

# 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**

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.

$$\exists x \in \mathbb{R}. a < x < b \quad \rightsquigarrow \quad a < b$$

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

$$\begin{aligned} \alpha \text{ fm} &= \text{TrueF} \mid \text{FalseF} \mid \text{Atom } \alpha \\ &\mid \text{And } (\alpha \text{ fm}) (\alpha \text{ fm}) \\ &\mid \text{Or } (\alpha \text{ fm}) (\alpha \text{ fm}) \\ &\mid \text{Neg } (\alpha \text{ fm}) \\ &\mid \text{ExQ } (\alpha \text{ fm}) \end{aligned}$$

Quantifiers: *de Bruijn notation!*

$$\begin{aligned} &\text{ExQ } (\text{ExQ } \dots 0 \dots 1 \dots) \\ &\approx \exists x_1. \exists x_0. \dots x_0 \dots x_1 \dots \end{aligned}$$

Abbreviations:  $\text{AllQ } \varphi = \text{Neg}(\text{ExQ}(\text{Neg } \varphi)), \dots$

# Auxiliary functions

*list-conj* ::  $\alpha$  fm list  $\Rightarrow$   $\alpha$  fm

*list-disj* ::  $\alpha$  fm list  $\Rightarrow$   $\alpha$  fm

# DNF

*dnf* ::  $\alpha$  fm  $\Rightarrow$   $\alpha$  list list

*dnf* TrueF = [[]]

*dnf* FalseF = []

*dnf* (Atom  $\varphi$ ) = [[ $\varphi$ ]]

*dnf* (Or  $\varphi_1$   $\varphi_2$ ) = *dnf*  $\varphi_1$  @ *dnf*  $\varphi_2$

*dnf* (And  $\varphi_1$   $\varphi_2$ ) =

[ $d_1$  @  $d_2$ .  $d_1 \leftarrow$  *dnf*  $\varphi_1$ ,  $d_2 \leftarrow$  *dnf*  $\varphi_2$ ]

# 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

# Locale ATOM

Parameters:

$l_a \quad :: \alpha \Rightarrow \beta \text{ list} \Rightarrow \text{bool}$

$aneg \quad :: \alpha \Rightarrow \alpha \text{ fm}$

...

# Interpretation

$I :: \alpha \text{ fm} \Rightarrow \beta \text{ list} \Rightarrow \text{bool}$

$I (\text{Atom } a) \text{ xs} = I_a \ a \ \text{xs}$

$I (\text{And } \varphi_1 \ \varphi_2) \ \text{xs} = (I \ \varphi_1 \ \text{xs} \wedge I \ \varphi_2 \ \text{xs})$

$I (\text{ExQ } \varphi) \ \text{xs} = (\exists x. I \ \varphi \ (x \cdot \text{xs}))$

...

Example:

$I (\text{ExQ } (\text{And } (\text{Atom } a_1) (\text{Atom } a_2))) \ \text{xs} =$   
 $(\exists x. I_a \ a_1 \ (x \cdot \text{xs}) \wedge I_a \ a_2 \ (x \cdot \text{xs}))$

# Assumptions

Locale ATOM has assumptions:

$$I(\text{aneg } a) \text{ } xs = (\neg I_a \text{ } a \text{ } xs)$$

...

Must be discharged when locale is instantiated

# NNF

$nnf :: \alpha \text{ fm} \Rightarrow \alpha \text{ fm}$

$nnf (\text{And } \varphi_1 \varphi_2) = \text{And } (nnf \varphi_1) (nnf \varphi_2)$

$nnf (\text{Neg } (\text{Atom } a)) = \text{aneg } a$

...

