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.

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

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

Syntax

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

Quantifiers: *de Bruijn notation!*

$$\begin{array}{l} \mathit{ExQ}\ (\mathit{ExQ}\ \ldots\ \mathit{0}\ \ldots\ \mathit{1}\ \ldots) \\ \approx\ \exists x_1.\exists x_0.\ \ldots x_0 \ldots x_1 \ldots \end{array}$$

Abbreviations: $\mathit{AllQ}\ \varphi = \mathit{Neg}(\mathit{ExQ}(\mathit{Neg}\ \varphi))$, ...

Auxiliary functions

$$\textit{list-conj} :: \alpha\ \textit{fm list} \Rightarrow \alpha\ \textit{fm}$$
$$\textit{list-disj} :: \alpha\ \textit{fm list} \Rightarrow \alpha\ \textit{fm}$$

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

$$\begin{aligned} &\textit{and TrueF}\ \varphi &&= \varphi \\ &\textit{and}\ \varphi\ \textit{TrueF} &&= \varphi \\ &\textit{and}\ \varphi_1\ \varphi_2 &&= \textit{And}\ \varphi_1\ \varphi_2 \end{aligned}$$

DNF

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

$$
\begin{aligned}
&dnf\ TrueF = [[]]\\
&dnf\ FalseF = []\\
&dnf\ (Atom\ \varphi) = [[\varphi]]\\
&dnf\ (Or\ \varphi_1\ \varphi_2) = dnf\ \varphi_1\ @\ dnf\ \varphi_2\\
&dnf\ (And\ \varphi_1\ \varphi_2) =\\
&\quad [d_1\ @\ d_2.\ d_1 \leftarrow dnf\ \varphi_1,\ d_2 \leftarrow dnf\ \varphi_2]
\end{aligned}
$$

Assumes negation normal form!

More normal forms

in-nnf :: α *fm* $\Rightarrow$ *bool*
"Does not contain *Neg*"

Note: $\not\leq \rightsquigarrow >$

qfree :: α *fm* $\Rightarrow$ *bool*
"Does not contain *ExQ*"

Atoms

- More than a type parameter α.
- 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:

I_a	$:: \alpha \Rightarrow \beta\ \mathit{list} \Rightarrow \mathit{bool}$	
aneg	$:: \alpha \Rightarrow \alpha\ \mathit{fm}$	
adepends	$:: \alpha \Rightarrow \mathit{bool}$	"Depends on x_0?"
adecr	$:: \alpha \Rightarrow \alpha$	"$x_{i+1} \mapsto x_i$"

Interpretation

$I :: \alpha\ fm \Rightarrow \beta\ list \Rightarrow bool$

$I\ (Atom\ a)\ xs = I_a\ a\ xs$
$I\ (And\ \varphi_1\ \varphi_2)\ xs = (I\ \varphi_1\ xs \wedge I\ \varphi_2\ xs)$
$I\ (ExQ\ \varphi)\ xs = (\exists\, x.\ I\ \varphi\ (x{\cdot}xs))$
...

Example:

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

Assumptions

Locale ATOM has assumptions:

$I\ (aneg\ a)\ xs = (\neg\ I_a\ a\ xs)$
$in\text{-}nnf\ (aneg\ a)$
$\ldots$

Must be discharged when locale is instantiated

NNF

$nnf :: \alpha\ fm \Rightarrow \alpha\ fm$

$$nnf\ (And\ \varphi_1\ \varphi_2) = And\ (nnf\ \varphi_1)\ (nnf\ \varphi_2)$$
$$nnf\ (Neg\ (Atom\ a)) = aneg\ a$$
$$nnf\ (Neg\ (And\ \varphi_1\ \varphi_2)) =$$
$$\quad Or\ (nnf\ (Neg\ \varphi_1))\ (nnf\ (Neg\ \varphi_2))$$
$$nnf\ (Neg\ (Neg\ \varphi)) = nnf\ \varphi$$

...

