Towards Practical Provably Correct Algorithms for Real Quantifier Elimination

Katherine Cordwell

Carnegie Mellon University

This material in this talk is based upon work supported by the NSF under Grant No. CNS-1739629, the NSF Graduate Research Fellowship Program under Grants Nos. DGE1252522 and DGE1745016, by A*STAR Singapore, and by the AFOSR under grant number FA9550-16-1-0288. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF, AFOSR, or A*STAR Singapore.

Problem

- Real arithmetic questions involving the **∃ (exists) and ∀ (for all) quantifiers** (ranging over the reals) are difficult for computers
- **Quantifier elimination (QE)**: The process of transforming a quantified statement into a ***logically equivalent*** quantifier-free statement

Examples

Example

$$\forall x.\ x^2 + 1 > 0$$

QE

True

Example*

$$\forall x \forall y.\ ((x^2 + ay^2 \leq 1) \Rightarrow (ax^2 - a^2xy + 2 \geq 0))$$

QE

$$(a \geq 0) \text{ and } (a^3 - 8a - 16 \leq 0)$$

QE is identifying exactly what conditions on a will make the original formula true!

*This example is taken from some of Pablo Parrilo's lecture notes (Lecture 18 of his 2006 course, "Algebraic Techniques and Semidefinite Optimization"). Accessible through his webpage: https://www.mit.edu/~parrilo/index.html

A Miraculous Result

- Algorithms for QE exist (Tarski, 1930)
- Algorithms for QE are complicated

Alfred Tarski

Terminology

- **Formulas:** Conjunctions and disjunctions of polynomial inequalities and equations (with rational coefficients)
- If a formula in a QE problem involves only one variable, we call it a **univariate** QE problem. Else it is a **multivariate** QE problem
- **Decision problems** are problems where all variables are quantified

Examples, Revisited

Example

$\forall x.\ x^2 + 1 > 0$

⬇ QE

True

A univariate decision problem

Example*

$$\forall x \forall y.\ ((x^2 + ay^2 \leq 1) \Rightarrow (ax^2 - a^2xy + 2 \geq 0))$$

⬇ QE

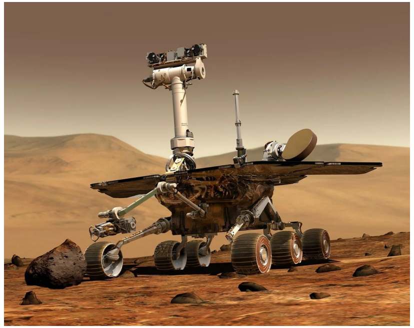
$$(a \geq 0) \text{ and } (a^3 - 8a - 16 \leq 0)$$

A multivariate QE question
***Not* a decision problem**

*This example is taken from some of Pablo Parrilo's lecture notes (Lecture 18 of his 2006 course, "Algebraic Techniques and Semidefinite Optimization"). Accessible through his webpage: https://www.mit.edu/~parrilo/index.html

Motivation

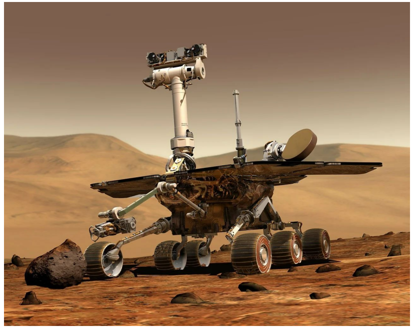
- Quantified statements arise in a number of applications
  - Geometry proofs
  - Stability analysis
  - Verification of cyber-physical systems (like robots!)



For more information, see:
Sturm, T. A Survey of Some Methods for Real Quantifier Elimination, Decision, and Satisfiability and Their Applications. *Math.Comput.Sci.* 11, 483–502 (2017).

Motivation

- Quantified statements arise in a number of applications
  - Geometry proofs
  - Stability analysis
  - Verification of cyber-physical systems (like robots!)
- Two conclusions
  - We want to know how to do QE
  - We want to be sure that we know how to do QE **correctly**



Doing QE correctly

- Formally verified QE algorithms
    - Implemented in theorem provers
    - Have proofs of correctness
    - Significantly more trustworthy than unverified algorithms

Doing QE correctly

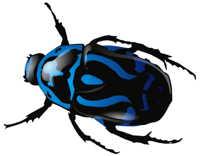
- Formally verified QE algorithms
  - Implemented in theorem provers
  - Have proofs of correctness
  - Significantly more trustworthy than unverified algorithms

There are QE algorithms (Tarski), we'll just verify them and be done…?

Doing QE correctly

- **Challenge:** Verified QE is much more difficult than unverified QE
- **Problem:** Dearth of efficient verified QE support
  - CPS theorem prover KeYmaera X outsources QE to unverified software
  - This can introduce bugs





Related Work

✓ = YES

✗ = NO

■ = IN BETWEEN

	Efficient?	Verified?	Multivariate case builds directly on univariate?
Cohen-Hörmander	✗	✓	✓
Tarski	✗	✓	✓
CAD	✓	■	✗

Our Approach

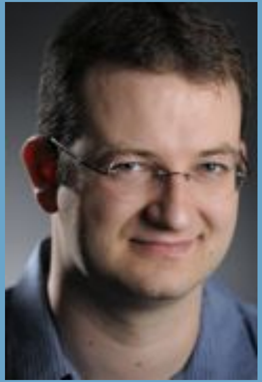
Twofold Approach

- Verify the **Ben-Or, Kozen, and Reif (BKR)** decision procedure (and its extension to a full QE algorithm by Renegar), **which fits in a sweet spot in between practicality and ease of formalization**
- Verify **virtual substitution (VS)**, an extremely efficient QE algorithm that works on a fragment of QE problems

Our Approach: Verifying Virtual Substitution (VS)

Matias Scharager

Stefan Mitsch



André Platzer

Fabian Immler

M. Scharager, **K. Cordwell**, S. Mitsch, and A. Platzer. Verified Quadratic Virtual Substitution for Real Arithmetic. Accepted to Formal Methods (FM) 2021, to appear.

What is Virtual Substitution?

- A **highly efficient** QE method that works on a fragment of QE problems
- Targets problems with **low-degree polynomials** (linear or quadratic)
- Two flavors: Equality VS and General VS

Equality Virtual Substitution

- Works when a formula has a linear or quadratic **equation**:

$$\exists\, x.\ (ax^2 + bx + c = 0 \wedge F)$$

- Can we directly substitute $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ into F?

