Quantifier elimination for counting extensions of Presburger arithmetic

Dmitry Chistikov[1], Christoph Haase[2], **Alessio Mansutti**[2]

[1] University of Warwick, UK   [2] University of Oxford, UK

erc

ARIAT
advanced reasoning in arithmetic theories

Presburger arithmetic

The first-order theory of $\langle \mathbb{Z}, 0, 1, +, \leq \rangle$

"Every integer is either even or odd"

$$\forall x\, \exists y : x = 2y \vee x = 2y + 1$$

M. Presburger

Why Presburger arithmetic?

- Number theory is (highly) undecidable
- Presburger arithmetic is decidable [Presburger, '29]
- Wide range of applications in verification, program synthesis, compiler optimisation...
- Starting point of several algorithmic paradigms

Algorithmic paradigms for Presburger Arithmetic

Quantifier elimination [Presburger, '29]

$$\exists x : \varphi_{\mathsf{QF}}(x, \mathbf{y}) \ \equiv \ \psi_{\mathsf{QF}}(\mathbf{y})$$

QF: quantifier-free

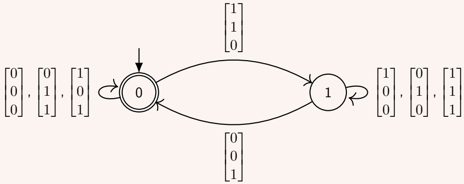
Algorithmic paradigms for Presburger Arithmetic

Quantifier elimination [Presburger, '29]

$$\exists x : \varphi_{\mathsf{QF}}(x, \mathbf{y}) \;\equiv\; \psi_{\mathsf{QF}}(\mathbf{y}) \qquad\qquad \text{QF: quantifier-free}$$

Automata techniques [Büchi, '60]

$$z = x + y \quad \rightsquigarrow \quad \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{x+y} \end{bmatrix}:$$



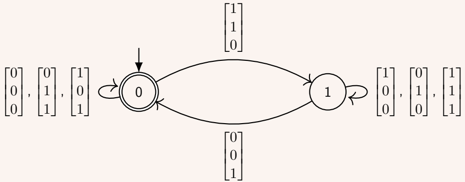
Algorithmic paradigms for Presburger Arithmetic

Quantifier elimination [Presburger, '29]

$$\exists x : \varphi_{\mathsf{QF}}(x, \mathbf{y}) \;\equiv\; \psi_{\mathsf{QF}}(\mathbf{y}) \qquad\qquad \text{QF: quantifier-free}$$

Automata techniques [Büchi, '60]

$z = x + y \quad \rightsquigarrow \quad \begin{bmatrix} \mathsf{x} \\ \mathsf{y} \\ \mathsf{x+y} \end{bmatrix}:$



Semilinear sets [Ginsburg and Spanier, '66]

$$\begin{bmatrix} \mathsf{x} \\ \mathsf{y} \\ \mathsf{x+y} \end{bmatrix}: \qquad \begin{bmatrix}0\\0\\0\end{bmatrix} + \begin{bmatrix}1\\0\\1\end{bmatrix} \cdot \mathbb{N} + \begin{bmatrix}0\\1\\1\end{bmatrix} \cdot \mathbb{N}$$

Algorithmic paradigms for Presburger Arithmetic

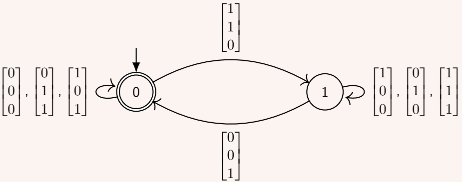
Quantifier elimination [Presburger, '29]

$$\exists x : \varphi_{\mathsf{QF}}(x, \mathbf{y}) \;\equiv\; \psi_{\mathsf{QF}}(\mathbf{y})$$

QF: quantifier-free

Automata techniques [Büchi, '60]

$z = x + y \quad \rightsquigarrow \quad \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{x+y} \end{bmatrix}:$



