Real functions are continuous, continuously, computationally

Sewon Park (Kyoto University) 

based on j.w.w.

Holger Thies (Kyoto U.) and Michal Konečný (Aston U.)

The Fourth **Korea Logic Day** 2025
January 13–15, 2025
Changwon, Korea

Real functions are continuous, continuously, computationally

- **Part I:** Introduce exact real-number computation: What does it mean to compute real numbers and functions?
- **Part II:** (1) Introduce a (constructive) dependent type theory as a language of expressing and reasoning about "are"

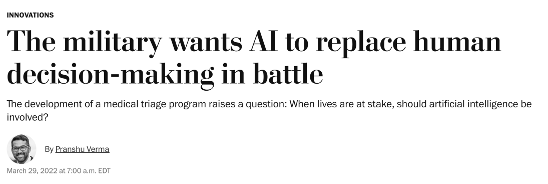
  (2) And present an axiomatic formalization of real numbers and functions (whose interpretation corresponds to the *exact real-number computation*, "computing real functions")
- **Part III:** Prove that all real functions are continuously continuous (in the type theory) and discuss possible applications

Part I

Motivation: Correct Numerical Computations

- Computers model and make decisions for real-world problems interacting with the physical world.

▼ WP article



INNOVATIONS

The military wants AI to replace human decision-making in battle

The development of a medical triage program raises a question: When lives are at stake, should artificial intelligence be involved?

By Pranshu Verma

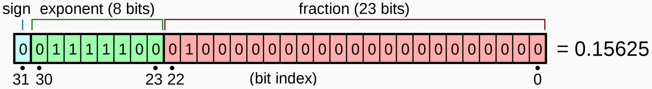
March 29, 2022 at 7:00 a.m. EDT

- Correctness in safety-critical applications; e.g., Ariane 5 ▲
- Infinite data such as real numbers, functions, spaces ubiquitously used to represent physical quantities such as distances, temperature, trajectory, areas, etc

▷ *Correctness in real number (and higher) computations becomes more and more important!*

Floating-Point Arithmetic

Common practice: `Floating-point` computation.



Inevitable round-off errors:

```
>>> x = (0.1 + 0.2) + 0.3
>>> y = 0.1 + (0.2 + 0.3)
>>> x == y
False
```

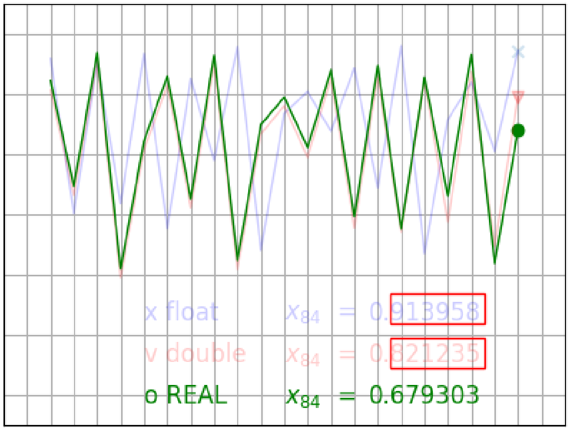
```
>>> x
0.6000000000000001
>>> y
0.6
```

due to fundamental limitation in expressivity of finite precision

- *Discrepancy* between intuitive semantics and actual machine semantics makes it challenging to obtain correct programs
- When round-off errors accumulate (*e.g. in an iterative function system*) computation can be totally meaningless

Example - Logistic Map

$$x_{n+1} = 3.75 \cdot x_n \cdot (1 - x_n) \quad \text{when} \quad x_0 = 0.5$$



Instead: Exact Real Computation

- **Infinite representations** for real numbers [Wei00]

  E.g., rationals $q_1, q_2, \cdots$ expresses $x \Leftrightarrow \forall i.\ |x - q_i| \leq 2^{-i}$

  exact computations by **type-2 machines**

  E.g., $x + y$ is realized by $(p_i)_i, (q_i)_i \mapsto (p_{i+1} + q_{i+1})_i$
- Hide representation-specific details

  $\rightsquigarrow$ ***Abstract data type*** *for exact real numbers*:

```
>>> print(pi, 10) # print 2^{-10} approximation of π
3.14159 ± 2^{-10}
>>> print(pi, 100)
3.14159265358979323846 ··· ± 2^{-100} # for high-precision result
>>> pi + sqrt(2) # evaluates exactly to π + √2
>>> print(pi + sqrt(2), p) # prints 2^{-p} approx. to π + √2
```