Equality Virtual Substitution

- Works when a formula has a linear or quadratic **equation**:

$$\exists x. (ax^2 + bx + c = 0 \land F)$$

- Can we directly substitute $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ into F?

  Not without leaving the first-order logic of real arithmetic (FOL_R)

  Instead: "virtually" substitute

Equality Virtual Substitution

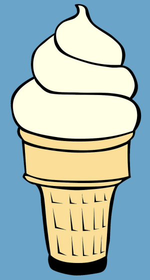
- Example: $\exists x. (x > 0 \land x^2 = 2 \land xy = 1)$
- We'd like to virtually substitute $x = \sqrt{2}$ into $xy = 1$

Equality Virtual Substitution

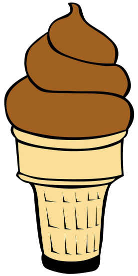
- Example: $\exists$ **x. (x > 0 ∧ x^2 = 2 ∧ xy = 1)**
- We'd like to virtually substitute x = $\sqrt{2}$ into xy = 1
- An appropriate FOL_R formula: **y > 0 ∧ y^2 = 1/2**

Key Takeaways

- VS "simulates" direct substitution in that it captures all of the logical meaning in direct substitution, but maintains formulas in FOL_R
- The presence of low-degree equalities automatically gives us a finite number of points to virtually substitute

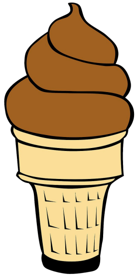


General Virtual Substitution



- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

General Virtual Substitution



- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

$$\exists x. (x^2 - 4 > 0 \wedge x^2 - 4x + 3 < 0)$$

General Virtual Substitution

- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

$$\exists\, x.\ (x^2 - 4 > 0 \wedge x^2 - 4x + 3 < 0)$$

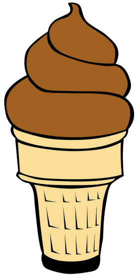
General Virtual Substitution

- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

$$\exists\, x.\ (x^2 - 4 > 0 \wedge x^2 - 4x + 3 < 0)$$

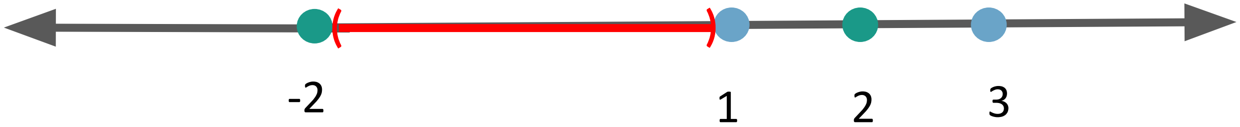
In every interval between the roots, the polynomials have constant sign

General Virtual Substitution



- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

$$\exists x. (x^2 - 4 > 0 \wedge x^2 - 4x + 3 < 0)$$

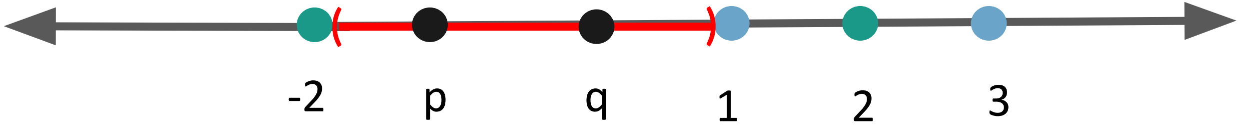


In every interval between the roots, the polynomials have constant sign

General Virtual Substitution

- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

$$\exists\, x.\ (x^2 - 4 > 0 \wedge x^2 - 4x + 3 < 0)$$



$p^2 - 4$ and $q^2 - 4$ have the same sign

$p^2 - 4p + 3$ and $q^2 - 4q + 3$ have the same sign

General Virtual Substitution

- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

$$\exists x. (x^2 - 4 > 0 \wedge x^2 - 4x + 3 < 0)$$

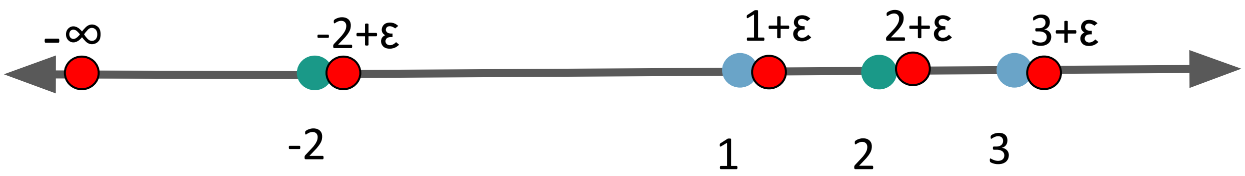
ANY point from this interval captures information for the entire interval!

This allows us to discretize with sample points

General Virtual Substitution

- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

$$\exists\, x.\ (x^2 - 4 > 0 \land x^2 - 4x + 3 < 0)$$

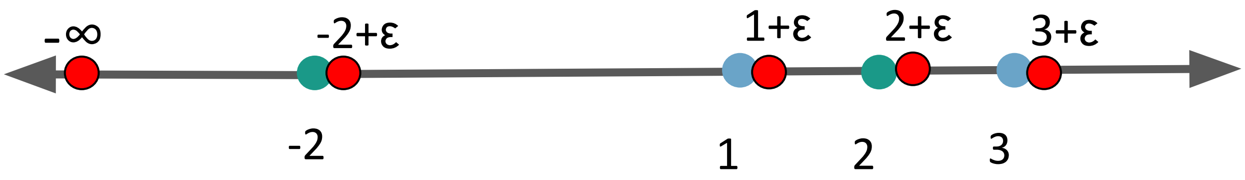


Here are the sample points VS will pick (in red)

General Virtual Substitution

- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

$$\exists x. (x^2 - 4 > 0 \wedge x^2 - 4x + 3 < 0)$$

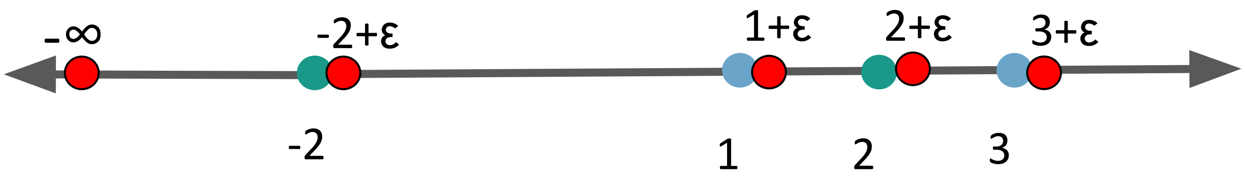


Why $-\infty$ and the ε's? Why not -3, 0, 1.5, 2.5, and 4?