Semilinear sets [Ginsburg and Spanier, '66]

Kleene star

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{x+y} \end{bmatrix}: \quad \begin{bmatrix}0\\0\\0\end{bmatrix} + \begin{bmatrix}1\\0\\1\end{bmatrix} \cdot \mathbb{N} + \begin{bmatrix}0\\1\\1\end{bmatrix} \cdot \mathbb{N}$$

Algorithmic paradigms for Presburger Arithmetic

Quantifier elimination [Presburger, '29]

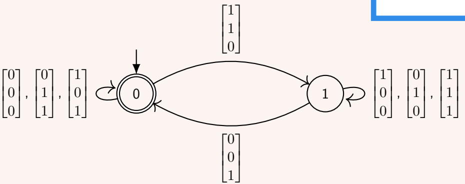
$$\exists x : \varphi_{\mathsf{QF}}(x, \mathbf{y}) \;\equiv\; \psi_{\mathsf{QF}}(\mathbf{y})$$

QF: quantifier-free

Automata techniques [Büchi, '60]

Büchi arithmetic

$z = x + y \quad \rightsquigarrow \quad \begin{bmatrix} \mathsf{x} \\ \mathsf{y} \\ \mathsf{x+y} \end{bmatrix}:$



Semilinear sets [Ginsburg and Spanier, '66]

Kleene star

$$\begin{bmatrix} \mathsf{x} \\ \mathsf{y} \\ \mathsf{x+y} \end{bmatrix}: \quad \begin{bmatrix}0\\0\\0\end{bmatrix} + \begin{bmatrix}1\\0\\1\end{bmatrix} \cdot \mathbb{N} + \begin{bmatrix}0\\1\\1\end{bmatrix} \cdot \mathbb{N}$$

Algorithmic paradigms for Presburger Arithmetic

Quantifier elimination [Presburger, '29]

Counting quantifiers

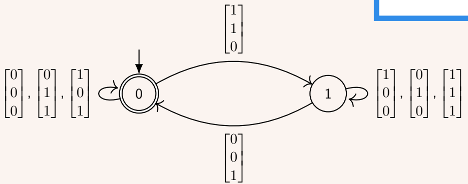
$$\exists x : \varphi_{\mathsf{QF}}(x, \mathbf{y}) \;\equiv\; \psi_{\mathsf{QF}}(\mathbf{y})$$

QF: quantifier-free

Automata techniques [Büchi, '60]

Büchi arithmetic

$z = x + y \quad \rightsquigarrow \quad \begin{bmatrix} \mathsf{x} \\ \mathsf{y} \\ \mathsf{x{+}y} \end{bmatrix}:$



Semilinear sets [Ginsburg and Spanier, '66]

Kleene star

$$\begin{bmatrix} \mathsf{x} \\ \mathsf{y} \\ \mathsf{x{+}y} \end{bmatrix}: \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \cdot \mathbb{N} + \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \cdot \mathbb{N}$$

Algorithmic paradigms for Presburger Arithmetic

Counting quantifiers

Quantifier elimination [Presburger, '29]

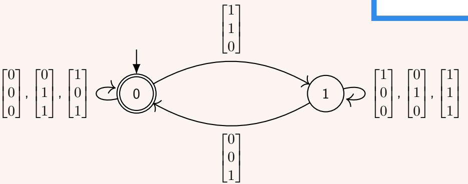
$$\exists x : \varphi_{\mathsf{QF}}(x, \mathbf{y}) \;\equiv\; \psi_{\mathsf{QF}}(\mathbf{y})$$

QF: quantifier-free

Automata techniques [Büchi, '60]

Büchi arithmetic

$z = x + y \quad \rightsquigarrow \quad \begin{bmatrix} \mathsf{x} \\ \mathsf{y} \\ \mathsf{x+y} \end{bmatrix}:$



Semilinear sets [Ginsburg and Spanier, '66]

