Quantifier Elimination for Presburger Arithmetic

Viktor Kunčak

Formal Verification Methodology

We can view formal verification of program as a three-step approach:

1. Express properties in logic or code
   (assertions, preconditions, post-conditions, invariants, run-time error conditions)
2. Compile program meaning to **logical formulas**
   (verification-condition generator, symbolic execution)
3. Develop and use an automated theorem prover for generated conditions
   (SAT and SMT solving, resolution-based theorem proving, proof assistants)

Which logic to use? Today: integer linear arithmetic (Presburger arithmetic)

References:

- Haase, Christoph. A survival guide to presburger arithmetic. ACM SIGLOG News. `https://dl.acm.org/doi/10.1145/3242953.3242964`.
- `https://en.wikipedia.org/wiki/Presburger_arithmetic`
- The Calculus of Computation (Bradley, Manna, Springer-Verlag 2007), Section 7.2

Presburger arithmetic

Integer arithmetic with logical operations (and, or, not), quantifiers, only addition as an arithmetic operation, and $<$ and $=$ as a relation.

- minimalistically one can define a variant of it over non-negative natural numbers as having $\wedge, \neg, \forall, +, =$ as the only symbols

One of the earliest theories shown decidable. Mojżesz Presburger gave an algorithm for quantifier elimination in 1929.

- a student of an influential logician and model theorist Alfred Tarski
- Tarski assigned this question to Mojżesz for his MSc thesis and he solved it

The result at this time was of interest to foundations of mathematics (giving algorithmic meaning to quantifiers).
Subsequently, it found applications in automated reasoning, including building first program verifiers (Cooper 1972, Derek C. Oppen - STOC 1973).

Presburger Arithmetic for Verification

```
res = 0
i = x
while // invariant I(res,i): res + 2*i == 2*x && 0 <= i
(i > 0) {
    i = i - 1
    res = res + 2
}
```

Verification condition (VC) for preservation of loop invariant:

$$\big[I(res, i) \wedge i' = i - 1 \wedge res' = res + 2 \wedge 0 < i\big] \rightarrow I(res', i')$$

To prove that this VC is valid, we check whether its *negation*

$$I(res, i) \wedge i' = i - 1 \wedge res' = res + 2 \wedge 0 < i \wedge \neg I(res', i')$$

is *satisfiable*, i.e. whether this PA formula is true:

$$\exists x, res, i, res', i'.\ \big[res + 2i = 2x\ \wedge\ 0 \leq i\ \wedge\ 0 < i\ \wedge \\ i' = i - 1\ \wedge\ res' = res + 2\ \wedge\ \neg(res' + 2i' = 2x\ \wedge\ 0 \leq i')\big]$$

Introducing: One-Point Rule

If $\bar{y}$ is a tuple of variables not containing x, then

$$\exists x.(x = t(\bar{y}) \wedge F(x, \bar{y})) \iff F(t(\bar{y}), \bar{y})$$

Proof:

- $\rightarrow$: Consider the values of $\bar{y}$ such that there exists x, say x_1, for which $x_1 = t(\bar{y}) \wedge F(x_1, \bar{y})$. Because $F(x_1, \bar{y})$ evaluates to true and the values of x_1 and $t(\bar{y})$ are the same, $F(t, \bar{y})$ also evaluates to true.
- $\leftarrow$: Let $\bar{y}$ be such that $F(t, \bar{y})$ holds. Let x be the value of $t(\bar{y})$. Then of course $x = t(\bar{y})$ evaluates to true and so does $F(x, \bar{y})$. So there exists x for which $x = t(\bar{y}) \wedge F(x, \bar{y})$ holds.

One point rule:
replaces left side (LHS) of equivalence by the right side (RHS).
Flattening, used when t is complex, replaces RHS by LHS.

Dual One-Point Rule for $\forall$

$$\forall x.(x = t(\bar{y}) \rightarrow F(x, \bar{y})) \iff F(t(\bar{y}), \bar{y})$$

To prove it, negate both sides:

$$\exists x.(x = t(\bar{y}) \wedge \neg F(x, \bar{y})) \iff \neg F(t(\bar{y}), \bar{y})$$

so it reduces to the rule for $\exists$.