General Virtual Substitution

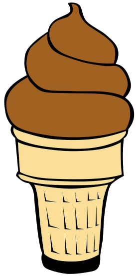
- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

$$\exists\, x.\ (x^2 - 4 > 0 \wedge x^2 - 4x + 3 < 0)$$



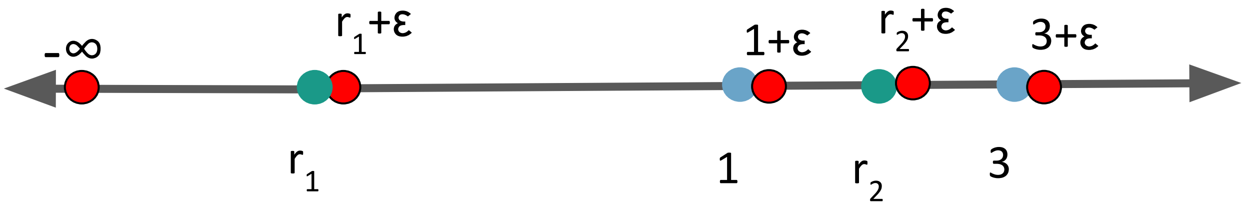
Why $-\infty$ and the ε's? Why not -3, 0, 1, 1.5, 2.5, 3, and 4?
VS needs to be able to generalize to arbitrary examples; can't overfit for the current example

General Virtual Substitution



- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

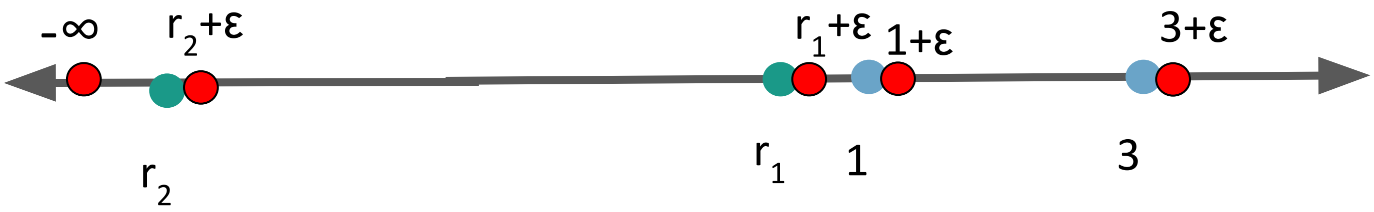
$$\exists\, x.\ (ax^2 + bx + c > 0 \wedge x^2 - 4x + 3 < 0)$$



General Virtual Substitution

- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

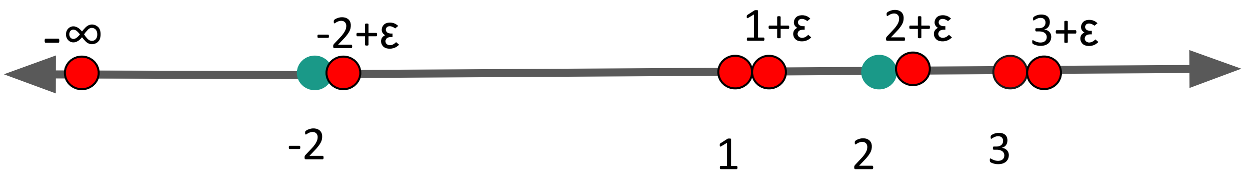
$$\exists\, x.\ (ax^2 + bx + c > 0 \wedge x^2 - 4x + 3 < 0)$$



General Virtual Substitution

- General VS allows for the presence of inequalities
- ...but then what points do we substitute?

$$\exists x. (x^2 - 4 > 0 \wedge x^2 - 4x + 3 < 0)$$



Note: We can only use general VS when we know all of the roots of the polynomials in our formula: i.e. when all of the polynomials are linear or quadratic in the variable of interest.

Formalizing this in Isabelle/HOL!

Substituting -∞

```
lemma infinity_evalUni: shows "(∃y. ∀x<y. aEvalUni At x) =
  (evalUni (substNegInfinityUni At) x)"
```

The intuition: VS for -∞ should be equivalent to sampling from in the leftmost interval on the number line

Substituting -∞

```
lemma infinity_evalUni: shows "(∃y. ∀x<y. aEvalUni At x) =
  (evalUni (substNegInfinityUni At) x)"
```

Checks whether ax^2 + bx + c satisfies the sign condition specified by At

At is a triple of real numbers (a, b, c) and a sign condition: <, =, ≤, or ≠

Substituting -∞

```
lemma infinity_evalUni: shows "(∃y. ∀x<y. aEvalUni At x) =
  (evalUni (substNegInfinityUni At) x)"
```

Decide: Is there some sufficiently negative y so that, for all $x < y$, $ax^2 + bx + c$ satisfies the sign condition specified by *At*?

Substituting -∞

```
lemma infinity_evalUni: shows "(∃y. ∀x<y. aEvalUni At x) =
  (evalUni (substNegInfinityUni At) x)"
```

Evaluate a formula at a point

Given *At*, virtually substitute -∞

Decide: Is there some sufficiently negative y so that, for all x < y, ax^2 + bx + c satisfies the sign condition specified by *At*?

Substituting -∞

```
lemma infinity_evalUni: shows "(∃y. ∀x<y. aEvalUni At x) =
   (evalUni (substNegInfinityUni At) x)"
```

Evaluate a formula at a point

Given *At*, virtually substitute -∞

Decide: Is there some sufficiently negative y so that ax^2 + bx + c satisfies the sign condition specified by *At*?

The intuition: VS for -∞ should be equivalent to sampling from in the leftmost interval on the number line

Substituting -∞

```
lemma infinity_evalUni: shows "(∃y. ∀x<y. aEvalUni At x) =
  (evalUni (substNegInfinityUni At) x)"
```

Why is this substitution lemma only stated for univariate polynomials?

Substituting -∞

```
lemma infinity_evalUni: shows "(∃y. ∀x<y. aEvalUni At x) =
  (evalUni (substNegInfinityUni At) x)"
```

Why is this substitution lemma only stated for univariate polynomials?