Kleene star

$$\begin{bmatrix} \mathsf{x} \\ \mathsf{y} \\ \mathsf{x+y} \end{bmatrix}: \quad \begin{bmatrix}0\\0\\0\end{bmatrix} + \begin{bmatrix}1\\0\\1\end{bmatrix} \cdot \mathbb{N} + \begin{bmatrix}0\\1\\1\end{bmatrix} \cdot \mathbb{N}$$

Counting in Presburger arithmetic

Applications in compiler optimisation [Verdoolaege et al., Algorithmica 48, '07]**:**

"How many times is a loop executed?" $\rightsquigarrow \varphi(\mathbf{z}, x) \;\equiv\; x = \#\{\mathbf{y} \in \mathbb{Z}^d \mid A\mathbf{y} \geq B\mathbf{z} + \mathbf{c}\}$

Counting in Presburger arithmetic

Applications in compiler optimisation [Verdoolaege et al., Algorithmica 48, '07]**:**

"How many times is a loop executed?" $\rightsquigarrow \varphi(\mathbf{z}, x) \;\equiv\; x = \#\{\mathbf{y} \in \mathbb{Z}^d \mid A\mathbf{y} \geq B\mathbf{z} + \mathbf{c}\}$

Counting quantifiers:

$\exists^{=x}\mathbf{y} : \varphi(\mathbf{y}, \mathbf{z}, x)$ *"there are x many distinct vectors $\mathbf{y}$ that satisfy φ"*

Counting in Presburger arithmetic

Applications in compiler optimisation [Verdoolaege et al., Algorithmica 48, '07]**:**

"How many times is a loop executed?" $\rightsquigarrow \varphi(\mathbf{z}, x) \;\equiv\; x = \#\{\mathbf{y} \in \mathbb{Z}^d \mid A\mathbf{y} \geq B\mathbf{z} + \mathbf{c}\}$

Counting quantifiers:

$\exists^{=x}\mathbf{y} : \varphi(\mathbf{y}, \mathbf{z}, x)$ *"there are x many distinct vectors $\mathbf{y}$ that satisfy φ"*

Theorem (folklore)

Presburger arithmetic enriched with counting quantifiers is undecidable.

Counting in Presburger arithmetic

Applications in compiler optimisation [Verdoolaege et al., Algorithmica 48, '07]:

"How many times is a loop executed?" $\rightsquigarrow \varphi(\mathbf{z}, x) \;\equiv\; x = \#\{\mathbf{y} \in \mathbb{Z}^d \mid A\mathbf{y} \geq B\mathbf{z} + \mathbf{c}\}$

Counting quantifiers:

$$\exists^{=x}\mathbf{y} : \varphi(\mathbf{y}, \mathbf{z}, x) \qquad \text{“there are } x \text{ many distinct vectors } \mathbf{y} \text{ that satisfy } \varphi\text{”}$$

Theorem (folklore)

Presburger arithmetic enriched with counting quantifiers is undecidable.

Proof.

$\varphi(x, y, z) \stackrel{\text{def}}{=} \exists^{=x}(a, b) : 1 \leq a \leq y \wedge 1 \leq b \leq z$ if and only if $x = y \cdot z$. □

Counting in Presburger arithmetic

Applications in compiler optimisation [Verdoolaege et al., Algorithmica 48, '07]**:**

"How many times is a loop executed?" $\rightsquigarrow \varphi(\mathbf{z}, x) \;\equiv\; x = \#\{\mathbf{y} \in \mathbb{Z}^d \mid A\mathbf{y} \geq B\mathbf{z} + \mathbf{c}\}$

Counting quantifiers:

$\exists^{=x}\mathbf{y} : \varphi(\mathbf{y}, \mathbf{z}, x)$ *"there are x many distinct vectors $\mathbf{y}$ that satisfy φ"*

Unary **counting quantifiers:**

$\exists^{=x}y : \varphi(x, y, \mathbf{z})$ *"there are x many distinct integers y that satisfy φ"*

Theorem (Apelt, '66; Schweikardt, '05)

Presburger arithmetic enriched with unary counting quantifiers is decidable.