Lemma  $l (nnf \varphi) xs = l \varphi xs$

# 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  $qe \varphi$ ,  $\text{ExQ}$  and 0 implicit

Apply  $qe$  bottom up:

$$\text{ExQ } \varphi \rightsquigarrow \text{ExQ } \psi \rightsquigarrow \psi'$$

# QE via DNF

informally

Put into DNF first:

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

Apply *qe* to conjunction of atoms  
all of which depend on  $x$ :

$$= (\bigvee_i A_i \wedge (\exists x. B_i(x)))$$

# QE via DNF

formally

*lift-dnf-qe* :: ( $\alpha$  list  $\Rightarrow$   $\alpha$  fm)  $\Rightarrow$   $\alpha$  fm  $\Rightarrow$   $\alpha$  fm

*lift-dnf-qe* qe (ExQ  $\varphi$ ) =  
(let djs = dnf (nnf (lift-dnf-qe qe  $\varphi$ ))  
in list-disj (map (qelim qe) djs))

*qelim* qe as =  
(let qf = qe [a  $\leftarrow$  as. a depends a];  
indep = [Atom (adecr a). a  $\leftarrow$  as,  $\neg$  a depends 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).

# Complexity

Conversion to DNF may (unavoidably!) cause exponential blowup

## QE via NNF

*lift-nnf-qe* ::  $(\alpha \text{ fm} \Rightarrow \alpha \text{ fm}) \Rightarrow \alpha \text{ fm} \Rightarrow \alpha \text{ fm}$

*lift-nnf-qe qe* (ExQ  $\varphi$ ) = *qe* (*nnf* (*lift-nnf-qe qe*  $\varphi$ ))

...

More efficient, but trickier for *qe*

- 1 Logical Framework
- 2 Dense Linear Orders  
Logic  
Reflection
- 3 Linear Real Arithmetic
- 4 Presburger Arithmetic
- 5 Beyond

- 1 Logical Framework
- 2 Dense Linear Orders  
Logic  
Reflection
- 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.*

Martin Davis. American Logic in the 1920s.  
*JSL* 1995.

# Quantifier elimination

informally

Example:

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

In general:

$$\begin{aligned} & \exists x. \left( \bigwedge_i l_i < x \right) \wedge \left( \bigwedge_j x < u_j \right) \\ &= \left( \max_i l_i < \min_j u_j \right) = \left( \bigwedge_{ij} l_i < u_j \right) \end{aligned}$$

# Atoms

formally

**datatype** *atom* = *Less nat nat*

$$\textit{Less } m \ n \ \approx \ x_m < x_n$$

Interpretation:  $\textit{I}_{dlo} (\textit{Less } i \ j) \ xs = (xs_{[i]} < xs_{[j]})$

# Quantifier elimination

formally

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

*qe-less as =*

*(if Less 0 0  $\in$  as then FalseF else*

*let lbs = [m-1. Less m 0  $\leftarrow$  as];*

*ubs = [n-1. Less 0 n  $\leftarrow$  as];*

*pairs = [Atom(Less m n). m  $\leftarrow$  lbs, n  $\leftarrow$  ubs]*

*in list-conj pairs)*

## Adding “=”

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

*qe-less-eq as =*

*(let bs = filter ( $\lambda a. a \neq \text{Eq } 0\ 0$ ) as in  
case filter is-Eq bs of []  $\Rightarrow$  qe-less bs  
| Eq i j · eqs  $\Rightarrow$  ... subst ...*

# Instantiating locales

functions and thms



Locale



functions and thms

# Instantiating locale ATOM

*DLO*: *ATOM*[ $\alpha \mapsto atom, I_a \mapsto I_{dlo}, \dots$ ]

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

Define: *dlo-qe* = *DLO.lift-dnf-qe qe-less-eq*

Obtain: *DLO.I (dlo-qe  $\varphi$ ) xs = DLO.I  $\varphi$  xs*

- 1 Logical Framework
- 2 Dense Linear Orders  
Logic  
Reflection
- 3 Linear Real Arithmetic
- 4 Presburger Arithmetic
- 5 Beyond

## Reflection in action

$$\exists x. s < x \wedge x < t$$

by def of *DLO.I* (reversed)

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

by *DLO.I* (*dlo-qe*  $\varphi$ )  $xs = DLO.I \varphi xs$

$$= DLO.I (dlo-qe (ExQ \dots)) [s,t]$$

by evaluation of *dlo-qe*

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

by def of *DLO.I*

$$= s < t$$

# 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  
 $\implies$  non-elementary

Decision problem PSPACE complete  
(Kozen, *Theory of Computation*, 2006)