Using One-Point Rule on Negated Verification Condition

$$\exists x, res, i, res', \underline{\mathbf{i'}}.\ \big[res + 2i = 2x\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge$$
$$\underline{\mathbf{i' = i - 1}}\ \wedge\ res' = res + 2\ \wedge$$
$$\neg(res' + 2i' = 2x\ \wedge\ 0 \le i')\big]$$

Using One-Point Rule on Negated Verification Condition

$$\exists x, res, i, res', \underline{\mathbf{i'}}.\ \big[res + 2i = 2x\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \underline{\mathbf{i' = i - 1}}\ \wedge\ res' = res + 2\ \wedge\ \neg(res' + 2i' = 2x\ \wedge\ 0 \le i')\big]$$

$$\exists x, res, i, \underline{\mathbf{res'}}.\ \big[res + 2i = 2x\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \underline{\mathbf{res' = res + 2}}\ \wedge\ \neg(res' + 2(i-1) = 2x\ \wedge\ 0 \le i-1)\big]$$

Using One-Point Rule on Negated Verification Condition

$$\exists x, res, i, res', \underline{\mathbf{i'}}.\ \big[res + 2i = 2x\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \underline{\mathbf{i' = i - 1}}\ \wedge\ res' = res + 2\ \wedge\ \neg(res' + 2i' = 2x\ \wedge\ 0 \le i')\big]$$

$$\exists x, res, i, \underline{\mathbf{res'}}.\ \big[res + 2i = 2x\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \underline{\mathbf{res' = res + 2}}\ \wedge\ \neg(res' + 2(i-1) = 2x\ \wedge\ 0 \le i-1)\big]$$

$$\exists x, res, i.\ \big[\underline{res + 2i = 2x}\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \neg(res + 2 + 2(i-1) = 2x\ \wedge\ 0 \le i-1)\big]$$

Using One-Point Rule on Negated Verification Condition

$$\exists x, res, i, res', \underline{\mathbf{i'}}.\ \big[res + 2i = 2x\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \underline{\mathbf{i' = i - 1}}\ \wedge\ res' = res + 2\ \wedge\ \neg(res' + 2i' = 2x\ \wedge\ 0 \le i')\big]$$

$$\exists x, res, i, \underline{\mathbf{res'}}.\ \big[res + 2i = 2x\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \underline{\mathbf{res' = res + 2}}\ \wedge\ \neg(res' + 2(i-1) = 2x\ \wedge\ 0 \le i-1)\big]$$

$$\exists x, res, i.\ \big[\underline{res + 2i = 2x}\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \neg(res + 2 + 2(i-1) = 2x\ \wedge\ 0 \le i-1)\big]$$

$$\exists x, \underline{\mathbf{res}}, i.\ \big[\underline{\mathbf{res = 2x - 2i}}\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \neg(res + 2 + 2(i-1) = 2x\ \wedge\ 0 \le i-1)\big]$$

Using One-Point Rule on Negated Verification Condition

$$\exists x, res, i, res', \underline{\mathbf{i'}}.\ \big[res + 2i = 2x\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \underline{\mathbf{i' = i - 1}}\ \wedge\ res' = res + 2\ \wedge\ \neg(res' + 2i' = 2x\ \wedge\ 0 \le i')\big]$$

$$\exists x, res, i, \underline{\mathbf{res'}}.\ \big[res + 2i = 2x\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \underline{\mathbf{res' = res + 2}}\ \wedge\ \neg(res' + 2(i-1) = 2x\ \wedge\ 0 \le i-1)\big]$$

$$\exists x, res, i.\ \big[\underline{res + 2i = 2x}\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \neg(res + 2 + 2(i-1) = 2x\ \wedge\ 0 \le i-1)\big]$$

$$\exists x, \underline{\mathbf{res}}, i.\ \big[\underline{\mathbf{res = 2x - 2i}}\ \wedge\ 0 \le i\ \wedge\ 0 < i\ \wedge\ \neg(res + 2 + 2(i-1) = 2x\ \wedge\ 0 \le i-1)\big]$$

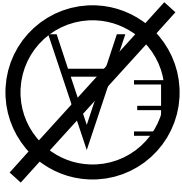
$$\exists x, i.\ \big[0 \le i\ \wedge\ 0 < i\ \wedge\ \neg(2x - 2i + 2 + 2(i-1) = 2x\ \wedge\ 0 \le i-1)\big]$$

Simplifies to $\exists x, i.\ 0 < i \wedge \neg(0 \le i-1)$ and then to false.

But there is more

One-point rule is one of the many steps used in **quantifier elimination** procedures.