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

Lifting quantifier elimination

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

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

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

Apply qe bottom up:

$$ExQ\ \varphi \rightsquigarrow ExQ\ \psi \rightsquigarrow \psi'$$

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:

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

QE via DNF

formally

$lift\text{-}dnf\text{-}qe :: (\alpha\ list \Rightarrow \alpha\ fm) \Rightarrow \alpha\ fm \Rightarrow \alpha\ fm$

$$lift\text{-}dnf\text{-}qe\ qe\ (And\ \varphi_1\ \varphi_2) =$$
$$and\ (lift\text{-}dnf\text{-}qe\ qe\ \varphi_1)\ (lift\text{-}dnf\text{-}qe\ qe\ \varphi_2)$$

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

$$qelim\ qe\ as =$$
$$(let\ qf = qe\ [a \leftarrow as.\ adepends\ a];$$
$$\quad indep = [Atom(adecr\ a).\ a \leftarrow as, \neg\ adepends\ a]$$
$$\ in\ and\ qf\ (list\text{-}conj\ indep))$$

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

Problematic case: quantifier alternation:

$$\forall\exists\bigvee\bigwedge = \forall\bigvee\exists\bigwedge = \forall\bigvee\bigwedge =$$

$$\neg\exists\neg\bigvee\bigwedge = \neg\exists\bigwedge\bigvee = \neg\exists\bigvee\bigwedge$$

Conversion to NNF is linear

QE via NNF

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

$\textit{lift-nnf-qe}\ \textit{qe}\ (\textit{ExQ}\ \varphi) = \textit{qe}\ (\textit{nnf}\ (\textit{lift-nnf-qe}\ \textit{qe}\ \varphi))$

...

More efficient, but trickier for *qe*

Dense Linear Orders

without endpoints

Atoms: $x < y$

Axioms:

Dense:	$x < z \Longrightarrow \exists\, y.\ x < y < z$
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:

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

$$= (\max_i l_i < \min_j u_j) = (\bigwedge_{ij} l_i < u_j)$$

Atoms
formally

datatype *atom* = *Less nat nat*

$$Less\ m\ n \quad \approx \quad x_m < x_n$$

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

Quantifier elimination

formally

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

$$
\begin{array}{l}
\textit{qe-less as} = \\
\quad (\textit{if Less 0 0} \in \textit{as then FalseF else} \\
\quad\;\; \textit{let lbs} = [\textit{m}-\textit{1. Less m 0} \leftarrow \textit{as}]; \\
\quad\quad\;\; \textit{ubs} = [\textit{n}-\textit{1. Less 0 n} \leftarrow \textit{as}]; \\
\quad\quad\;\; \textit{pairs} = [\textit{Atom}(\textit{Less m n}).\ \textit{m} \leftarrow \textit{lbs},\ \textit{n} \leftarrow \textit{ubs}] \\
\quad\;\; \textit{in list-conj pairs})
\end{array}
$$

Adding "="

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

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

Instantiating locales

functions and thms

$\downarrow$

Locale

$\downarrow$

functions and thms

Intstantiating locale ATOM

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

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

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

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

Reflection in action

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

by def of *DLO.I* (reversed)

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

by $\mathit{DLO.I}\ (\mathit{dlo\text{-}qe}\ \varphi)\ xs = \mathit{DLO.I}\ \varphi\ xs$

$= \ \mathit{DLO.I}\ (\mathit{dlo\text{-}qe}\ (\mathit{ExQ}\ \ldots))\ [s,t]$

by evaluation of *dlo-qe*

$= \ \mathit{DLO.I}\ (\mathit{Less}\ 0\ 1)\ [s,t]$

by def of *DLO.I*

$= \ s < t$

Reflection abstractly

form

by def of I (reversed)

$=$ I rep [subterms]

by correctness of *simp*

$=$ I (*simp*(rep)) [subterms]

by evaluation of *simp*

$=$ I rep' [subterms]

by def of I

$=$ 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
$\Longrightarrow$ non-elementary

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

Quantifier elimination TIME($2^{p(n)}$) (?)

Certificates for DPs

1. Unverified computation of certificate C for formula ϕ (external, fast)
2. Verified check that C indeed proves ϕ (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: QE($\exists\overline{x}.\phi$) yields *False* iff it constructs a cycle ($x < x$)

DP for unquantified formulae: To prove ϕ, prove unsatisfiability of each disjunct of DNF($\neg\phi$)

Certificates for DLO

Formally

Certificate checkers:

$\mathit{cycle}\ [a_1, \ldots, a_m]\ [i_1, \ldots, i_n]$

iff $[a_{i_1}, \ldots, a_{i_n}]$ forms a cycle.

