Reflecting Quantifier Elimination: From Dense Linear Orders to Presburger Arithmetic

Tobias Nipkow
(jww Amine Chaieb)

Technische Universität München
Aims

General  How to extend theorem provers safely with decision procedures (DP)

Application  Linear Arithmetic (+, <, not *)

Focus  Not just DPs but Quantifier Elimination
Which theorem provers?

**Foundational**  Small trusted inference kernel

**Extensible**  Logic or meta-language must be able to express proof procedures

Yes:  Coq, HOLs, Isabelle, (PVS, ACL2)

No:  E, Spass, Vampire, Simplify, zChaff, . . .

Not considered:  DPs as trusted black boxes

Unless they return a checkable certificate
Isabelle/HOL

Isabelle A generic interactive theorem prover and logical framework (Paulson, N., Wenzel)

Isabelle/HOL An instance supporting HOL

HOL Church’s Higher Order Logic:
a classical logic of total polymorphic higher order functions

HOL = Functional Programming + Quantifiers
All algorithms in this talk have been programmed and verified in Isabelle/HOL
Decision procedures for and in theorem provers

LCF approach
- program proof search in meta-language (ML)
- reduce proof to rules of the logic

Reflection
- describe decision procedure in the logic
- show soundness (and completeness)
- execute decision procedure on formulae in the logic
Comparison

LCF approach
- no meta-theory, just do it
- produces proof every time
- slow
- tricky to write, often incomplete
- hard to maintain

Reflection
- meta-theoretic proofs
- correctness proof only once
- fast (if executed efficiently)
- completeness proof
- easy to maintain
We focus on reflection
Quantifier elimination

QE takes quantified formula and produces *equivalent* unquantified formula.

\[ \exists x \in \mathbb{R}. \ a < x < b \implies a < b \]

If ground atoms are decidable, QE yields DP:

1. start with sentence
2. eliminate quantifiers
3. decide ground formula
Aims

- Present the essence of the algorithms and their formalization.
- Show similarities via a unified framework.
- Explain and demo reflection.
Related theorem proving work

- Norrish: Presburger in HOL (LCF approach)
- Harrison: *Introduction to Logic and ATP*
- Reflection in Nqthm (Boyer&Moore) and Coq
- *Locales* (Ballarin, Kammüller, Wenzel, Paulson)
1. Logical Framework
2. Dense Linear Orders
3. Linear Real Arithmetic
4. Presburger Arithmetic
5. Beyond
1. Logical Framework
   - Logic
   - Quantifier Elimination

2. Dense Linear Orders

3. Linear Real Arithmetic

4. Presburger Arithmetic

5. Beyond
1 Logical Framework
   Logic
      Quantifier Elimination

2 Dense Linear Orders

3 Linear Real Arithmetic

4 Presburger Arithmetic

5 Beyond
Syntax

\[
\alpha \text{ fm} = \begin{array}{c}
\text{TrueF} \mid \text{FalseF} \mid \text{Atom } \alpha \\
\mid \text{And } (\alpha \text{ fm}) (\alpha \text{ fm}) \\
\mid \text{Or } (\alpha \text{ fm}) (\alpha \text{ fm}) \\
\mid \text{Neg } (\alpha \text{ fm}) \\
\mid \text{ExQ } (\alpha \text{ fm})
\end{array}
\]

Quantifiers: de Bruijn notation!

\[
\text{ExQ } (\text{ExQ } \ldots 0 \ldots 1 \ldots) \\
\approx \exists x_1. \exists x_0. \ldots x_0 \ldots x_1 \ldots
\]

Abbreviations: \(\text{AllQ} \varphi = \text{Neg}(\text{ExQ}(\text{Neg } \varphi))\), \ldots
Auxiliary functions

\[
\text{list-conj} : \alpha \ \text{fm} \ \text{list} \Rightarrow \alpha \ \text{fm}
\]
\[
\text{list-disj} : \alpha \ \text{fm} \ \text{list} \Rightarrow \alpha \ \text{fm}
\]

\[
\text{list-conj} \ [\varphi_1, \ldots, \varphi_n] = \text{and} \ \varphi_1 \ (\text{and} \ \ldots \ \varphi_n)
\]