Quantifier Elimination (QE)



Given a formula $F(\bar{y})$ containing quantifiers find a formula $G(\bar{y})$

- **equivalent** to $F(\bar{y})$
- that has **no quantifiers**
- and has a **subset (or equal set) of free variables** of F

Note

- Equivalence: For all $\bar{y}$, $F(\bar{y})$ and $G(\bar{y})$ have same truth value
  $\rightsquigarrow$ we can use $G(\bar{y})$ instead of $F(\bar{y})$
- No quantifiers: easier to check satisfiability of $G(\bar{y})$

$\bar{y}$ is a possibly empty tuple of variables

We are lucky when a theory has ("admits") QE

Suppose F has no free variables (all variables are quantified).
What is the result of applying QE to F?

We are lucky when a theory has ("admits") QE

Suppose F has no free variables (all variables are quantified).
What is the result of applying QE to F?
Are there any variables in the resulting formula?

We are lucky when a theory has ("admits") QE

Suppose F has no free variables (all variables are quantified).
What is the result of applying QE to F?
Are there any variables in the resulting formula?

- No free variables: they are a subset of the original, empty set
- No quantified variables: because it has no quantifiers ☺

Formula without any variables! Example:

$$(2+4=7) \vee (1+1=2)$$

We check the truth value of such formula by simply evaluating it!

Using QE for Deciding Satisfiability/Validity

- To check satisfiability of $H(\bar{y})$: eliminate the quantifiers from $\exists \bar{y}.H(\bar{y})$ and evaluate.
- Validity: eliminate quantifiers from $\forall \bar{y}.H(\bar{y})$ and evaluate

We can even check formulas like this:

$$\forall x, y, r.\ \exists z.\ (5 \leq r \wedge x + r \leq y) \rightarrow (x < z \wedge z < y \wedge 3|z)$$

Here $3|z$ denotes that z is divisible by 3.

Does Presburger Arithmetic admit QE?

Does Presburger Arithmetic admit QE?

Depends on the particular set of symbols!
(Recall objective: given $F(\bar{y})$ containing quantifiers find a formula $G(\bar{y})$

- **equivalent** to $F(\bar{y})$
- that has **no quantifiers**
- and has a **subset (or equal set) of free variables** of F)

If we lack some operations that can be expressed using quantifiers, there may be no equivalent formula without quantifiers.

- $\exists y.x = y + y + y$, so we better have divisibility

Quantifier elimination says: if you can define some relationship between variables using an arbitrary, possibly quantified, formula F,

$$r \stackrel{def}{=} \{(x, y) \mid F(x, y)\}$$

then you can also define same r using another quantifier-free formula G.

Presburger Arithmetic (PA)

We look at the theory of integers with addition.

- introduce constant for each integer constant
- to be able to restrict values to natural numbers when needed, and to compare them, we introduce $<$
- introduce not only addition but also subtraction
- to conveniently express certain expressions, introduce function m_K for each $K \in \mathscr{Z}$, to be interpreted as multiplication by a constant, $m_K(x) = K \cdot x$. We write m_K as $K \cdot x$.
  Note: there is *no multiplication between variables* in PA
- to enable quantifier elimination from $\exists x.y = K \cdot x$ introduce for each K predicate $K|y$ (divisibility, $y\%K = 0$)

The resulting language has these function and relation symbols:
$\{+,-,=,<\} \cup \{K \mid K \in \mathbb{Z}\} \cup \{(K \cdot \_) \mid K \in \mathbb{Z}\} \cup \{(K|\_) \mid K \in \mathbb{Z}\}$ We also have, as usual:
$\wedge, \vee, \neg, \rightarrow$ and also: $\exists, \forall$

Example

Eliminate y from this formula:

$$\exists y.\ \ 3y-2w+1>-w \wedge 2y-6<z \wedge 4 \mid 5y+1$$

What should we do first?

Example

Eliminate y from this formula:

$$\exists y.\ \ 3y-2w+1>-w \wedge 2y-6<z \wedge 4 \mid 5y+1$$

What should we do first?
Simplify/normalize what we can using properties of integer operations:

$$\exists y.\ \ 0<-w+3y+1 \ \ \wedge \ \ 0<-2y+z+6 \ \wedge \ 4 \mid 5y+1$$

Example

Eliminate y from this formula:

$$\exists y.\ \ 3y-2w+1>-w \wedge 2y-6<z \wedge 4 \mid 5y+1$$

What should we do first?
Simplify/normalize what we can using properties of integer operations:

$$\exists y.\ \ 0<-w+3y+1 \ \ \wedge \ \ 0<-2y+z+6 \ \wedge \ 4 \mid 5y+1$$

First we will consider only eliminating existential from a **conjunction of literals**.

