02917 Advanced Topics in Embedded Systems

Presburger Arithmetic: Cooper's algorithm

Michael R. Hansen

DTU Informatics
Department of Informatics and Mathematical Modelling

Introduction

Presburger Arithmetic (introduced by Mojzesz Presburger in 1929), is the first-order theory of natural numbers with addition.

Examples of formulas are: $\exists x.2x = y$ and $\exists x.\forall y.x + y > z$.

Unlike Peano Arithmetic, which also includes multiplication, Presburger Arithmetic is a decidable theory.

We shall consider the algorithm introduced by D.C Cooper in 1972.

The presentation is based on: Chapter 7: Quantified Linear Arithmetic of *The Calculus of Computation* by Bradley and Manna.

Introduction

Presburger Arithmetic (introduced by Mojzesz Presburger in 1929), is the first-order theory of natural numbers with addition.

Examples of formulas are: $\exists x.2x = y$ and $\exists x.\forall y.x + y > z$.

Unlike Peano Arithmetic, which also includes multiplication, Presburger Arithmetic is a decidable theory.

We shall consider the algorithm introduced by D.C Cooper in 1972.

The presentation is based on: Chapter 7: Quantified Linear Arithmetic of *The Calculus of Computation* by Bradley and Manna.

Introduction

Presburger Arithmetic (introduced by Mojzesz Presburger in 1929), is the first-order theory of natural numbers with addition.

Examples of formulas are: $\exists x.2x = y$ and $\exists x.\forall y.x + y > z$.

Unlike Peano Arithmetic, which also includes multiplication, Presburger Arithmetic is a decidable theory.

We shall consider the algorithm introduced by D.C Cooper in 1972.

The presentation is based on: Chapter 7: Quantified Linear Arithmetic of *The Calculus of Computation* by Bradley and Manna.

Introduction

Presburger Arithmetic (introduced by Mojzesz Presburger in 1929), is the first-order theory of natural numbers with addition.

Examples of formulas are: $\exists x.2x = y$ and $\exists x.\forall y.x + y > z$.

Unlike Peano Arithmetic, which also includes multiplication, Presburger Arithmetic is a decidable theory.

We shall consider the algorithm introduced by D.C Cooper in 1972.

The presentation is based on: Chapter 7: Quantified Linear Arithmetic of *The Calculus of Computation* by Bradley and Manna.

Presburger terms

The terms are generated from

- integer constants $\ldots, -2, -1, 0, 1, 2, \ldots$ and
- variables $x, y, z, \ldots$

using the following operations:

- addition $+$ and subtraction $-$ and
- multiplication by constants: $\ldots,\ -2\cdot,\ -1\cdot,\ 0\cdot,\ 1\cdot,\ 2\cdot,\ \ldots$

Notice

- terms are interpreted over integers
- the terms do not really allow multiplication as, e.g. $3 \cdot x$ is equal to $x + x + x$
- a term like $3 \cdot x$ is usually written $3x$

Presburger terms

The terms are generated from

- integer constants $\ldots, -2, -1, 0, 1, 2, \ldots$ and
- variables $x, y, z, \ldots$

using the following operations:

- addition $+$ and subtraction $-$ and
- multiplication by constants: $\ldots,\ -2\cdot,\ -1\cdot,\ 0\cdot,\ 1\cdot,\ 2\cdot,\ \ldots$

Notice

- terms are interpreted over integers
- the terms do not really allow multiplication as, e.g. $3 \cdot x$ is equal to $x + x + x$
- a term like $3 \cdot x$ is usually written $3x$

Presburger terms

The terms are generated from

- integer constants $\ldots, -2, -1, 0, 1, 2, \ldots$ and
- variables $x, y, z, \ldots$

using the following operations:

- addition $+$ and subtraction $-$ and
- multiplication by constants: $\ldots,\ -2\cdot,\ -1\cdot,\ 0\cdot,\ 1\cdot,\ 2\cdot,\ \ldots$

Notice

- terms are interpreted over integers
- the terms do not really allow multiplication as, e.g. $3 \cdot x$ is equal to $x + x + x$
- a term like $3 \cdot x$ is usually written $3x$

Presburger terms

The terms are generated from

- integer constants $\ldots, -2, -1, 0, 1, 2, \ldots$ and
- variables $x, y, z, \ldots$

using the following operations:

- addition $+$ and subtraction $-$ and
- multiplication by constants: $\ldots,\ -2\cdot,\ -1\cdot,\ 0\cdot,\ 1\cdot,\ 2\cdot,\ \ldots$

Notice

- terms are interpreted over integers
- the terms do not really allow multiplication as, e.g. $3 \cdot x$ is equal to $x + x + x$
- a term like $3 \cdot x$ is usually written $3x$

Formulas

We consider formulas $F, G, F_1, G_1, F_2 \ldots$ of the following forms:

- $s = t, s < t, s > t, s \leq t$, and $s \geq t$ (comparisons)
- $1|s, 2|s, 3|s, \ldots$ (divisibility constraints)
- $\top$ (true) and $\bot$ (false) (propositional constants)
- $F \vee G$ (disjunction), $F \wedge G$ (conjunction) and $\neg F$ (negation) (propositional connectives)
- $\exists x.F$ (reads "there exists an x such that F") and $\forall x.F$ (reads "for all x: F") (first-order fragment)

where s and t are terms and x is a variable.

Furthermore, We allow brackets in formulas.

Notice: The formulas are interpreted over the integers.

Formulas

We consider formulas $F, G, F_1, G_1, F_2 \ldots$ of the following forms:

- $s = t, s < t, s > t, s \leq t$, and $s \geq t$ (comparisons)
- $1|s, 2|s, 3|s, \ldots$ (divisibility constraints)
- $\top$ (true) and $\bot$ (false) (propositional constants)
- $F \vee G$ (disjunction), $F \wedge G$ (conjunction) and $\neg F$ (negation) (propositional connectives)
- $\exists x.F$ (reads "there exists an x such that F") and $\forall x.F$ (reads "for all x: F") (first-order fragment)

where s and t are terms and x is a variable.

Furthermore, We allow brackets in formulas.

Notice: The formulas are interpreted over the integers.

Formulas

We consider formulas $F, G, F_1, G_1, F_2 \ldots$ of the following forms:

- $s = t, s < t, s > t, s \leq t$, and $s \geq t$ (comparisons)
- $1|s, 2|s, 3|s, \ldots$ (divisibility constraints)
- $\top$ (true) and $\bot$ (false) (propositional constants)
- $F \vee G$ (disjunction), $F \wedge G$ (conjunction) and $\neg F$ (negation) (propositional connectives)
- $\exists x.F$ (reads "there exists an x such that F") and $\forall x.F$ (reads "for all x: F") (first-order fragment)

where s and t are terms and x is a variable.

Furthermore, We allow brackets in formulas.

