Linear arithmetic theories:
quantifier elimination procedures

Christoph Haase **Alessio Mansutti**

Linear arithmetic theories

- FO variables:

$$x, y, z, x_1, \ldots, x_n \in X$$

- Atomic formulas:

$$a_1 \cdot x_1 + \cdots + a_n \cdot x_n \leq b$$

- Boolean connectives:

$$\neg \qquad \wedge \qquad \vee \qquad \rightarrow$$

- Quantifiers:

$$\exists x : \Phi(x) \qquad \forall x : \Phi(x)$$

Linear arithmetic theories

- FO variables:

$$x, y, z, x_1, \ldots, x_n \in X$$

- Atomic formulas:

$$a_1 \cdot x_1 + \cdots + a_n \cdot x_n \leq b$$

- Boolean connectives:

$$\neg \qquad \wedge \qquad \vee \qquad \rightarrow$$

- Quantifiers:

$$\exists x : \Phi(x) \qquad \forall x : \Phi(x)$$

Linear integer arithmetic
(a.k.a. Presburger arithmetic)

$$(\mathbb{Z}, 0, 1, +, \leq)$$

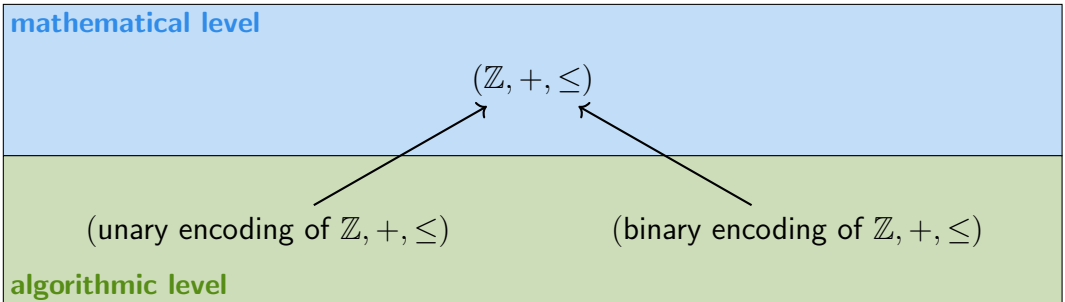
Proven decidable by
Mojżesz Presburger
in 1929

Today's lecture: quantifier elimination procedures

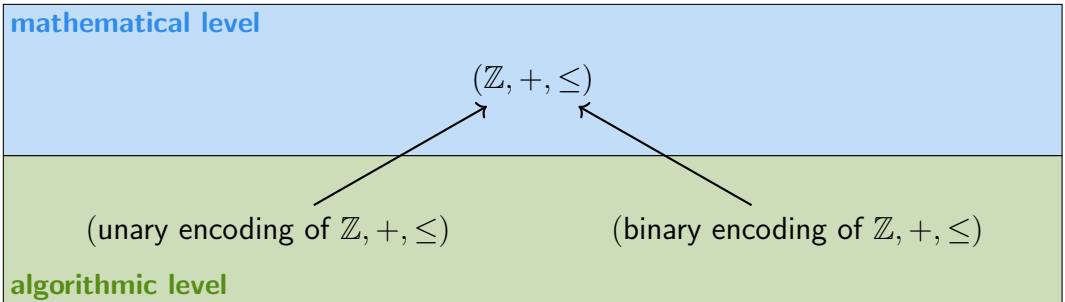
- Algorithmic paradigms for linear arithmetic theories
- Quantifier elimination for linear rational arithmetic
- Quantifier elimination for linear integer arithmetic (a.k.a. Presburger arithmetic)
- Linear integer arithmetic with unary counting quantifiers
- Tool presentation: REDLOG

Algorithmic paradigms for arithmetic theories

Representations of logical structures



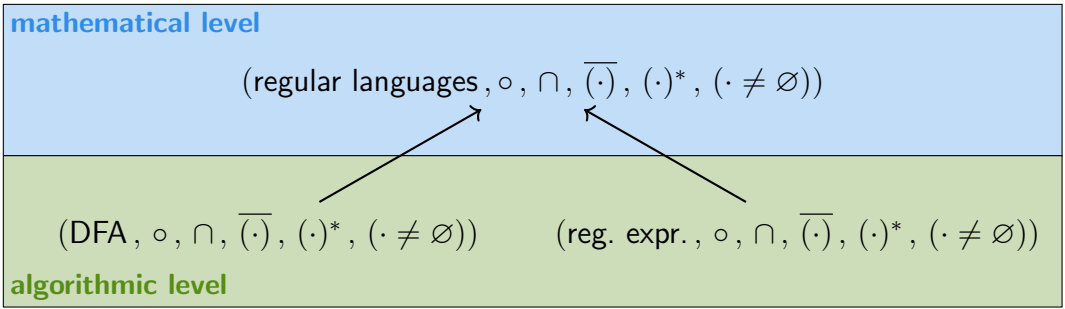
Representations of logical structures



For algorithms, we need "tangible logical structures", i.e. representations

- elements have a notion of size
- functions and relations are interpreted with computable functions

Representations of logical structures



For algorithms, we need "tangible logical structures", i.e. representations

- elements have a notion of size
- functions and relations are interpreted with computable functions

Distinct representations behave algorithmically differently

Representations and algorithmic paradigms for linear arithmetic theories

Quantifier elimination: (representation: formulae)

$$\exists x : \Phi_{\mathsf{qf}}(x, \boldsymbol{y}) \;\equiv\; \Psi_{\mathsf{qf}}(\boldsymbol{y}) \qquad \text{qf: quantifier-free}$$

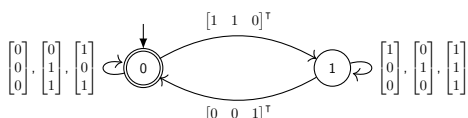
Representations and algorithmic paradigms for linear arithmetic theories

Quantifier elimination: (representation: formulae)

$$\exists x : \Phi_{\mathsf{qf}}(x, \boldsymbol{y}) \;\equiv\; \Psi_{\mathsf{qf}}(\boldsymbol{y}) \qquad \text{qf: quantifier-free}$$