\[
\text{and} \ TrueF \ \varphi = \varphi
\]
\[
\text{and} \ \varphi \ TrueF = \varphi
\]
\[
\text{and} \ \varphi_1 \ \varphi_2 = \text{And} \ \varphi_1 \ \varphi_2
\]
\textbf{DNF}

dnf :: \(\alpha\) fm \(\Rightarrow\) \(\alpha\) list list

dnf TrueF = [[]]
dnf FalseF = []
dnf (Atom \(\varphi\)) = [[\varphi]]
dnf (Or \(\varphi_1\ \varphi_2\)) = dnf \(\varphi_1\) \(\odot\) dnf \(\varphi_2\)

dnf (And \(\varphi_1\ \varphi_2\)) =
\[d_1 \odot d_2. \ d_1 \leftarrow \text{dnf } \varphi_1, \ d_2 \leftarrow \text{dnf } \varphi_2\]

Assumes negation normal form!
More normal forms

\[ \text{in-nnf} :: \alpha \text{ fm} \Rightarrow \text{ bool} \]

“Does not contain \textit{Neg}”

Note: \( \not\leq \iff > \)

\[ \text{qfree} :: \alpha \text{ fm} \Rightarrow \text{ bool} \]

“Does not contain \textit{ExQ}”
Atoms

- More than a type parameter $\alpha$.
- Atoms come with an *interpretation*, a *negation* etc.
- Functions on atoms are *parameters* of the generic development.
- Parameters form a named context (Isabelle: *locale*)
- Parameters can be instantiated later on
Parameters:

\( l_a \) :: \( \alpha \Rightarrow \beta \) \ list \Rightarrow \ bool

\( \text{aneg} \) :: \( \alpha \Rightarrow \alpha \) \ fm

\( \text{adepends} \) :: \( \alpha \Rightarrow \bool \) “Depends on \( x_0 \)?”

\( \text{adecr} \) :: \( \alpha \Rightarrow \alpha \) “\( x_{i+1} \mapsto x_i \)”
Interpretation

\[ l :: \alpha \text{ fm} \Rightarrow \beta \text{ list} \Rightarrow \text{ bool} \]

\[ l (\text{Atom } a) \; xs = l_a a \; xs \]
\[ l (\text{And } \varphi_1 \varphi_2) \; xs = (l \varphi_1 \; xs \land l \varphi_2 \; xs) \]
\[ l (\text{ExQ } \varphi) \; xs = (\exists x. l \varphi (x \cdot xs)) \]

... 

Example:

\[ l (\text{ExQ} (\text{And} (\text{Atom } a_1) (\text{Atom } a_2))) \; xs = (\exists x. l_a a_1 (x \cdot xs) \land l_a a_2 (x \cdot xs)) \]
Locale ATOM has assumptions:

\[ l(\text{aneg } a) \; xs = (\neg l_a \; a \; xs) \]

\textit{in-nnf} (\text{aneg } a)

\ldots

Must be discharged when locale is instantiated
NNF

\[ \text{nnf} :: \alpha \text{ fm} \Rightarrow \alpha \text{ fm} \]

\[ \text{nnf} \left( \text{And} \ \varphi_1 \ \varphi_2 \right) = \text{And} \left( \text{nnf} \ \varphi_1 \right) \left( \text{nnf} \ \varphi_2 \right) \]
\[ \text{nnf} \left( \text{Neg} \left( \text{Atom} \ a \right) \right) = \text{aneg} \ a \]
\[ \text{nnf} \left( \text{Neg} \left( \text{And} \ \varphi_1 \ \varphi_2 \right) \right) = \]
\[ \quad \text{Or} \left( \text{nnf} \left( \text{Neg} \ \varphi_1 \right) \right) \left( \text{nnf} \left( \text{Neg} \ \varphi_2 \right) \right) \]
\[ \text{nnf} \left( \text{Neg} \left( \text{Neg} \ \varphi \right) \right) = \text{nnf} \ \varphi \]

\[ \ldots \]

Lemma 1 \( \text{I} \left( \text{nnf} \ \varphi \right) \ xs = \text{I} \ \varphi \ xs \)
1 Logical Framework
   Logic
   Quantifier Elimination

2 Dense Linear Orders

3 Linear Real Arithmetic

4 Presburger Arithmetic

5 Beyond
Lifting quantifier elimination

If you can eliminate one of them, you can eliminate them all!

