An application of constructive dependent type theory to certified computation over the reals

Holger Thies (Kyoto University)
j.w.w. Michal Konečný (Aston U) and Sewon Park (Kyoto U)

Jan 14, 2022

Second Korea Logic Day
Online

Exact Real Computation

- In exact real computation, real numbers are seen as primitive data-types and users do not have to worry about their representation.

Exact Real Computation

- In exact real computation, real numbers are seen as primitive data-types and users do not have to worry about their representation.
- Arithmetical operations are computed exactly, without any rounding errors.

Exact Real Computation

- In exact real computation, real numbers are seen as primitive data-types and users do not have to worry about their representation.
- Arithmetical operations are computed exactly, without any rounding errors.
- Realistic in the sense that programs can be executed on a computer.

Exact Real Computation

- In exact real computation, real numbers are seen as primitive data-types and users do not have to worry about their representation.
- Arithmetical operations are computed exactly, without any rounding errors.
- Realistic in the sense that programs can be executed on a computer.
- Many frameworks exist (iRRAM, AERN).

No Rounding errors

Rump's example

$$R(x, y) = (333.75 - x^2)y^6 + x^2(11x^2y^2 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y}$$

evaluated at $x = 77617$ and $y = 33096$.

No Rounding errors

Rump's example

$$R(x, y) = (333.75 - x^2)y^6 + x^2(11x^2y^2 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y}$$

evaluated at $x = 77617$ and $y = 33096$.

- Can be implemented in ERC frameworks directly
- Output up to any desired precision
- Simple mathematical proofs of correctness

Subtleties of Exact Real Computation: Limits

Fast converging Cauchy sequence: $|a_n - a| \leq 2^{-n}$
One of the most fundamental operations: Computing a limit of a fast converging sequence:

Subtleties of Exact Real Computation: Limits

Fast converging Cauchy sequence: $|a_n - a| \leq 2^{-n}$

One of the most fundamental operations: Computing a limit of a fast converging sequence:

```
sqrt_approx x n =
 let heron_step y = (y + x/y)/2 in
 (iterate heron_step 1) !! n
```

Subtleties of Exact Real Computation: Limits

Fast converging Cauchy sequence: $|a_n - a| \leq 2^{-n}$

One of the most fundamental operations: Computing a limit of a fast converging sequence:

```
sqrt_approx x n =
 let heron_step y = (y + x/y)/2 in
 (iterate heron_step 1) !! n

sqrt_approx_fast x n =
 sqrt_approx x (1 + (integerLog2 (n+1)))
```

Subtleties of Exact Real Computation: Limits

Fast converging Cauchy sequence: $|a_n - a| \leq 2^{-n}$

One of the most fundamental operations: Computing a limit of a fast converging sequence:

```
sqrt_approx x n =
 let heron_step y = (y + x/y)/2 in
 (iterate heron_step 1) !! n

sqrt_approx_fast x n =
 sqrt_approx x (1 + (integerLog2 (n+1)))

restr_sqrt x = -- restricted to 0.25 < x < 2
  limit $
    \n -> sqrt_approx_fast x n
```

Semi-decidable comparisons

Comparison is partial. Kleenean comparison used instead:

```
...> pi > 0
{?(prec 36): CertainTrue}

...> pi == pi
{?(prec 36): TrueOrFalse}

...> pi == pi + 2^(-100)
{?(prec 36): TrueOrFalse}

...> (pi == pi + 2^(-100)) ? (prec 1000)
CertainFalse
```

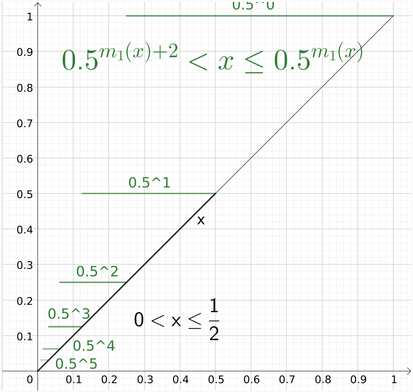
Multivalued selection

- Branching over semi-decidable comparison can be problematic
- `select` increases the precision until one of the Kleenans becomes `CertainTrue`.

```
max_nondeterministic x y =
 limit $ \n ->
   let e = 0.5^n in
   if select (x > y - e) (y > x - e)
        then x
        else y
```

Non-extensionality

```
magnitude1 x =
 integer $ fromJust $ List.findIndex id $ map test [0..]
 where
 test n = select (0.5^(n+2) < x) (x < 0.5^(n+1))
```



Certified exact real computation

Why certified exact real computation?
Limits, Non-determinism, etc. can easily go wrong.