Notice: The formulas are interpreted over the integers.

Formulas

We consider formulas $F, G, F_1, G_1, F_2 \ldots$ of the following forms:

- $s = t, s < t, s > t, s \leq t$, and $s \geq t$ (comparisons)
- $1|s, 2|s, 3|s, \ldots$ (divisibility constraints)
- $\top$ (true) and $\bot$ (false) (propositional constants)
- $F \vee G$ (disjunction), $F \wedge G$ (conjunction) and $\neg F$ (negation) (propositional connectives)
- $\exists x.F$ (reads "there exists an x such that F") and $\forall x.F$ (reads "for all x: F") (first-order fragment)

where s and t are terms and x is a variable.

Furthermore, We allow brackets in formulas.

Notice: The formulas are interpreted over the integers.

Formulas

We consider formulas $F, G, F_1, G_1, F_2 \ldots$ of the following forms:

- $s = t, s < t, s > t, s \leq t$, and $s \geq t$ (comparisons)
- $1|s, 2|s, 3|s, \ldots$ (divisibility constraints)
- $\top$ (true) and $\bot$ (false) (propositional constants)
- $F \vee G$ (disjunction), $F \wedge G$ (conjunction) and $\neg F$ (negation) (propositional connectives)
- $\exists x.F$ (reads ”there exists an x such that F”) and $\forall x.F$ (reads ”for all x: F”) (first-order fragment)

where s and t are terms and x is a variable.

Furthermore, We allow brackets in formulas.

Notice: The formulas are interpreted over the integers.

Formulas

We consider formulas $F, G, F_1, G_1, F_2 \ldots$ of the following forms:

- $s = t, s < t, s > t, s \leq t$, and $s \geq t$ (comparisons)
- $1|s, 2|s, 3|s, \ldots$ (divisibility constraints)
- $\top$ (true) and $\bot$ (false) (propositional constants)
- $F \vee G$ (disjunction), $F \wedge G$ (conjunction) and $\neg F$ (negation) (propositional connectives)
- $\exists x.F$ (reads "there exists an x such that F") and $\forall x.F$ (reads "for all x: F") (first-order fragment)

where s and t are terms and x is a variable.

Furthermore, We allow brackets in formulas.

Notice: The formulas are interpreted over the integers.

Formulas

We consider formulas $F, G, F_1, G_1, F_2 \ldots$ of the following forms:

- $s = t, s < t, s > t, s \leq t$, and $s \geq t$ (comparisons)
- $1|s, 2|s, 3|s, \ldots$ (divisibility constraints)
- $\top$ (true) and $\bot$ (false) (propositional constants)
- $F \vee G$ (disjunction), $F \wedge G$ (conjunction) and $\neg F$ (negation) (propositional connectives)
- $\exists x.F$ (reads "there exists an x such that F") and $\forall x.F$ (reads "for all x: F") (first-order fragment)

where s and t are terms and x is a variable.

Furthermore, We allow brackets in formulas.

Notice: The formulas are interpreted over the integers.

Notational conventions

- Relative precedence: $\neg$ (highest – binds tightest), $\wedge$, $\vee$ (lowest)
- The quantifiers $\forall x$ and $\exists x$ extend as far as possible to the right.
- $\forall x_1.\forall x_2.\ldots.\forall x_n.F$ is abbreviated to $\forall x_1, x_2, \ldots, x_n.F$

Example:

$$\forall x.\exists y.\neg x + 1 = 2y \wedge x > 0 \vee y < 2$$

means

$$\forall x.\exists y.(((\neg x + 1 = 2y) \wedge x > 0) \vee y < 2)$$

Notational conventions

- Relative precedence: $\neg$ (highest – binds tightest), $\wedge$, $\vee$ (lowest)
- The quantifiers $\forall x$ and $\exists x$ extend as far as possible to the right.
- $\forall x_1.\forall x_2.\ldots.\forall x_n.F$ is abbreviated to $\forall x_1, x_2, \ldots, x_n.F$

Example:

$$\forall x.\exists y.\neg x + 1 = 2y \wedge x > 0 \vee y < 2$$

means

$$\forall x.\exists y.(((\neg x + 1 = 2y) \wedge x > 0) \vee y < 2)$$

Notational conventions

- Relative precedence: $\neg$ (highest – binds tightest), $\wedge$, $\vee$ (lowest)
- The quantifiers $\forall x$ and $\exists x$ extend as far as possible to the right.
- $\forall x_1.\forall x_2.\dots.\forall x_n.F$ is abbreviated to $\forall x_1, x_2, \dots, x_n.F$

Example:

$$\forall x.\exists y.\neg x + 1 = 2y \wedge x > 0 \vee y < 2$$

means

$$\forall x.\exists y.(((\neg x + 1 = 2y) \wedge x > 0) \vee y < 2)$$

Notational conventions

- Relative precedence: $\neg$ (highest – binds tightest), $\wedge$, $\vee$ (lowest)
- The quantifiers $\forall x$ and $\exists x$ extend as far as possible to the right.
- $\forall x_1.\forall x_2.\ldots.\forall x_n.F$ is abbreviated to $\forall x_1, x_2, \ldots, x_n.F$

Example:

$$\forall x.\exists y.\neg x + 1 = 2y \wedge x > 0 \vee y < 2$$

means

$$\forall x.\exists y.(((\neg x + 1 = 2y) \wedge x > 0) \vee y < 2)$$

Notational conventions

- Relative precedence: $\neg$ (highest – binds tightest), $\wedge$, $\vee$ (lowest)
- The quantifiers $\forall x$ and $\exists x$ extend as far as possible to the right.
- $\forall x_1.\forall x_2.\dots.\forall x_n.F$ is abbreviated to $\forall x_1, x_2, \dots, x_n.F$

Example:

$$\forall x.\exists y.\neg x + 1 = 2y \wedge x > 0 \vee y < 2$$

means

$$\forall x.\exists y.(((\neg x + 1 = 2y) \wedge x > 0) \vee y < 2)$$

Some concepts

- In $\forall x.F$:
  - x is called the quantified variable
  - $\forall x$ is called the quantifier
  - F is the scope of the quantifier

  The case for $\exists x.F$ is similar.

- An occurrence of a variable x in a formula F is a bound occurrence if it occurs in the scope of a quantifier $\forall x$ or $\exists x$ in F. Otherwise, that occurrence of x is free in F.

- x is a free variable of F if there is some free occurrence of x in F.

- A formula is called closed if it contains no free variables; otherwise it is called open.

Some concepts

- In $\forall x.F$:
  - x is called the quantified variable
  - $\forall x$ is called the quantifier
  - F is the scope of the quantifier

  The case for $\exists x.F$ is similar.

- An occurrence of a variable x in a formula F is a bound occurrence if it occurs in the scope of a quantifier $\forall x$ or $\exists x$ in F. Otherwise, that occurrence of x is free in F.