Automata techniques: (representation: automata)

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



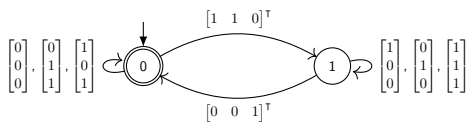
Representations and algorithmic paradigms for linear arithmetic theories

Quantifier elimination: (representation: formulae)

$$\exists x : \Phi_{\mathsf{qf}}(x, \boldsymbol{y}) \;\equiv\; \Psi_{\mathsf{qf}}(\boldsymbol{y}) \qquad \text{qf: quantifier-free}$$

Automata techniques: (representation: automata)

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



Geometric procedures: (representation: collections of vectors)

$$\begin{bmatrix} \mathtt{x} \\ \mathtt{y} \\ \mathtt{x{+}y} \end{bmatrix} : \quad \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{base}} + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \cdot \mathbb{N} + \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \cdot \mathbb{N}}_{\text{periods}}$$

Representations and algorithmic paradigms for linear arithmetic theories

Quantifier elimination: (representation: formulae)

$$\exists x : \Phi_{\mathsf{qf}}(x, \boldsymbol{y}) \;\equiv\; \Psi_{\mathsf{qf}}(\boldsymbol{y}) \qquad \text{qf: quantifier-free}$$

Automata techniques: (representation: automata)

$z = x + y \quad \rightsquigarrow$

Geometric procedures: (representation: collections of vectors)

What do we mean by quantifier elimination procedure?

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

What do we mean by quantifier elimination procedure?

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

$$\exists x_5 \exists x_4 \forall x_3 \forall x_2 \exists x_1 : \Phi(x_1, x_2, x_3, x_4, x_5)$$

What do we mean by quantifier elimination procedure?

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

$$\exists x_5 \exists x_4 \forall x_3 \forall x_2 \exists x_1 : \Phi(x_1, x_2, x_3, x_4, x_5)$$

What do we mean by quantifier elimination procedure?

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

$$\exists x_5 \exists x_4 \forall x_3 \forall x_2 \Phi_1(x_2, x_3, x_4, x_5)$$

What do we mean by quantifier elimination procedure?

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

$$\exists x_5 \exists x_4 \neg \exists x_3 \exists x_2 \neg \Phi_1(x_2, x_3, x_4, x_5)$$

What do we mean by quantifier elimination procedure?

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

$$\exists x_5 \exists x_4 \neg \exists x_3 \Phi_2(x_3, x_4, x_5)$$

What do we mean by quantifier elimination procedure?

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

$$\exists x_5 \exists x_4 \neg \Phi_3(x_4, x_5)$$

What do we mean by quantifier elimination procedure?

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

$\exists x_5 \Phi_4(x_5)$

What do we mean by quantifier elimination procedure?

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

Φ_5

What do we mean by quantifier elimination procedure?

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

Φ_5 $\leftarrow$ No variables, so we can evaluate it!

e.g. $1 < 2 \wedge 2 + 3 > 4 \wedge 2 \neq 2$ evaluates to **false**

What do we mean by quantifier elimination procedure?

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

Techniques from model theory are often able to show that Ψ exists.
Here we want a **procedure** that computes Ψ

What do we mean by quantifier elimination procedure?

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

Techniques from model theory are often able to show that Ψ exists.
Here we want a **procedure** that computes Ψ

Most quantifier elimination procedures are **axiomatic**:
they manipulate $\exists x\, \Phi(x, \boldsymbol{y})$ using tautologies of the logic

Quantifier elimination procedures for linear arithmetics

Quantifier elimination for the linear **rational** arithmetic $(\mathbb{Q}, 0, 1, +, \leq)$

Fourier-Motzkin quantifier elimination (1826,1936)

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

Quantifier elimination for the linear **rational** arithmetic $(\mathbb{Q}, 0, 1, +, \leq)$

Fourier-Motzkin quantifier elimination (1826,1936)

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

Suppose that $\Phi(x, \boldsymbol{y})$ is a conjunction of inequalities

$$f(x, \boldsymbol{y}) \leq g(x, \boldsymbol{y}) \text{ or } f(x, \boldsymbol{y}) > g(x, \boldsymbol{y}).$$

Quantifier elimination for the linear **rational** arithmetic $(\mathbb{Q}, 0, 1, +, \leq)$

Fourier-Motzkin quantifier elimination (1826,1936)

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

Suppose that $\Phi(x, \boldsymbol{y})$ is a conjunction of inequalities

$$f(x, \boldsymbol{y}) \leq g(x, \boldsymbol{y}) \text{ or } f(x, \boldsymbol{y}) > g(x, \boldsymbol{y}).$$

Update each inequality so that it has one of the following forms:

$$\underbrace{\ell(\boldsymbol{y}) < x, \quad \ell(\boldsymbol{y}) \leq x}_{\text{lower bounds}}, \quad \underbrace{x \leq u(\boldsymbol{y}), \quad x < u(\boldsymbol{y})}_{\text{upper bounds}}, \quad \underbrace{h(\boldsymbol{y}) \geq 0, \quad h(\boldsymbol{y}) < 0}_{\text{free from } x}.$$

Quantifier elimination for the linear **rational** arithmetic $(\mathbb{Q}, 0, 1, +, \leq)$

FOURIER-MOTZKIN QUANTIFIER ELIMINATION (1826,1936)

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

Suppose that $\Phi(x, \boldsymbol{y})$ is a conjunction of inequalities

$$f(x, \boldsymbol{y}) \leq g(x, \boldsymbol{y}) \text{ or } f(x, \boldsymbol{y}) > g(x, \boldsymbol{y}).$$

Update each inequality so that it has one of the following forms:

$$\underbrace{\ell(\boldsymbol{y}) < x, \quad \ell(\boldsymbol{y}) \leq x}_{\text{lower bounds}}, \quad \underbrace{x \leq u(\boldsymbol{y}), \quad x < u(\boldsymbol{y})}_{\text{upper bounds}}, \quad \underbrace{h(\boldsymbol{y}) \geq 0, \quad h(\boldsymbol{y}) < 0}_{\text{free from } x}.$$