Approaches to certified exact real computation:

- Program verification
  - ERC, Incone, ...
- Program extraction from (constructive) proofs
  - CorN, IFP, Minlog, **cAERN**, ...

Our approach to certified exact real computation

- Efficient implementations of exact real computation exist (e.g. iRRAM, AERN, Ariadne)

Our approach to certified exact real computation

- Efficient implementations of exact real computation exist (e.g. iRRAM, AERN, Ariadne)
- Verifying those implementations from scratch is difficult

Our approach to certified exact real computation

- Efficient implementations of exact real computation exist (e.g. iRRAM, AERN, Ariadne)
- Verifying those implementations from scratch is difficult
- Fully verified implementations exist, but are usually much slower

Our approach to certified exact real computation

- Efficient implementations of exact real computation exist (e.g. iRRAM, AERN, Ariadne)
- Verifying those implementations from scratch is difficult
- Fully verified implementations exist, but are usually much slower
- Our approach: "Axiomatize" some of the basic operations of ERC (arithmetic, limits, . . .)

Our approach to certified exact real computation

- Efficient implementations of exact real computation exist (e.g. iRRAM, AERN, Ariadne)
- Verifying those implementations from scratch is difficult
- Fully verified implementations exist, but are usually much slower
- Our approach: "Axiomatize" some of the basic operations of ERC (arithmetic, limits, . . .)
- Extract axiomatized operations to corresponding operations in an ERC implementation (AERN)

Our approach to certified exact real computation

- Efficient implementations of exact real computation exist (e.g. iRRAM, AERN, Ariadne)
- Verifying those implementations from scratch is difficult
- Fully verified implementations exist, but are usually much slower
- Our approach: "Axiomatize" some of the basic operations of ERC (arithmetic, limits, ...)
- Extract axiomatized operations to corresponding operations in an ERC implementation (AERN)

Our approach to certified exact real computation

- Efficient implementations of exact real computation exist (e.g. iRRAM, AERN, Ariadne)
- Verifying those implementations from scratch is difficult
- Fully verified implementations exist, but are usually much slower
- Our approach: "Axiomatize" some of the basic operations of ERC (arithmetic, limits, . . .)
- Extract axiomatized operations to corresponding operations in an ERC implementation (AERN)

Michal Konečný, Sewon Park, and Holger Thies.
Axiomatic Reals and Certified Efficient Exact Real Computation.
WoLLIC 2021. Springer, Cham, 2021.

Aims of our development

- Reliability

Aims of our development

- Reliability
  - Readable specification

Aims of our development

- Reliability
  - Readable specification
  - Small trusted base

Aims of our development

- Reliability
  - Readable specification
  - Small trusted base
- Smooth development

Aims of our development

- Reliability
    - Readable specification
    - Small trusted base
- Smooth development
    - Specification, algorithms and proofs

Aims of our development

- Reliability
  - Readable specification
  - Small trusted base
- Smooth development
  - Specification, algorithms and proofs
  - Readable algorithms

Aims of our development

- Reliability
  - Readable specification
  - Small trusted base
- Smooth development
  - Specification, algorithms and proofs
  - Readable algorithms
- Fast execution

Aims of our development

- Reliability
  - Readable specification
  - Small trusted base
- Smooth development
  - Specification, algorithms and proofs
  - Readable algorithms
- Fast execution
  - Comparable to iRRAM, Ariadne, AERN

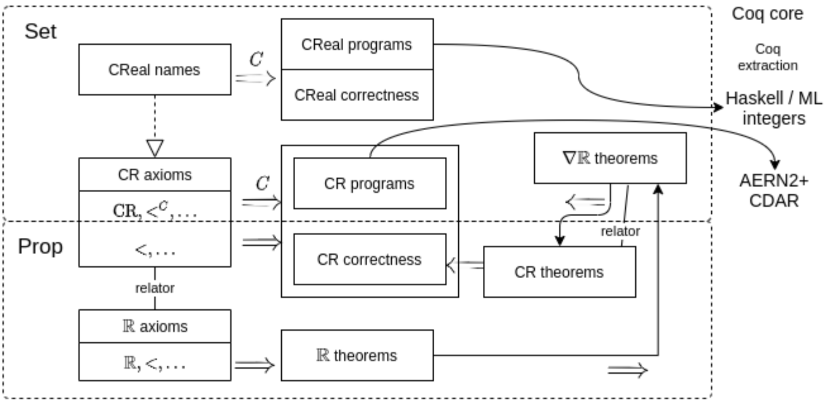
Aims of our development

- Reliability
  - Readable specification
  - Small trusted base
- Smooth development
  - Specification, algorithms and proofs
  - Readable algorithms
- Fast execution
  - Comparable to iRRAM, Ariadne, AERN