- x is a free variable of F if there is some free occurrence of x in F.

- A formula is called closed if it contains no free variables; otherwise it is called open.

Some concepts

- In $\forall x.F$:
  - x is called the quantified variable
  - $\forall x$ is called the quantifier
  - F is the scope of the quantifier

  The case for $\exists x.F$ is similar.

- An occurrence of a variable x in a formula F is a bound occurrence if it occurs in the scope of a quantifier $\forall x$ or $\exists x$ in F. Otherwise, that occurrence of x is free in F.

- x is a free variable of F if there is some free occurrence of x in F.

- A formula is called closed if it contains no free variables; otherwise it is called open.

Some concepts

- In $\forall x.F$:
  - x is called the quantified variable
  - $\forall x$ is called the quantifier
  - F is the scope of the quantifier

  The case for $\exists x.F$ is similar.

- An occurrence of a variable x in a formula F is a bound occurrence if it occurs in the scope of a quantifier $\forall x$ or $\exists x$ in F. Otherwise, that occurrence of x is free in F.

- x is a free variable of F if there is some free occurrence of x in F.

- A formula is called closed if it contains no free variables; otherwise it is called open.

Semantics

Let $\mathbb{Z}$ denote the set of integers $\ldots, -2, -1, 0, 1, 2, \ldots$.

The operations $+$ and $-$ and the relations $=, <, \leq, >, \geq$ have their standard meaning.

A interpretation I assigns an integer $I(x) \in \mathbb{Z}$ to every variable x.

Let $I \lhd \{x \mapsto v\}$ be the x-variant of I which is as I except that v is assigned to x.

Semantics

Let $\mathbb{Z}$ denote the set of integers $\ldots, -2, -1, 0, 1, 2, \ldots$.

The operations $+$ and $-$ and the relations $=, <, \leq, >, \geq$ have their standard meaning.

A interpretation I assigns an integer $I(x) \in \mathbb{Z}$ to every variable x.

Let $I \lhd \{x \mapsto v\}$ be the x-variant of I which is as I except that v is assigned to x.

Semantics

Let $\mathbb{Z}$ denote the set of integers $\ldots, -2, -1, 0, 1, 2, \ldots$.

The operations $+$ and $-$ and the relations $=, <, \leq, >, \geq$ have their standard meaning.

A interpretation I assigns an integer $I(x) \in \mathbb{Z}$ to every variable x.

Let $I \lhd \{x \mapsto v\}$ be the x-variant of I which is as I except that v is assigned to x.

Semantics

Let $\mathbb{Z}$ denote the set of integers $\ldots, -2, -1, 0, 1, 2, \ldots$.

The operations $+$ and $-$ and the relations $=, <, \leq, >, \geq$ have their standard meaning.

A interpretation I assigns an integer $I(x) \in \mathbb{Z}$ to every variable x.

Let $I \lhd \{x \mapsto v\}$ be the x-variant of I which is as I except that v is assigned to x.

Semantics of terms

Let an assignment I be given.

The semantics of a term s is an integer $\hat{I}(s) \in \mathbb{Z}$ defined as follows:

$$
\begin{array}{lcll}
\hat{I}(x) & = & I(x) & \\
\hat{I}(a) & = & a & \text{where } a \in \mathbb{Z} \\
\hat{I}(s+t) & = & \hat{I}(s) + \hat{I}(t) & \\
\hat{I}(s-t) & = & \hat{I}(s) - \hat{I}(t) & \\
\hat{I}(a \cdot s) & = & a \cdot \hat{I}(s) & \text{where } a \in \mathbb{Z}
\end{array}
$$

Semantics of formulas

Let an assignment I be given.

The semantic relation $I \models F$ is defined by structural induction on formulas:

$I \models s < t$	iff	$\hat{I}(s) < \hat{I}(t)$	other relations are similar
$I \models a \mid s$	iff	a divides $\hat{I}(s)$	where $a \in \mathbb{Z}$
$I \models \neg F$	iff	not ($I \models F$)	
$I \models F \vee G$	iff	$I \models F$ or $I \models G$	
$I \models F \wedge G$	iff	$I \models F$ and $I \models G$	
$I \models \forall x.F$	iff	$I \lhd \{x \mapsto v\} \models F$	for every $v \in \mathbb{Z}$
$I \models \exists x.F$	iff	$I \lhd \{x \mapsto v\} \models F$	for some $v \in \mathbb{Z}$

Concepts

A formula *F* is satisfiable if there is some assignment *I* for which the formula is true. Otherwise it is unsatisfiable.

A formula is valid if it is true for all assignments.

Notice: The truth value of a closed formula is independent of the chosen assignment. It is either valid (true for all assignments), or unsatisfiable (false for all assignments).

Concepts

A formula *F* is satisfiable if there is some assignment *I* for which the formula is true. Otherwise it is unsatisfiable.

A formula is valid if it is true for all assignments.

Notice: The truth value of a closed formula is independent of the chosen assignment. It is either valid (true for all assignments), or unsatisfiable (false for all assignments).

Concepts

A formula F is satisfiable if there is some assignment I for which the formula is true. Otherwise it is unsatisfiable.

A formula is valid if it is true for all assignments.

Notice: The truth value of a closed formula is independent of the chosen assignment. It is either valid (true for all assignments), or unsatisfiable (false for all assignments).

Examples

- $\exists y . x = 2y$ — (x is even) is satisfiable but not valid
  also expressible as $2|x$
- $\exists y . x = 2y \vee x = 2y + 1$ — (x is even or x is odd) is valid
  also expressible as $2|x \vee 2|x + 1$
- $\exists x . \forall y . x \leq y$ is unsatifiable (false)
- $\exists x . \forall y . x + y = y$ is valid (true)

Examples

- $\exists y . x = 2y$ — (x is even) is satisfiable but not valid
  also expressible as $2 | x$
- $\exists y . x = 2y \vee x = 2y + 1$ — (x is even or x is odd) is valid
  also expressible as $2 | x \vee 2 | x + 1$
- $\exists x . \forall y . x \leq y$ is unsatifiable (false)
- $\exists x . \forall y . x + y = y$ is valid (true)

Examples

- $\exists y. x = 2y$ — (x is even) is satisfiable but not valid
  also expressible as $2|x$
- $\exists y. x = 2y \lor x = 2y + 1$ — (x is even or x is odd) is valid
  also expressible as $2|x \lor 2|x + 1$
- $\exists x. \forall y. x \leq y$ is unsatifiable (false)
- $\exists x. \forall y. x + y = y$ is valid (true)

Examples

- $\exists y.x = 2y$ — (x is even) is satisfiable but not valid
  also expressible as $2|x$
- $\exists y.x = 2y \vee x = 2y + 1$ — (x is even or x is odd) is valid
  also expressible as $2|x \vee 2|x + 1$
- $\exists x.\forall y.x \leq y$ is unsatifiable (false)
- $\exists x.\forall y.x + y = y$ is valid (true)

Quantifier elimination

In the theory of real numbers an example of quantifier elimination is:

$$\exists x . ax^2 + bx + c = 0 \quad \text{is equivalent to} \quad b^2 - 4ac \geq 0$$

where $a, b, c \in \mathbb{R}$ and $a \neq 0$.