E.g. $-2x + y \leq 5 \quad \rightarrow \quad y - 5 \leq 2x \quad \rightarrow \quad \frac{1}{2} \cdot y - \frac{5}{2} \leq x.$

Quantifier elimination for the linear **rational** arithmetic $(\mathbb{Q}, 0, 1, +, \leq)$

Fourier-Motzkin quantifier elimination (1826,1936)

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(x, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \ell_j(\boldsymbol{y}) < x \qquad \wedge \bigwedge_{j=1}^{r_2} x < u_j(\boldsymbol{y}) \qquad \wedge \bigwedge_{j=1}^{r_3} h(\boldsymbol{y}) < 0$$

$$\bigwedge_{j=1}^{s_1} \ell_j(\boldsymbol{y}) \leq x \qquad \wedge \bigwedge_{j=1}^{s_2} x \leq u_j(\boldsymbol{y}) \qquad \wedge \bigwedge_{j=1}^{s_3} h(\boldsymbol{y}) \leq 0$$

Quantifier elimination for the linear **rational** arithmetic $(\mathbb{Q}, 0, 1, +, \leq)$

FOURIER-MOTZKIN QUANTIFIER ELIMINATION (1826,1936)

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(x, \boldsymbol{y})$ is translated into a system of the form

$$\begin{array}{lll} \bigwedge_{j=1}^{r_1} \ell_j(\boldsymbol{y}) < x & \wedge \bigwedge_{j=1}^{r_2} x < u_j(\boldsymbol{y}) & \wedge \bigwedge_{j=1}^{r_3} h(\boldsymbol{y}) < 0 \\ \bigwedge_{j=1}^{s_1} \ell_j(\boldsymbol{y}) \leq x & \wedge \bigwedge_{j=1}^{s_2} x \leq u_j(\boldsymbol{y}) & \wedge \bigwedge_{j=1}^{s_3} h(\boldsymbol{y}) \leq 0 \end{array}$$

eliminate x by combining lower and upper bounds

$$\ell(\boldsymbol{y}) \leq x \wedge x < u(\boldsymbol{y}) \quad \rightarrow \quad \ell(\boldsymbol{y}) < u(\boldsymbol{y})$$

Quantifier elimination for the linear **rational** arithmetic $(\mathbb{Q}, 0, 1, +, \leq)$

FOURIER-MOTZKIN QUANTIFIER ELIMINATION (1826,1936)

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

- $\bigvee_{i=1}^{n} \Phi_i(x, \boldsymbol{y}) \;\coloneqq$ translate $\Phi(x, \boldsymbol{y})$ into disjunctive normal form
- for every $i \in [1, n]$, eliminate x in Φ_i :
  - massage every inequality $f(x, \boldsymbol{y}) \leq g(x, \boldsymbol{y})$ to make them of the form

    $$\ell(\boldsymbol{y}) \leq x, \quad x \leq u(\boldsymbol{y}), \quad h(\boldsymbol{y}) \leq 0, \quad \text{(or strict variants with } <\text{)}$$

  - and rewrite $\ell_j(\boldsymbol{y}) \leq x \wedge x \leq u_k(\boldsymbol{y})$ into $\ell_j(\boldsymbol{y}) \leq u_k(\boldsymbol{y})$
- **output** the resulting formula

Towards Presburger's quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

Towards Presburger's quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

Suppose that $\Phi(x, \boldsymbol{y})$ is a conjunction of inequalities $f(x, \boldsymbol{y}) \leq g(x, \boldsymbol{y})$

Over $\mathbb{Q}$ we also had $f(x, \boldsymbol{y}) > g(x, \boldsymbol{y})$. But over $\mathbb{Z}$ we can rewrite strict inequalities:

$$f(x, \boldsymbol{y}) > g(x, \boldsymbol{y}) \quad \iff \quad f(x, \boldsymbol{y}) \geq g(x, \boldsymbol{y}) + 1$$

Towards Presburger's quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

Suppose that $\Phi(x, \boldsymbol{y})$ is a conjunction of inequalities $f(x, \boldsymbol{y}) \leq g(x, \boldsymbol{y})$

Over $\mathbb{Q}$, next step was to generate the lower and upper bounds

E.g. $-2x + y \leq 5 \quad \rightarrow \quad y - 5 \leq 2x \quad \rightarrow \quad \frac{1}{2} \cdot y - \frac{5}{2} \leq x.$

Towards Presburger's quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

Suppose that $\Phi(x, \boldsymbol{y})$ is a conjunction of inequalities $f(x, \boldsymbol{y}) \leq g(x, \boldsymbol{y})$

Over $\mathbb{Q}$, next step was to generate the lower and upper bounds

E.g. $-2x + y \leq 5 \quad \rightarrow \quad y - 5 \leq 2x \quad \rightarrow \quad \frac{1}{2} \cdot y - \frac{5}{2} \leq x.$

Towards Presburger's quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

Suppose that $\Phi(x, \boldsymbol{y})$ is a conjunction of inequalities $f(x, \boldsymbol{y}) \leq g(x, \boldsymbol{y})$

Over $\mathbb{Q}$, next step was to generate the lower and upper bounds

E.g. $-2x + y \leq 5 \quad \rightarrow \quad y - 5 \leq 2x \quad \rightarrow \quad \frac{1}{2} \cdot y - \frac{5}{2} \leq x.$

Idea: What if we modify Φ so that x only has **one** coefficient?

Towards Presburger's quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(x, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \ell_j(\boldsymbol{y}) \leq a_j \cdot x \quad \wedge \bigwedge_{j=1}^{r_2} b_j \cdot x \leq u_j(\boldsymbol{y}) \quad \wedge \Phi'(\boldsymbol{y})$$

where each a_j and b_j is a positive integer.

Towards Presburger's quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(x, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \ell_j(\boldsymbol{y}) \leq a_j \cdot x \quad \wedge \bigwedge_{j=1}^{r_2} b_j \cdot x \leq u_j(\boldsymbol{y}) \quad \wedge \Phi'(\boldsymbol{y})$$

where each a_j and b_j is a positive integer.

