A Modular, Efficient Formalisation of Real Algebraic Numbers

Wenda Li
in joint work with Larry Paulson

University of Cambridge

wl302@cam.ac.uk

January 17, 2016

What is real algebraic number

Real algebraic numbers ($\mathbb{R}_{alg}$) are real roots of non-zero polynomials with rational or integer coefficients.

- For example, $\sqrt{2}, \sqrt{3}, \sqrt{2}+\sqrt{3}$
- $$\mathbb{Q} \subset \mathbb{R}_{alg} \subset \mathbb{R}$$
- decidable arithmetic and ordering operations

Structure of our formalisation in Isabelle/HOL

We formalise $\mathbb{R}_{alg}$ on two levels:

- abstraction:
  - more mathematical and not considering executability
  - define an abstract type
  - show that it forms an ordered field
- implementation: restore executability

Abstraction: abstract type for $\mathbb{R}_{alg}$

We follow the classical definition:

typedef *alg* = *"{x::real. ∃p::int poly. p≠0*
∧ poly (of_int_poly p) x = 0}"

- *of_int_poly* converts coefficients of a polynomial from *int* to *real*
- *poly p x* means evaluating polynomial p at x

Abstraction: closed under arithmetic operations

To show $\mathbb{R}_{alg}$ is closed under addition, we need to show: given $\alpha, \beta \in \mathbb{R}$ and $P, Q \in \mathbb{Z}[x]$,

$$\frac{\begin{array}{c} P \neq 0 \wedge P(\alpha) = 0 \\ Q \neq 0 \wedge Q(\beta) = 0 \end{array}}{\exists R \in \mathbb{Z}[x]. \quad R \neq 0 \wedge R(\alpha + \beta) = 0}$$

- Constructive proof (based on resultants) exists.
- We choose to follow a classical (but simpler) proof in abstract algebra, as we do not need executability on this level.

Abstraction: closed under arithmetic operations (2)

The proof can be finished if we can show some linear dependence

$$\frac{\exists n. \quad \{(\alpha+\beta)^k \mid k \leq n\} \text{ is linearly dependent}}{\exists R \in \mathbb{Z}[x]. \quad R \neq 0 \wedge R(\alpha+\beta) = 0}$$

A set of vectors $\{\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_n}\}$ is *linearly dependent* if there exist scalars $a_1, a_2, ..., a_n$, at least one of which is non-zero, such that

$$a_1\mathbf{v_1} + a_2\mathbf{v_2} + ... + a_n\mathbf{v_n} = \mathbf{0}$$

Abstraction: closed under arithmetic operations (3)

$$\frac{\begin{array}{c} P \neq 0 \wedge P(\alpha) = 0 \\ Q \neq 0 \wedge Q(\beta) = 0 \\ \forall k. \quad (\alpha+\beta)^k \in \mathrm{span}\{\alpha^i \beta^j \mid i\ j \in \mathbb{N}. i \leq \mathsf{deg}(P) \wedge j \leq \mathsf{deg}(Q)\} \end{array}}{\exists n. \quad \{(\alpha+\beta)^k \mid k \leq n\} \text{ is linearly dependent}}$$

derive linear dependence, and hence complete the proof (of $\mathbb{R}_{alg}$ being closed under addition).

$\mathrm{span}(\{\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_n}\})$ is defined as

$$\{w \mid w = a_1\mathbf{v_1} + a_2\mathbf{v_2} + ... + a_n\mathbf{v_n}, \text{and } a_1,...,a_n \text{ are scalars}\}$$

Abstraction: $\mathbb{R}_{alg}$ forms an ordered field

- Similarly, we show $\mathbb{R}_{alg}$ is closed under subtraction and multiplication.
- Multiplicative inverse: reversing the list of polynomial coefficients
- Division: composition of multiplication and inverse
- Inherits ordering operations from $\mathbb{R}$

Finally, we show $\mathbb{R}_{alg}$ forms an ordered field

instantiation `alg :: linordered_field`

Implementation: finite encoding and pseudo-constructor

Encoding of $\mathbb{R}_{alg}$ = polynomial + root selection strategy (e.g. isolation interval, root index and Thom encoding)
We choose the isolation interval approach:

$$\text{Alg} :: \text{int poly} \Rightarrow \underbrace{\text{rat} \Rightarrow \text{rat}}_{\text{isolation interval}} \Rightarrow \textit{real}$$

`Alg p lb ub` is a pseudo-constructor and encodes the only root of polynomial p within the interval `(lb,ub)`.

