

- All languages of polynomial growth [Haase and Różycki 2021]
- Some non-regular languages (see above)
- But not $\{01, 10\}^*$ [Starchak 2024]
- A class of string constraints with the length function [Draghici et al. 2024]



Michael Benedikt
(Oxford)

Alessio Mansutti
(IMDEA Software Institute)

Christoph Haase
(Oxford)

Mikhail Starchak
(MPI-SWS)

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.

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.

Thank you!

`https://warwick.ac.uk/chdir`