$C :=$ product of all a_j and b_j

Towards Presburger's quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(x, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \frac{C}{a_j} \cdot \ell_j(\boldsymbol{y}) \leq \frac{C}{a_j} \cdot a_j \cdot x \qquad \wedge \bigwedge_{j=1}^{r_2} \frac{C}{b_j} \cdot b_j \cdot x \leq \frac{C}{b_j} \cdot u_j(\boldsymbol{y}) \qquad \wedge \Phi'(\boldsymbol{y})$$

where each a_j and b_j is a positive integer.

$C \coloneqq$ product of all a_j and b_j

Towards Presburger's quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(x, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \frac{C}{a_j} \cdot \ell_j(\boldsymbol{y}) \leq C \cdot x \qquad \wedge \bigwedge_{j=1}^{r_2} C \cdot x \leq \frac{C}{b_j} \cdot u_j(\boldsymbol{y}) \qquad \wedge \Phi'(\boldsymbol{y})$$

where each a_j and b_j is a positive integer.

$C \coloneqq$ product of all a_j and b_j

Towards Presburger's quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(x, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \frac{C}{a_j} \cdot \ell_j(\boldsymbol{y}) \leq C \cdot x \qquad \wedge \bigwedge_{j=1}^{r_2} C \cdot x \leq \frac{C}{b_j} \cdot u_j(\boldsymbol{y}) \qquad \wedge \Phi'(\boldsymbol{y})$$

where each a_j and b_j is a positive integer.

$C \coloneqq$ product of all a_j and b_j

Combining lower and upper bounds is problematic:

$\ell(\boldsymbol{y}) \leq C \cdot x \ \wedge \ C \cdot x \leq u(\boldsymbol{y})$ is implicitly saying that between $\ell(\boldsymbol{y})$ and $u(\boldsymbol{y})$ we can find a multiple of C.

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$ is not possible!

Issue: The formula $\Phi(y) \coloneqq \exists x.\, y = 3 \cdot x$ states that 3 divides y.
There is no quantifier-free formula equivalent to $\Phi(y)$!

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$ is not possible!

Issue: The formula $\Phi(y) \coloneqq \exists x.\, y = 3 \cdot x$ states that 3 divides y.
There is no quantifier-free formula equivalent to $\Phi(y)$!

Solution [Presburger, '29]**:** Enrich the structure $(\mathbb{Z}, 0, 1, +, \leq)$ with a relation $d|\cdot$ for every d positive integer

for every $n \in \mathbb{Z}$, $\quad d \,|\, n \iff d$ divides n

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, \leq)$ is not possible!

Issue: The formula $\Phi(y) \coloneqq \exists x.\, y = 3 \cdot x$ states that 3 divides y.
There is no quantifier-free formula equivalent to $\Phi(y)$!

Solution [Presburger, '29]**:** Enrich the structure $(\mathbb{Z}, 0, 1, +, \leq)$ with a relation $d|\cdot$ for every d positive integer

$$\text{for every } n \in \mathbb{Z}, \quad d \,|\, n \iff d \text{ divides } n$$

The structure $(\mathbb{Z}, 0, 1, +, (d \mid \cdot)_{d \in \mathbb{Z}_+}, \leq)$ admits quantifier elimination!

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, (d \mid \cdot)_{d \in \mathbb{Z}_+}, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, (d \mid \cdot)_{d \in \mathbb{Z}_+}, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

Suppose $\Phi(x, \boldsymbol{y})$ conjunction of constraints $f(x, \boldsymbol{y}) \leq g(x, \boldsymbol{y})$ and $d \mid h(x, \boldsymbol{y})$

Note: $\neg(d \mid h(x, \boldsymbol{y})) \iff \bigvee_{r=1}^{d-1} d \mid h(x, \boldsymbol{y}) + r$

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, (d \mid \cdot)_{d \in \mathbb{Z}_+}, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(x, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \frac{C}{a_j} \cdot \ell_j(\boldsymbol{y}) \leq C \cdot x \ \wedge \ \bigwedge_{j=1}^{r_2} C \cdot x \leq \frac{C}{b_j} \cdot u_j(\boldsymbol{y}) \wedge \bigwedge_{j=1}^{r_3} \quad d_j \mid (\quad c_j \cdot x + \quad h_j(\boldsymbol{y})) \wedge \Phi'(\boldsymbol{y})$$

where each a_j and b_j is a positive integer.

$C :=$ product of all a_j and b_j

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, (d \mid \cdot)_{d \in \mathbb{Z}_+}, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(x, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \frac{C}{a_j} \cdot \ell_j(\boldsymbol{y}) \leq C \cdot x \ \wedge \ \bigwedge_{j=1}^{r_2} C \cdot x \leq \frac{C}{b_j} \cdot u_j(\boldsymbol{y}) \wedge \bigwedge_{j=1}^{r_3} C \cdot d_j \mid (C \cdot c_j \cdot x + C \cdot h_j(\boldsymbol{y})) \wedge \Phi'(\boldsymbol{y})$$

where each a_j and b_j is a positive integer.

$C \coloneqq$ product of all a_j and b_j

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, (d \mid \cdot)_{d \in \mathbb{Z}_+}, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(\widetilde{x}, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \frac{C}{a_j} \cdot \ell_j(\boldsymbol{y}) \leq \widetilde{x} \wedge \bigwedge_{j=1}^{r_2} \widetilde{x} \leq \frac{C}{b_j} \cdot u_j(\boldsymbol{y}) \wedge \bigwedge_{j=1}^{r_3} C \cdot d_j \mid (c_j \cdot \widetilde{x} + C \cdot h_j(\boldsymbol{y})) \wedge C \mid \widetilde{x} \wedge \Phi'(\boldsymbol{y})$$

where each a_j and b_j is a positive integer.