Conjunctions of Literals

Atomic formula: a relation applied to argument.
Here, relations are: $=$, $<$, $K|\_$. So, atomic formulas are:
$t_1 = t_2, \quad t_1 < t_2, \quad K \mid t$

Conjunctions of Literals

Atomic formula: a relation applied to argument.
Here, relations are: $=$, $<$, $K|\_$. So, atomic formulas are:
$t_1 = t_2, \quad t_1 < t_2, \quad K \mid t$
Literal: Atomic formula or its negation. Example: $\neg(x = y + 1)$
Conjunction of literals: $L_1 \wedge \ldots \wedge L_n$

- no disjunctions, no implications
- negation only applies to atomic formulas

We first consider the quantifier elimination problem of the form:

$$\exists y.\ L_1 \wedge \ldots \wedge L_n$$

This will prove to be sufficient to eliminate all quantifiers!

Eliminating $\exists$ from conjunction of literals suffices

Can we eliminate $\exists$ from any **quantifier-free formula**?

$$\exists x.F(x,\bar{y})$$

where F is quantifier-free?

Eliminating $\exists$ from conjunction of literals suffices

Can we eliminate $\exists$ from any **quantifier-free formula**?

$$\exists x.F(x,\bar{y})$$

where F is quantifier-free?
Formula without quantifiers has $\wedge,\vee,\neg$ applied to atomic formulas.

Eliminating $\exists$ from conjunction of literals suffices

Can we eliminate $\exists$ from any **quantifier-free formula**?

$$\exists x.F(x,\bar{y})$$

where F is quantifier-free?
Formula without quantifiers has $\wedge,\vee,\neg$ applied to atomic formulas.
Convert F to **disjunctive normal form**:

$$F \iff \bigvee_{i=1}^{m} C_i$$

each C_i is a **conjunction of literals**.

Eliminating $\exists$ from conjunction of literals suffices

Can we eliminate $\exists$ from any **quantifier-free formula**?

$$\exists x.F(x,\bar{y})$$

where F is quantifier-free?
Formula without quantifiers has $\wedge,\vee,\neg$ applied to atomic formulas.
Convert F to **disjunctive normal form**:

$$F \iff \bigvee_{i=1}^{m} C_i$$

each C_i is a **conjunction of literals**.

$$\left[\exists x. \bigvee_{i=1}^{m} C_i\right] \iff \bigvee_{i=1}^{m} (\exists x. C_i)$$

How does disjunctive normal form (DNF) transformation work?

Which steps should we use?

How does disjunctive normal form (DNF) transformation work?

Which steps should we use?

Negation propagation:

$$\neg(p \wedge q) \rightsquigarrow (\neg p) \vee (\neg q)$$

$$\neg(p \vee q) \rightsquigarrow (\neg p) \wedge (\neg q)$$

$$\neg\neg p \rightsquigarrow p$$

Result is **negation-normal form**, NNF

NNF transformation is polynomial (exercise!)

How does disjunctive normal form (DNF) transformation work?

Which steps should we use?

Negation propagation:

$$\neg(p \wedge q) \rightsquigarrow (\neg p) \vee (\neg q)$$

$$\neg(p \vee q) \rightsquigarrow (\neg p) \wedge (\neg q)$$

$$\neg\neg p \rightsquigarrow p$$

Result is **negation-normal form**, NNF

NNF transformation is polynomial (exercise!)

Distributivity

$$a \wedge (b_1 \vee b_2) \rightsquigarrow (a \wedge b_1) \vee (a \wedge b_2)$$

This can lead to exponential explosion.

Can we obtain *equivalent* DNF formula without explosion?

How does disjunctive normal form (DNF) transformation work?

Which steps should we use?

Negation propagation:

$$\neg(p \wedge q) \rightsquigarrow (\neg p) \vee (\neg q)$$

$$\neg(p \vee q) \rightsquigarrow (\neg p) \wedge (\neg q)$$

$$\neg\neg p \rightsquigarrow p$$

Result is **negation-normal form**, NNF

NNF transformation is polynomial (exercise!)