Aims of our development

- Reliability
  - Readable specification
  - Small trusted base
- Smooth development
  - Specification, algorithms and proofs
  - Readable algorithms
- Fast execution
  - Comparable to iRRAM, Ariadne, AERN

cAERN



Set
Coq core
CReal names
C
CReal programs
CReal correctness
Coq extraction
Haskell / ML integers
CR axioms
CR, $<^C, \ldots$
C
CR programs
$\nabla\mathbb{R}$ theorems
AERN2+ CDAR
relator
Prop
$<, \ldots$
CR correctness
CR theorems
relator
$\mathbb{R}$ axioms
$\mathbb{R}, <, \ldots$
$\mathbb{R}$ theorems

Background

- All our theory is implemented in the Coq proof assistant.
- However, we can look at the results in a simpler type-theoretical framework.

Background

- All our theory is implemented in the Coq proof assistant.
- However, we can look at the results in a simpler type-theoretical framework.

We assume to work in a dependent type theory with

- Basic Types $0, 1, 2, \mathbb{N}, \mathbb{Z}$,
- An impredicative universe Prop of classical propositions closed under $\to, \wedge, \vee, \exists, \Pi$,
- A predicative universe Type with type constructors $\to, \times, +, \Sigma, \Pi$

Background

Prop is classical

- Law of excluded middle $\Pi(P : \text{Prop}).\ P \vee \neg P$
- Propositional extensionality

$$\Pi(P, Q : \text{Prop}).\ (P \leftrightarrow Q) \rightarrow P = Q$$

- Countable choice

$$\begin{gathered}\Pi(A : \text{Type}).\ \Pi(P : \mathbb{N} \rightarrow A \rightarrow \text{Prop}).\\ (\Pi(n : \mathbb{N}).\ \exists(x : X).\ P\, n\, x) \rightarrow \exists(f : \mathbb{N} \rightarrow A).\ \Pi(n : \mathbb{N}).\ P\, n\, (f\ n)\end{gathered}$$

- Functional Extensionality, Markov principle

Axiomatized Reals

Field axioms:

$$
\begin{aligned}
&R : \text{Type} \\
&0 : R \\
&1 : R \\
&+ : R \to R \to R \\
&\quad\vdots
\end{aligned}
$$

Axiomatized Reals

Field axioms:

$$R : \text{Type}$$

$$0 : R$$

$$1 : R$$

$$+ : R \to R \to R$$

$$\vdots$$

Classical order:

$$<: R \to R \to \text{Prop}$$

$$\Pi(x, y : R).\ x < y \lor x = y \lor x > y$$

$$\vdots$$

Semi-decidability

We assume the exsistence of a type K of Kleeneans with two distinct constants $\mathsf{true}, \mathsf{false} : K$.

Semi-decidability

We assume the exsistence of a type K of Kleeneans with two distinct constants true, false : K.
We call a proposition P : Prop semi-decidable if there is a Kleenean t : K that identifies P.

Semi-decidability

We assume the exsistence of a type K of Kleeneans with two distinct constants true, false : K.
We call a proposition P : Prop semi-decidable if there is a Kleenean t : K that identifies P.
For example, we assume semi-decidability of comparisons:

$$\Pi(x, y : R).\ \Sigma(t : \mathsf{K}).\ x < y \leftrightarrow t = \mathit{true}$$

Nondeterminism

Nondeterminsm monad: For any type X : *Type* we automatically get a type $\mathsf{M}X$: *Type*.

$$A \to \mathsf{M}B \Rightarrow \text{nondeterministic function from } A \text{ to } B$$

$$\mathsf{M}(A + B) \Rightarrow \text{nondeterministic decision if } A \text{ or } B$$

Nondeterminism

Nondeterminsm monad: For any type X : *Type* we automatically get a type $\mathsf{M}X$: *Type*.

$$A \to \mathsf{M}B \Rightarrow \text{nondeterministic function from } A \text{ to } B$$

$$\mathsf{M}(A + B) \Rightarrow \text{nondeterministic decision if } A \text{ or } B$$

For any two semi-decidable decisions $x, y : \mathsf{K}$, if promised that either of x or y holds classically, we can nondeterministically decide whether x holds or y holds:

$$\Pi(x, y : \mathsf{K}).\ (x = \mathsf{true} \lor y = \mathsf{true}) \to \mathsf{M}\big(x = \mathsf{true} + y = \mathsf{true}\big)$$