- 1 Logical Framework
- 2 Dense Linear Orders
- 3 Linear Real Arithmetic**  
Fourier-Motzkin  
Ferrante and Rackoff
- 4 Presburger Arithmetic
- 5 Beyond

- 1 Logical Framework
- 2 Dense Linear Orders
- 3 Linear Real Arithmetic**  
Fourier-Motzkin  
Ferrante and Rackoff
- 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

$$r < c_0x_0 + \cdots + c_nx_n$$

where  $r, c_0, \dots, c_n \in \mathbb{R}$

# Fourier-Motzkin elimination

in general

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

where  $c_i > 0, c'_j < 0$

$$= \max_i ((r_i - t_i) / c_i) < \min_j ((r'_j - t'_j) / c'_j)$$

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

# Formalization

Atoms: *Less*  $r [c_0, \dots, c_n]$

Note:

- Variables are indexed by de Bruijn notation
- Conversion into normal form omitted

# Lists as vectors

## Addition and subtraction

$$[c_0, \dots] + [d_0, \dots] = [c_0 + d_0, \dots]$$

$$[c_0, \dots] - [d_0, \dots] = [c_0 - d_0, \dots]$$

## Multiplication with scalar

$$r *_s [c_0, \dots] = [r*c_0, \dots]$$

## Inner product

$$[c_0, \dots] \odot [d_0, \dots] = c_0*d_0 + \dots$$

# Interpreting atoms

$I_R :: atom \Rightarrow real\ list \Rightarrow bool$

$I_R (Less\ r\ cs)\ xs = (r < cs \odot xs)$

Instantiating ATOM:

$R: ATOM[I_a \mapsto I_R, \dots]$

# Fourier-Motzkin elimination

formally

*qe-less :: atom list  $\Rightarrow$  atom fm*

*qe-less as =*

*(let lbs = [(r,c,cs). Less r (c·cs)  $\leftarrow$  as, c>0];  
ubs = [(r,c,cs). Less r (c·cs)  $\leftarrow$  as, c<0];  
pairs = [Atom(combine p q). p $\leftarrow$ lbs, q $\leftarrow$ ubs]  
in list-conj pairs)*

*combine (r<sub>1</sub>, c<sub>1</sub>, cs<sub>1</sub>) (r<sub>2</sub>, c<sub>2</sub>, cs<sub>2</sub>) =*

*Less (c<sub>1</sub> \* r<sub>2</sub> - c<sub>2</sub> \* r<sub>1</sub>) (c<sub>1</sub> \*<sub>s</sub> cs<sub>2</sub> - c<sub>2</sub> \*<sub>s</sub> cs<sub>1</sub>)*

# Correctness

As for DLO:

Prove:  $\dots \implies R.I (qe-less-eq\ as)\ xs =$   
 $(\exists x.\forall a \in as. I_R a (x \cdot xs))$

Define:  $lin-qe = R.lift-dnf-qe\ qe-less-eq$

Obtain:  $R.I (lin-qe\ \varphi)\ xs = R.I\ \varphi\ xs$

- 1 Logical Framework
- 2 Dense Linear Orders
- 3 Linear Real Arithmetic**  
Fourier-Motzkin  
Ferrante and Rackoff
- 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  $u$ )

Q-free  $P(x)$  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  $(-\infty, u)$

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

# Ferrante and Rackoff

idea

- 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  $l < y < u$   
 $\implies P((l + u)/2)$

# Ferrante and Rackoff

informal

$$(\exists x. P(x)) = (P(-\infty) \vee P(+\infty) \vee \bigvee_{I, u \in P} P((I + u)/2))$$