$C \coloneqq$ product of all a_j and b_j

apply change of variable $C \cdot x \ \rightarrow \ \widetilde{x}$, where $\widetilde{x}$ is a fresh variable

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, (d \mid \cdot)_{d \in \mathbb{Z}_+}, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(\widetilde{x}, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \frac{C}{a_j} \cdot \ell_j(\boldsymbol{y}) \leq \widetilde{x} \wedge \bigwedge_{j=1}^{r_2} \widetilde{x} \leq \frac{C}{b_j} \cdot u_j(\boldsymbol{y}) \wedge \bigwedge_{j=1}^{r_3} C \cdot d_j \mid (c_j \cdot \widetilde{x} + C \cdot h_j(\boldsymbol{y})) \wedge C \mid \widetilde{x} \wedge \Phi'(\boldsymbol{y})$$

wh

C :

app

Eliminate $\widetilde{x}$:

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, (d \mid \cdot)_{d \in \mathbb{Z}_+}, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(\widetilde{x}, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \frac{C}{a_j} \cdot \ell_j(\boldsymbol{y}) \leq \widetilde{x} \wedge \bigwedge_{j=1}^{r_2} \widetilde{x} \leq \frac{C}{b_j} \cdot u_j(\boldsymbol{y}) \wedge \bigwedge_{j=1}^{r_3} C \cdot d_j \mid (c_j \cdot \widetilde{x} + C \cdot h_j(\boldsymbol{y})) \wedge C \mid \widetilde{x} \wedge \Phi'(\boldsymbol{y})$$

wh

C :

app

Eliminate $\widetilde{x}$:

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, (d \mid \cdot)_{d \in \mathbb{Z}_+}, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(\widetilde{x}, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \frac{C}{a_j} \cdot \ell_j(\boldsymbol{y}) \leq \widetilde{x} \wedge \bigwedge_{j=1}^{r_2} \widetilde{x} \leq \frac{C}{b_j} \cdot u_j(\boldsymbol{y}) \wedge \bigwedge_{j=1}^{r_3} C \cdot d_j \mid (c_j \cdot \widetilde{x} + C \cdot h_j(\boldsymbol{y})) \wedge C \mid \widetilde{x} \wedge \Phi'(\boldsymbol{y})$$

wh

C :

app

Eliminate $\widetilde{x}$:

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, (d \mid \cdot)_{d \in \mathbb{Z}_+}, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(\widetilde{x}, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \frac{C}{a_j} \cdot \ell_j(\boldsymbol{y}) \leq \widetilde{x} \wedge \bigwedge_{j=1}^{r_2} \widetilde{x} \leq \frac{C}{b_j} \cdot u_j(\boldsymbol{y}) \wedge \bigwedge_{j=1}^{r_3} C \cdot d_j \mid (c_j \cdot \widetilde{x} + C \cdot h_j(\boldsymbol{y})) \wedge C \mid \widetilde{x} \wedge \Phi'(\boldsymbol{y})$$

wh

C :

app

Eliminate $\widetilde{x}$:

multiple of $D := C \cdot \mathrm{lcm}(d_j : j \in [1, r_3])$

Quantifier elimination for $(\mathbb{Z}, 0, 1, +, (d \mid \cdot)_{d\in\mathbb{Z}_+}, \leq)$

Input: $\Phi(x, \boldsymbol{y})$ q.f.; we want to eliminate x

$\Phi(\widetilde{x}, \boldsymbol{y})$ is translated into a system of the form

$$\bigwedge_{j=1}^{r_1} \frac{C}{a_j} \cdot \ell_j(\boldsymbol{y}) \leq \widetilde{x} \wedge \bigwedge_{j=1}^{r_2} \widetilde{x} \leq \frac{C}{b_j} \cdot u_j(\boldsymbol{y}) \wedge \bigwedge_{j=1}^{r_3} C \cdot d_j \mid (c_j \cdot \widetilde{x} + C \cdot h_j(\boldsymbol{y})) \wedge C \mid \widetilde{x} \wedge \Phi'(\boldsymbol{y})$$

wh

C :

app

Eliminate $\widetilde{x}$: $\exists \widetilde{x}\, \Phi(\widetilde{x}, \boldsymbol{y})$ is equivalent to

$$\left(\bigvee_{j=1}^{r_1} \bigvee_{r=0}^{D-1} \Phi(\frac{C}{a_j} \cdot \ell_j(\boldsymbol{y}) + r, \boldsymbol{y}) \right) \vee \left(\bigvee_{j=1}^{r_2} \bigvee_{r=0}^{D-1} \Phi(\frac{C}{b_j} \cdot u_j(\boldsymbol{y}) - r, \boldsymbol{y}) \right)$$

Quantifier elimination for linear integer arithmetic (recap)

- Put the formula in disjunctive normal form
- modify inequalities so that x only has one coefficient C
- variable substitution $C \cdot x \ \rightarrow \ \widetilde{x}$; add the constraint $C \mid \widetilde{x}$
- "guess" a value for $\widetilde{x}$ that is "close" to a lower or upper bound;
  how close depend on C and on the divisors d of constraints $d \mid f(x, \boldsymbol{y})$

Quantifier elimination for linear integer arithmetic (recap)

- ~~Put the formula in disjunctive normal form~~
- modify inequalities so that x only has one coefficient C
- variable substitution $C \cdot x \ \rightarrow \ \widetilde{x}$; add the constraint $C \mid \widetilde{x}$
- "guess" a value for $\widetilde{x}$ that is "close" to a lower or upper bound;
  how close depend on C and on the divisors d of constraints $d \mid f(x, \boldsymbol{y})$

Quantifier elimination for linear integer arithmetic (recap)

Quantifier elimination procedure: on input formula $\Phi(x, \boldsymbol{y})$

1. let $T \coloneqq \{(a, f(\boldsymbol{y})) : a \cdot x \sim f(\boldsymbol{y}) \text{ in } \Phi, \text{ with } \sim \in \{\leq, \geq\} \text{ and } a \in \mathbb{N}_+\}$
2. let $C \coloneqq \prod\{a : (a, f) \in T\}$
3. let $D \coloneqq C \cdot \operatorname{lcm}\{d : \ d \mid f(x, \boldsymbol{y}) \text{ in } \Phi\}$
4. replace in Φ every $a \cdot x \sim f(\boldsymbol{y})$ with $\widetilde{x} \sim \frac{C}{a} \cdot f(\boldsymbol{y}) \leq 0$
5. **return** $\displaystyle\bigvee_{(a, f(\boldsymbol{y})) \in T} \bigvee_{r=-D}^{D} \left(\Phi(r - \tfrac{C}{a} \cdot f(\boldsymbol{y}), \boldsymbol{y}) \wedge C \mid r - \tfrac{C}{a} \cdot f(\boldsymbol{y})\right)$

The complexity of linear integer arithmetic

Theorem

There is a quantifier elimination procedure for linear integer arithmetic running in $\mathbf{3EXPTIME}$.