Given $qe :: \alpha \text{ fm} \Rightarrow \alpha \text{ fm}$ such that $I(qe \varphi) = I(\text{ExQ} \varphi)$ if $qfree \varphi$

Not $qe (\text{ExQ} \varphi)$, just $qe \varphi$, ExQ and 0 implicit

Apply $qe$ bottom up:

$\text{ExQ} \varphi \leadsto \text{ExQ} \psi \leadsto \psi'$
Put into DNF first:

\[(\exists x. \phi) = (\exists x. \bigvee_i \bigwedge_j a_{ij}) = (\bigvee_i \exists x. \bigwedge_j a_{ij})\]

Apply \textit{qe} to conjunction of atoms all of which depend on \(x\):

\[= (\bigvee_i A_i \land (\exists x. B_i(x)))\]
QE via DNF
formally

lift-dnf-qe :: (α list ⇒ α fm) ⇒ α fm ⇒ α fm

lift-dnf-qe qe (And φ₁ φ₂) =
and (lift-dnf-qe qe φ₁) (lift-dnf-qe qe φ₂)

lift-dnf-qe qe (ExQ φ) =
(let djs = dnf (nnf (lift-dnf-qe qe φ)))
in list-disj (map (qelim qe) djs)

qelim qe as =
(let qf = qe [a ← as. adepends a];
indep = [Atom(adecr a). a ← as, ¬ adepends a]
in and qf (list-conj indep))
Correctness

**Theorem** If $qe$ eliminates one existential (while preserving the interpretation), then $lift-dnf-qe qe$ eliminates all quantifiers (while preserving the interpretation).
Conversion to DNF may (unavoidably!) cause exponential blowup

Problematic case: quantifier alternation:

\[ \forall \exists \land = \forall \lor \exists \land = \forall \lor \land = \]
\[ \neg \exists \neg \lor \land = \neg \exists \land \lor = \neg \exists \lor \land = \]

Conversion to NNF is linear
QE via NNF

\[ \text{lift-nnf-qe :: } (\alpha \text{ fm} \Rightarrow \alpha \text{ fm}) \Rightarrow \alpha \text{ fm} \Rightarrow \alpha \text{ fm} \]

\[ \text{lift-nnf-qe qe } (\text{ExQ } \varphi) = \text{qe } (\text{nnf } (\text{lift-nnf-qe qe } \varphi)) \]

\[
...
\]

More efficient, but trickier for \text{qe}
1 Logical Framework

2 Dense Linear Orders
   Logic
   Reflection
   Certificates

3 Linear Real Arithmetic

4 Presburger Arithmetic

5 Beyond
1 Logical Framework

2 Dense Linear Orders
   Logic
   Reflection
   Certificates

3 Linear Real Arithmetic

4 Presburger Arithmetic

5 Beyond
Dense Linear Orders
without endpoints

Atoms: $x < y$

Axioms:
- Dense: $x < z \implies \exists y. x < y < z$
- No endpoints: $\exists x z. x < y < z$
Langford [1927] developed what has come to be known as the method of elimination of quantifiers to solve the decision problem for the first order theory of dense linear orders. However, despite this very important technical contribution, Langford remained badly confused.

Example:

$$(\exists y. \ x < y \land y < z) = (x < z)$$

In general:

$$\exists x. \ (\bigwedge_i l_i < x) \land (\bigwedge_j x < u_j)$$

$$= (\max_i l_i < \min_j u_j) = (\bigwedge_{ij} l_i < u_j)$$
datatype $atom = \text{Less } \text{nat } \text{nat}$

$\text{Less } m \ n \ ≈ \ x_m < x_n$

Interpretation: $l_{d\mathcal{O}} (\text{Less } i \ j) \ xs = (xs[i] < xs[j])$
Quantifier elimination
formally