- **Expectation**: intuitive reasoning with reals as in textbooks

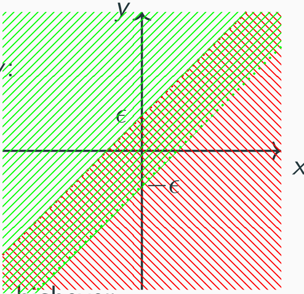
Computable and Uncomputable Primitives

- **Infinite representations** for real numbers [Wei00]

$$(q_1, q_2, \ldots) \in \mathbb{Q}^{\mathbb{N}} \text{ expresses } x \Leftrightarrow \forall i.\ |x - q_i| \leq 2^{-i}$$

- The arithmetical operations $(+, -, \times, \div)$ are computable (exactly without rounding errors.)
- However, computing $x < y$ fails when $x = y$:

```
(p_i)_{i∈ℕ} < (q_i)_{i∈ℕ} =
  for i = 0 → ∞:
    if p_i <_ℚ q_i − 2^{−n}: return True
    else if q_i <_ℚ p_i − 2^{−n}: return False
    else: continue
```



More precisely, $x < y$ *diverges* when $x = y$ whichever representation and whichever algorithm is used.

- Parallel evaluation is used:

$$x <_\epsilon y := (x < y + \epsilon) \| (y < x + \epsilon)$$

to **nondeterministically**, but totally approximate $x < y$

Verification in Exact Real Computation

- Programming with real numbers (as they were the familiar abstract entities in the textbooks) carefully dealing with *partial comparisons* and *nondeterminism*.
  - **In imperative paradigm**: `iRRAM` (C++), `Ariadne` (C++ and Python), `Clerical`, ...
  - **In functional paradigm**: `AERN` (Haskell), ...
- **Imperative programs**: Verification reduces to the theory of real numbers (with help of domain theory)

  P et. al: **Semantics, Specification Logic, and Hoare Logic of Exact Real Computation (2024)**. *Logical Methods in Computer Science*

  Andrej Bauer, P, Alex Simpson: **An Imperative Language for Verified Exact Real-Number Computation (2024)**. *(submitted)*
- **Functional programs**: From a (constructive) proof from mathematical analysis, extract a correct program

  Michal Konečný, P, Holger Thies: **Extracting efficient exact real number computation from proofs in constructive type theory (2024)**, *Journal of Logic and Computation*

cAERN

- Introduces types for computational real numbers, partiality, nondeterminism, ... and primitive operations in a constructive dependent type theory
- A constructive proofs get extracted to verified *Exact Real Computation* user programs

  E.g., Intermediate Value Theorem $\rightsquigarrow$ Root-finding program

- A realizability interpretation as a metatheorem to prove soundness of our axiomatization.
- Implementation as the Coq library **cAERN**
  - `https://github.com/holgerthies/coq-aern`
  - Approximately 12,000 lines of code.
- Program extraction to Haskell, using the AERN library for basic operations on real numbers.

Part II : Dependent Type Theory

Dependent Type Theory

- **Base types**: 0 (*empty*), 1 (*unit*), N (*numbers*) are types.
  When A, B are types, $A \times B$ (*product*), $A + B$ (*sum*), $A \to B$ (*mapping*) are types.

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \mathrm{inL}\ a : A + B} \qquad \frac{\Gamma \vdash b : B}{\Gamma \vdash \mathrm{inR}\ b : A + B}$$

- *When $B(x)$ is a type indexed by $x : A$*
  $\Pi(x : A).\, B(x)$ (dependent function; the space of sections) and $\Sigma(x : A).\, B(x)$ (dependent pair; the total space) are types

$$\frac{\Gamma \vdash a{:}A \qquad \Gamma \vdash b{:}B[a/x]}{\Gamma \vdash \langle a, b \rangle : \Sigma(x{:}A).\, B(x)}$$

- Interpret types as propositions, $A + B$ as $A \vee B$, $\Pi(x : A).\, B(x)$ as $\forall x : A.\, B(x)$, and $\Sigma(x : A).\, B(x)$ as $\exists x : A.\, B(x) \rightsquigarrow$ language for constructive mathematics.