Nondeterminism

Nondeterminsm monad: For any type X : *Type* we automatically get a type $\mathsf{M}X$: *Type*.

$$A \to \mathsf{M}B \Rightarrow \text{nondeterministic function from } A \text{ to } B$$

$$\mathsf{M}(A + B) \Rightarrow \text{nondeterministic decision if } A \text{ or } B$$

For any two semi-decidable decisions $x, y : \mathsf{K}$, if promised that either of x or y holds classically, we can nondeterministically decide whether x holds or y holds:

$$\Pi(x, y : \mathsf{K}).\ (x = \mathsf{true} \vee y = \mathsf{true}) \to \mathsf{M}\big(x = \mathsf{true} + y = \mathsf{true}\big)$$

Nondeterministic soft comparison:

$$\Pi(x, y : R).\ \Pi(n : \mathbb{N}).\ \mathsf{M}(x < y + 2^{-n} + y < x + 2^{-n})$$

Nondeterminsim and Limits

Consider the following three cases where we might want to compute a limit

1. A deterministic sequence of reals converges to a deterministic point: $f : \mathbb{N} \to R \rightsquigarrow x : R$

Nondeterminsim and Limits

Consider the following three cases where we might want to compute a limit

1. A deterministic sequence of reals converges to a deterministic point: $f : \mathbb{N} \to R \rightsquigarrow x : R$
2. A sequence of nondeterministic reals converges to a deterministic point: $f : \mathbb{N} \to \mathrm{M}R \rightsquigarrow x : R$

Nondeterminsim and Limits

Consider the following three cases where we might want to compute a limit

1. A deterministic sequence of reals converges to a deterministic point: $f : \mathbb{N} \to R \rightsquigarrow x : R$
2. A sequence of nondeterministic reals converges to a deterministic point: $f : \mathbb{N} \to \mathsf{M}R \rightsquigarrow x : R$
3. A sequence of nondeterministic reals converges to a nondeterministic point: $f : \mathbb{N} \to \mathsf{M}R \rightsquigarrow x : \mathsf{M}R$

Deterministic Limits

We assume constructive metric completeness:
Whenever we have a sequence $f : \mathbb{N} \to R$ that is fast Cauchy, i.e.,

$$\Pi(n, m : \mathbb{N}).\ -2^{-n-m} \leq f\ n - f\ m \leq 2^{-n-m}$$

there is a limit point of the sequence, i.e.

$$\Sigma(x : R).\ \Pi(n : \mathbb{N}).\ -2^{-n} \leq f\ n - x \leq 2^{-n}$$

Example: Real square root

- Consider the square root operation $\sqrt{}$ on nonnegative real numbers.

Example: Real square root

- Consider the square root operation $\sqrt{}$ on nonnegative real numbers.
- For $0.25 \leq x \leq 1$ the Heron method converges quadratically

$$0.25 \leq x \leq 1 \to \Sigma(y : R).\ y \cdot y = x$$

Example: Real square root

- Consider the square root operation $\sqrt{}$ on nonnegative real numbers.
- For $0.25 \leq x \leq 1$ the Heron method converges quadratically

$$0.25 \leq x \leq 1 \to \Sigma(y : R).\ y \cdot y = x$$

- For $x \neq 0$, we can nondeterministically find a scaling factor n

$$x \neq 0 \to \mathrm{M}\Sigma(z : \mathbb{Z}).\ 4^{z} \leq x \leq 4^{z+2}$$

Example: Real square root

- Consider the square root operation $\sqrt{}$ on nonnegative real numbers.
- For $0.25 \leq x \leq 1$ the Heron method converges quadratically

$$0.25 \leq x \leq 1 \to \Sigma(y : R).\ y \cdot y = x$$

- For $x \neq 0$, we can nondeterministically find a scaling factor n

$$x \neq 0 \to \mathrm{M}\Sigma(z : \mathbb{Z}).\ 4^z \leq x \leq 4^{z+2}$$

- $0.25 \leq 4^{-z}x \leq 1$ and $\sqrt{x} = 2^z\sqrt{4^{-z}x}$

Real square root

A nondeterministic sequence that converges to $\sqrt{x}$: For $n \in \mathbb{N}$

1. Nondeterministically check $x < 2^{-2n}$ or $x > 0$.

Real square root

A nondeterministic sequence that converges to $\sqrt{x}$: For $n \in \mathbb{N}$

1. Nondeterministically check $x < 2^{-2n}$ or $x > 0$.
2. In the first case, return 0

Real square root

A nondeterministic sequence that converges to $\sqrt{x}$: For $n \in \mathbb{N}$

1. Nondeterministically check $x < 2^{-2n}$ or $x > 0$.
2. In the first case, return 0
3. In the second case, scale and return the exact square root