Input: list (conjunction) of atoms, all containing 0

\[
\text{qe-less as} = \begin{cases} 
\text{False} & \text{if } \text{Less } 0 \ 0 \in \text{as} \\
\text{let } \text{lbs} = [m-1. \ \text{Less } m \ 0 \leftarrow \text{as}] \\
\text{let } \text{ubs} = [n-1. \ \text{Less } 0 \ n \leftarrow \text{as}] \\
\text{pairs} = [\text{Atom} (\text{Less } m \ n). \ m \leftarrow \text{lbs}, \ n \leftarrow \text{ubs}] \\
\text{in } \text{list-conj pairs}\end{cases}
\]
Adding “=”

\[(\exists x. \ x = t \land \phi) = \phi[t/x] \quad \text{if} \ x \notin t\]

\[\text{qe-less-eq as } = \]
\[(\text{let } bs = \text{filter } (\lambda a. \ a \neq \text{Eq } 0 \ 0) \ \text{as in case filter is-Eq bs of } [] \Rightarrow \text{qe-less bs} \]
\[\mid \text{Eq } i \ j \cdot \text{eqs } \Rightarrow\]
\[(\text{let } \text{ineqs } = \text{filter } (\text{not } \circ \text{is-Eq}) \ bs; \]
\[\quad v = (\text{if } i=0 \ \text{then } j \ \text{else } i) \]
\[\quad \text{cs } = \text{map } (\text{Atom } \circ \text{subst } v) \ (\text{eqs } \& \text{ ineqs}) \]
\[\quad \text{in list-conj cs})\]
Instantiating locales

functions and thms

↓

Locale

↓

functions and thms
Instantiating locale ATOM

\[ DLO: \text{ATOM}[\alpha \mapsto \text{atom}, \ I_a \mapsto \ I_{\text{dlo}}, \ldots] \]

Prove: \ldots \implies \ \text{DLO.I (qe-less-eq as) xs} = 
\quad (\exists x. \ \forall a \in \text{as.} \ I_{\text{dlo}} a (x \cdot \text{xs}))

Define: \ \text{dlo-qe} = \ DLO.\text{lift-dnf-qe qe-less-eq}

Obtain: \ \text{DLO.I (dlo-qe } \varphi \text{) xs} = \ DLO.I \varphi \ \text{xs}
Logical Framework

Dense Linear Orders
  Logic
  Reflection
  Certificates

Linear Real Arithmetic

Presburger Arithmetic

Beyond
Reflection in action

\[ \exists x. \ s < x \land x < t \]

by def of \( DLO.I \) (reversed)

\[ DLO.I (ExQ (And (Less 1 0) (Less 0 2))) [s,t] \]

by \( DLO.I (dlo-qe \ \varphi) \ \chi s = DLO.I \ \varphi \ \chi s \)

\[ DLO.I (dlo-qe (ExQ \ldots)) [s,t] \]

by evaluation of \( dlo-qe \)

\[ DLO.I (Less 0 1) [s,t] \]

by def of \( DLO.I \)

\[ s < t \]
Reflection abstractly

form

    by def of \( I \) (reversed)

\[ = I \; rep \; [\text{subterms}] \]

    by correctness of \( simp \)

\[ = I \; (simp(rep)) \; [\text{subterms}] \]

    by evaluation of \( simp \)

\[ = I \; rep' \; [\text{subterms}] \]

    by def of \( I \)

\[ = \text{form'} \]
Evaluation

- by proof (e.g. rewriting) — slow
- by proof-free execution — fast
  - compilation to abstract machine code (Coq)
  - compilation to ML (Isabelle) or Lisp (ACL2)
Demo
Worst case complexity

Algorithm  Exponential blowup for every quantifier alternation

Decision problem  PSPACE complete

Quantifier elimination  \( \text{TIME}(2^{p(n)}) \) (?)
1 Logical Framework

2 Dense Linear Orders
   Logic
   Reflection
   Certificates

3 Linear Real Arithmetic

4 Presburger Arithmetic

5 Beyond
Certificates for DPs

1. Unverified computation of certificate $C$ for formula $\phi$ (external, fast)
2. Verified check that $C$ indeed proves $\phi$ (internal)

Works well for problems in NP and more

Example Propositional unsatisfiability

1. Find refutation proof (SAT solver)
2. Check refutation proof (TP)
Certificates for DLO

The idea

Certificate for unsatisfiability of $\bigwedge_i x_{l(i)} < x_{r(i)} =: \phi$

cycle $x_k < \cdots < x_k$

Soundness and completeness: $\text{QE}(\exists \overline{x}. \phi)$ yields $False$ iff it constructs a cycle $(x < x)$