Presburger developed a method, which for an arbitrary Presburger formula F gives to an equivalent quantifier-free formula G.

If F is a closed formula, then the truth value of G can be computed.

For example, Cooper's algorithm for Presburger Arithmetic transforms:

$$\exists x . (3x + 1 < 10 \vee 7x - 6 > 7) \wedge 2 | x$$

to the following variable- and quantifier-free formula:

$$\bigvee_{j=1}^{42} (42 | j \wedge 21 | j)$$
$$\vee$$
$$\bigvee_{j=1}^{42} ((39 + j < 63 \vee 39 < 39 + j) \wedge 42 | 39 + j \wedge 21 | 39 + j)$$

Quantifier elimination

In the theory of real numbers an example of quantifier elimination is:

$$\exists x. ax^2 + bx + c = 0 \quad \text{is equivalent to} \quad b^2 - 4ac \geq 0$$

where $a, b, c \in \mathbb{R}$ and $a \neq 0$.

Presburger developed a method, which for an arbitrary Presburger formula F gives to an equivalent quantifier-free formula G.

If F is a closed formula, then the truth value of G can be computed.

For example, Cooper's algorithm for Presburger Arithmetic transforms:

$$\exists x.(3x + 1 < 10 \vee 7x - 6 > 7) \wedge 2|x$$

to the following variable- and quantifier-free formula:

$$\bigvee_{j=1}^{42}(42|j \wedge 21|j)$$
$$\vee$$
$$\bigvee_{j=1}^{42}((39 + j < 63 \vee 39 < 39 + j) \wedge 42|39 + j \wedge 21|39 + j)$$

Quantifier elimination

In the theory of real numbers an example of quantifier elimination is:

$$\exists x. ax^2 + bx + c = 0 \quad \text{is equivalent to} \quad b^2 - 4ac \geq 0$$

where $a, b, c \in \mathbb{R}$ and $a \neq 0$.

Presburger developed a method, which for an arbitrary Presburger formula F gives to an equivalent quantifier-free formula G.

If F is a closed formula, then the truth value of G can be computed.

For example, Cooper's algorithm for Presburger Arithmetic transforms:

$$\exists x.(3x + 1 < 10 \vee 7x - 6 > 7) \wedge 2|x$$

to the following variable- and quantifier-free formula:

$$\bigvee_{j=1}^{42}(42|j \wedge 21|j)$$
$$\vee$$
$$\bigvee_{j=1}^{42}((39 + j < 63 \vee 39 < 39 + j) \wedge 42|39 + j \wedge 21|39 + j)$$

Quantifier elimination

In the theory of real numbers an example of quantifier elimination is:

$$\exists x. ax^2 + bx + c = 0 \quad \text{is equivalent to} \quad b^2 - 4ac \geq 0$$

where $a, b, c \in \mathbb{R}$ and $a \neq 0$.

Presburger developed a method, which for an arbitrary Presburger formula F gives to an equivalent quantifier-free formula G.

If F is a closed formula, then the truth value of G can be computed.

For example, Cooper's algorithm for Presburger Arithmetic transforms:

$$\exists x.(3x + 1 < 10 \vee 7x - 6 > 7) \wedge 2|x$$

to the following variable- and quantifier-free formula:

$$\bigvee_{j=1}^{42} (42|j \wedge 21|j)$$
$$\vee$$
$$\bigvee_{j=1}^{42} ((39 + j < 63 \vee 39 < 39 + j) \wedge 42|39 + j \wedge 21|39 + j)$$

Quantifier elimination

In the theory of real numbers an example of quantifier elimination is:

$$\exists x . ax^2 + bx + c = 0 \quad \text{is equivalent to} \quad b^2 - 4ac \geq 0$$

where $a, b, c \in \mathbb{R}$ and $a \neq 0$.

Presburger developed a method, which for an arbitrary Presburger formula F gives to an equivalent quantifier-free formula G.

If F is a closed formula, then the truth value of G can be computed.

For example, Cooper's algorithm for Presburger Arithmetic transforms:

$$\exists x . (3x + 1 < 10 \vee 7x - 6 > 7) \wedge 2 | x$$

to the following variable- and quantifier-free formula:

$$\begin{array}{l} \bigvee_{j=1}^{42} (42 | j \wedge 21 | j) \\ \vee \\ \bigvee_{j=1}^{42} ((39 + j < 63 \vee 39 < 39 + j) \wedge 42 | 39 + j \wedge 21 | 39 + j) \end{array}$$

Quantifier Elimination (II)

Excluding divisible predicates $a|s$ from the Presburger Formulas quantifier elimination is not possible.

Lemma: If $F(y)$ is a quantifier-free formula with one free variable y. Let

$$S = \{n \in \mathbb{Z} \mid F(n) \text{ is valid}\}$$

Then either

$$S \cap \mathbb{Z}^+ \quad \text{or} \quad \mathbb{Z}^+ \setminus S$$

is finite

Consider the formula: $\exists x.2x = y$.

- $S \cap \mathbb{Z}^+$ is the infinite set of positive even numbers
- $\mathbb{Z}^+ \setminus S$ is the infinite set of positive odd numbers

The addition of divisibility predicates makes quantifier elimination possible for Presburger Arithmetic, and, e.g.

$$\exists x.2x = y \quad \text{is equivalent to} \quad 2|y$$

Quantifier Elimination (II)

Excluding divisible predicates $a|s$ from the Presburger Formulas quantifier elimination is not possible.

Lemma: If $F(y)$ is a quantifier-free formula with one free variable y. Let

$$S = \{n \in \mathbb{Z} \mid F(n) \text{ is valid}\}$$

Then either

$$S \cap \mathbb{Z}^+ \qquad \text{or} \qquad \mathbb{Z}^+ \setminus S$$

is finite

Consider the formula: $\exists x.2x = y$.

- $S \cap \mathbb{Z}^+$ is the infinite set of positive even numbers
- $\mathbb{Z}^+ \setminus S$ is the infinite set of positive odd numbers

The addition of divisibility predicates makes quantifier elimination possible for Presburger Arithmetic, and, e.g.

$$\exists x.2x = y \qquad \text{is equivalent to} \qquad 2|y$$

Quantifier Elimination (II)

Excluding divisible predicates $a|s$ from the Presburger Formulas quantifier elimination is not possible.