Real square root

A nondeterministic sequence that converges to $\sqrt{x}$: For $n \in \mathbb{N}$

1. Nondeterministically check $x < 2^{-2n}$ or $x > 0$.
2. In the first case, return 0
3. In the second case, scale and return the exact square root

Real square root

A nondeterministic sequence that converges to $\sqrt{x}$: For $n \in \mathbb{N}$

1. Nondeterministically check $x < 2^{-2n}$ or $x > 0$.
2. In the first case, return 0
3. In the second case, scale and return the exact square root

The sequence converges to a square root, thus we should be able to show

$$\Sigma(y : R).\ x \geq 0 \implies y \cdot y = x$$

Example: Complex square root

- Now consider the square root operation on complex numbers.

Example: Complex square root

- Now consider the square root operation on complex numbers.
- Each $z \neq 0$ has exactly two square roots.

Example: Complex square root

- Now consider the square root operation on complex numbers.
- Each $z \neq 0$ has exactly two square roots.
- There is no total, continuous mapping $\sqrt{\ }: \mathbb{C} \to \mathbb{C}$, the complex square root is inherently multivalued.

Example: Complex square root

- Now consider the square root operation on complex numbers.
- Each $z \neq 0$ has exactly two square roots.
- There is no total, continuous mapping $\sqrt{} : \mathbb{C} \to \mathbb{C}$, the complex square root is inherently multivalued.
- However, the nondeterministic function computing any of the two square roots is computable.

Complex square root

We can reduce complex square roots to real square roots:

Let $z = a + ib$, then

$$\sqrt{\frac{\sqrt{a^2+b^2}+a}{2}} + i\,\mathrm{sgn}(b)\sqrt{\frac{\sqrt{a^2+b^2}-a}{2}}$$

is one of the square roots of z.

Complex square root

We can reduce complex square roots to real square roots:
Let $z = a + ib$, then

$$\sqrt{\frac{\sqrt{a^2 + b^2} + a}{2}} + i\,\mathrm{sgn}(b)\sqrt{\frac{\sqrt{a^2 + b^2} - a}{2}}$$

is one of the square roots of z.
Problem: sgn is not computable in 0.

Complex square root

We can reduce complex square roots to real square roots:
Let $z = a + ib$, then

$$\sqrt{\frac{\sqrt{a^2 + b^2} + a}{2}} + i\,\mathrm{sgn}(b)\sqrt{\frac{\sqrt{a^2 + b^2} - a}{2}}$$

is one of the square roots of z.
Problem: sgn is not computable in 0.

- $z \neq 0 \to \mathsf{M}(a < 0 + a > 0 + b < 0 + b > 0)$
- Nondeterministically choose one of the four cases and adapt the formula to the case:

$$z \neq 0 \to \mathsf{M}\Sigma(x : C).\ x \cdot x = z$$

Square root as limit

Given $z \in \mathbb{C}$, consider the following recursively defined (nondeterministic) sequence. For each n and given a previous approximation x_{n-1},

- Start by nondeterministically choosing one of the two cases $\|z\| \leq 2^{-2(n+2)}$ or $\|z\| > 0$

Square root as limit

Given $z \in \mathbb{C}$, consider the following recursively defined (nondeterministic) sequence. For each n and given a previous approximation x_{n-1},

- Start by nondeterministically choosing one of the two cases $\|z\| \leq 2^{-2(n+2)}$ or $\|z\| > 0$
- In the first one, return 0

Square root as limit

Given $z \in \mathbb{C}$, consider the following recursively defined (nondeterministic) sequence. For each n and given a previous approximation x_{n-1},

- Start by nondeterministically choosing one of the two cases $\|z\| \leq 2^{-2(n+2)}$ or $\|z\| > 0$
- In the first one, return 0
- In the second one, nondeterministcally return one of the square roots of z

Square root as limit

Given $z \in \mathbb{C}$, consider the following recursively defined (nondeterministic) sequence. For each n and given a previous approximation x_{n-1},

- Start by nondeterministically choosing one of the two cases $\|z\| \leq 2^{-2(n+2)}$ or $\|z\| > 0$
- In the first one, return 0
- In the second one, nondeterministcally return one of the square roots of z
- Once the second case is chosen, always return the previous approximation x_{n-1}

Square root as limit

Given $z \in \mathbb{C}$, consider the following recursively defined (nondeterministic) sequence. For each n and given a previous approximation x_{n-1},