DP for unquantified formulae: To prove $\phi$, prove unsatisfiability of each disjunct of $\text{DNF}(\neg \phi)$
Certificates for DLO

Formally

Certificate checkers:

\[
\text{cycle } [a_1, \ldots, a_m] [i_1, \ldots, i_n] \\
\text{iff } [a_{i_1}, \ldots, a_{i_n}] \text{ forms a cycle.}
\]

\[
\text{cyclic-dnf } [as_1, \ldots, as_n] \\
\text{iff } \exists is_1, \ldots, is_n. \text{ cycle } as_1 is_1 \land \ldots \land \text{ cycle } as_n is_n
\]

Correctness theorem:

\[
qfree \varphi \land \text{cyclic-dnf } (\text{dnf}(DLO.nnf \varphi)) \iff \neg DLO.I \varphi xs
\]
Demo
1 Logical Framework

2 Dense Linear Orders

3 Linear Real Arithmetic
   Fourier-Motzkin
   Ferrante and Rackoff
   Quantifier free case

4 Presburger Arithmetic

5 Beyond
Works for . . .

- $\mathbb{R}$
- $\mathbb{Q}$
- Ordered, divisible, torsion free Abelian groups
  (divisible & torsion free $=$
   has division by positive integers)
1 Logical Framework

2 Dense Linear Orders

3 Linear Real Arithmetic
   Fourier-Motzkin
   Ferrante and Rackoff
   Quantifier free case

4 Presburger Arithmetic

5 Beyond
Linear real arithmetic

Atoms: $s < t$ (and $s = t$)

where $s$ and $t$ are expressions involving

- constants
- variables
- addition
- multiplication with constants

Eg: $2.7(x + 0.5y) < x + 3.1$
Normal form

\[ r < c_0 x_0 + \cdots + c_n x_n \]

where \( r, c_0, \ldots, c_n \in \mathbb{R} \)
Fourier-Motzkin elimination by example

\[ \exists x. \ 3 < 2x + s \land 5 < -3x + t \]
\[ = \exists x. \ 9 < 6x + 3s \land 10 < -6x + 2t \]
\[ = 19 < 3s + 2t \]

Lower/upper bound view:
\[ \exists x. \ 3 < 2x + s \land 5 < -3x + t \]
\[ = \exists x. \ 9 - 3s < 6x \land 6x < 2t - 10 \]
\[ = 9 - 3s < 2t - 10 \]

Combine +/- atoms (lower/upper bounds) by unifying leading coefficients
Fourier-Motzkin elimination

in general

\[ \exists x. \left( \bigwedge_i r_i < c_i x + t_i \right) \land \left( \bigwedge_j r'_j < c'_j x + t'_j \right) \]

where \( c_i > 0, c'_j < 0 \)

\[ = \max_i \left( \frac{(r_i - t_i)}{c_i} \right) < \min_j \left( \frac{(r'_j - t'_j)}{c'_j} \right) \]

\[ = \bigwedge_{ij} c'_j r_i - c_i r'_j < c'_j t_i - c_i t'_j \]
Formalization

Atoms: \( \text{Less } r [c_0, \ldots, c_n] \)

Note:
- Variables are indexed by de Bruijn notation
- Conversion into normal form omitted
**Lists as vectors**

**Addition and subtraction**

\[
\begin{align*}
[c_0, \ldots] + [d_0, \ldots] &= [c_0 + d_0, \ldots] \\
[c_0, \ldots] - [d_0, \ldots] &= [c_0 - d_0, \ldots]
\end{align*}
\]

**Multiplication with scalar**

\[
r \ast_s [c_0, \ldots] = [r \ast c_0, \ldots]
\]

**Inner product**

\[
[c_0, \ldots] \circ [d_0, \ldots] = c_0 \ast d_0 + \ldots
\]
Interpreting atoms

\[ l_R \::\ atom \Rightarrow real\ list \Rightarrow bool \]

\[ l_R \ (Less\ r\ cs)\ xs = (r < cs \odot xs) \]