Lemma: If $F(y)$ is a quantifier-free formula with one free variable y. Let

$$S = \{n \in \mathbb{Z} \mid F(n) \text{ is valid}\}$$

Then either

$$S \cap \mathbb{Z}^+ \quad \text{or} \quad \mathbb{Z}^+ \setminus S$$

is finite

Consider the formula: $\exists x.2x = y$.

- $S \cap \mathbb{Z}^+$ is the infinite set of positive even numbers
- $\mathbb{Z}^+ \setminus S$ is the infinite set of positive odd numbers

The addition of divisibility predicates makes quantifier elimination possible for Presburger Arithmetic, and, e.g.

$$\exists x.2x = y \quad \text{is equivalent to} \quad 2|y$$

Quantifier Elimination (II)

Excluding divisible predicates $a|s$ from the Presburger Formulas quantifier elimination is not possible.

Lemma: If $F(y)$ is a quantifier-free formula with one free variable y. Let

$$S = \{n \in \mathbb{Z} \mid F(n) \text{ is valid}\}$$

Then either

$$S \cap \mathbb{Z}^+ \qquad \text{or} \qquad \mathbb{Z}^+ \setminus S$$

is finite

Consider the formula: $\exists x.2x = y$.

- $S \cap \mathbb{Z}^+$ is the infinite set of positive even numbers
- $\mathbb{Z}^+ \setminus S$ is the infinite set of positive odd numbers

The addition of divisibility predicates makes quantifier elimination possible for Presburger Arithmetic, and, e.g.

$$\exists x.2x = y \qquad \text{is equivalent to} \qquad 2|y$$

Quantifier Elimination (II)

Excluding divisible predicates $a|s$ from the Presburger Formulas quantifier elimination is not possible.

Lemma: If $F(y)$ is a quantifier-free formula with one free variable y. Let

$$S = \{n \in \mathbb{Z} \mid F(n) \text{ is valid}\}$$

Then either

$$S \cap \mathbb{Z}^+ \qquad \text{or} \qquad \mathbb{Z}^+ \setminus S$$

is finite

Consider the formula: $\exists x.2x = y$.

- $S \cap \mathbb{Z}^+$ is the infinite set of positive even numbers
- $\mathbb{Z}^+ \setminus S$ is the infinite set of positive odd numbers

The addition of divisibility predicates makes quantifier elimination possible for Presburger Arithmetic, and, e.g.

$$\exists x.2x = y \qquad \text{is equivalent to} \qquad 2|y$$

Quantifier Elimination (II)

Excluding divisible predicates $a|s$ from the Presburger Formulas quantifier elimination is not possible.

Lemma: If $F(y)$ is a quantifier-free formula with one free variable y. Let

$$S = \{n \in \mathbb{Z} \mid F(n) \text{ is valid}\}$$

Then either

$$S \cap \mathbb{Z}^+ \qquad \text{or} \qquad \mathbb{Z}^+ \setminus S$$

is finite

Consider the formula: $\exists x.2x = y$.

- $S \cap \mathbb{Z}^+$ is the infinite set of positive even numbers
- $\mathbb{Z}^+ \setminus S$ is the infinite set of positive odd numbers

The addition of divisibility predicates makes quantifier elimination possible for Presburger Arithmetic, and, e.g.

$$\exists x.2x = y \qquad \text{is equivalent to} \qquad 2|y$$

Cooper's procedure - main idea

Consider a formula $\exists x.F[x]$, where F is quantifier free.
Main steps:

- Put $F[x]$ on negation normal form, yielding $F_1[x]$
- Normalize $F_1[x]$ to use $<$ as the only comparison operator, yielding $F_2[x]$
- Normalize $F_2[x]$ so that atomic formulas have one occurrence of x (at most), yielding $F_3[x]$
- Normalize $F_3[x]$ so that the coefficients of x is 1 (in atomic formulas containing x), yielding $F_4[x']$
- Construct from $F_4[x']$ a quantifier-free formula F_5 which is equivalent to $\exists x.F[x]$

Cooper's procedure - main idea

Consider a formula $\exists x.F[x]$, where F is quantifier free.
Main steps:

- Put $F[x]$ on negation normal form, yielding $F_1[x]$
- Normalize $F_1[x]$ to use $<$ as the only comparison operator, yielding $F_2[x]$
- Normalize $F_2[x]$ so that atomic formulas have one occurrence of x (at most), yielding $F_3[x]$
- Normalize $F_3[x]$ so that the coefficients of x is 1 (in atomic formulas containing x), yielding $F_4[x']$
- Construct from $F_4[x']$ a quantifier-free formula F_5 which is equivalent to $\exists x.F[x]$

Cooper's procedure - main idea

Consider a formula $\exists x.F[x]$, where F is quantifier free.
Main steps:

- Put $F[x]$ on negation normal form, yielding $F_1[x]$
- Normalize $F_1[x]$ to use $<$ as the only comparison operator, yielding $F_2[x]$
- Normalize $F_2[x]$ so that atomic formulas have one occurrence of x (at most), yielding $F_3[x]$
- Normalize $F_3[x]$ so that the coefficients of x is 1 (in atomic formulas containing x), yielding $F_4[x']$
- Construct from $F_4[x']$ a quantifier-free formula F_5 which is equivalent to $\exists x.F[x]$

Cooper's procedure - main idea

Consider a formula $\exists x.F[x]$, where F is quantifier free.
Main steps:

- Put $F[x]$ on negation normal form, yielding $F_1[x]$
- Normalize $F_1[x]$ to use $<$ as the only comparison operator, yielding $F_2[x]$
- Normalize $F_2[x]$ so that atomic formulas have one occurrence of x (at most), yielding $F_3[x]$
- Normalize $F_3[x]$ so that the coefficients of x is 1 (in atomic formulas containing x), yielding $F_4[x']$
- Construct from $F_4[x']$ a quantifier-free formula F_5 which is equivalent to $\exists x.F[x]$

Cooper's procedure - main idea

Consider a formula $\exists x.F[x]$, where F is quantifier free.
Main steps:

- Put $F[x]$ on negation normal form, yielding $F_1[x]$
- Normalize $F_1[x]$ to use $<$ as the only comparison operator, yielding $F_2[x]$
- Normalize $F_2[x]$ so that atomic formulas have one occurrence of x (at most), yielding $F_3[x]$
- Normalize $F_3[x]$ so that the coefficients of x is 1 (in atomic formulas containing x), yielding $F_4[x']$
- Construct from $F_4[x']$ a quantifier-free formula F_5 which is equivalent to $\exists x.F[x]$

Negation Normal Form

Input: A quantifier-free formula $F[x]$.

Output: A formula $F_1[x]$, where negation is used on literals only.

Technique: Apply de Morgan's laws

$$\begin{array}{lcl} \neg(F \vee G) & \Longleftrightarrow & \neg F \wedge \neg G \\ \neg(F \wedge G) & \Longleftrightarrow & \neg F \vee \neg G \end{array}$$

from left to right, together with:

$$\begin{array}{lcl} \neg\neg F & \Longleftrightarrow & F \\ \neg\top & \Longleftrightarrow & \bot \\ \neg\bot & \Longleftrightarrow & \top \end{array}$$

until no further application is possible.