A clever trick: The multivariate VS proof proceeds by valuation. So we can state most of our correctness lemmas for univariate polynomials. Then later, naturally extend them to multivariate.

Substituting ε

```
lemma infinitesimal_quad:
  fixes A B C D:: "real"
  assumes "D≠0"
  assumes "C≥0"
  shows "(∃y::real>((A+B * sqrt(C))/(D)).
      ∀x::real ∈{((A+B * sqrt(C))/(D))<..y}. aEvalUni At x)
      = (evalUni (substInfinitesimalQuadraticUni A B C D At) x
```

This is stated for a quadratic polynomial, with a root (A + B*sqrt(C))/D

Substituting ε

```
lemma infinitesimal_quad:
  fixes A B C D:: "real"
  assumes "D≠0"
  assumes "C≥0"
  shows "(∃y::real>((A+B * sqrt(C))/(D)).
      ∀x::real ∈{((A+B * sqrt(C))/(D))<..y}. aEvalUni At x)
      = (evalUni (substInfinitesimalQuadraticUni A B C D At) x
```

This is stated for a quadratic polynomial, with a root (A + B*sqrt(C))/D

VS of (A + B*sqrt(C)/D) + ε is equivalent to **sampling from the interval directly "above" (A + B*sqrt(C)/D)**

Formalizing VS: Related Work

- We formalize both Equality VS and General VS
- Related work: Tobias Nipkow (linear VS), Amine Chaeib (quadratic equality VS)
  - Nipkow's work is more theoretically oriented
  - Chaeib's formalization is not publicly available; we chose not to build on it

Formalizing VS: Related Work

- We formalize both Equality VS and General VS
- Related work: Tobias Nipkow (linear VS), Amine Chaeib (quadratic equality VS)
  - Nipkow's work is more theoretically oriented
  - Chaeib's formalization is not publicly available; we chose not to build on it

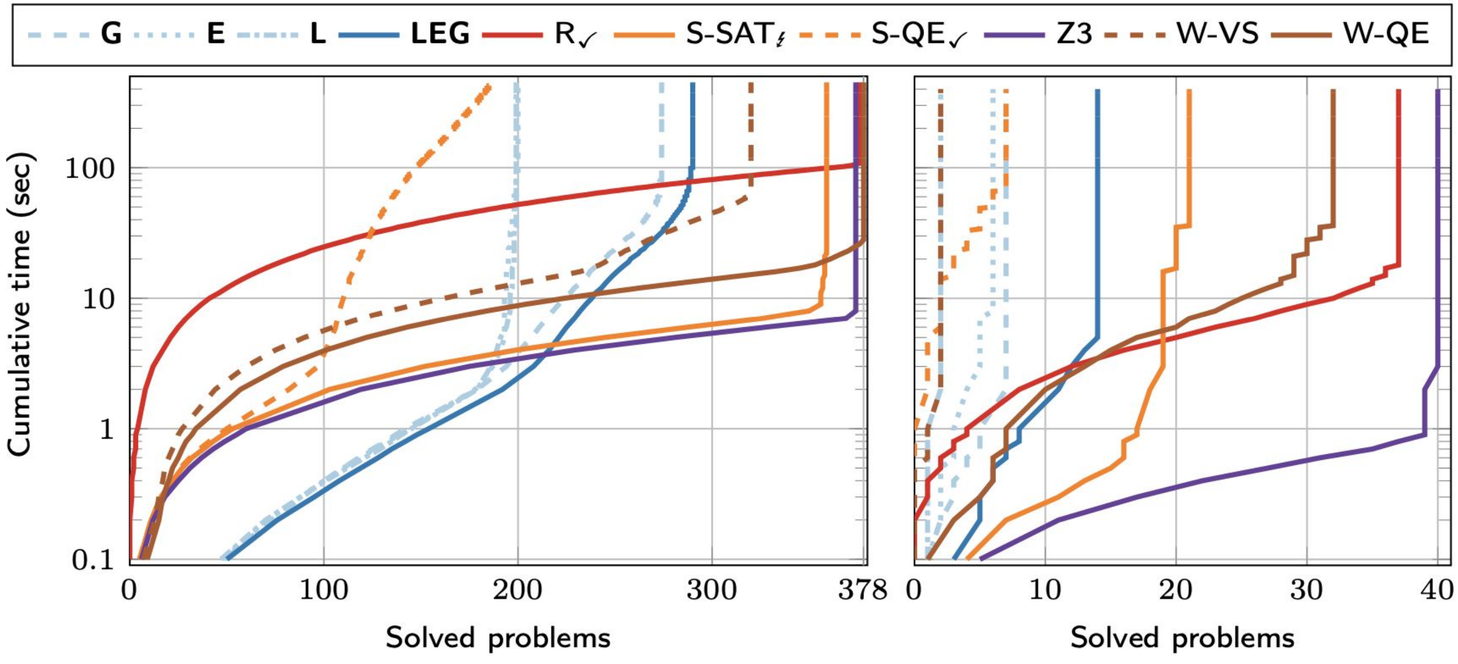
Different goals: We want practical verified real QE

Code Export and Experiments

Formalizing VS

- We export our code to SML for experimentation
  - 378 benchmarks from the literature
  - Compare to Mathematica, Z3, Redlog, SMT-RAT

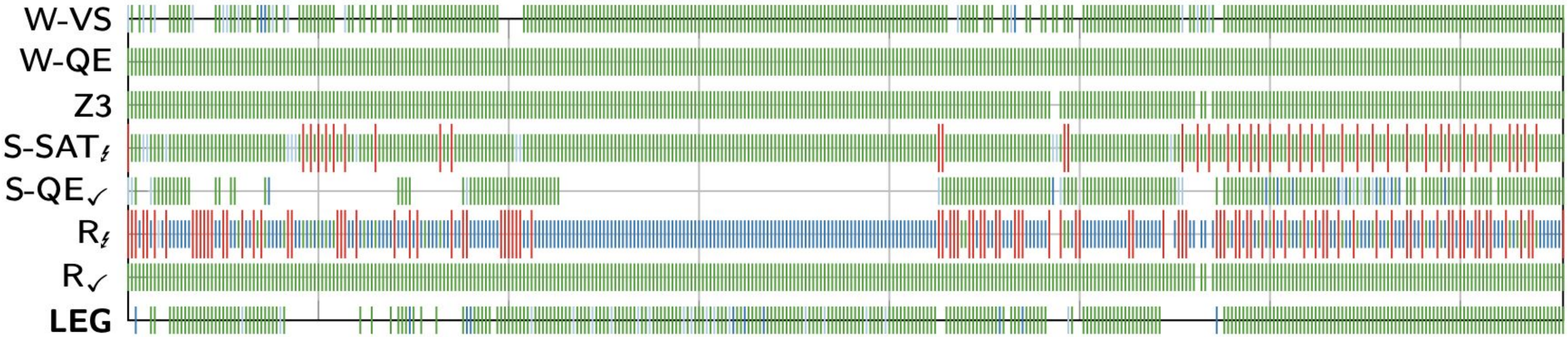
Some Experimental Results

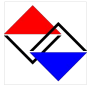


(a) CADE09 (378 examples) (b) Economics (45 examples)

Some Experimental Results

We find longstanding errors in existing tools with a consistency comparison:



W:  Z3:  S:  R:  **LEG**: us!