Instantiating ATOM:
\[ R: ATOM[l_a \mapsto l_R, \ldots] \]
Fourier-Motzkin elimination
formally

\[ \text{qe-less :: atom list } \Rightarrow \text{ atom fm} \]

\[ \text{qe-less as } = \]
\[
\quad (\text{let } lbs = [(r, c, cs). \text{ Less } r (c \cdot cs) \leftarrow as, c > 0];
\quad ubs = [(r, c, cs). \text{ Less } r (c \cdot cs) \leftarrow as, c < 0];
\quad \text{pairs } = [\text{Atom(combine p q). } p \leftarrow lbs, q \leftarrow ubs]
\quad \text{in list-conj pairs})
\]

\[ \text{combine } (r_1, c_1, cs_1) (r_2, c_2, cs_2) = \]
\[ \text{Less } (c_1 \ast r_2 - c_2 \ast r_1) (c_1 \ast_s cs_2 - c_2 \ast_s cs_1) \]
Adding $Eq \ r \ cs$

As for DLO:

$qe$-less-$eq$ as $=  \\
\begin{align*}
(case \ filter \ is-Eq \ as \ of \ [] & \Rightarrow qe$-less $as \\
| Eq \ r \ (c \cdot cs) \cdot eqs & \Rightarrow \ldots \ subst \ldots
\end{align*}$
Correctness

As for DLO:

Prove: \[\ldots \implies R.I \ (qe\text{-}less\text{-}eq \ as) \ xs = (\exists x. \forall a \in as. \ I_R a \ (x \cdot xs))\]

Define: \(lin\text{-}qe = R.lift\text{-}dnf\text{-}qe \ qe\text{-}less\text{-}eq\)

Obtain: \(R.I \ (lin\text{-}qe \ \varphi) \ xs = R.I \ \varphi \ xs\)
1 Logical Framework

2 Dense Linear Orders

3 Linear Real Arithmetic
   Fourier-Motzkin
   Ferrante and Rackoff
   Quantifier free case

4 Presburger Arithmetic

5 Beyond
Prolegomena

- For simplicity: only $<$
- View all atoms involving $x$ as $l < x$ or $x < u$
  ($x$ not in $l$ or $u$)

$P(x)$ in NNF can be put into DNF: $\bigvee_i \bigwedge_j a_{ij}$

$\implies P(x)$ is a finite union of finite intersections of intervals $(l, +\infty)$ and $(-\infty, u)$

$\implies$ For each valuation of the other variables, $
\{x \mid P(x)\}$ looks like this:

```
(   ) (   ) (   )
```

Every interval has upper/lower bound in $P(x)$

Problem: $l$s and $u$s are symbolic
Put formula into NNF $P(x)$ — no DNF!

If $P(x)$ for some $x$, then either
- there is no lower bound ($P(-\infty)$), or
- there is no upper bound ($P(+\infty)$), or
- $l < x < u$ for some $l$ and $u$ in $P(x)$ such that $P(y)$ for any $l < y < u$ implies $P((l + u)/2)$
(\exists x. P(x)) = (P(-\infty) \lor P(+\infty) \lor \bigvee_{l,u \in P} P((l + u)/2))

\[ \begin{align*}
P(-\infty) & \text{ replace } l < x \text{ by False, } x < u \text{ by True} \\
P(+\infty) & \text{ replace } l < x \text{ by True, } x < u \text{ by False}
\end{align*} \]

Example
\[ P(x) = x < y \land y < z \implies P(-\infty) = y < z \]
\[ P(+\infty) = False \]
Consider three sets of terms:

lower bounds \( l \) in \( l < x \)
upper bounds \( u \) in \( x < u \)
equalities \( t \) in \( x = t \)
Ferrante and Rackoff formalized

\[
fr \varphi = \\
(\text{let } as = \text{atoms } \varphi; \\
\quad lbs = \text{lbounds as}; \; ubs = \text{ubounds as}; \\
\quad \text{bet} = [\text{subst} (\text{between } p \; q) \varphi \cdot p \leftarrow lbs, \; q \leftarrow ubs]; \\
\quad \text{eqs} = [\text{subst } p \varphi \cdot p \leftarrow \text{ebounds as}] \\
\text{in } \text{list-disj} (\inf_- \varphi \cdot \inf_+ \varphi \cdot \text{bet } \odot \text{eqs}))
\]

\[
fr\text{-qe} = R.lift-nnf-qe fr
\]

\[
R.l (fr\text{-qe } \varphi) \; xs = R.l \varphi \; xs
\]
Worst case complexity of algorithms