Examples

"The number of y satisfying φ is congruent to r modulo q"

$$\exists x : (\exists z : x - r = q \cdot z) \wedge \exists^{=x} y : \varphi$$

Examples

"The number of y satisfying φ is congruent to r modulo q"

$$\exists x : (\exists z : x - r = q \cdot z) \wedge \exists^{=x} y : \varphi$$

"The number of y satisfying φ is the product of all primes in the interval $[2, 2^{2^n}]$"

$$\exists x : \ell_n(x) \wedge \exists^{=x} y : \varphi$$

($\ell_n(x)$: x is the product of all primes in $[2, 2^{2^n}]$; formula polynomial in n)

On the complexity of Presburger arithmetic with counting quantifiers

Input: A Presburger formula φ featuring unary counting quantifiers.
Output: A solution for φ, or $\perp$ if no solution exists.

Upper bound: $\textsc{Tower}$ [Schweikardt, '05]

- quantifier elimination procedure
- $\exists^{=x} y : \varphi_{\mathsf{QF}} \rightsquigarrow \psi_{\mathsf{PA}} \rightsquigarrow \psi_{\mathsf{QF}}$
- each "$\rightsquigarrow$" costs one exponential

Lower bound: $2\mathrm{A}\textsc{Exp}_{\textsc{Poly}}$

- same as (standard) Presburger [Fischer and Rabin, '74]

On the complexity of Presburger arithmetic with counting quantifiers

Input:	A Presburger formula φ featuring unary counting quantifiers.
Output:	A solution for φ, or $\perp$ if no solution exists.

Upper bound: $\textsc{Tower}$ [Schweikardt, '05]

- quantifier elimination procedure
- $\exists^{=x} y : \varphi_{\mathsf{QF}} \rightsquigarrow \psi_{\mathsf{PA}} \rightsquigarrow \psi_{\mathsf{QF}}$
- each "$\rightsquigarrow$" costs one exponential

Lower bound: $2\mathrm{A}\textsc{Exp}_{\textsc{Poly}}$

- same as (standard) Presburger [Fischer and Rabin, '74]

***Modulo* counting quantifiers:**

$$\exists^{(q,r)} y : \varphi \qquad \equiv \qquad \exists x : x \equiv_q r \wedge \exists^{=x} y : \varphi$$

"The number of y satisfying φ is congruent to r modulo q"

On the complexity of Presburger arithmetic with counting quantifiers

Input:	A Presburger formula φ featuring unary counting quantifiers.
Output:	A solution for φ, or $\perp$ if no solution exists.

Upper bound: $\textsc{Tower}$ [Schweikardt, '05]

- quantifier elimination procedure
- $\exists^{=x} y : \varphi_{\mathsf{QF}} \rightsquigarrow \psi_{\mathsf{PA}} \rightsquigarrow \psi_{\mathsf{QF}}$
- each "$\rightsquigarrow$" costs one exponential

Lower bound: $2\mathrm{AE}\textsc{xp}_{\textsc{Poly}}$

- same as (standard) Presburger [Fischer and Rabin, '74]

Theorem 1 (Habermehl and Kuske, FOSSACS'15)

Presburger arithmetic enriched with $\exists^{(q,r)}$ is decidable in $2\textsc{ExpSpace}$.

On the complexity of Presburger arithmetic with counting quantifiers

Input:	A Presburger formula φ featuring unary counting quantifiers.
Output:	A solution for φ, or $\perp$ if no solution exists.