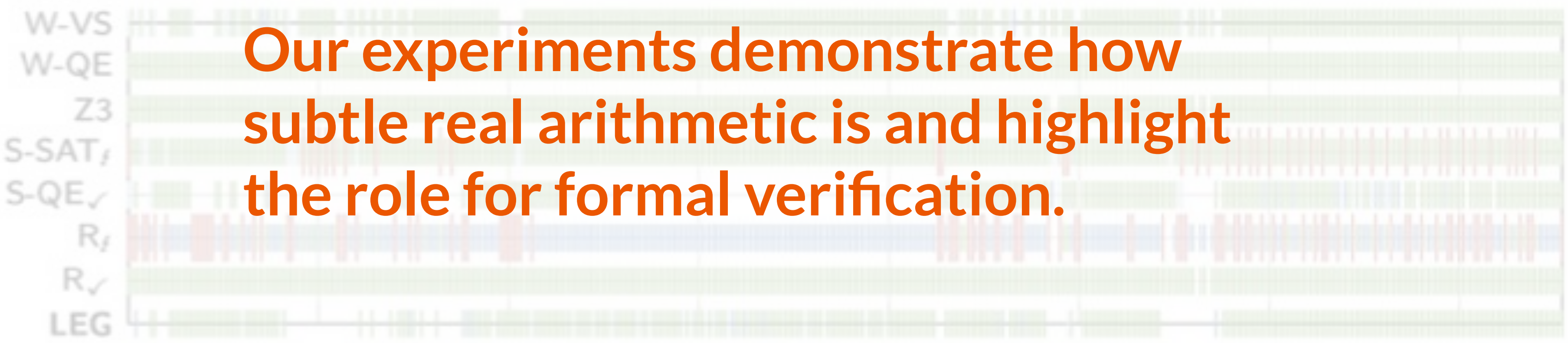
Green: consistent

Blue: only one solved

Red: inconsistent

Some Experimental Results

We find longstanding errors in existing tools with a consistency comparison:



Our experiments demonstrate how subtle real arithmetic is and highlight the role for formal verification.

W: Z3: S: R: **LEG**: us!

Green: consistent Blue: only one solved Red: inconsistent

Our Approach: Verifying BKR/Renegar

Yong Kiam Tan

André Platzer

Related Work

✓ = YES
✗ = NO
▭ = IN BETWEEN

	Efficient?	Verified?	Multivariate case builds directly on univariate?
Cohen-Hörmander	✗	✓	✓
Tarski	✗	✓	✓
CAD	✓	▭	✗
BKR & Renegar Potential sweet spot!	▭	✗	✓

We formally verify* the univariate cases of BKR and Renegar in Isabelle/HOL.

K. Cordwell, Y. K. Tan, and A. Platzer. A Verified Decision Procedure for Univariate Real Arithmetic with the BKR Algorithm. Interactive Theorem Proving (ITP) 2021.

*Available on the Archive of Formal Proofs at: https://www.isa-afp.org/entries/BenOr_Kozen_Reif.html

High-level Context

- ~7000 LOC
    - Algorithm: ~110 LOC
    - Matrix library extensions: ~1800 LOC

High-level Context

- ~7000 LOC
  - Algorithm: ~110 LOC
  - Matrix library extensions: ~1800 LOC
- Why Isabelle/HOL?
  - Well-suited to formalizing mathematics
  - Strong math libraries
  - Sledgehammer

Univariate BKR: Bird's Eye View

- Transform the problem:
  1. Decision problems to sign determination
  2. Sign determination to restricted sign determination
  3. To solve restricted sign determination, set up a matrix equation.

The main formalization challenge

Step 1: Decision to Sign Determination

- Solve decision problems by finding the ***consistent sign assignments (CSAs)*** for a set of polynomials (sign determination)

Definition (sign assignment for $\{g_1, \ldots, g_n\}$). A mapping $\sigma: \{g_1, \ldots, g_n\} \to \{+, -, 0\}$
σ is **consistent** if there is a real x where, for all i, the sign of $g_i(x)$ matches $\sigma(g_i)$.

Step 1: Decision to Sign Determination

- Solve decision problems by finding the ***consistent sign assignments (CSAs)*** for a set of polynomials (sign determination)

Decision Problem:
$\exists x. (x^2+1 \geq 0 \land 3x < 0)$

Find all consistent sign assignments for $x^2 + 1$ and $3x$

CSAs: (+, -), (+, 0), (+, +)

CSA (+, -) indicates the existence of a point k with $(k^2+1 \geq 0 \land 3k < 0)$

Correctness Results for Step 1

```
theorem decision_procedure:
  "(∀x::real. fml_sem fml x) ⟷ decide_universal fml"
  "(∃x::real. fml_sem fml x) ⟷ decide_existential fml"
```

Canonical semantics for formulas (defines what it means for a formula to hold at x in the standard way)