All numbers in the resulting formula have a doubly exponential bit length

The complexity of linear integer arithmetic

Theorem

There is a quantifier elimination procedure for linear integer arithmetic running in $\mathbf{3EXPTIME}$.

All numbers in the resulting formula have a doubly exponential bit length

Corollary

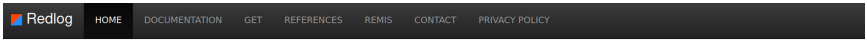
There is a computable function f *from formulae to positive integers such that*

- $f(\Phi) \leq \exp\exp\langle\Phi\rangle$, $\langle\Phi\rangle :=$ *bit length of* Φ
- $\exists x : \Phi(x, \boldsymbol{y})$ *is equivalent to* $\exists x \in [-2^{f(\Phi)}, 2^{f(\Phi)}] : \Phi(x, \boldsymbol{y})$.

Hence, satisfiability of linear integer arithmetic is in $\mathbf{2EXPSPACE}$.

Tool presentation: Redlog

Demo of the tool Redlog



Interpreted First-Order Logic

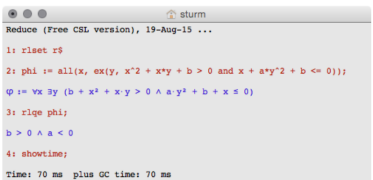
Redlog generally works with interpreted first-order logic in contrast to free first-order logic. Each first-order formula in Redlog must exclusively contain atoms from one particular Redlog-supported domain, which corresponds to a choice of admissible functions and relations with fixed semantics. Redlog-supported domains include nonlinear real arithmetic (Tarski Algebra), Presburger arithmetic, parametric quantified Boolean formulas, and many more.

Quantifier Elimination

Effective quantifier elimination procedures for the various supported theories play a central role in Redlog: Given a first-order formula φ, Redlog computes an equivalent formula φ' that does not contain any quantifier. The screenshot shows an example in the real domain.

Symbolic Computation

Redlog provides a productive interactive environment for constructing, analyzing, and manipulating formulas. This includes various normal form computations, advanced logical and algebraic simplification techniques, and quantifier elimination and decision procedures. All this can be combined with the symbolic computation power of the host computer algebra system REDUCE.



Decision Methods

Redlog's quantifier elimination procedures are also decision procedures. For instance, Redlog's quantifier elimination computes "false" for the decision problem ∀a∃b(φ) with φ as above. Some further decision methods in Redlog are not based on quantifier elimination.

Connectivity

REDUCE has been designed as an interactive computer algebra system. The REDUCE source distribution includes a C library for managing REDUCE processes and communicating with these processes. Besides communication via REDUCE command strings, e.g., an XML-based communication protocol can be easily realized.

License

Both REDUCE and Redlog are open-source and freely distributed under a very liberal FreeBSD License.

Demo of the tool Redlog

1. **rlset** **theory** **;**
   - **integers** : Presburger arithmetic ($+$ some support for multiplication)
   - **reals** : real closed field $(\mathbb{R}, 0, 1, +, \cdot, \leq)$
   - **boolean**, **complex**, **differential**, **queues**, ...
2. **phi** $:=$ **all(**x**, ex(**y**,** $x + 3y < 5$ **and** $x - 3y + z \leq 6$**)) ;**
3. **rlvarl phi ;** list free and bound variables $\{\{z\}, \{x, y\}\}$
4. **rlqe phi ;** generic quantifier elimination procedure
5. **cad phi ;** q.e. for specific theories. E.g., **cad** is only for real closed field

Linear integer arithmetic with unary counting quantifiers

Counting in Presburger arithmetic

Counting quantifiers:

$\exists^{=x}\boldsymbol{y} : \Phi(\boldsymbol{z}, x, \boldsymbol{y})$ *"there are x many distinct vectors $\boldsymbol{y}$ that satisfy Φ"*

Counting in Presburger arithmetic

Counting quantifiers:

$\exists^{=x} \boldsymbol{y} : \Phi(\boldsymbol{z}, x, \boldsymbol{y})$ *"there are x many distinct vectors $\boldsymbol{y}$ that satisfy Φ"*

Theorem (folklore)

Presburger arithmetic enriched with counting quantifiers is undecidable.

Counting in Presburger arithmetic

Counting quantifiers:

$\exists^{=x}\boldsymbol{y} : \Phi(\boldsymbol{z}, x, \boldsymbol{y})$ *"there are x many distinct vectors $\boldsymbol{y}$ that satisfy Φ"*

Theorem (folklore)

Presburger arithmetic enriched with counting quantifiers is undecidable.

Proof.

$\Phi(x, y, z) := \exists^{=x}(a, b) : 1 \leq a \leq y \wedge 1 \leq b \leq z$ if and only if $x = y \cdot z$. ■

Counting in Presburger arithmetic

Counting quantifiers:

$\exists^{=x}\boldsymbol{y} : \Phi(\boldsymbol{z}, x, \boldsymbol{y})$ *"there are x many distinct vectors $\boldsymbol{y}$ that satisfy Φ"*

Unary counting quantifiers:

$\exists^{=x} y : \Phi(\boldsymbol{z}, x, y)$ *"there are x many distinct integers y that satisfy Φ"*

Theorem (Apelt, 1966; Schweikardt, 2005)

Presburger arithmetic enriched with unary counting quantifiers is decidable.