Constructive Proofs are Programs

"Any natural number is either odd or even"

Define $(n\!:\!\mathsf{N})$-indexed families of types:

- $\mathsf{isEven}(n\!:\!\mathsf{N}) := \Sigma(k : \mathsf{N}).\, n = k + k$
- $\mathsf{isOdd}(n\!:\!\mathsf{N}) := \Sigma(k : \mathsf{N}).\, n = k + k + 1$

Then, the type below corresponds to the above statement:

$$\Pi(n : \mathsf{N}).\, \mathsf{isOdd}(n) + \mathsf{isEven}(n)$$

The type is the space of sections:

$$f : \mathbb{N} \ni n \mapsto \begin{cases} \mathsf{inL}\ \langle k, \cdot \rangle & \text{if } n = 2k \\ \mathsf{inR}\ \langle k, \cdot \rangle & \text{if } n = 2k + 1 \end{cases}$$

a program that tells us whether n is even **or** odd and **why**.

Classical Types

Though, not all types are constructive:

- 0, 1, $x = y$ do not carry any computational structure
- Define $\neg A := A \to 0$.
- A type A is a *classical proposition* if $A \cong \neg\neg A$.

Assume that there is a universe Prop of classical propositions that is closed under $\tilde{\exists}$ and $\tilde{\vee}$.

Assume $\Pi(P : \mathrm{Prop}).\, P \mathbin{\tilde{\vee}} \neg P$ **but not** $\Pi(P : \mathrm{Prop}).\, P + \neg P$.

***Idea**: put algorithms in the usual type-level,*
and write verification-related specifications in Prop.

Further assume classical propositional extensionality, functional extensionality, etc.

Naïve Reals in Constructive Type Theory

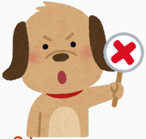
- Constructive dependent type theory:

$$A + B \text{ is valid } \cong \text{deciding } A \text{ or } B \text{ is computable}$$
$$\Sigma(x : A).\ B(x) \text{ is valid } \cong \text{finding } x : A \text{ s.t. } B(x) \text{ is computable}$$

- Certified program extraction:

$$\Pi(x : A).\ \Sigma(y : B).\ R(x, y)$$

yields a program $\mathcal{P} : A \to B$ s.t. $\forall(x : A).\ R(x, \mathcal{P}(x))$



- Classical axiomatization of reals is **invalid**:

$$\text{Trichotomy} : \Pi(x : \mathrm{R}).\ (x < 0) + (x = 0) + (x > 0)$$

The sign test of reals is not computable

- Axiomatization of exact reals s.t.
  proofs $\cong$ programs in ERC framework (viz. AERN in Haskell)

"Constructive" Axiomatic Reals and Partiality

Axiom 1: There is a type R.

Axiom 2: There are terms for $(0, 1, \ldots, +, -, \times, \div)$

Axiom 3: Given x, y : R, we have

$$(x < y) : \mathsf{S}$$

where S is another axiomatic type for partial computations

Axiom 4: $\downarrow$:S is for termination and $\uparrow$:S is for nontermination

- $(x < y) = \downarrow$ *(or write $(x < y)\downarrow$)* when $x < y$.
- $(x < y) = \uparrow$ *(or write $(x < y)\uparrow$)* when $x \geq y$.

We can **prove** $\Pi(x : \mathsf{R}).\ (x < 0)\downarrow \tilde{\vee}\ (x = 0)\ \tilde{\vee}\ (x > 0)\downarrow$: Prop

But cannot **prove** $\Pi(x : \mathsf{R}).\ (x < 0)\downarrow + (x = 0) + (x > 0)\downarrow$: Type

Axiomatic Nondeterminism

Axiom: there is a monad $\mathsf{M} : \mathsf{Type} \to \mathsf{Type}$ for nondeterminism

$$\Pi(s_1, s_2 : \mathsf{S}).\,(s_1{\downarrow}\ \tilde{\vee}\ s_2{\downarrow}) \to \mathsf{M}\big(s_1{\downarrow} + s_2{\downarrow}\big)$$

Given two partial computations s_1, s_2, given classically that s_1 or s_2 terminates, we can nondeterministically decide which terminates.