$P(-\infty)$  replace  $I < x$  by *False*,  $x < u$  by *True*

$P(+\infty)$  replace  $I < x$  by *True*,  $x < u$  by *False*

# Ferrante and Rackoff

optimized

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

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

$fr-qe = R.lift-nnf-qe fr$

**Theorem**  $R.I (fr-qe \varphi) xs = R.I \varphi xs$

# Worst case complexity of algorithms

Fourier-Motzkin Exponential blowup  
for every quantifier alternation  
 $\implies$  non-elementary

Ferrante&Rackoff Quadratic blowup  
for every quantifier  
 $\implies 2^{2^{cn}}$

- 1 Logical Framework
- 2 Dense Linear Orders
- 3 Linear Real Arithmetic
- 4 Presburger Arithmetic
  - “Presburger’s algorithm”
  - Cooper’s algorithm
  - Complexity
- 5 Beyond

# Atoms

$$\begin{aligned}i &\leq k_0 * x_0 + \cdots k_n * x_n \\d &| i + k_0 * x_0 + \cdots k_n * x_n\end{aligned}$$

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
- 5 Beyond

# Presburger/Enderton

by example

$$\begin{aligned} & \exists x. l \leq 2x \wedge 3x \leq u \\ = & \exists x. 3l \leq 6x \leq 2u \\ = & \exists y. 3l \leq y \leq 2u \wedge 6|y \\ = & \bigvee_{n=0}^5 3l + n \leq 2u \wedge 6|3l + n \end{aligned}$$

# Presburger/Enderton

informally

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

- Set all coefficients of  $x$  to the lcm of all coefficients of  $x$  (by  $*$ )  $\rightsquigarrow Q(m * x)$
- $R(x) := Q(x) \wedge m|x$
- Let  $\delta$  be the lcm of all divisors  $d$  ( $d|_ - \in R(x)$ )
- If  $x$  has lower bounds  $ls$  in  $R(x)$ :  $\bigvee_{t \in T} R(t)$   
where  $T = \{l + n \mid l \in ls \wedge 0 \leq n < \delta\}$
- Otherwise  $\bigvee_{t \in T} R'(t)$  where  
 $R'$  is  $R$  w/o  $\leq$ -atoms and  $T = \{n \mid 0 \leq n < \delta\}$

# Presburger/Enderton

The core, formally

*qe as =*  
*(let d = lcm (map divisor as); ls = lbounds as in*  
*if ls = []*  
*then let ds = filter (not ∘ is-Le) as in*  
*Disj [0..<d] (λn. [list-conj (map (subst n []) ds]))*  
*else*  
*Disj [0..<d] (λn.*  
*Disj ls (λ(li,lks).*  
*list-conj (map (subst (li+n) lks) as))))*  
  
*Disj is f = list-disj (map f is)*

- 1 Logical Framework
- 2 Dense Linear Orders
- 3 Linear Real Arithmetic
- 4 Presburger Arithmetic**
  - “Presburger’s algorithm”
  - Cooper’s algorithm
  - Complexity
- 5 Beyond

# Cooper's algorithm

No DNF, just NNF

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

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

[Cooper 72]  $\rightsquigarrow$  [Ferrante & Rackoff 75]

- 1 Logical Framework
- 2 Dense Linear Orders
- 3 Linear Real Arithmetic
- 4 Presburger Arithmetic**
  - “Presburger’s algorithm”
  - Cooper’s algorithm
  - Complexity**
- 5 Beyond

# Worst case complexity

of algorithms

Presburger Exponential blowup  
for every quantifier alternation

$\implies$  non-elementary

Cooper  $2^{2^{2^{cn}}}$  [Oppen 73/78]

- ① Logical Framework
- ② Dense Linear Orders
- ③ Linear Real Arithmetic
- ④ Presburger Arithmetic
- ⑤ Beyond

# Beyond

Mixed integer/real linear arithmetic ( $\lfloor \rfloor$ )

Algorithm: Weisspfenning

Reflection: Chaieb

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

Algorithms: Tarski, Cohen/Hörmander,  
Collins (CAD, doubly exponential)

LCF tactic: McLaughlin&Harrison

Reflection: Mahboubi (CAD, *partial!*)

**The future is bright for reflection**

But optimization is of the essence

Imperative features?