Counting in Presburger arithmetic

Counting quantifiers:

$\exists^{=x}\boldsymbol{y} : \Phi(\boldsymbol{z}, x, \boldsymbol{y})$ *"there are x many distinct vectors $\boldsymbol{y}$ that satisfy Φ"*

Unary counting quantifiers:

$\exists^{=x} y : \Phi(\boldsymbol{z}, x, y)$ *"there are x many distinct integers y that satisfy Φ"*

Theorem (Apelt, 1966; Schweikardt, 2005)

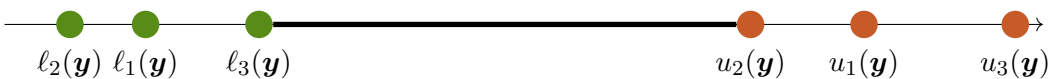
Presburger arithmetic enriched with unary counting quantifiers is decidable.

Idea: Eliminate unary counting quantifiers in favour of standard first-order quantifiers.

Eliminating unary counting quantifiers: conjunctive case

$\exists^{=x} y : \Phi(\boldsymbol{z}, x, y)$

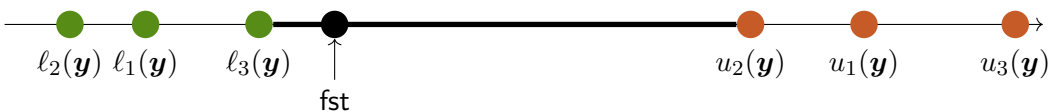
where $\Phi(\boldsymbol{z}, x, y) \coloneqq \bigwedge_{j=1}^{r_1} \ell_j(x, \boldsymbol{z}) \leq y \wedge \bigwedge_{j=1}^{r_2} y \leq u_j(x, \boldsymbol{z}) \wedge C \mid y - r \wedge \Psi(\boldsymbol{z}, x)$



Eliminating unary counting quantifiers: conjunctive case

$$\exists^{=x} y : \Phi(\boldsymbol{z}, x, y)$$

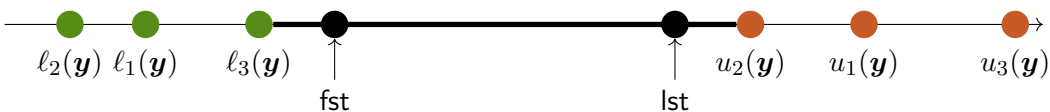
where $\Phi(\boldsymbol{z}, x, y) \coloneqq \bigwedge_{j=1}^{r_1} \ell_j(x, \boldsymbol{z}) \leq y \wedge \bigwedge_{j=1}^{r_2} y \leq u_j(x, \boldsymbol{z}) \wedge C \mid y - r \wedge \Psi(\boldsymbol{z}, x)$



Eliminating unary counting quantifiers: conjunctive case

$\exists^{=x} y : \Phi(\boldsymbol{z}, x, y)$

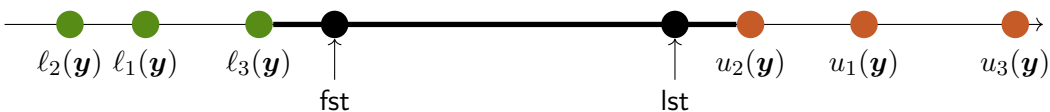
where $\Phi(\boldsymbol{z}, x, y) \coloneqq \bigwedge_{j=1}^{r_1} \ell_j(x, \boldsymbol{z}) \leq y \wedge \bigwedge_{j=1}^{r_2} y \leq u_j(x, \boldsymbol{z}) \wedge C \mid y - r \wedge \Psi(\boldsymbol{z}, x)$



Eliminating unary counting quantifiers: conjunctive case

$$\exists^{=x} y : \Phi(\boldsymbol{z}, x, y)$$

where $\Phi(\boldsymbol{z}, x, y) \coloneqq \bigwedge_{j=1}^{r_1} \ell_j(x, \boldsymbol{z}) \leq y \wedge \bigwedge_{j=1}^{r_2} y \leq u_j(x, \boldsymbol{z}) \wedge C \mid y - r \wedge \Psi(\boldsymbol{z}, x)$



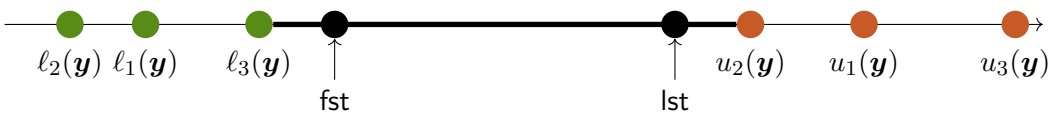
$$\Phi_{\min}(y, \boldsymbol{z}, x) \coloneqq \Phi(y, \boldsymbol{z}, x) \wedge \forall w : \Phi(w, \boldsymbol{z}, x) \implies y \leq w$$

$$\Phi_{\max}(y, \boldsymbol{z}, x) \coloneqq \Phi(y, \boldsymbol{z}, x) \wedge \forall w : \Phi(w, \boldsymbol{z}, x) \implies w \leq y$$

Eliminating unary counting quantifiers: conjunctive case

$$\exists^{=x} y : \Phi(\boldsymbol{z}, x, y)$$

$$\text{where} \quad \Phi(\boldsymbol{z}, x, y) \coloneqq \bigwedge_{j=1}^{r_1} \ell_j(x, \boldsymbol{z}) \leq y \wedge \bigwedge_{j=1}^{r_2} y \leq u_j(x, \boldsymbol{z}) \wedge C \mid y - r \wedge \Psi(\boldsymbol{z}, x)$$



$$\Phi_{\min}(y, \boldsymbol{z}, x) \coloneqq \Phi(y, \boldsymbol{z}, x) \wedge \forall w : \Phi(w, \boldsymbol{z}, x) \implies y \leq w$$

$$\Phi_{\max}(y, \boldsymbol{z}, x) \coloneqq \Phi(y, \boldsymbol{z}, x) \wedge \forall w : \Phi(w, \boldsymbol{z}, x) \implies w \leq y$$

$$\exists^{=x} y : \Phi(\boldsymbol{z}, x, y) \quad \iff \quad \exists \mathsf{fst}\, \exists \mathsf{lst} : \Phi_{\min}(\mathsf{fst}, \boldsymbol{z}, x) \wedge \Phi_{\max}(\mathsf{lst}, \boldsymbol{z}, x) \wedge x = \frac{\mathsf{lst} - \mathsf{fst}}{C} + 1$$