Our algorithms

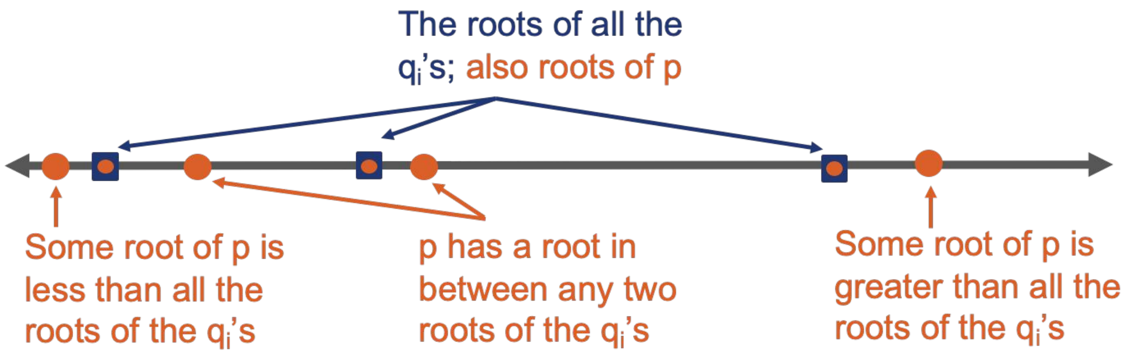
Step 2: Restricted Sign Determination

- Restrict sign determination to finding all **CSAs** for a set of univariate polynomials $\{q_1, \ldots, q_n\}$ ***at the roots of*** an auxiliary nonzero polynomial p

Technical detail: BKR imposes some conditions on $\{q_1, \ldots, q_n\}$, p

Step 2: Restricted Sign Determination

- Restrict sign determination to finding all **CSAs** for a set of univariate polynomials $\{q_1, \ldots, q_n\}$ ***at the roots of*** an auxiliary nonzero polynomial p



Correctness Results for Step 2

```
definition roots :: "real poly ⇒ real set" where "roots p = {x. poly p x = 0}"

definition consistent_signs_at_roots :: "real poly ⇒ real poly list ⇒ rat list set"
where "consistent_signs_at_roots p qs = (sgn_vec qs) ` (roots p)"
```

Plug in the roots to the q_i's, take the resulting signs

Solve for the roots of a polynomial

Correctness Results for Step 2

```
definition roots :: "real poly ⇒ real set" where "roots p = {x. poly p x = 0}"

definition consistent_signs_at_roots :: "real poly ⇒ real poly list ⇒ rat list set"
where "consistent_signs_at_roots p qs = (sgn_vec qs) ` (roots p)"

theorem find_consistent_signs_at_roots:
assumes "p ≠ 0"
assumes "⋀q. q ∈ set qs ⟹ coprime p q"
shows "set (find_consistent_signs_at_roots p qs) = consistent_signs_at_roots p qs"
```

our (constructive) algorithm

the nonconstructive definition

Step 3: The Matrix Equation

- Stores all relevant information for sign determination
- Idea dates back to Tarski; similarities to Cohen and Mahboubi's formalization*
- **But BKR does it efficiently**

Alfred Tarski

*Cyril Cohen and Assia Mahboubi. Formal proofs in real algebraic geometry: from ordered fields to quantifier elimination. Log. Methods Comput. Sci., 8(1), 2012. doi:10.2168/ LMCS-8(1:2)2012.

Step 3: The Matrix Equation

Find sign assignments to $q_1, \dots, q_n$ at the roots of p

Tarski

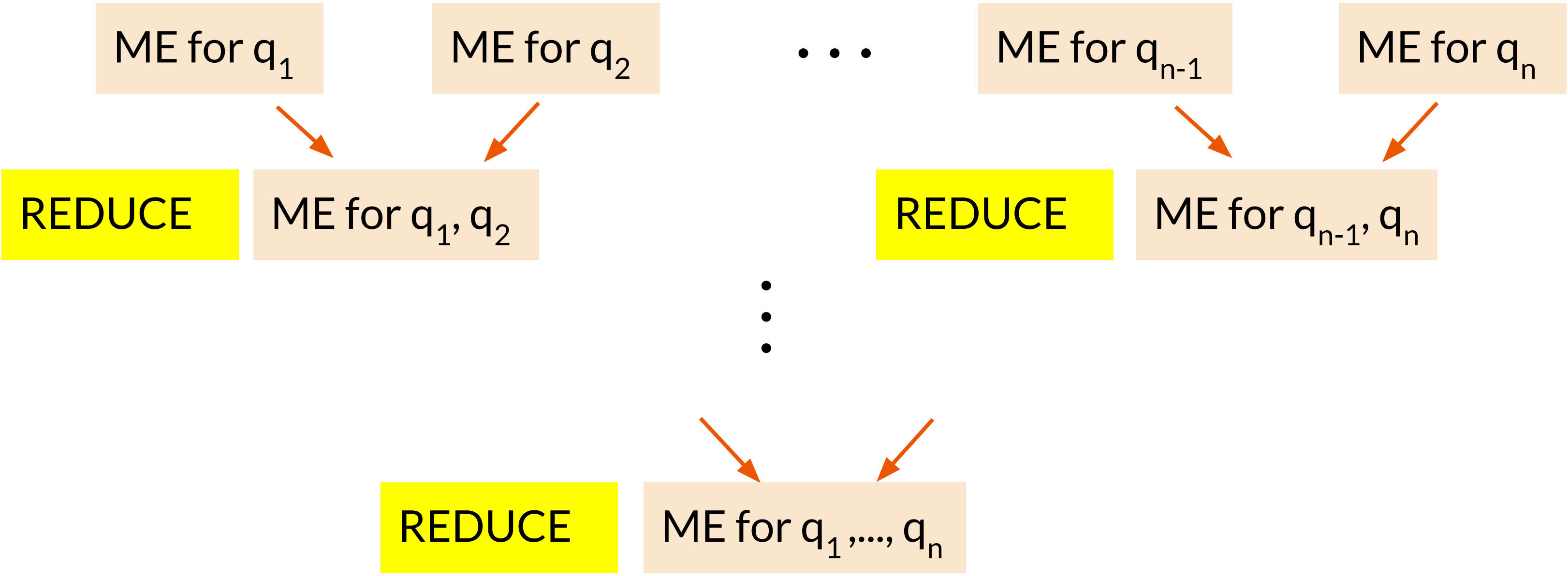
TQ stands for "Tarski query", refers to invoking the (computational) Sturm-Tarski theorem