$\mathit{cyclic\text{-}dnf}\ [\mathit{as}_1, \ldots, \mathit{as}_n]$

iff $\exists\ \mathit{is}_1, \ldots, \mathit{is}_n.\ \mathit{cycle}\ \mathit{as}_1\ \mathit{is}_1 \wedge \ldots \wedge \mathit{cycle}\ \mathit{as}_n\ \mathit{is}_n$

Correctness theorem:

$\mathit{qfree}\ \varphi \wedge \mathit{cyclic\text{-}dnf}\ (\mathit{dnf}(\mathit{DLO.nnf}\ \varphi)) \Longrightarrow$
$\neg\ \mathit{DLO.I}\ \varphi\ \mathit{xs}$

Demo

Works for . . .

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

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

$$
\begin{array}{rll}
 & \exists x.\;\; 3 < 2x + s & \wedge\; 5 < -3x + t \\
= & \exists x.\;\; 9 < 6x + 3s & \wedge\; 10 < -6x + 2t \\
= & 19 < 3s + 2t &
\end{array}
$$

Lower/upper bound view:

$$
\begin{array}{rll}
 & \exists x.\;\; 3 < 2x + s & \wedge\; 5 < -3x + t \\
= & \exists x.\;\; 9 - 3s < 6x & \wedge\; 6x < 2t - 10 \\
= & 9 - 3s < 2t - 10 &
\end{array}
$$

Combine $+/-$ atoms (lower/upper bounds)
by unifying leading coefficients

Fourier-Motzkin elimination

in general

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

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: $\mathit{Less}\ r\ [c_0, \ldots, c_n]$

Note:

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

Lists as vectors

Addition and subtraction

$$[c_0, \ldots] + [d_0, \ldots] = [c_0 + d_0, \ldots]$$
$$[c_0, \ldots] - [d_0, \ldots] = [c_0 - d_0, \ldots]$$

Multiplication with scalar

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

Inner product

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

Interpreting atoms

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

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

Instantiating ATOM:
R: $ATOM[I_a \mapsto I_R, \ldots]$

Fourier-Motzkin elimination

formally

qe-less :: *atom list* ⇒ *atom fm*

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

$combine\ (r_1,\ c_1,\ cs_1)\ (r_2,\ c_2,\ cs_2) =$
$Less\ (c_1 * r_2 - c_2 * r_1)\ (c_1 *_s cs_2 - c_2 *_s cs_1)$

Adding $Eq\ r\ cs$

As for DLO:

$$\begin{array}{l} qe\text{-}less\text{-}eq\ as = \\ \quad (case\ filter\ is\text{-}Eq\ as\ of\ [] \Rightarrow qe\text{-}less\ as \\ \quad\ \ |\ Eq\ r\ (c{\cdot}cs) \cdot eqs \Rightarrow \ldots\ subst\ \ldots \end{array}$$

Correctness

As for DLO:

Prove: $\ldots \Longrightarrow$ $R.I\ (\mathit{qe\text{-}less\text{-}eq}\ \mathit{as})\ \mathit{xs} =$
$(\exists\, x. \forall\, a \in \mathit{as}.\ I_R\ a\ (x \cdot \mathit{xs}))$

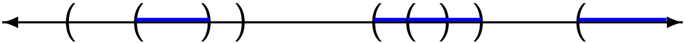
Define: $\mathit{lin\text{-}qe} = R.\mathit{lift\text{-}dnf\text{-}qe}\ \mathit{qe\text{-}less\text{-}eq}$

Obtain: $R.I\ (\mathit{lin\text{-}qe}\ \varphi)\ \mathit{xs} = R.I\ \varphi\ \mathit{xs}$

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: $\bigvee_i \bigwedge_j a_{ij}$
$\Longrightarrow P(x)$ is a finite union of finite intersections of intervals $(l, +\infty)$ and $(-\infty, u)$
$\Longrightarrow$ For each valuation of the other variables, $\{x \mid P(x)\}$ looks like this:



Every interval has upper/lower bound in $P(x)$
Problem: ls and us are symbolic

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

Ferrante and Rackoff

informal

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

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

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

Example

$$P(x) = x < y \wedge y < z \implies \begin{array}{l} P(-\infty) = y < z \\ P(+\infty) = \mathit{False} \end{array}$$