Distributivity

$$a \wedge (b_1 \vee b_2) \rightsquigarrow (a \wedge b_1) \vee (a \wedge b_2)$$

This can lead to exponential explosion.

Can we obtain *equivalent* DNF formula without explosion?

No! We can prove this (no equivalent DNF formula exists), unrelated to NP vs P

Eliminating from quantifier free formulas

$$\exists x.F \iff \left[\exists x. \bigvee_{i=1}^{m} C_i\right] \iff \bigvee_{i=1}^{m} (\exists x. C_i)$$

Nested Existential Quantifiers

$$\exists x_1.\exists x_2.\underline{\exists \mathbf{x_3}}.F_0(x_1, x_2, x_3, \bar{y})$$

Nested Existential Quantifiers

$$\exists x_1.\exists x_2.\underline{\exists \mathbf{x_3}}.F_0(x_1, x_2, x_3, \bar{y})$$

$$\exists x_1.\underline{\exists \mathbf{x_2}}.F_1(x_1, x_2, \bar{y})$$

Nested Existential Quantifiers

$$\exists x_1.\exists x_2.\underline{\exists \mathbf{x_3}}.F_0(x_1, x_2, x_3, \bar{y})$$

$$\exists x_1.\underline{\exists \mathbf{x_2}}.F_1(x_1, x_2, \bar{y})$$

$$\underline{\exists \mathbf{x_1}}.F_2(x_1, \bar{y})$$

Nested Existential Quantifiers

$$\exists x_1.\exists x_2.\underline{\exists \mathbf{x_3}}.F_0(x_1, x_2, x_3, \bar{y})$$

$$\exists x_1.\underline{\exists \mathbf{x_2}}.F_1(x_1, x_2, \bar{y})$$

$$\underline{\exists \mathbf{x_1}}.F_2(x_1, \bar{y})$$

$$F_3(\bar{y})$$

Universal Quantifiers

If $F_0(x, \bar{y})$ is quantifier-free, how to eliminate

$$\forall x. F_0(x, \bar{y})$$

Universal Quantifiers

If $F_0(x, \bar{y})$ is quantifier-free, how to eliminate

$$\forall x. F_0(x, \bar{y})$$

Note this equivalence (F_0 universally holds if there is no counterexample):

$$\forall x. F_0(x, \bar{y}) \iff \neg\big[\exists x. \neg F_0(x, \bar{y})\big]$$

It thus suffices to handle:

$$\neg\big[\exists x. \neg F_0(x, \bar{y})\big]$$

Note that $\neg F_0(x, \bar{y})$ is quantifier-free, so we know how to eliminate $\exists x. \neg F_0(x, \bar{y})$:

$$\exists x. \neg F_0(x, \bar{y}) \quad \rightsquigarrow \quad F_1(\bar{y})$$

Then the result of the elimination is the quantifier-free formula:

$$\neg F_1(\bar{y})$$

Removing any **alternation** of quantifiers: illustration

Alternation: switch between existentials and universals

$$\exists x_1.\forall x_2.\forall x_3.\exists x_4.F_0(x_1,x_2,x_3,x_4,\bar{y})$$

$$\exists x_1.\neg\exists x_2.\exists x_3.\neg\underline{\exists x_4.F_0(x_1,x_2,x_3,x_4,\bar{y})}$$

$$\exists x_1.\neg\exists x_2.\underline{\exists x_3.\neg F_1(x_1,x_2,x_3,\bar{y})}$$

$$\exists x_1.\neg\underline{\exists x_2.F_2(x_1,x_2,\bar{y})}$$

$$\exists x_1.\neg F_3(x_1,\bar{y})$$

$$F_4(\bar{y})$$

Each quantifier alternation involves a disjunctive normal form transformation.
In practice, we do not have many alternations.

Back to Presburger Arithmetic

Consider the quantifier elimination problem of the form:

$$\exists y.\ L_1 \wedge \ldots \wedge L_n$$

where L_i are literals from PA.
Note that, for integers:

- $\neg(x < y) \iff y \leq x$
- $x < y \iff x + 1 \leq y$
- $x \leq y \iff x < y + 1$

We use these observations below.
Instead of $\leq$ we choose to use $<$ only.
We do not write $x > y$ but only $y < x$.