Example: for any positive $\epsilon : \mathsf{R}$, we can prove

$$\Pi(x, y : \mathsf{R}).\,\mathsf{M}\big((x < y + \epsilon){\downarrow} + (y < x + \epsilon){\downarrow}\big)$$

but cannot prove

$$\Pi(x, y : \mathsf{R}).\,(x < y + \epsilon){\downarrow} + (y < x + \epsilon){\downarrow}$$

Axiom: Subsingletons are deterministic

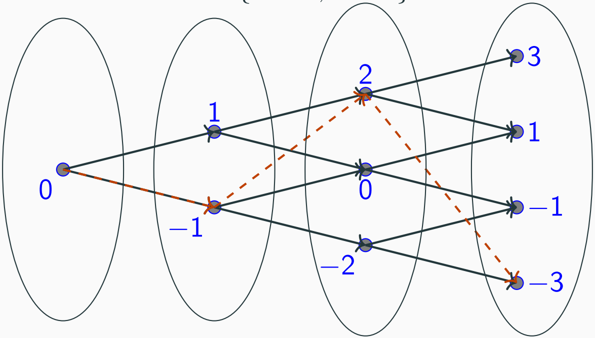
$$\Pi(A : \mathsf{Type}).\,\big(\Pi(a, b : A).\,a = b\big) \to \mathsf{M}\ A \cong A$$

Example: we can prove $\Pi(x, y).\,(x \neq y) \to (x < y){\downarrow} + (y < x){\downarrow}$

Nondeterministic Dependent Choice

Axiom: For a nondeterministic procedure $f : A \to \mathrm{M}\ A$, iterating it on $a : A$, *nondeterministically yields* a deterministic section $h : \mathrm{N} \to A$ of $f^\omega : (n \mapsto f^n) : \mathrm{N} \to \mathrm{M}\ A$ that are precisely traces

Example: Consider $f : x \mapsto \{x - 1, x + 1\}$



Naive iteration yields $f^\omega(n) = \{-n, -n+2, \ldots, n\}$ whereas traces are $\{h \mid h(0) = 0,\ h(n+1) = h(n) \pm 1\}$

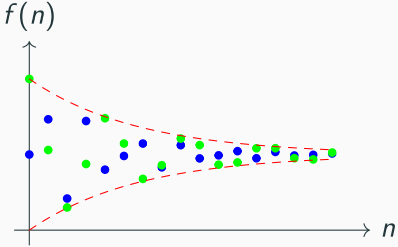
Nondeterministic Completeness

Example: Given the unique classical description of a real number $P : \mathsf{R} \to \mathsf{Prop}$. Write $x \approx_n P$ for the classical proposition saying $x : \mathsf{R}$ approximates P up to 2^{-n}. Then we have:

$$\big(\Pi(n : \mathsf{N}).\, \mathsf{M}\Sigma(y : \mathsf{R}).\, y \approx_n P\big) \to \Sigma(x : \mathsf{R}).\, P\ x$$

Proof Idea:

$f : \mathsf{N} \to \mathsf{M}\mathsf{R}$ converges to $y : \mathsf{R}$



- Each nondeterministic section $h : \mathsf{N} \to \mathsf{R}$ of f is a Cauchy sequence
- M-lifting of *lim* on h yields M R limits
- As there is at most one limit, Subsingleton-elimination yields the limit y

Nice Example

Define $\texttt{isMax}(z, x, y) :\equiv (x \geq y \to z = x) \wedge (y \geq x \to z = y)$
as a classical predicate and prove:

$$\Pi(x, y : \mathsf{R}).\ \Sigma(z : \mathsf{R}).\ \texttt{isMax}(z, x, y)$$

Proof.

- limit as $n \to \infty$:
- $\quad$ *assume* $\mathsf{M}\big((x < y + 2^{-n}) + (y < x + 2^{-n})\big) \quad \leftarrow$ axiom
- $\qquad$ *assume* $\big((x < y + 2^{-n}) + (y < x + 2^{-n})\big)$
- $\qquad\quad$ *case 1:* $x < y + 2^{-n}$, y approximates the max by 2^{-n}
- $\qquad\quad$ *case 2:* $y < x + 2^{-n}$, x approximates the max by 2^{-n}
- $\qquad$ $\Sigma(z : \mathsf{R}).\ z$ approximates the max by 2^{-n}
- $\quad$ $\mathsf{M}\Sigma(z : \mathsf{R}).\ z$ approximates the max by $2^{-n} \quad \leftarrow$ **M**-lift
- $\Sigma(z : \mathbb{R}).\ \texttt{isMax}(z, x, y) \quad \leftarrow$ nondeterministic completeness $\square$