Eliminating unary counting quantifiers

Input: $\exists^{=x} y : \Phi(\boldsymbol{z}, x, y)$, where $\Phi(\boldsymbol{z}, x, y)$ is quantifier-free

Eliminating unary counting quantifiers

Input: $\exists^{=x} y : \Phi(\boldsymbol{z}, x, y)$, where $\Phi(\boldsymbol{z}, x, y)$ is quantifier-free

1. Translate $\Phi(\boldsymbol{z}, x, y)$ into an equivalent formula $\bigvee_{i=1}^{k} \Phi_i(\boldsymbol{z}, x, y)$ where
   - $\Phi_i(\boldsymbol{z}, x, y)$ is of the form $\bigwedge_{j=1}^{r_1} \ell_j(x, \boldsymbol{z}) \leq y \wedge \bigwedge_{j=1}^{r_2} y \leq u_j(x, \boldsymbol{z}) \wedge C \mid y - r \wedge \Psi(x, \boldsymbol{z})$
   - for every $i \neq j$ in $[1, k]$, $\Phi_i \wedge \Phi_j$ is unsatisfiable

Eliminating unary counting quantifiers

Input: $\exists^{=x} y : \Phi(\boldsymbol{z}, x, y)$, where $\Phi(\boldsymbol{z}, x, y)$ is quantifier-free

1. Translate $\Phi(\boldsymbol{z}, x, y)$ into an equivalent formula $\bigvee_{i=1}^{k} \Phi_i(\boldsymbol{z}, x, y)$ where
   - $\Phi_i(\boldsymbol{z}, x, y)$ is of the form $\bigwedge_{j=1}^{r_1} \ell_j(x, \boldsymbol{z}) \leq y \wedge \bigwedge_{j=1}^{r_2} y \leq u_j(x, \boldsymbol{z}) \wedge C \mid y - r \wedge \Psi(x, \boldsymbol{z})$
   - for every $i \neq j$ in $[1, k]$, $\Phi_i \wedge \Phi_j$ is unsatisfiable
2. look at $\exists x_1, \ldots, x_k : x = x_1 + \cdots + x_k \wedge \bigwedge_{i=1}^{k} \exists^{=x_i} y : \Phi_i(\boldsymbol{z}, x, y)$ ...
3. ... and eliminate every $\exists^{=x_i} y$ in favour of first-order quantifiers

Eliminating unary counting quantifiers

Input: $\exists^{=x} y : \Phi(\boldsymbol{z}, x, y)$, where $\Phi(\boldsymbol{z}, x, y)$ is quantifier-free

1. Translate $\Phi(\boldsymbol{z}, x, y)$ into an equivalent formula $\bigvee_{i=1}^{k} \Phi_i(\boldsymbol{z}, x, y)$ where
   - $\Phi_i(\boldsymbol{z}, x, y)$ is of the form $\bigwedge_{j=1}^{r_1} \ell_j(x, \boldsymbol{z}) \leq y \wedge \bigwedge_{j=1}^{r_2} y \leq u_j(x, \boldsymbol{z}) \wedge C \mid y - r \wedge \Psi(x, \boldsymbol{z})$
   - for every $i \neq j$ in $[1, k]$, $\Phi_i \wedge \Phi_j$ is unsatisfiable
2. look at $\exists x_1, \ldots, x_k : x = x_1 + \cdots + x_k \wedge \bigwedge_{i=1}^{k} \exists^{=x_i} y : \Phi_i(\boldsymbol{z}, x, y)$ ...
3. ... and eliminate every $\exists^{=x_i} y$ in favour of first-order quantifiers
4. eliminate every first-order quantifier; **output** the resulting formula

The complexity of unary counting in Presburger arithmetic

Theorem (Schweikardt, 2005)

There is a quantifier elimination procedure for linear integer arithmetic enriched with unary counting quantifiers running in $\mathbf{TOWER}$.

Best known lower bound is the same as standard Presburger arithmetic

i.e. hard for the class of problems decidable by an alternating Turing machine running in $\mathbf{2EXPTIME}$ and performing a linear amount of alternations

(between $\mathbf{2NEXPTIME}$ and $\mathbf{2EXPSPACE}$)

The complexity of unary counting in Presburger arithmetic

Theorem (Schweikardt, 2005)

There is a quantifier elimination procedure for linear integer arithmetic enriched with unary counting quantifiers running in $\mathbf{TOWER}$.

Best known lower bound is the same as standard Presburger arithmetic

i.e. hard for the class of problems decidable by an alternating Turing machine running in $\mathbf{2EXPTIME}$ and performing a linear amount of alternations

(between $\mathbf{2NEXPTIME}$ and $\mathbf{2EXPSPACE}$)

Theorem (Chistikov, Haase & Mansutti, 2022)

$\mathbf{2EXPSPACE}$ *if counting quantifiers only appear as* $\exists x\big(\Phi(\boldsymbol{z}, x) \wedge \exists^{=x} y : \Psi(\boldsymbol{z}, y)\big)$.

Summary of today's lecture

Quantifier elimination procedure

Input: a quantifier-free (q.f.) formula $\Phi(x, \boldsymbol{y})$ from a first-order logic

Output: a q.f. formula $\Psi(\boldsymbol{y})$ that is equivalent to $\exists x\, \Phi(x, \boldsymbol{y})$

- Linear rational arithmetic
- Linear integer arithmetic
- Linear integer arithmetic with unary counting quantifiers $\exists^{=x} y$

Summary of today's lecture

Takeaway messages:

Pros:

- quantifier-elimination is a simple yet powerful technique
- it yields algorithms that are often optimal
- these algorithms are relatively easy to implement and prove correct: they are based on formula manipulations using tautologies of the logic

Cons:

- signature of the theory might need to be enriched (see $d \mid \cdot$) ...
- ... so it is difficult to design a q.e. procedure for logics where infinitely many relations need to be added to the signature (e.g. Büchi arithmetic)

Agenda

Tomorrow Automata-based procedures

Friday Geometric procedures