Upper bound: $\textsc{Tower}$ [Schweikardt, '05]

- quantifier elimination procedure
- $\exists^{=x} y : \varphi_{\mathsf{QF}} \rightsquigarrow \psi_{\mathsf{PA}} \rightsquigarrow \psi_{\mathsf{QF}}$
- each "$\rightsquigarrow$" costs one exponential

Lower bound: $2\textsc{AExp}_{\textsc{Poly}}$

- same as (standard) Presburger [Fischer and Rabin, '74]

In this paper: Revised quantifier elimination procedure for PAC ($=$ PA $+ \exists^{=x}$):

- elimination of a single $\exists^{=x}$ costs one exponential
- shows $2\textsc{ExpSpace}$ membership for a fragment of PAC that generalizes $\exists^{(q,r)}$.

Quantifier elimination: some ingredients

Input:	A set T of linear terms in d variables.
Output:	A tautology $\bigvee_{i=1}^{o} O_i$ where O_i is an ordering of terms.

Ordering: $t_1 \leq t_2 \leq \cdots \leq t_n$ with $T = \{t_1, \ldots, t_n\}$.

Quantifier elimination: some ingredients

Input:	A set T of linear terms in d variables.
Output:	A tautology $\bigvee_{i=1}^{o} O_i$ where O_i is an ordering of terms.

Ordering: $t_1 \leq t_2 \leq \cdots \leq t_n$ with $T = \{t_1, \ldots, t_n\}$.

Lemma

The number of orderings is bounded by $n^{O(d)}$.

Quantifier elimination: some ingredients

Input:	A set T of linear terms in d variables.
Output:	A tautology $\bigvee_{i=1}^{o} O_i$ where O_i is an ordering of terms.

Ordering: $t_1 \leq t_2 \leq \cdots \leq t_n$ with $T = \{t_1, \ldots, t_n\}$.

Lemma

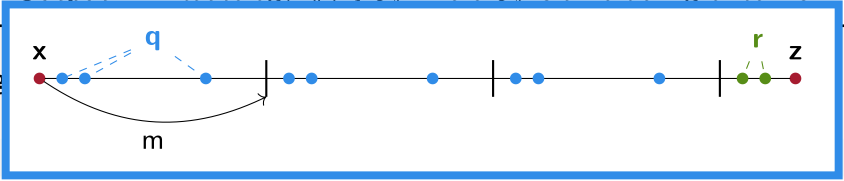
The number of orderings is bounded by $n^{O(d)}$.

Input:	A Boolean combination φ of constraints $y = r_j \mod m$.
Output:	A function $f(x, z)$ that returns $\#\{y \in [x, z] \mid y \text{ satisfies } \varphi\}$.

Note: given residue classes of x and z mod m, $f(x, z)$ is of the form $q\lfloor \frac{z-x}{m} \rfloor + r$.

Quantifier elimination: some ingredients

Input: A set T of linear terms in d variables.
Output: A tautology $\bigvee_{i=1}^{o} O_i$ where O_i is an ordering of terms.



Ordering

Lemma

The number of orderings is bounded by $n^{O(d)}$.

Input: A Boolean combination φ of constraints $y = r_j \mod m$.
Output: A function $f(x, z)$ that returns $\#\{y \in [x, z] \mid y \text{ satisfies } \varphi\}$.

Note: given residue classes of x and z mod m, $f(x, z)$ is of the form $q\lfloor \frac{z-x}{m} \rfloor + r$.

Quantifier elimination: result of the procedure

Our QE procedure translate $\exists^{=x_1} y : \varphi(y, x_1, \dots, x_d)$ into

Quantifier elimination: result of the procedure

Our QE procedure translate $\exists^{=x_1} y : \varphi(y, x_1, \dots, x_d)$ into

- $\bot$ (e.g. if there are infinitely many y satisfying φ), or

Quantifier elimination: result of the procedure

Our QE procedure translate $\exists^{=x_1} y : \varphi(y, x_1, \dots, x_d)$ into

- $\perp$ (e.g. if there are infinitely many y satisfying φ), or
- a formula of the form

$$\bigvee_{\substack{i \in [1,o] \\ f : \{x_1, \dots, x_d\} \to [0, m-1]}} t_1^{(i)} \leq \dots \leq t_n^{(i)} \wedge \Big(\bigwedge_{j=1}^{d} x_j \equiv_m f(x_j) \Big) \wedge m \cdot x_1 = \sum_{k=1}^{n-1} q_k^{(i,f)} (t_{k+1}^{(i)} - t_k^{(i)}) + r_k^{(i,f)}$$