Negation Normal Form

Input: A quantifier-free formula $F[x]$.

Output: A formula $F_1[x]$, where negation is used on literals only.

Technique: Apply de Morgan's laws

$$\begin{array}{lcl} \neg(F \vee G) & \iff & \neg F \wedge \neg G \\ \neg(F \wedge G) & \iff & \neg F \vee \neg G \end{array}$$

from left to right, together with:

$$\begin{array}{lcl} \neg\neg F & \iff & F \\ \neg\top & \iff & \bot \\ \neg\bot & \iff & \top \end{array}$$

until no further application is possible.

Construction of $F_2[x]$

Output: A formula $F_2[x]$ containing comparison $<$ only, and where negation is applied to divisibility constraints only.

Technique: Use

$$\begin{array}{lcl} s = t & \iff & s < t + 1 \wedge t < s + 1 \\ \neg(s = t) & \iff & s < t \vee t < s \\ \neg(s < t) & \iff & t < s + 1 \end{array}$$

The other comparisons $\leq, \geq, >$ can also be treated.

Construction of $F_3[x]$

Output: A formula $F_3[x]$, where atomic formulas contain one occurrence of x at most.

Technique: Use linear arithmetic to bring each atomic formula containing x on the form

$$hx < t \qquad \text{or} \qquad t < hx \qquad \text{or} \qquad k | hx + t$$

where $h, k \in \mathbb{Z}^+$ and x does not occur in t.

Example:

$$6x + z < 4x + 3y - 5$$

is transformed to

$$2x < 3y - z - 5$$

Construction of $F_3[x]$

Output: A formula $F_3[x]$, where atomic formulas contain one occurrence of x at most.

Technique: Use linear arithmetic to bring each atomic formula containing x on the form

$$hx < t \qquad \text{or} \qquad t < hx \qquad \text{or} \qquad k | hx + t$$

where $h, k \in \mathbb{Z}^+$ and x does not occur in t.

Example:

$$6x + z < 4x + 3y - 5$$

is transformed to

$$2x < 3y - z - 5$$

Construction of $F_4[x']$

Output: A formula $F_4[x']$, where coefficients to x' are all 1 and

$$\exists x.F[x] \quad \text{is equivalent to} \quad \exists x'.F[x']$$

Let δ be the least common multiple (lcm) of all coefficients to x.

Normalize each constraint so that δ is the coefficient of x. The resulting formula is $F_3'[\delta x]$. $F_4[x']$ is $F_3'[x'] \wedge \delta | x'$

Example:

$$2x < z + 6 \wedge y - 1 < 3x \wedge 4|5x + 1$$

is transformed to $F_3'[30x]$

$$30x < 15z + 90 \ \wedge \ 10y - 10 < 30x \ \wedge \ 24|30x + 6$$

as $30 = \text{lcm}\{2, 3, 5\}$, and $F_4[x']$ is

$$x' < 15z + 90 \ \wedge \ 10y - 10 < x' \ \wedge \ 24|x' + 6 \ \wedge 30|x'$$

Construction of $F_4[x']$

Output: A formula $F_4[x']$, where coefficients to x' are all 1 and

$$\exists x.F[x] \qquad \text{is equivalent to} \qquad \exists x'.F[x']$$

Let δ be the least common multiple (lcm) of all coefficients to x.

Normalize each constraint so that δ is the coefficient of x. The resulting formula is $F_3'[\delta x]$. $F_4[x']$ is $F_3'[x'] \wedge \delta | x'$

Example:

$$2x < z + 6 \wedge y - 1 < 3x \wedge 4|5x + 1$$

is transformed to $F_3'[30x]$

$$30x < 15z + 90 \;\wedge\; 10y - 10 < 30x \;\wedge\; 24|30x + 6$$

as $30 = \mathrm{lcm}\{2,3,5\}$, and $F_4[x']$ is

$$x' < 15z + 90 \;\wedge\; 10y - 10 < x' \;\wedge\; 24|x' + 6 \;\wedge 30|x'$$

Construction of $F_4[x']$

Output: A formula $F_4[x']$, where coefficients to x' are all 1 and

$$\exists x.F[x] \qquad \text{is equivalent to} \qquad \exists x'.F[x']$$

Let δ be the least common multiple (lcm) of all coefficients to x.

Normalize each constraint so that δ is the coefficient of x. The resulting formula is $F_3'[\delta x]$. $F_4[x']$ is $F_3'[x'] \wedge \delta | x'$

Example:

$$2x < z + 6 \wedge y - 1 < 3x \wedge 4 | 5x + 1$$

is transformed to $F_3'[30x]$

$$30x < 15z + 90 \;\wedge\; 10y - 10 < 30x \;\wedge\; 24 | 30x + 6$$

as $30 = \text{lcm}\{2, 3, 5\}$, and $F_4[x']$ is

$$x' < 15z + 90 \;\wedge\; 10y - 10 < x' \;\wedge\; 24 | x' + 6 \;\wedge 30 | x'$$

Construction of $F_4[x']$

Output: A formula $F_4[x']$, where coefficients to x' are all 1 and

$$\exists x.F[x] \qquad \text{is equivalent to} \qquad \exists x'.F[x']$$

Let δ be the least common multiple (lcm) of all coefficients to x.

Normalize each constraint so that δ is the coefficient of x. The resulting formula is $F_3'[\delta x]$. $F_4[x']$ is $F_3'[x'] \wedge \delta | x'$

Example:

$$2x < z + 6 \wedge y - 1 < 3x \wedge 4 | 5x + 1$$

is transformed to $F_3'[30x]$

$$30x < 15z + 90 \ \wedge \ 10y - 10 < 30x \ \wedge \ 24 | 30x + 6$$

as $30 = \text{lcm}\{2, 3, 5\}$, and $F_4[x']$ is

$$x' < 15z + 90 \ \wedge \ 10y - 10 < x' \ \wedge \ 24 | x' + 6 \ \wedge 30 | x'$$

Construction of F_5 – part 1

Output: A quantifier-free formula F_5 which is equivalent to $\exists x.F[x]$ (and to $\exists x'.F_4[x']$)

Each literal in $F_4[x']$ containing x' has one of the forms:
(A) $x' < a$, (B) $b < x'$, (C) $h | x' + c$, (D) $\neg(h | x' + c)$

We distinguish two cases.

Case 1: there are infinitely many small satisfying assignments to x'.

Let $F_{-\infty}[x']$ be obtained from $F_4[x']$ by replacing:

- (A)-literals by $\top$ and
- (B)-literals by $\bot$.