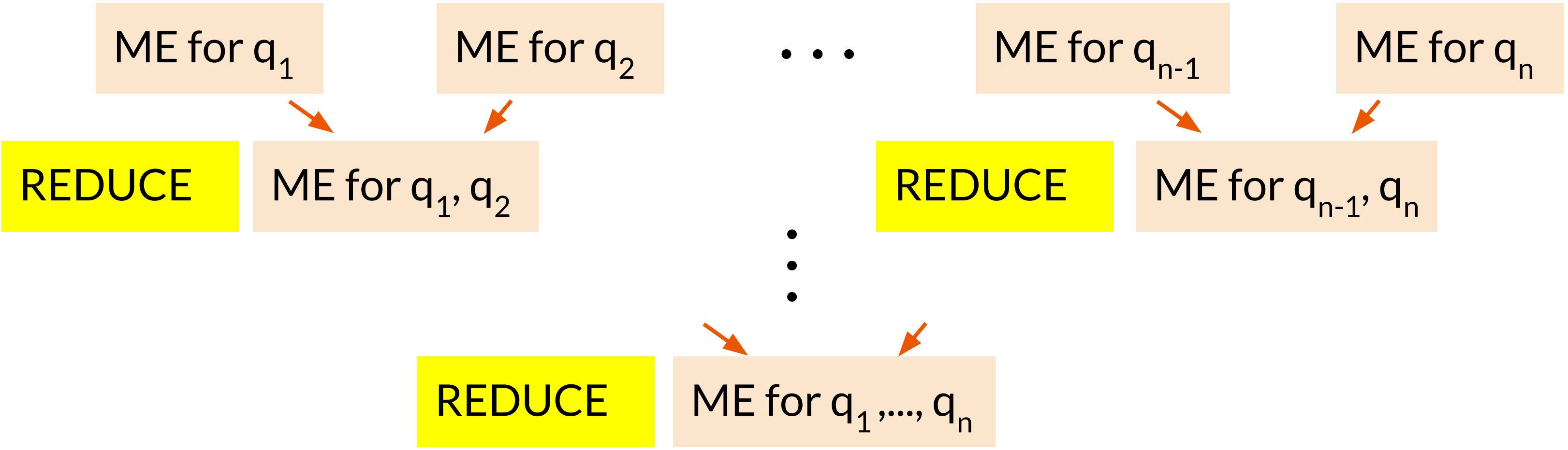
$$\begin{pmatrix} \text{\# of } (+,\dots,+,+) \\ \text{\# of } (+,\dots,+,-) \\ \vdots \\ \text{\# of } (-,\dots,-,-) \end{pmatrix} = M^{-1} * \begin{pmatrix} \text{TQ subset } 1 \\ \text{TQ subset } 2 \\ \vdots \\ \text{TQ subset } 2^n \end{pmatrix}$$

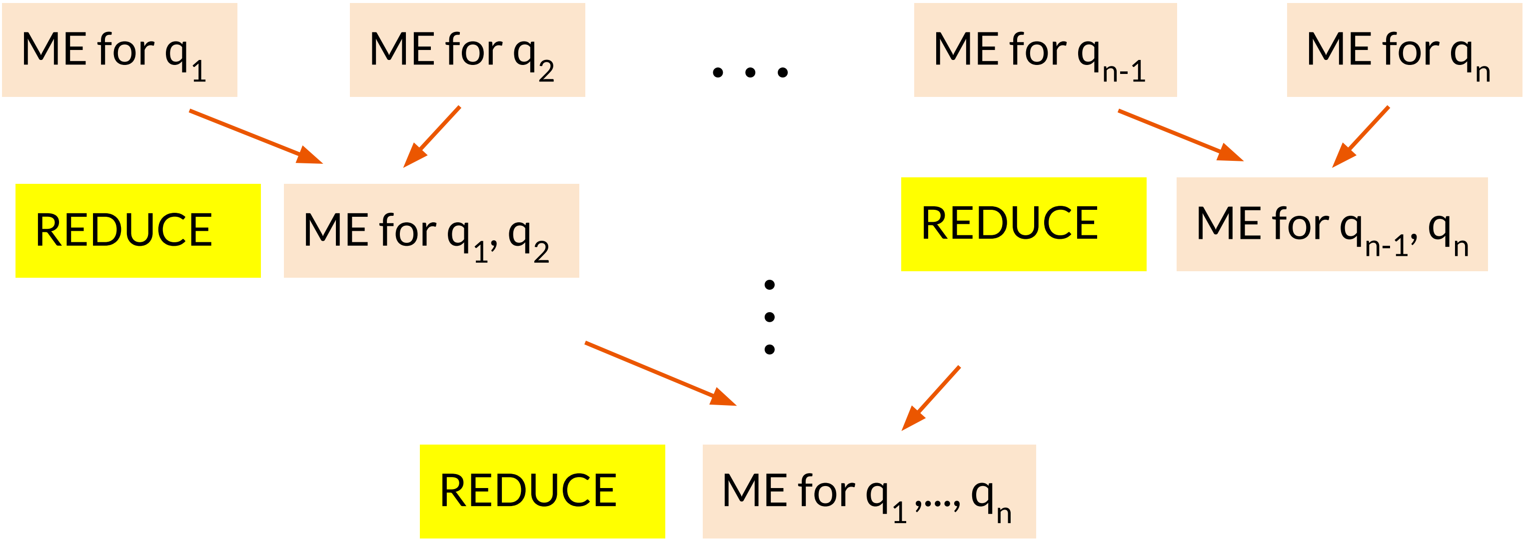
Invertible matrix
Size $2^n \times 2^n$
Can be computed

Step 3: The Matrix Equation

Find sign assignments to $q_1, \ldots, q_n$ at the roots of p
BKR builds its matrix equation (ME) inductively




ME for q_1
ME for q_2
. . .
ME for q_{n-1}
ME for q_n
REDUCE
ME for q_1, q_2
REDUCE
ME for q_{n-1}, q_n
REDUCE
ME for $q_1 ,..., q_n$


ME for q_1
ME for q_2
. . .
ME for q_{n-1}
ME for q_n
REDUCE
ME for q_1, q_2
REDUCE
ME for q_{n-1}, q_n
REDUCE
ME for q_1 ,..., q_n

Step 3: The Matrix Equation

After each combination, remove all inconsistent sign assignments
(reduction step)

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 1 \\ -1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}$$

Signs: ++, + - , - +, --

Signs: ++, + - , - +

Reflections on Formalizing the Matrix Equation

- Inductive construction, inductive proof!
    - It took some work to identify the right inductive invariant
    - The reduction step poses the biggest challenge
- The reduction step requires extra proofs

*Wenda Li. The Sturm-Tarski theorem. Archive of Formal Proofs, September 2014. https://isa-afp.org/entries/Sturm_Tarski.html, Formal proof development.

Reflections on Formalizing the Matrix Equation

- Isabelle/HOL has well-developed libraries
  - The Sturm-Tarski theorem is already formalized* (the key computational tool for the matrix equation)
  - A number of linear algebra libraries are available

*Wenda Li. The Sturm-Tarski theorem. Archive of Formal Proofs, September 2014. https://isa-afp.org/entries/Sturm_Tarski.html, Formal proof development.