**Fourier-Motzkin**  Exponential blowup for every quantifier alternation
\[ \implies \text{non-elementary} \]

**Ferrante & Rackoff**  Quadratic blowup for every quantifier
\[ \implies 2^{2^c n} \]
Worst case complexity of problems

Decision problem
\[ \text{NTIME}(2^{cn}) < \text{DP}(\mathbb{R}, +) \leq \text{SPACE}(2^{dn}) \]
[Fischer & Rabin 74] [Ferrante & Rackoff 75]

Quantifier elimination
\[ \text{SPACE}(2^{2^{cn}}) \leq \text{QE}(\mathbb{R}, +) \leq \text{SPACE}(2^{2^{cn}}) \]
\[ \text{TIME}(2^{2^{cn}}) \leq \text{QE}(\mathbb{R}, +) \leq \text{TIME}(2^{2^{cn}}) \]
[Weisspfenning 88]
Corollary There are no short \((\leq 2^{cn})\) certificates (proofs) that can be checked quickly (in polynomial time).
1 Logical Framework

2 Dense Linear Orders

3 Linear Real Arithmetic
   Fourier-Motzkin
   Ferrante and Rackoff
   Quantifier free case

4 Presburger Arithmetic

5 Beyond
Applications

- Most theorem provers
- Proof Carrying Code
- Certified program analysis
Quantifier free case

Remember:

\( \phi \) true iff each disjunct of DNF(\( \neg \phi \)) is unsatisfiable.

**Lemma** \( \bigwedge_{i=1}^{n} a_i \) is unsatisfiable iff there is a non-negative linear combination \( \sum_{i=1}^{n} c_i \ast a_i \) that is *contradictory* (e.g. \( 0 \leq -1 \)).

**Example** \( \neg(2 \leq x \land 1 \leq -3x) \)
because \( 3(2 \leq x) + (1 \leq -3x) = (7 \leq 0) \)

Certificate: \((c_1, \ldots, c_n)\)
Finding the certificate

- By Fourier-Motzkin elimination (⇒ Lemma)
- By Linear Programming: \( \bigwedge_{i=1}^{n} r_i \leq cs_i \odot xs \)
  \( \sim \) \( Ax \geq b \) with \( b \in \mathbb{R}^n, A \in \mathbb{R}^{n \times m}, x \in \mathbb{R}^m \)

**Lemma** (Farkas)
- Either \( \exists x. Ax \geq b \)
- or \( \exists y \geq 0. A^T y = 0 \land b^T y < 0. \)

The system has no solution \( (x) \) iff there is an unsatisfiability certificate \( (y) \).

Find certificate by (eg) Simplex
Corollary Implications \( (\bigwedge_{i=1}^{n} a_i) \rightarrow a \) of linear inequalities can be proved in polynomial time.
Checking the certificate

check as \( y = ( (\forall c \in y. \; c \geq 0) \land \) \\
(let \( b = \text{map \; lhs \; as;} \)) \\
A = \text{map \; rhs \; as;} \\
by = b \odot y; \\
Ay = [cs \odot y. \; cs \leftarrow A]; \\
in (\forall c \in Ay. \; c = 0) \land \\
(by < 0 \lor (\forall a \in as. \; \text{is-Eq} \; a) \land by \neq 0))

Lemma \( \text{check \; as \; cs} \iff \exists a \in as. \; \neg l_R \; a \; xs \)
1 Logical Framework

2 Dense Linear Orders

3 Linear Real Arithmetic

4 Presburger Arithmetic
   Presburger’s algorithm
   Cooper’s algorithm
   Complexity and more

5 Beyond
where \( d, i, k_n, x_n \in \mathbb{Z} \)
1 Logical Framework
2 Dense Linear Orders
3 Linear Real Arithmetic
4 Presburger Arithmetic
   Presburger’s algorithm
   Cooper’s algorithm
   Complexity and more
5 Beyond
Presburger’s algorithm
by example

\[ \exists i. \, l \leq 2i \land 3i \leq u \]
\[ = \exists i. \, 3l \leq 6i \leq 2u \]
\[ = \exists j. \, 3l \leq j \leq 2u \land 6|j \]
\[ = \bigvee_{n=0}^{5} 3l + n \leq 2u \land 6|3l + n \]
Presburger’s algorithm(?)