- Start by nondeterministically choosing one of the two cases $\|z\| \leq 2^{-2(n+2)}$ or $\|z\| > 0$
- In the first one, return 0
- In the second one, nondeterministcally return one of the square roots of z
- Once the second case is chosen, always return the previous approximation x_{n-1}

Square root as limit

Given $z \in \mathbb{C}$, consider the following recursively defined (nondeterministic) sequence. For each n and given a previous approximation x_{n-1},

- Start by nondeterministically choosing one of the two cases $\|z\| \leq 2^{-2(n+2)}$ or $\|z\| > 0$
- In the first one, return 0
- In the second one, nondeterministcally return one of the square roots of z
- Once the second case is chosen, always return the previous approximation x_{n-1}

Any such sequence is a Cauchy sequence converging against one of the square roots of z, thus we expect to nondeterministically have a limit point.

Nondeterministic dependent choice

- Suppose we have $x_0 : A$ and $f : \mathbb{N} \to A \to \mathrm{M}A$.
- We can get a nondeterministic sequence by repeatedly applying $x_{n+1} := f\ n\ x_n$.

Example:

- $x_0 := 0.5$

Nondeterministic dependent choice

- Suppose we have $x_0 : A$ and $f : \mathbb{N} \to A \to \mathsf{M}A$.
- We can get a nondeterministic sequence by repeatedly applying $x_{n+1} := f\ n\ x_n$.

Example:

- $x_0 := 0.5$
- $f\ n\,x := \begin{cases} 0 \text{ or } 1 & \text{if } n = 0, \\ x & \text{otherwise} \end{cases}$

Nondeterministic dependent choice

- Suppose we have $x_0 : A$ and $f : \mathbb{N} \to A \to \mathsf{M}A$.
- We can get a nondeterministic sequence by repeatedly applying $x_{n+1} := f\ n\ x_n$.

Example:

- $x_0 := 0.5$
- $f\,n\,x := \begin{cases} 0 \text{ or } 1 & \text{if } n = 0, \\ x & \text{otherwise} \end{cases}$
- Primitive recursion: $\{0.5\} :: \{0,1\} :: \{0,1\} :: \{0,1\} :: \cdots$

Nondeterministic dependent choice

- Suppose we have $x_0 : A$ and $f : \mathbb{N} \to A \to MA$.
- We can get a nondeterministic sequence by repeatedly applying $x_{n+1} := f\ n\ x_n$.

Example:

- $x_0 := 0.5$
- $f\ n\, x := \begin{cases} 0 \text{ or } 1 & \text{if } n = 0, \\ x & \text{otherwise} \end{cases}$
- Primitive recursion: $\{0.5\} :: \{0, 1\} :: \{0, 1\} :: \{0, 1\} :: \cdots$
- Want: $\{0.5 :: 0 :: 0 :: 0 :: \ldots, 0.5 :: 1 :: 1 :: 1 :: \ldots\}$

Nondeterministic dependent choice

Definition (Nondeterministic dependent choice)

Given a sequence $R : \mathbb{N} \to A \to A \to \mathrm{Prop}$, $x_0 : A$ and $f : \Pi(n : \mathbb{N}).\ \Pi(x_n : A).\ \mathrm{M}\Sigma(x_{n+1} : A).\ R\ n\ x_n\ x_{n+1}$.
There is $F : \mathrm{M}(\mathbb{N} \to A)$ such that for any $f \in F$ and $n \in \mathbb{N}$, $R\ n\ (f\ n)\ (f\ (n+1))$ holds.

Nondeterministic dependent choice

Definition (Nondeterministic dependent choice)

Given a sequence $R : \mathbb{N} \to A \to A \to \mathrm{Prop}$, $x_0 : A$ and
$f : \Pi(n : \mathbb{N}).\ \Pi(x_n : A).\ \mathrm{M}\Sigma(x_{n+1} : A).\ R\ n\ x_n\ x_{n+1}$.
There is $F : \mathrm{M}(\mathbb{N} \to A)$ such that for any $f \in F$ and $n \in \mathbb{N}$,
$R\ n\ (f\ \ n)\ (f\ \ (n+1))$ holds.

- $x_0 := 0.5$

Nondeterministic dependent choice

Definition (Nondeterministic dependent choice)

Given a sequence $R : \mathbb{N} \to A \to A \to \text{Prop}$, $x_0 : A$ and $f : \Pi(n : \mathbb{N}).\ \Pi(x_n : A).\ \text{M}\Sigma(x_{n+1} : A).\ R\ n\ x_n\ x_{n+1}$.
There is $F : \text{M}(\mathbb{N} \to A)$ such that for any $f \in F$ and $n \in \mathbb{N}$, $R\ n\ (f\ \ n)\ (f\ \ (n+1))$ holds.