Quantifier elimination: result of the procedure

Our QE procedure translate $\exists^{=x_1} y : \varphi(y, x_1, \dots, x_d)$ into

- $\bot$ (e.g. if there are infinitely many y satisfying φ), or
- a formula of the form

$$\bigvee_{\substack{i \in [1,o] \\ f : \{x_1, \dots, x_d\} \to [0, m-1]}} \underbrace{t_1^{(i)} \leq \dots \leq t_n^{(i)}}_{\text{ordering}} \land \left(\bigwedge_{j=1}^{d} x_j \equiv_m f(x_j) \right) \land m \cdot x_1 = \sum_{k=1}^{n-1} q_k^{(i,f)} (t_{k+1}^{(i)} - t_k^{(i)}) + r_k^{(i,f)}$$

Quantifier elimination: result of the procedure

Our QE procedure translate $\exists^{=x_1} y : \varphi(y, x_1, \dots, x_d)$ into

- $\bot$ (e.g. if there are infinitely many y satisfying φ), or
- a formula of the form

$$\bigvee_{\substack{i \in [1,o] \\ f : \{x_1,\dots,x_d\} \to [0,m-1]}} \underbrace{t_1^{(i)} \leq \dots \leq t_n^{(i)}}_{\text{ordering}} \land \underbrace{\left(\bigwedge_{j=1}^{d} x_j \equiv_m f(x_j) \right)}_{\text{residue classes}} \land m \cdot x_1 = \sum_{k=1}^{n-1} q_k^{(i,f)} (t_{k+1}^{(i)} - t_k^{(i)}) + r_k^{(i,f)}$$

Quantifier elimination: result of the procedure

Our QE procedure translate $\exists^{=x_1} y : \varphi(y, x_1, \dots, x_d)$ into

- $\bot$ (e.g. if there are infinitely many y satisfying φ), or
- a formula of the form

$$\bigvee_{\substack{i \in [1,o] \\ f : \{x_1, \dots, x_d\} \to [0, m-1]}} \underbrace{t_1^{(i)} \leq \dots \leq t_n^{(i)}}_{\text{ordering}} \wedge \underbrace{\Big(\bigwedge_{j=1}^{d} x_j \equiv_m f(x_j) \Big)}_{\text{residue classes}} \wedge \underbrace{m \cdot x_1 = \sum_{k=1}^{n-1} q_k^{(i,f)} (t_{k+1}^{(i)} - t_k^{(i)}) + r_k^{(i,f)}}_{\text{counting}}$$

Quantifier elimination: result of the procedure

Our QE procedure translate $\exists^{=x_1} y : \varphi(y, x_1, \dots, x_d)$ into

- $\bot$ (e.g. if there are infinitely many y satisfying φ), or
- a formula of the form

$$\bigvee_{\substack{i \in [1,o] \\ f : \{x_1,\dots,x_d\} \to [0,m-1]}} \underbrace{t_1^{(i)} \leq \dots \leq t_n^{(i)}}_{\text{ordering}} \land \underbrace{\left(\bigwedge_{j=1}^{d} x_j \equiv_m f(x_j) \right)}_{\text{residue classes}} \land \underbrace{m \cdot x_1 = \sum_{k=1}^{n-1} q_k^{(i,f)} (t_{k+1}^{(i)} - t_k^{(i)}) + r_k^{(i,f)}}_{\text{counting}}$$

The source of "towerness":

- First step of the procedure normalises the coefficients of y to -1 or 1.