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

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

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

$$R.I\ (fr\text{-}qe\ \varphi)\ xs = R.I\ \varphi\ xs$$

Worst case complexity
of algorithms

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

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

Worst case complexity
of problems

Decision problem

$\mathsf{NTIME}(2^{cn}) < \mathsf{DP}(\mathbb{R},+) \leq \mathsf{SPACE}(2^{dn})$

[Fischer & Rabin 74] [Ferrante & Rackoff 75]

Quantifier elimination

$\mathsf{SPACE}(2^{2^{cn}}) \leq \mathsf{QE}(\mathbb{R},+) \leq \mathsf{SPACE}(2^{2^{cn}})$

$\mathsf{TIME}(2^{2^{cn}}) \leq \mathsf{QE}(\mathbb{R},+) \leq \mathsf{TIME}(2^{2^{cn}})$

[Weisspfenning 88]

Certificates
general case

Corollary There are no short ($\leq 2^{cn}$) certificates (proofs) that can be checked quickly (in polynomial time).

Applications

- Most theorem provers
- Proof Carrying Code
- Certified program analysis

Quantifier free case

Remember:

ϕ 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
$\sum_{i=1}^{n} c_i * a_i$ that is *contradictory* (eg $0 \leq -1$).

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

Certificate: $(c_1, \ldots, c_n)$

Finding the certificate

- By Fourier-Motzkin elimination ($\Rightarrow$ Lemma)
- By Linear Programming: $\bigwedge_{i=1}^{n} r_i \leq cs_i \odot xs$
  $\rightsquigarrow Ax \geq b$ 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 $\exists y \geq \overline{0}.\ A^T y = \overline{0} \wedge b^T y < 0.$

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

Find certificate by (eg) Simplex

Complexity

Corollary Implications $(\bigwedge_{i=1}^{n} a_i) \longrightarrow a$ of linear inequalitities can be proved in polynomial time.

Checking the certificate

$$\begin{array}{l}
\textit{check as } y = ((\forall\, c \in y.\ c \geq 0) \wedge \\
(\textit{let } b = \textit{map lhs as}; \\
\qquad A = \textit{map rhs as}; \\
\qquad by = b \odot y; \\
\qquad Ay = [cs \odot y.\ cs \leftarrow A]; \\
\ \textit{in } (\forall\, c \in Ay.\ c = 0) \wedge \\
\quad (by < 0 \vee (\forall\, a \in as.\ \textit{is-Eq } a) \wedge by \neq 0))
\end{array}$$

Lemma $\textit{check as cs} \Longrightarrow \exists\, a{\in}as.\ \neg\ I_R\ a\ xs$

Atoms

$$i \leq k_0 * x_0 + \cdots k_n * x_n$$
$$d \mid i + k_0 * x_0 + \cdots k_n * x_n$$

where $d, i, k_n, x_n \in \mathbb{Z}$

Presburger's algorithm
by example

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

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 $*$) $\leadsto Q(m * x)$
- $R(x) := Q(x) \wedge m | x$
- Let δ 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's algorithm

The core, formally

```
qe as =
(let d = lcms(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)
```

Cooper's algorithm

No DNF, just NNF

$\exists i. P(i)$

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

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

Worst case complexity
of algorithms

Presburger Exponential blowup
for every quantifier alternation
$\Longrightarrow$ non-elementary

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

Worst case complexity

of problems

Decision problem

$\mathsf{NTIME}(2^{2^{cn}}) < \mathsf{DP}(\mathbb{Z},+) \leq \mathsf{SPACE}(2^{2^{dn}})$

[Fischer & Rabin 74] [Ferrante & Rackoff 75]

Quantifier elimination

$\mathsf{QE}(\mathbb{Z},+) \leq \mathsf{TIME}(2^{2^{2^{cn}}})$ [Oppen 78]

One exponential up from $\mathbb{R}$

Quantifier free case

ϕ is unsatisfiable over $\mathbb{Z}$
if it is unsatisfiable over $\mathbb{R}$

Popular!

Alternatives

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

$\mathrm{DP}(\mathbb{Z}, +)$ Finite automata

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

Reflection?

Beyond

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

Algorithm: Weisspfenning
Reflection: Chaieb

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

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