Implementation: need to prove some code equations

To enable executable addition, we need to prove something like

lemma *[code]:*
"Alg p1 lb1 ub1 + Alg p2 lb2 ub2 = Alg p3 lb3 ub3"

How can we calculate *(p3,lb3,ub3)* using *(p1,lb1,ub1)* and *(p2,lb2,ub2)*?

- Do all calculations within HOL logic
- Or use untrusted external procedures, and adopt the result after checking its correctness.

We choose the second approach.

Implementation: how to validate untrusted results?

Given $\alpha =$`Alg p1 lb1 ub1`, $\beta =$`Alg p2 lb2 ub2` and `(p3,lb3,ub3)`, how can we check that $\alpha + \beta$ is the only root of `p3` within the interval `(lb3,ub3)`?

First, we need to check if $p3(\alpha + \beta) = 0$: deciding the sign of a bivariate polynomial $p3(x + y) \in \mathbb{Z}[x, y]$ at (α, β).

Implementation: bivariate sign determination procedure

Previously, we have built a **univariate** sign determination procedure for $P(x) \in \mathbb{Q}[x]$ at $\alpha \in \mathbb{R}_{alg}$ based on the Sturm-Tarski theorem.

- need Euclidean/pseudo division of polynomials when calculating remainder sequences
- use only rational arithmetic

For example,

value `"sgn_at [:-1,1:] (Alg [:-2,0,1:] 1 2)"`

which stands for the sign of $(x-1)[x \to \sqrt{2}]$ and returns `1`.

Implementation: bivariate sign determination procedure (2)

To use the univariate procedure, can we pretend $P(\alpha, x) \in \mathbb{Q}(\alpha)[x]$ is a univariate polynomial? We need algebraic arithmetic when doing polynomial divisions.

To eliminate arithmetic in $\mathbb{Q}(\alpha)$: given $P, Q \in \mathbb{Q}[x, y]$ and $\alpha \in \mathbb{R}_{alg}$,

$$\frac{\begin{array}{c}\deg_x(Q) = \deg(Q[y \to \alpha]) \\ \deg_x(P) = \deg(P[y \to \alpha])\end{array}}{P[y \to \alpha] \underbrace{\operatorname{pmod}}_{\text{arithmetic in } \mathbb{Q}(\alpha)} Q[y \to \alpha] = (P \underbrace{\operatorname{pmod}}_{\text{arithmetic in } \mathbb{Q}} Q)[y \to \alpha]}$$

where pmod is pseudo-division of polynomials.

Implementation: bivariate sign determination procedure (3)

We finish the bivariate sign determination procedure:

```
value "bsgn_at [:[:0,-1:],[:1:]:] (Alg [:-2,0,1:] 1 2)
                                  (Alg [:-3,0,1:] 1 2)"
```

which stands for the sign of $(x - y)[x \to \sqrt{2}, y \to \sqrt{3}]$ and returns *-1*.

Implementation: revised code equation

lemma *[code]:*

```
"Alg p1 lb1 ub1 + Alg p2 lb2 ub2 =
   (let
     (p3,lb3,ub3) = (*untrusted code*)
   in
     (if (*(p3,lb3,ub3) is correct*) then
       Alg p3 lb3 ub3
     else
       (*throws an exception*)"
```

Implementation: executability restored

value *"Alg [:-2,0,1:] 1 2 / Alg [:-3,0,1:] 1 2*
+ Alg [:-5,0,1:] 2 3 > Alg [:-7,0,2:] 1 2"

which stands for $\sqrt{2}/\sqrt{3} + \sqrt{5} > \sqrt{7/2}$ and returns *true*.

Experiment

Verified algebraic arithmetic is approximately 2 to 15 times slower than the unverified version.

The bivariate sign determination procedure is very efficient.

Remarks

Exact algebraic arithmetic is expensive!

Related work

Cyril Cohen's formulation in Coq

- for theoretical interest
- constructive proofs based on the resultant theory

Conclusion

We formalize real algebraic numbers on two levels:

- abstraction
  - abstract without considering executability
  - classical proofs to show that $\mathbb{R}_{alg}$ forms an ordered field
- implementation
  - restore executability
  - use a verified bivariate sign determination procedure and untrusted code