- $x_0 := 0.5$
- $f\ n\, x := \begin{cases} 0 \text{ or } 1 & \text{if } n = 0, \\ x & \text{otherwise} \end{cases}$

Nondeterministic dependent choice

Definition (Nondeterministic dependent choice)

Given a sequence $R : \mathbb{N} \to A \to A \to \text{Prop}$, $x_0 : A$ and $f : \Pi(n : \mathbb{N}).\ \Pi(x_n : A).\ \text{M}\Sigma(x_{n+1} : A).\ R\ n\ x_n\ x_{n+1}$.
There is $F : \text{M}(\mathbb{N} \to A)$ such that for any $f \in F$ and $n \in \mathbb{N}$, $R\ n\ (f\ \ n)\ (f\ \ (n+1))$ holds.

- $x_0 := 0.5$
- $f\ n\,x := \begin{cases} 0 \text{ or } 1 & \text{if } n = 0, \\ x & \text{otherwise} \end{cases}$
- $R\ \ n\ \ x\ \ y := x > 0 \to x = y$

Nondeterministic limits

Brauße and Müller suggested the following nondeterministic limit based on refinements. To construct a multivalued limit $X : \mathrm{M}R$,

- Provide a 2^{-0} approximation to some limit $x \in X$
- Provide a nondeterministic refinement function $f : \mathbb{N} \to R \to \mathrm{M}R$
- Whenever x_n is a 2^{-n} approx. to some limit $x \in X$, any $x_{n+1} \in f\ n\ x_n$ is a $2^{-(n+1}$ approximation to some (not necessarily the same) $x \in X$ and $|x_n - x_{n+1}| \leq 2^{-(n+1)}$

Nondeterministic limits

Brauße and Müller suggested the following nondeterministic limit based on refinements. To construct a multivalued limit $X : \mathrm{M}R$,

- Provide a 2^{-0} approximation to some limit $x \in X$
- Provide a nondeterministic refinement function $f : \mathbb{N} \to R \to \mathrm{M}R$
- Whenever x_n is a 2^{-n} approx. to some limit $x \in X$, any $x_{n+1} \in f\ n\ x_n$ is a $2^{-(n+1}$ approximation to some (not necessarily the same) $x \in X$ and $|x_n - x_{n+1}| \leq 2^{-(n+1)}$

This limit is derivable from the nondeterministic dependent choice: Choose $R\ n\ x\ y := |x - y| \leq 2^{-(n+1)} \wedge (y \sim_n X)$

Nondeterministic limits with additional information

- Recall the sequence we defined for the complex square root.

Nondeterministic limits with additional information

- Recall the sequence we defined for the complex square root.
- Any sequence has the form $0, 0, 0, \ldots, \sqrt{z}, \sqrt{z}, \ldots$

Nondeterministic limits with additional information

- Recall the sequence we defined for the complex square root.
- Any sequence has the form $0, 0, 0, \ldots, \sqrt{z}, \sqrt{z}, \ldots$
- However, the sequence is not a refinement procedure in the previous sense, as we were required to refine any given previous approximation.

Nondeterministic limits with additional information

- Recall the sequence we defined for the complex square root.
- Any sequence has the form $0, 0, 0, \ldots, \sqrt{z}, \sqrt{z}, \ldots$
- However, the sequence is not a refinement procedure in the previous sense, as we were required to refine any given previous approximation.

Nondeterministic limits with additional information

- Recall the sequence we defined for the complex square root.
- Any sequence has the form $0, 0, 0, \ldots, \sqrt{z}, \sqrt{z}, \ldots$
- However, the sequence is not a refinement procedure in the previous sense, as we were required to refine any given previous approximation.

We define the following more informative version of the nondeterministic limit:

- Additionally we are given an invariant property $Q : \mathbb{N} \to R \to \mathit{Type}$ that is preserved through the refinements.

Nondeterministic limits with additional information

- Recall the sequence we defined for the complex square root.
- Any sequence has the form $0, 0, 0, \ldots, \sqrt{z}, \sqrt{z}, \ldots$
- However, the sequence is not a refinement procedure in the previous sense, as we were required to refine any given previous approximation.

We define the following more informative version of the nondeterministic limit:

- Additionally we are given an invariant property $Q : \mathbb{N} \to R \to \mathit{Type}$ that is preserved through the refinements.
- For example, for the square root example we choose $Q\ n\ x := (|z| \leq 2^{-(n+2)} \land x = 0) + (x \cdot x = z)$.