Normalizing Literals for PA

Normal Form of Terms: All *terms* are built from $K, +, -, K \cdot \_$, so using standard transformations they can be represented as: $K_0 + \sum_{i=1}^{n} K_i x_i$ We call such term a linear term.

Normal Form for Literals in PA:

$$
\begin{aligned}
\neg(t_1 < t_2) &\text{ becomes } t_2 < t_1 + 1 \\
\neg(t_1 = t_2) &\text{ becomes } t_1 < t_2 \vee t_2 < t_1 \\
t_1 = t_2 &\text{ becomes } t_1 < t_2 + 1 \wedge t_2 < t_1 + 1 \quad (*) \\
\neg(K \mid t) &\text{ becomes } \bigvee_{i=1}^{K-1} K \mid t + i \\
t_1 < t_2 &\text{ becomes } 0 < t_2 - t_1
\end{aligned}
$$

To remove disjunctions we generated, compute DNF again.

$(*)$ We transformed equalities just for simplicity. Usually we handle them directly.

Why one-point rule will not be enough

Note that we must handle inequalities, not merely equalities

If we have integers, we cannot always divide perfectly.
Variable to eliminate can occur not as y but as, e.g. $3y$

Exposing the Variable to Eliminate: Example

$$\exists y.\ \ 0 < -w + \underline{\mathbf{3y}} + 1 \ \ \wedge \ \ 0 < -\underline{\mathbf{2y}} + z + 6 \ \wedge \ 4 \mid \underline{\mathbf{5y}} + 1$$

Least common multiple of coefficients next to y, $M = lcm(3,2,5) = 30$
Make all occurrences of y in the body have this coefficient:

$$\exists y.\ \ 0 < -10w + \underline{\mathbf{30y}} + 10 \wedge 0 < -\underline{\mathbf{30y}} + 15z + 90 \wedge 24 \mid \underline{\mathbf{30y}} + 6$$

Now we are quantifying over y and using $30y$ everywhere.
Let x denote $30y$.
It is **not an arbitrary** x. It is divisible by 30.

$$\exists x.\ \ 0 < -10w + x + 10 \wedge \ 0 < -x + 15z + 90 \ \wedge \ 24 \mid x + 6 \ \wedge \ 30 \mid x$$

Exposing the Variable to Eliminate in General

Eliminating y from conjunction $F(y)$ of literals:

- $0 < t$
- $K \mid t$

where t is a linear term. To eliminate $\exists y$ from such conjunction, we wish to ensure that the coefficient next to y is one or minus one.
Observation:

- $0 < t$ is equivalent to $0 < c\,t$
- $K \mid t$ is equivalent to $c\,K \mid c\,t$

for c a positive integer.
Let $K_1, \ldots, K_n$ be all coefficients next to y in the formula.
Let M be a positive integer such that $K_i \mid M$ for all i, $1 \leq i \leq n$

- for example, let M be the **least common multiple**

$$M = lcm(K_1, \ldots, K_n)$$

Ensuring Coefficient One

Multiply each literal where y occurs in subterm $K_i y$ by constant $M/|K_i|$

- the point is, M is divisible by $|K_i|$ by construction

What is the coefficient next to y in the resulting formula?

Ensuring Coefficient One

Multiply each literal where y occurs in subterm $K_i y$ by constant $M/|K_i|$

- the point is, M is divisible by $|K_i|$ by construction

What is the coefficient next to y in the resulting formula?

$$M \text{ or } -M$$

Ensuring Coefficient One

Multiply each literal where y occurs in subterm $K_i y$ by constant $M/|K_i|$

- the point is, M is divisible by $|K_i|$ by construction

What is the coefficient next to y in the resulting formula?

$$M \text{ or } -M$$

We obtain a formula of the form $\exists y.F(M \cdot y)$.
Letting $x = My$, we conclude the formula is equivalent to

$$\exists x.\ F(x) \wedge (M \mid x)$$

What is the coefficient next to y in the resulting formula?