extracts to the maximum function in AERN

Code Extraction Example

```
Lemma real_max_prop :
  forall x y, {z | (x >= y → z = x)
                 ∧ (x < y → z = y)}.
  Proof.
    intros.
    apply real_mslimit_P_lt.
    + (* max is single-valued *)
      ...
    + (* construct limit *)
      intros.
      apply (mjoin (x>y − prec n)
                   (y>x − prec n)).
      ++ intros [c1|c2].
         +++ (* when x > y − 2^-n *)
         exists x.
         ...
         +++ (* when x < y − 2^-n *)
         exists y.
         ...
      ++ apply M_split.
         apply prec_pos.
  Defined.
```

```
real_max_prop ::
    AERN2.CReal ->
    AERN2.CReal ->
    AERN2.CReal
real_max_prop x y =
  AERN2.limit (\n ->
    Prelude.id (\h -> case h of {
                   P.True -> x;
                   P.False -> y})
      (m_split x y (prec n)))
```

What really is M? How are the axioms justified?

Assemblies

- Assembly $X = (|X|, \Vdash_X)$ is a pair of a set $|X|$ and a binary relation $\Vdash_X \subseteq \mathbb{N}^{\mathbb{N}} \times |X|$ that is surjective [Lon95]:

$$\forall (x \in |X|).\ \exists (\varphi \in \mathbb{N}^{\mathbb{N}}).\ \varphi \Vdash_X x$$

- $f : |X| \to |Y|$ is computable if there is a *type-2 machine* $\tau :\subseteq \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ that tracks f:

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ \Vdash_X \vdots & & \vdots \Vdash_Y \\ \mathbb{N}^{\mathbb{N}} & \xrightarrow{\tau} & \mathbb{N}^{\mathbb{N}} \end{array}$$

$$\forall (x \in |X|).\ \forall (\varphi \in \mathbb{N}^{\mathbb{N}}).\ \varphi \Vdash_X x \Rightarrow \tau(\varphi) \Vdash_Y f(x)$$

- Category of assemblies & computable functions $\mathrm{Asm}(\mathbb{N}^{\mathbb{N}})$ forms a locally Cartesian closed category modeling Dependent Type Theory [Bir95]

Validity of Some Axiomatization

- Standard Cauchy assembly $|\mathsf{R}| = \mathbb{R}$:

$$\varphi \Vdash_{\mathsf{R}} x \quad :\Longleftrightarrow \quad \varphi(n) \textit{ encodes } q_n \in \mathbb{Q}.\ |q_n - x| < 2^{-n} \text{ for all } n$$

- Nondeterminism monad $\mathsf{M} : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$

$$|\mathsf{M}\ X| :\equiv \{A \subseteq |X| \mid A \neq \emptyset\} \quad \varphi \Vdash_{\mathsf{M}\ X} A \quad :\Longleftrightarrow \quad \exists (x \in A).\ \varphi \Vdash_X x$$

- Sierpiński assembly $|\mathsf{S}| = \{\uparrow, \downarrow\}$:

$$\varphi \Vdash_{\mathsf{S}} \uparrow \ :\Longleftrightarrow\ \forall (i \in \mathbb{N}).\ \varphi(i) = 0$$
$$\varphi \Vdash_{\mathsf{S}} \downarrow \ :\Longleftrightarrow\ \exists (i \in \mathbb{N}).\ \varphi(i) \neq 0$$

- classifies opens/semi-decidable subsets:

$$f : \mathsf{R} \to \mathsf{S} \iff f \text{ characterize a semi-decidable } S \subseteq \mathbb{R}$$

- The axiom saying $x < y$ is semi-decidable is indeed valid.
- The set of axioms are valid and universal [Her99]

Part III: Continuity of Continuity

Continuity Principles

- In $\mathrm{Asm}(\mathbb{N}^{\mathbb{N}})$, a mapping $f : X \to Y$ is by def. computable and a function object Y^X consists of continuously realizable functions from X to Y
- (Hence) the statement

  all real functions are continuous as stated by Brouwer

  is a valid sentence in $\mathrm{Asm}(\mathbb{N}^{\mathbb{N}})$

  We can assume and use it for integration, derivation, etc

  How about other abstract spaces X other than R?
- The common approach is to assume the statement:

  all mappings $\mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ *are continuous*

  then study $\mathrm{N}^{\mathrm{N}} \hookrightarrow X$ to expand it to X.
- **Desired:** abstract and at the same time general enough continuity principle