Nondeterministic limits with additional information

- Recall the sequence we defined for the complex square root.
- Any sequence has the form $0, 0, 0, \ldots, \sqrt{z}, \sqrt{z}, \ldots$
- However, the sequence is not a refinement procedure in the previous sense, as we were required to refine any given previous approximation.

We define the following more informative version of the nondeterministic limit:

- Additionally we are given an invariant property $Q : \mathbb{N} \to R \to \mathit{Type}$ that is preserved through the refinements.
- For example, for the square root example we choose $Q\ n\ x := (|z| \leq 2^{-(n+2)} \wedge x = 0) + (x \cdot x = z)$.
- We not only need to refine the approximation but also construct a Boolean term in each step.

Program Extraction

When we prove a statement

$$\Pi(x : R).\ \Sigma(y : R).\ P\ x\ y$$

we get a computable function $f : \mathbb{R} \to \mathbb{R}$ computing numbers with the property P.

Program Extraction

When we prove a statement

$$\Pi(x : R).\ \Sigma(y : R).\ P\ x\ y$$

we get a computable function $f : \mathbb{R} \to \mathbb{R}$ computing numbers with the property P.

When we prove

$$\Pi(x : R).\ \mathrm{M}\Sigma(y : R).\ P\ x\ y$$

we get a multivalued function $f : \mathbb{R} \rightrightarrows \mathbb{R}$.

Program Extraction

When we prove a statement

$$\Pi(x : R).\ \Sigma(y : R).\ P\ x\ y$$

we get a computable function $f : \mathbb{R} \to \mathbb{R}$ computing numbers with the property P.

When we prove

$$\Pi(x : R).\ \mathrm{M}\Sigma(y : R).\ P\ x\ y$$

we get a multivalued function $f : \mathbb{R} \rightrightarrows \mathbb{R}$.
In the Coq implentation this is realized by mapping R to AERN's datatype `CReal` and axiomatized operations to primitive operations in AERN.

Quality of programs: Reliability

Need to trust:

- Coq core
- Coq extraction
- Haskell compiler, base libraries
- AERN

Quality of programs: Smooth development

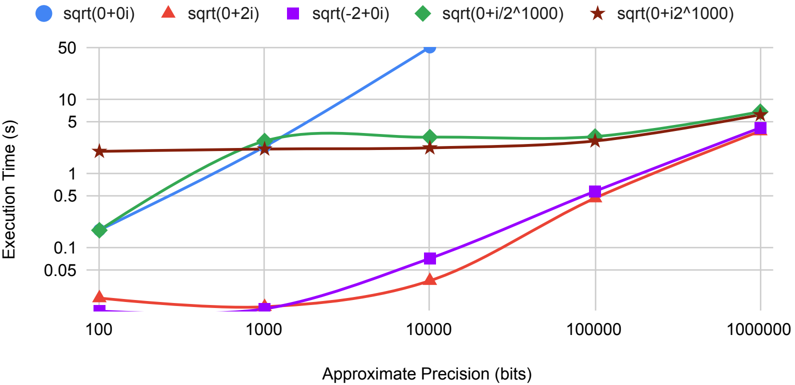
- Specifications readable
- Coq libraries can be used
- Algorithms readability still an issue

Quality of programs: Execution speed

Benchmark		Average execution time (s)			
Formula	Accuracy	Extracted	Hand-written	Native	iRRAM
$\max(0, \pi - \pi)$	10^6 bits	3.5	3.8	3.8	1.59
$\sqrt{2}$	10^6 bits	0.72	0.70	0.40	0.62
$\sqrt{\sqrt{2}}$	10^6 bits	1.52	1.38	0.85	1.15
$x - 0.5 = 0$	10^3 bits	1.44	0.32	—	0.03
$x(2-x) - 0.5 = 0$	10^3 bits	2.02	0.35	—	0.04
$\sqrt{x + 0.5} - 1 = 0$	10^3 bits	12.9	2.35	—	0.29

(i7-4710MQ CPU, 16GB RAM, Ubuntu 18.04, Haskell Stackage LTS 17.2)

Quality of programs: Execution speed



(i7-4710MQ CPU, 16GB RAM, Ubuntu 18.04, Haskell Stackage LTS 17.2)

Summary and Future Work

Summary:

- Real numbers are a primitive data-type in exact real computation frameworks
- We defined axiomatic reals in a dependent type theory
- We implemented the axioms in Coq and adjusted the extraction mechanism such that axiomatic reals are mapped to AERN's primitive data-type `CReal`, getting efficient and certified real number computation programs.

Thank you!