Quantifier elimination: result of the procedure

Our QE procedure translate $\exists^{=x_1} y : \varphi(y, x_1, \ldots, x_d)$ into

- $\bot$ (e.g. if there are infinitely many y satisfying φ), or
- a formula of the form

$$\bigvee_{\substack{i \in [1,o] \\ f : \{x_1,\ldots,x_d\} \to [0,m-1]}} \underbrace{t_1^{(i)} \leq \cdots \leq t_n^{(i)}}_{\text{ordering}} \wedge \underbrace{\Big(\bigwedge_{j=1}^{d} x_j \equiv_m f(x_j) \Big)}_{\text{residue classes}} \wedge \underbrace{m \cdot x_1 = \sum_{k=1}^{n-1} q_k^{(i,f)} (t_{k+1}^{(i)} - t_k^{(i)}) + r_k^{(i,f)}}_{\text{counting}}$$

The source of "towerness":

- First step of the procedure normalises the coefficients of y to -1 or 1.
- This adds a constraint $y \equiv_C 0$ where C is the LCM of all coefficients of y.

Quantifier elimination: result of the procedure

Our QE procedure translate $\exists^{=x_1} y : \varphi(y, x_1, \dots, x_d)$ into

- $\bot$ (e.g. if there are infinitely many y satisfying φ), or
- a formula of the form

$$\bigvee_{\substack{i \in [1,o] \\ f : \{x_1,\dots,x_d\} \to [0,m-1]}} \underbrace{t_1^{(i)} \leq \dots \leq t_n^{(i)}}_{\text{ordering}} \wedge \underbrace{\Big(\bigwedge_{j=1}^{d} x_j \equiv_m f(x_j) \Big)}_{\text{residue classes}} \wedge \underbrace{m \cdot x_1 = \sum_{k=1}^{n-1} q_k^{(i,f)} (t_{k+1}^{(i)} - t_k^{(i)}) + r_k^{(i,f)}}_{\text{counting}}$$

The source of "towerness":

- First step of the procedure normalises the coefficients of y to -1 or 1.
- This adds a constraint $y \equiv_C 0$ where C is the LCM of all coefficients of y.
- A priori, across all orderings and residue classes, there are a lot of distinct coefficients in the counting part of the output formula.

Monadically-guarded fragment of PAC

$$\exists x : \psi(x, \mathbf{z}) \land \exists^{=x} y : \varphi(y, \mathbf{z})$$

where ψ is **monadically decomposable on** x:

Monadically-guarded fragment of PAC

$$\exists x : \psi(x, \mathbf{z}) \wedge \exists^{=x} y : \varphi(y, \mathbf{z})$$

where ψ is **monadically decomposable on** x:

$$\psi(x, \mathbf{z}) \;\equiv\; \bigvee_{i \in I} \psi_i(x) \wedge \gamma_i(\mathbf{z}).$$

Monadically-guarded fragment of PAC

$$\exists x : \psi(x, \mathbf{z}) \land \exists^{=x} y : \varphi(y, \mathbf{z})$$

where ψ is **monadically decomposable on** x:

$$\psi(x, \mathbf{z}) \;\equiv\; \bigvee\nolimits_{i \in I} \psi_i(x) \land \gamma_i(\mathbf{z}).$$

Examples:

"The number of y satisfying φ is congruent to x' modulo q"

$$\exists x : x \equiv_q x' \land \exists^{=x} y : \varphi$$

Monadically-guarded fragment of PAC

$$\exists x : \psi(x, \mathbf{z}) \land \exists^{=x} y : \varphi(y, \mathbf{z})$$

where ψ is **monadically decomposable on** x:

$$\psi(x, \mathbf{z}) \;\equiv\; \bigvee_{i \in I} \psi_i(x) \land \gamma_i(\mathbf{z}).$$

Examples:

"The number of y satisfying φ is congruent to x' modulo q"

$$\exists x : \bigvee_{r \in [0, q-1]} x \equiv_q r \land r \equiv_q x' \land \exists^{=x} y : \varphi$$