Ensuring Coefficient One

Multiply each literal where y occurs in subterm $K_i y$ by constant $M/|K_i|$

- the point is, M is divisible by $|K_i|$ by construction

What is the coefficient next to y in the resulting formula?

$$M \text{ or } -M$$

We obtain a formula of the form $\exists y.F(M \cdot y)$.
Letting $x = My$, we conclude the formula is equivalent to

$$\exists x.\ F(x) \wedge (M \mid x)$$

What is the coefficient next to y in the resulting formula?

$$1 \text{ or } -1$$

Exposing the Variable to Eliminate: Example

$$\exists y.\ \ 0 < -w + \underline{\mathbf{3y}} + 1 \ \ \wedge \ \ 0 < -\underline{\mathbf{2y}} + z + 6 \ \wedge \ 4 \mid \underline{\mathbf{5y}} + 1$$

Least common multiple of coefficients next to y, $M = lcm(3,2,5) = 30$
Make all occurrences of y in the body have this coefficient:

$$\exists y.\ \ 0 < -10w + \underline{\mathbf{30y}} + 10 \wedge 0 < -\underline{\mathbf{30y}} + 15z + 90 \wedge 24 \mid \underline{\mathbf{30y}} + 6$$

Now we are quantifying over y and using $30y$ everywhere.
Let x denote $30y$.
It is **not an arbitrary** x. It is divisible by 30.

$$\exists x.\ \ 0 < -10w + x + 10 \wedge \ 0 < -x + 15z + 90 \ \wedge \ 24 \mid x + 6 \ \wedge \ 30 \mid x$$

Lower and upper bounds:

Consider the coefficient next to x in $0 < t$. If it is -1, move the term to left side. If it is 1, move the remaining terms to the left side. We obtain formula $F_1(x)$ of the form

$$\bigwedge_{i=1}^{L} a_i < x \wedge \bigwedge_{j=1}^{U} x < b_j \wedge \bigwedge_{i=1}^{D} K_i \mid (x + t_i)$$

Lower and upper bounds:

Consider the coefficient next to x in $0 < t$. If it is -1, move the term to left side. If it is 1, move the remaining terms to the left side. We obtain formula $F_1(x)$ of the form

$$\bigwedge_{i=1}^{L} a_i < x \wedge \bigwedge_{j=1}^{U} x < b_j \wedge \bigwedge_{i=1}^{D} K_i \mid (x + t_i)$$

If there are no divisibility constraints ($D = 0$), what is the formula equivalent to?

Lower and upper bounds:

Consider the coefficient next to x in $0 < t$. If it is -1, move the term to left side. If it is 1, move the remaining terms to the left side. We obtain formula $F_1(x)$ of the form

$$\bigwedge_{i=1}^{L} a_i < x \wedge \bigwedge_{j=1}^{U} x < b_j \wedge \bigwedge_{i=1}^{D} K_i \mid (x + t_i)$$

If there are no divisibility constraints ($D = 0$), what is the formula equivalent to?

$$\max_i a_i + 1 \leq \min_j b_j - 1 \text{ which is equivalent to } \bigwedge_{ij} a_i + 1 < b_j$$

Replacing variable by test terms

There is a an alternative way to express the above condition by replacing $F_1(x)$ with $\bigvee_k F_1(t_k)$ where t_k do not contain x. This is a common technique in quantifier elimination. Note that if $F_1(t_k)$ holds then certainly $\exists x.F_1(x)$.
What are example terms t_i when $D = 0$ and $L > 0$? Hint: ensure that at least one of them evaluates to $\max a_i + 1$.

$$\bigvee_{k=1}^{L} F_1(a_k + 1)$$

What if $D > 0$ i.e. we have additional divisibility constraints?

$$\bigvee_{k=1}^{L} \bigvee_{i=1}^{N} F_1(a_k + i)$$

What is N?

Replacing variable by test terms

There is a an alternative way to express the above condition by replacing $F_1(x)$ with $\bigvee_k F_1(t_k)$ where t_k do not contain x. This is a common technique in quantifier elimination. Note that if $F_1(t_k)$ holds then certainly $\exists x.F_1(x)$.
What are example terms t_i when $D = 0$ and $L > 0$? Hint: ensure that at least one of them evaluates to $\max a_i + 1$.

$$\bigvee_{k=1}^{L} F_1(a_k + 1)$$

What if $D > 0$ i.e. we have additional divisibility constraints?

$$\bigvee_{k=1}^{L} \bigvee_{i=1}^{N} F_1(a_k + i)$$

What is N? least common multiple of $K_1, \ldots, K_D$

Replacing variable by test terms

There is a an alternative way to express the above condition by replacing $F_1(x)$ with $\bigvee_k F_1(t_k)$ where t_k do not contain x. This is a common technique in quantifier elimination. Note that if $F_1(t_k)$ holds then certainly $\exists x.F_1(x)$.
What are example terms t_i when $D=0$ and $L>0$? Hint: ensure that at least one of them evaluates to $\max a_i + 1$.

$$\bigvee_{k=1}^{L} F_1(a_k+1)$$

What if $D>0$ i.e. we have additional divisibility constraints?

$$\bigvee_{k=1}^{L} \bigvee_{i=1}^{N} F_1(a_k+i)$$

What is N? least common multiple of $K_1,\ldots,K_D$
Correctness: note that if $F_1(u)$ holds then also $F_1(u-N)$ holds.