Let
$\delta = \text{lcm}\{h \mid h|x + c$ is a divisibility constraint in a (C) or (D) literal$\}$.

Let F_{51} be

$$\bigvee_{j=1}^{\delta} F_{-\infty}[j]$$

All possible combinations of divisibility constraints are tested.

Construction of F_5 – part 1

Output: A quantifier-free formula F_5 which is equivalent to $\exists x.F[x]$ (and to $\exists x'.F_4[x']$)

Each literal in $F_4[x']$ containing x' has one of the forms:
(A) $x' < a$, (B) $b < x'$, (C) $h|x' + c$, (D) $\neg(h|x' + c)$

We distinguish two cases.

Case 1: there are infinitely many small satisfying assignments to x'.

Let $F_{-\infty}[x']$ be obtained from $F_4[x']$ by replacing:

- (A)-literals by $\top$ and
- (B)-literals by $\bot$.

Let
$\delta = \text{lcm}\{h \mid h|x + c$ is a divisibility constraint in a (C) or (D) literal$\}$.

Let F_{51} be

$$\bigvee_{j=1}^{\delta} F_{-\infty}[j]$$

All possible combinations of divisibility constraints are tested.

Construction of F_5 – part 1

Output: A quantifier-free formula F_5 which is equivalent to $\exists x.F[x]$ (and to $\exists x'.F_4[x']$)

Each literal in $F_4[x']$ containing x' has one of the forms:
(A) $x' < a$, (B) $b < x'$, (C) $h|x' + c$, (D) $\neg(h|x' + c)$

We distinguish two cases.

Case 1: there are infinitely many small satisfying assignments to x'.

Let $F_{-\infty}[x']$ be obtained from $F_4[x']$ by replacing:

- (A)-literals by $\top$ and
- (B)-literals by $\bot$.

Let
$\delta = \text{lcm}\{h \mid h|x + c$ is a divisibility constraint in a (C) or (D) literal$\}$.

Let F_{51} be

$$\bigvee_{j=1}^{\delta} F_{-\infty}[j]$$

All possible combinations of divisibility constraints are tested.

Construction of F_5 – part 1

Output: A quantifier-free formula F_5 which is equivalent to $\exists x.F[x]$ (and to $\exists x'.F_4[x']$)

Each literal in $F_4[x']$ containing x' has one of the forms:
(A) $x' < a$, (B) $b < x'$, (C) $h | x' + c$, (D) $\neg(h | x' + c)$

We distinguish two cases.

Case 1: there are infinitely many small satisfying assignments to x'.

Let $F_{-\infty}[x']$ be obtained from $F_4[x']$ by replacing:

- (A)-literals by $\top$ and
- (B)-literals by $\bot$.

Let
$\delta = \text{lcm}\{h \mid h | x + c$ is a divisibility constraint in a (C) or (D) literal$\}$.

Let F_{51} be

$$\bigvee_{j=1}^{\delta} F_{-\infty}[j]$$

All possible combinations of divisibility constraints are tested.

Construction of F_5 – part 2

Each literal in $F_4[x']$ containing x' has one of the forms:
(A) $x' < a$, (B) $b < x'$, (C) $h | x' + c$, (D) $\neg(h | x' + c)$

Case 2: there is a least satisfying assignments to x'.

For that assignment an (B) literal is true and for smaller assignments to x' the formula is false.

Let $B = \{b | b < x' \text{ is a (B) literal}\}$

Then F_{52} is:

$$\bigvee_{j=1}^{\delta} \bigvee_{b \in B} F_4[b+j]$$

Then F_5 is $F_{51} \vee F_{52}$ i.e.

$$\bigvee_{j=1}^{\delta} F_{-\infty}[j] \vee \bigvee_{j=1}^{\delta} \bigvee_{b \in B} F_4[b+j]$$

Construction of F_5 – part 2

Each literal in $F_4[x']$ containing x' has one of the forms:
(A) $x' < a$, (B) $b < x'$, (C) $h | x' + c$, (D) $\neg(h | x' + c)$

Case 2: there is a least satisfying assignments to x'.

For that assignment an (B) literal is true and for smaller assignments to x' the formula is false.

Let $B = \{b | b < x' \text{ is a (B) literal}\}$

Then F_{52} is:

$$\bigvee_{j=1}^{\delta} \bigvee_{b \in B} F_4[b + j]$$

Then F_5 is $F_{51} \vee F_{52}$ i.e.

$$\bigvee_{j=1}^{\delta} F_{-\infty}[j] \vee \bigvee_{j=1}^{\delta} \bigvee_{b \in B} F_4[b + j]$$

Construction of F_5 – part 2

Each literal in $F_4[x']$ containing x' has one of the forms:
(A) $x' < a$, (B) $b < x'$, (C) $h|x' + c$, (D) $\neg(h|x' + c)$

Case 2: there is a least satisfying assignments to x'.

For that assignment an (B) literal is true and for smaller assignments to x' the formula is false.

Let $B = \{b|b < x' \text{ is a (B) literal}\}$

Then F_{52} is:

$$\bigvee_{j=1}^{\delta} \bigvee_{b \in B} F_4[b + j]$$

Then F_5 is $F_{51} \vee F_{52}$ i.e.

$$\bigvee_{j=1}^{\delta} F_{-\infty}[j] \vee \bigvee_{j=1}^{\delta} \bigvee_{b \in B} F_4[b + j]$$

Example (I)

$$\exists x. \underbrace{(3x + 1 < 10 \ \lor \ 7x - 6 > 7) \ \land \ 2|x}_{F[x]}$$

$F[x]$ is on Negation Normal Form. Isolate x and use $<$ only:

$$\exists x. \underbrace{(3x < 9 \ \lor \ 13 < 7x) \ \land \ 2|x}_{F_3[x]}$$

Normalize coefficient to x, part 1:

$$\exists x. \underbrace{(21x < 63 \ \lor \ 39 < 21x) \ \land \ 42|21x}_{F_3'[21x]}$$

Normalize coefficient to x, part 2:

$$\exists x'. \underbrace{(x' < 63 \ \lor \ 39 < x') \ \land \ 42|x' \ \land \ 21|x'}_{F_4[x']}$$

Example (I)

$$\exists x.\underbrace{(3x+1<10 \ \vee \ 7x-6>7) \ \wedge \ 2|x}_{F[x]}$$

$F[x]$ is on Negation Normal Form. Isolate x and use $<$ only:

$$\exists x.\underbrace{(3x<9 \ \vee \ 13<7x) \ \wedge \ 2|x}_{F_3[x]}$$

Normalize coefficient to x, part 1:

$$\exists x.\underbrace{(21x<63 \ \vee \ 39<21x) \ \wedge \ 42|21x}_{F_3'[21x]}$$

Normalize coefficient to x, part 2:

$$\exists x'.\underbrace{(x'<63 \ \vee \ 39<x') \ \wedge \ 42|x' \ \wedge \ 21|x'}_{F_4[x']}$$