Monadically-guarded fragment of PAC

$$\exists x : \psi(x, \mathbf{z}) \wedge \exists^{=x} y : \varphi(y, \mathbf{z})$$

where ψ is **monadically decomposable on** x:

$$\psi(x, \mathbf{z}) \;\equiv\; \bigvee_{i \in I} \psi_i(x) \wedge \gamma_i(\mathbf{z}).$$

Examples:

"The number of y satisfying φ is congruent to x' modulo q"

$$\exists x : x \equiv_q x' \wedge \exists^{=x} y : \varphi$$

"The number of y satisfying φ is the product of all primes in $[2, 2^{2^n}]$"

$$\exists x : \ell_n(x) \wedge \exists^{=x} y : \varphi$$

Monadically-guarded fragment of PAC

$$\exists x : \psi(x, \mathbf{z}) \land \exists^{=x} y : \varphi(y, \mathbf{z})$$

where ψ is **monadically decomposable on** x:

$$\psi(x, \mathbf{z}) \;\equiv\; \bigvee_{i \in I} \psi_i(x) \land \gamma_i(\mathbf{z}).$$

Examples:

"The number of y satisfying φ is congruent to x' modulo q"

$$\exists x : x \equiv_q x' \land \exists^{=x} y : \varphi$$

"The number of y satisfying φ is the product of all primes in $[2, 2^{2^n}]$"

$$\exists x : \ell_n(x) \land \exists^{=x} y : \varphi$$

Note I: any Presburger formula with one free variable is monadically decomposable.

Monadically-guarded fragment of PAC

$$\exists x : \psi(x, \mathbf{z}) \wedge \exists^{=x} y : \varphi(y, \mathbf{z})$$

where ψ is **monadically decomposable on** x:

$$\psi(x, \mathbf{z}) \ \equiv \ \bigvee_{i \in I} \psi_i(x) \wedge \gamma_i(\mathbf{z}).$$

Examples:

"The number of y satisfying φ is congruent to x' modulo q"

$$\exists x : x \equiv_q x' \wedge \exists^{=x} y : \varphi$$

"The number of y satisfying φ is the product of all primes in $[2, 2^{2^n}]$"

$$\exists x : \ell_n(x) \wedge \exists^{=x} y : \varphi$$

Note II: monadic decomposability for PA is decidable [Libkin, TOCL'03].

Monadically-guarded fragment of PAC

$$\exists x : \psi(x, \mathbf{z}) \land \exists^{=x} y : \varphi(y, \mathbf{z})$$

where ψ is **monadically decomposable on** x:

Theorem

The monadically-guarded fragment of PAC is in $2\textsc{ExpSpace}$.

Monadically-guarded fragment of PAC

$$\exists x : \psi(x, \mathbf{z}) \land \exists^{=x} y : \varphi(y, \mathbf{z})$$

where ψ is **monadically decomposable on** x.

Theorem

The monadically-guarded fragment of PAC is in $2\textsc{ExpSpace}$.

What about $\mathbf{2AExp_{Poly}}$-complete fragments? Yes, as long as each monadic guard ψ has all solutions of magnitude doubly exponential in $|\psi|$.

Conclusion

In this paper: New quantifier elimination procedure for PAC:

- elimination of a single $\exists^{=x}$ costs one exponential
- shows $2\textsc{ExpSpace}$ membership for the monadically-guarded fragment.

Conclusion

In this paper: New quantifier elimination procedure for PAC:

- elimination of a single $\exists^{=x}$ costs one exponential
- shows $2\textsc{ExpSpace}$ membership for the monadically-guarded fragment.

One take-away message: The concept of monadic decomposition brings strong complexity advantages (see also [Libkin, TOCL'03]).

Conclusion

In this paper: New quantifier elimination procedure for PAC:

- elimination of a single $\exists^{=x}$ costs one exponential
- shows $2\textsc{ExpSpace}$ membership for the monadically-guarded fragment.

One take-away message: The concept of monadic decomposition brings strong complexity advantages (see also [Libkin, TOCL'03]).

Wide open problem: Exact complexity of PAC is still not known.