Back to Example

$$\exists x.\ -10+10w<x\ \wedge\ x<90+15z\ \wedge\ 24\,|\,x+6\ \wedge\ 30\,|\,x$$

Back to Example

$$\exists x.\ -10+10w<x\ \wedge\ x<90+15z\ \wedge\ 24\,|\,x+6\ \wedge\ 30\,|\,x$$

$$\bigvee_{i=1}^{120} 10w+i<100+15z\ \wedge\ 24\,|\,10w-4+i\ \wedge\ 30\,|\,10w-10+i$$

No lower bounds

Now consider the case $L = 0$. We cannot try all lower bounds, as there are none:

$$\bigwedge_{j=1}^{U} x < b_j \wedge \bigwedge_{i=1}^{D} K_i \mid (x + t_i)$$

No lower bounds

Now consider the case $L = 0$. We cannot try all lower bounds, as there are none:

$$\bigwedge_{j=1}^{U} x < b_j \wedge \bigwedge_{i=1}^{D} K_i \mid (x + t_i)$$

We first drop all constraints except divisibility, obtaining $F_2(x)$:

$$\bigwedge_{i=1}^{D} K_i \mid (x + t_i)$$

and then eliminate quantifier as

$$\bigvee_{i=1}^{N} F_2(i)$$

Exercise: prove equivalence of this result with $\exists x.F_1(x)$ in this case.

An Example with No Lower Bounds

Eliminate quantifier from this formula:

$$\exists x.\ x < 5 + 7z \ \wedge\ 2 \mid x \ \wedge\ 3 \mid x + 2$$

Wrap Up

This completes the description of a quantifier elimination algorithm for Presburger Arithmetic (PA).

Wrap Up

This completes the description of a quantifier elimination algorithm for Presburger Arithmetic (PA).
This algorithm and its correctness prove that:

- PA admits quantifier elimination
- Satisfiability, validity, entailment, equivalence of PA formulas is decidable
  We can use the algorithm to prove verification conditions.
  Even if not the most efficient way, it gives us insights on which we can later build to come up with better algorithms.
- Quantified and quantifier-free formulas have the same expressive power

Approaches to Making QE for PA More Efficient in Practice

Avoid transforming to conjunctions of literals: work directly on negation-normal form.
The technique is similar to what we described for conjunctive normal form.

- \+ no need for DNF
- \- we may end up trying irrelevant bounds

This is the Cooper's algorithm:

- Reddy, Loveland: Presburger Arithmetic with Bounded Quantifier Alternation. (Gives a slight improvement of the original Cooper's algorithm.)
- Section 7.2 of the Calculus of Computation Textbook

An alternative direction for improvement: handle a system of equalities more efficiently, without introducing inequalities and divisibility constraints too eagerly, using Euclid's GCD algorithm for solving linear Diophantine equations.
(Hermite normal form of a matrix.)
Weak quantifier elimination: only eliminate alternations of quantifiers.

Complexity of Deciding Quantified PA Formulas

Regardless how we proceed, we cannot escape inherent computational complexity of deciding if quantified Presburger arithmetic formula is true, which has been characterized using alternating complexity classes by Berman.

Complexity of Deciding Quantified PA Formulas

Regardless how we proceed, we cannot escape inherent computational complexity of deciding if quantified Presburger arithmetic formula is true, which has been characterized using alternating complexity classes by Berman.

A set A is in $STA(s(n), t(n), a(n))$ iff there exists a single-tape *alternating* Turing machine M_i (making $\wedge$ as well as $\vee$ choices) that accepts A and runs within space $s(n)$, time $t(n)$ and performs at most $a(n)$ alternations between $\wedge$ and $\vee$ choices.

Deciding of a PA formula of size n with m quantifiers is in

$$STA(*, 2^{n^{O(m)}}, m)$$

Complexity is particularly sensitive to m.