Extending the Matrix Libraries

- We build on a matrix library by Thiemann and Yamada*
- Our additions (~1800 LOC):
  - A computational notion of the Kronecker product
  - An algorithm to extract a basis from the rows of a matrix
    - Involved proving that row rank equals column rank

*Rene Theimann and Akihisa Yamada. Matrices, Jordan normal forms, and spectral radius theory. *Archive of Formal Proofs,* August 2015.

Code Export and Experiments

Experiments with SML code

- We export our formally verified algorithm to SML for experimentation
- Compare to:
    - A naive (unverified) version of Tarski's algorithm
    - Li, Passmore, and Paulson*

*Wenda Li, Grant Olney Passmore, and Lawrence C. Paulson. Deciding univariate polynomial problems using untrusted certificates in Isabelle/HOL. J. Autom. Reason., 62(1):69–91, 2019.

Experiments with SML code

- We export our formally verified algorithm to SML for experimentation
- Compare to:
  - A naive (unverified) version of Tarski's algorithm
  - Li, Passmore, and Paulson*
- Li et. al is faster:
  - CAD is generally faster than BKR
  - Their procedure is highly optimized
  - They use Mathematica as an untrusted oracle

*Wenda Li, Grant Olney Passmore, and Lawrence C. Paulson. Deciding univariate polynomial problems using untrusted certificates in Isabelle/HOL. J. Autom. Reason., 62(1):69–91, 2019.

Experiments with SML code

*Compiled with mlton
*Run on a laptop
*Dashes indicate timeout
*Times in seconds

Benchmarks from [18]

Formula	#Poly	$\#N(p,q)$ (Naive)	$\#N(p,q)$ (BKR)	Time (Naive)	Time (BKR)	Time ([18])
ex1	4 (12)	20	31	0.003	0.006	3.020
ex2	5 (6)	576	180	5.780	0.442	3.407
ex3	4 (22)	112	120	1794.843	1865.313	3.580
ex4	5 (3)	112	95	0.461	0.261	3.828
ex5	8 (3)	576	219	28.608	8.333	3.806
ex6	22 (9)	50331648	-	-	-	6.187
ex7	10 (12)	6144	-	-	-	-
ex1 $\wedge$ 2	9 (12)	2816	298	317.432	3.027	3.033
ex1 $\wedge$ 2 $\wedge$ 4	13 (12)	28672	555	-	51.347	3.848
ex1 $\wedge$ 2 $\wedge$ 5	16 (12)	131072	826	-	436.575	3.711

3s startup time for Mathematica

[18] Wenda Li, Grant Olney Passmore, and Lawrence C. Paulson. Deciding univariate polynomial problems using untrusted certificates in Isabelle/HOL. J. Autom. Reason., 62(1):69–91, 2019.

Experiments with SML code

*Compiled with mlton
*Run on a laptop
*Dashes indicate timeout
*Times in seconds

Formula	#Poly	$\#N(p,q)$ (Naive)	$\#N(p,q)$ (BKR)	Time (Naive)	Time (BKR)	Time ([18])
ex1	4 (12)	20	31	0.003	0.006	3.020
ex2	5 (6)	576	180	5.780	0.442	3.407
ex3	4 (22)	112	120	1794.843	1865.313	3.580
ex4	5 (3)	112	95	0.461	0.261	3.828
ex5	8 (3)	576	219	28.608	8.333	3.806
ex6	22 (9)	50331648	-	-	-	6.187
ex7	10 (12)	6144	-	-	-	-
ex1 $\wedge$ 2	9 (12)	2816	298	317.432	3.027	3.033
ex1 $\wedge$ 2 $\wedge$ 4	13 (12)	28672	555	-	51.347	3.848
ex1 $\wedge$ 2 $\wedge$ 5	16 (12)	131072	826	-	436.575	3.711

[18] Wenda Li, Grant Olney Passmore, and Lawrence C. Paulson. Deciding univariate polynomial problems using untrusted certificates in Isabelle/HOL. J. Autom. Reason., 62(1):69–91, 2019.

Experiments with SML code

*Compiled with mlton
*Run on a laptop
*Dashes indicate timeout
*Times in seconds

Formula	#Poly	$\#N(p,q)$ (Naive)	$\#N(p,q)$ (BKR)	Time (Naive)	Time (BKR)	Time ([18])
ex1	4 (12)	20	31	0.003	0.006	3.020
ex2	5 (6)	576	180	5.780	0.442	3.407
ex3	4 (22)	112	120	1794.843	1865.313	3.580
ex4	5 (3)	112	95	0.461	0.261	3.828
ex5	8 (3)	576	219	28.608	8.333	3.806
ex6	22 (9)	50331648	-	-	-	6.187
ex7	10 (12)	6144	-	-	-	-
ex1 $\wedge$ 2	9 (12)	2816	298	317.432	3.027	3.033
ex1 $\wedge$ 2 $\wedge$ 4	13 (12)	28672	555	-	51.347	3.848
ex1 $\wedge$ 2 $\wedge$ 5	16 (12)	131072	826	-	436.575	3.711

[18] Wenda Li, Grant Olney Passmore, and Lawrence C. Paulson. Deciding univariate polynomial problems using untrusted certificates in Isabelle/HOL. J. Autom. Reason., 62(1):69–91, 2019.

Putting it all together: Future directions

Future Directions

BKR

- Optimize univariate BKR
- Formally verified complexity analysis (ambitious!)
- Formalizing multivariate BKR

VS

- Continue to optimize
- Add support for division
- Extend to higher-degree?

Future Directions

BKR

- Optimize univariate BKR
- Formally verified complexity analysis (ambitious!)
- Formalizing multivariate BKR

+

VS

- Continue to optimize
- Add support for division
- Extend to higher-degree?

Towards a practical verified QE method: link together BKR, VS

Conclusion

- We have formally verified the univariate case of BKR's QE algorithm
  - BKR hits a potential **sweet spot** in between practicality and ease of verification
- We have formally verified linear and quadratic VS, a **highly effective** (but limited) QE method
  - Our experiments demonstrate the role of verification for QE

Conclusion

- We have formally verified the univariate case of BKR's QE algorithm
  - BKR hits a potential **sweet spot** in between practicality and ease of verification
- We have formally verified linear and quadratic VS, a **highly effective** (but limited) QE method
  - Our experiments demonstrate the role of verification for QE

$\exists$ Questions?