Informally

Input $P(x)$: Conjunction of atoms (DNF!)

- Set all coefficients of $x$ to the lcm of all coefficients of $x$ (by $\ast$) $\leadsto Q(m \ast x)$
- $R(x) := Q(x) \land m \mid x$
- Let $\delta$ be the lcm of all divisors $d$ ($d \mid \_ \in R(x)$)
- If $x$ has lower bounds $ls$ in $R(x)$: $\bigvee_{t \in T} R(t)$
  where $T = \{l + n \mid l \in ls \land 0 \leq n < \delta\}$
- Otherwise $\bigvee_{t \in T} R'(t)$ where $R'$ is $R$ w/o $\leq$-atoms and $T = \{n \mid 0 \leq n < \delta\}$
Presburger’s algorithm
The core, formally

\[ qe = (let d = \text{lcm}(\text{map divisor as}); \; ls = \text{lbounds as in}
\begin{align*}
\text{if } ls &= [] \\
\text{then let } ds &= \text{filter}(\text{not} \circ \text{is-Le}) \text{ as in} \\
\text{Disj } [0..<d] (\lambda n. [\text{list-conj}(\text{map (subst } n []) ds)]) \\
\text{else} \\
\text{Disj } [0..<d] (\lambda n. \\
\text{Disj } ls (\lambda (li,lks). \\
\text{list-conj}(\text{map (subst } (li+n) lks) as))))
\end{align*}
\text{Disj is } f = \text{list-disj (map } f \text{ is) \} \]
1 Logical Framework

2 Dense Linear Orders

3 Linear Real Arithmetic

4 Presburger Arithmetic
   Presburger’s algorithm
   Cooper’s algorithm
   Complexity and more

5 Beyond
Cooper’s algorithm

No DNF, just NNF

$$\exists i. P(i)$$

$$\left( \bigvee_{n=0}^{d-1} P_{-\infty}(n) \right) \lor \left( \bigvee_{n=0}^{d-1} \bigvee_{l} P(l + n) \right)$$

[Cooper 72] ⇝ [Ferrante & Rackoff 75]
1 Logical Framework

2 Dense Linear Orders

3 Linear Real Arithmetic

4 Presburger Arithmetic
   Presburger’s algorithm
   Cooper’s algorithm
   Complexity and more

5 Beyond
Worst case complexity of algorithms

Presburger: Exponential blowup for every quantifier alternation

\[ 2^{2^{2cn}} \] [Oppen 73/78]
Worst case complexity
of problems

Decision problem
\[ \text{NTIME}(2^{2^{cn}}) < \text{DP} (\mathbb{Z}, +) \leq \text{SPACE}(2^{2^{dn}}) \]
[Fischer & Rabin 74] [Ferrante & Rackoff 75]

Quantifier elimination
\[ \text{QE}(\mathbb{Z}, +) \leq \text{TIME}(2^{2^{2^{cn}}}) \] [Oppen 78]

One exponential up from \( \mathbb{R} \)
Quantifier free case

\( \phi \) is unsatisfiable over \( \mathbb{Z} \) if it is unsatisfiable over \( \mathbb{R} \)

Popular!
Alternatives

\[ \text{QE}(\mathbb{Z}, +) \] Omega [Pugh 92]

\[ \text{DP}(\mathbb{Z}, +) \] Finite automata

Solutions to Presburger formulae (viewed as bitstrings) are regular sets

Reflection?
Logical Framework

Dense Linear Orders

Linear Real Arithmetic

Presburger Arithmetic

Beyond
Beyond

Mixed integer/real linear arithmetic ($\mathbb{Z} \sqcup \mathbb{R}$)
Algorithm: Weisspfenning
Reflection: Chaieb

($\mathbb{R}, +, \ast$)

Algorithms: Tarski, Cohen/Hörmander, Collins (CAD)
LCF tactic: McLaughlin&Harrison
Reflection: Mahboubi (CAD, partial!)
The future is bright for reflection

But optimization is of the essence