Example (I)

$$\exists x . \underbrace{(3x + 1 < 10 \ \lor \ 7x - 6 > 7) \ \land \ 2|x}_{F[x]}$$

$F[x]$ is on Negation Normal Form. Isolate x and use $<$ only:

$$\exists x . \underbrace{(3x < 9 \ \lor \ 13 < 7x) \ \land \ 2|x}_{F_3[x]}$$

Normalize coefficient to x, part 1:

$$\exists x . \underbrace{(21x < 63 \ \lor \ 39 < 21x) \ \land \ 42|21x}_{F_3'[21x]}$$

Normalize coefficient to x, part 2:

$$\exists x' . \underbrace{(x' < 63 \ \lor \ 39 < x') \ \land \ 42|x' \ \land \ 21|x'}_{F_4[x']}$$

Example (I)

$$\exists x.\underbrace{(3x+1<10 \ \lor \ 7x-6>7) \ \land \ 2|x}_{F[x]}$$

$F[x]$ is on Negation Normal Form. Isolate x and use $<$ only:

$$\exists x.\underbrace{(3x<9 \ \lor \ 13<7x) \ \land \ 2|x}_{F_3[x]}$$

Normalize coefficient to x, part 1:

$$\exists x.\underbrace{(21x<63 \ \lor \ 39<21x) \ \land \ 42|21x}_{F_3'[21x]}$$

Normalize coefficient to x, part 2:

$$\exists x'.\underbrace{(x'<63 \ \lor \ 39<x') \ \land \ 42|x' \ \land \ 21|x'}_{F_4[x']}$$

Example (II)

$$\exists x'.\underbrace{(x' < 63 \lor 39 < x') \land 42|x' \land 21|x'}_{F_4[x']}$$

Eliminate the quantifier:

$$F_{-\infty}[x'] : (\top \lor \bot) \land 42|x' \land 21|x'$$

$$\delta = \text{lcm}\{21, 42\} = 42$$

$$B = \{39\}$$

$$\bigvee_{j=1}^{42}(42|j \land 21|j)$$
$$\lor$$
$$\bigvee_{j=1}^{42}((39+j < 63 \lor 39 < 39+j) \land 42|39+j \land 21|39+j)$$

This formula is *true* and so is

$$\exists x.\underbrace{(3x+1 < 10 \lor 7x-6 > 7) \land 2|x}_{F[x]}$$

Example (II)

$$\exists x'.\underbrace{(x' < 63 \ \vee \ 39 < x') \ \wedge \ 42|x' \ \wedge \ 21|x'}_{F_4[x']}$$

Eliminate the quantifier:

$$F_{-\infty}[x'] : (\top \ \vee \ \bot) \ \wedge \ 42|x' \ \wedge 21|x'$$

$$\delta = \text{lcm}\{21, 42\} = 42$$

$$B = \{39\}$$

$$\bigvee_{j=1}^{42}(42|j \wedge 21|j)$$
$$\vee$$
$$\bigvee_{j=1}^{42}((39+j < 63 \vee 39 < 39+j) \wedge 42|39+j \wedge 21|39+j)$$

This formula is *true* and so is

$$\exists x.\underbrace{(3x+1 < 10 \ \vee \ 7x-6 > 7) \ \wedge \ 2|x}_{F[x]}$$

Example (II)

$$\exists x'. \underbrace{(x' < 63 \ \vee \ 39 < x') \ \wedge \ 42|x' \ \wedge \ 21|x'}_{F_4[x']}$$

Eliminate the quantifier:

$$F_{-\infty}[x'] : (\top \ \vee \ \bot) \ \wedge \ 42|x' \ \wedge 21|x'$$

$$\delta = \text{lcm}\{21, 42\} = 42$$

$$B = \{39\}$$

$$\bigvee_{j=1}^{42}(42|j \wedge 21|j)$$
$$\vee$$
$$\bigvee_{j=1}^{42}((39 + j < 63 \vee 39 < 39 + j) \wedge 42|39 + j \wedge 21|39 + j)$$

This formula is *true* and so is

$$\exists x. \underbrace{(3x + 1 < 10 \ \vee \ 7x - 6 > 7) \ \wedge \ 2|x}_{F[x]}$$

Summary (I)

- Cooper's algorithm can decide arbitrary formulas of Presbruger Arithmetic – even in the presence of arbitrary quantifications.
- The problem has a double exponential lower bound and a triple exponential upper bound.
- Cooper's algorithm has a triple exponential upper bound.
- Many optimizations are possible.

Summary (I)

- Cooper's algorithm can decide arbitrary formulas of Presbruger Arithmetic – even in the presence of arbitrary quantifications.
- The problem has a double exponential lower bound and a triple exponential upper bound.
- Cooper's algorithm has a triple exponential upper bound.
- Many optimizations are possible.

Summary (I)

- Cooper's algorithm can decide arbitrary formulas of Presbruger Arithmetic – even in the presence of arbitrary quantifications.
- The problem has a double exponential lower bound and a triple exponential upper bound.
- Cooper's algorithm has a triple exponential upper bound.
- Many optimizations are possible.

Summary (I)

- Cooper's algorithm can decide arbitrary formulas of Presbruger Arithmetic – even in the presence of arbitrary quantifications.
- The problem has a double exponential lower bound and a triple exponential upper bound.
- Cooper's algorithm has a triple exponential upper bound.
- Many optimizations are possible.

Summary (II)

The advantage with Cooper's algorithm is that it does not require normal form, as some other decision methods do. Quantifier elimination in connection with DNF or CNF hurts a lot.

A disadvantage with Cooper's algorithm is that constants obtained using lcm may be large.

Ongoing work: Experiments with a declarative implementation of the algorithm including many optimizations aiming at:

- including techniques from other decision methods, and
- a parallel implementation on a multi-core platform

striving for a very efficient backend for DC-modelchecking.

Summary (II)

The advantage with Cooper's algorithm is that it does not require normal form, as some other decision methods do. Quantifier elimination in connection with DNF or CNF hurts a lot.

A disadvantage with Cooper's algorithm is that constants obtained using lcm may be large.

Ongoing work: Experiments with a declarative implementation of the algorithm including many optimizations aiming at:

- including techniques from other decision methods, and
- a parallel implementation on a multi-core platform

striving for a very efficient backend for DC-modelchecking.

Summary (II)

The advantage with Cooper's algorithm is that it does not require normal form, as some other decision methods do. Quantifier elimination in connection with DNF or CNF hurts a lot.

A disadvantage with Cooper's algorithm is that constants obtained using lcm may be large.

Ongoing work: Experiments with a declarative implementation of the algorithm including many optimizations aiming at:

- including techniques from other decision methods, and
- a parallel implementation on a multi-core platform

striving for a very efficient backend for DC-modelchecking.