Continuity Principle

Axiom(Continuity): For any partial computation over sequences,

$$f : X^{\mathrm{N}} \to \mathrm{S}$$

and for any sequence $x : X^{\mathrm{N}}$, when $f\ x$ terminates ($(f\ x)\downarrow$) there nondeterministically exists an index $n : \mathrm{N}$ that f cannot distinguish:

$$\Pi(n{:}\mathrm{N}).\, \bar{x}_n = \bar{y}_n \to (f\ y)\downarrow$$

The axiom can be realized by:

```
function continuity (f : X^N → S, x : X^N):
  var n : N := 0;
  local function x_(m : N) =
    n := max(n, m);
    return x_m
  let _ := f(x_);
  return n;
```

Continuous Continuity

Lemma: Any $f : \mathsf{N}^\mathsf{N} \to \mathsf{S}$ is continuously continuous;
i.e., there *nondeterministically is* $\mu : \mathsf{N}^\mathsf{N} \to \mathsf{N}$ s.t.

1. μ is a modulus of continuity:
$$\Pi(x : \mathsf{N}^\mathsf{N}).\,(f\ x){\downarrow} \to \Pi(y : \mathsf{N}^\mathsf{N}).\,\bar{y}_{\mu\ x} = \bar{x}_{\mu\ x} \to (f\ y){\downarrow}$$
For any sequence x, when $f\ x$ terminates, it is okay to read only $\mu\ x$ entries around x.
2. and μ is again continuous:
$$\Pi(x : \mathsf{N}^\mathsf{N}).\,(f\ x){\downarrow} \to \Sigma(n : \mathsf{N}).\,\Pi(y : \mathsf{N}^\mathsf{N}).\,\bar{y}_n = \bar{x}_n \to \mu\ y = \mu\ x$$
For any sequence x, when $f\ x$ terminates, there nondeterministically is a number of entries n where μ should give a consistent answer.

Proof Idea: prove continuity of $h : \mathsf{N}^\mathsf{N} \to \mathsf{N}$ by reducing it to a Sierpinski-valued function. Then the result follows.

Main Result

Lemma: Any real function $f : \mathsf{R} \to \mathsf{R}$ is point-wise continuous:

$$\Pi(x\!:\!\mathsf{R}, n\!:\!\mathsf{N}).\, \mathsf{M}\Sigma(\mu_{x,n}\!:\!\mathsf{N}).\, \Pi(y\!:\!\mathsf{R}).\, |x-y| \le 2^{-\mu_{x,n}} \to |f\ x - f\ y| \le 2^{-n}$$

Moreover, it is continuously continuous on $\mathsf{Q} \hookrightarrow \mathsf{R}$;
i.e., there *nondeterministically is* $\mu : \mathsf{Q} \times \mathsf{N} \to \mathsf{N}$ s.t.

1. μ is a modulus of continuity:
$$\Pi(q\!:\!\mathsf{Q}, n\!:\!\mathsf{N}).\, \Pi(y\!:\!\mathsf{R}).\, |q-y| \le 2^{-\mu(q,n)} \to |f\ q - f\ y| \le 2^{-n}$$
2. and μ is again continuous:
$$\Pi(q\!:\!\mathsf{Q}, n\!:\!\mathsf{N}).\, \Sigma(m\!:\!\mathsf{N}).\, \Pi(r\!:\!\mathsf{R}).\, |q-r| \le 2^{-m} \to \mu(q,n) = \mu(r,n)$$

However, Q *cannot be replaced with* R*. In this sense, any real function* $f : \mathsf{R} \to \mathsf{R}$ *is continuously continuous.*

Proof Idea

Prove it for $f : \mathsf{R} \to \mathsf{S}$:

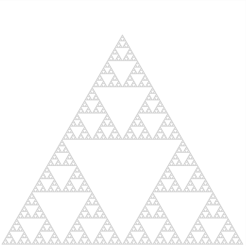
$$\begin{array}{ccccccc} & & & & \mathsf{R} & \xrightarrow{f} & \mathsf{S} \\ & & & \nearrow^{\delta} & \downarrow{\scriptstyle \alpha} & & \\ \mathsf{Q} & \xrightarrow{q \mapsto q^{\omega}} & \mathsf{Q}^{\mathsf{N}} & \xrightarrow{\eta} & \mathsf{M}\ \mathsf{Q}^{\mathsf{N}} & & \end{array}$$

- Get continuous modulus of continuity μ for $f \circ \delta : \mathsf{Q}^{\mathsf{N}} \to \mathsf{S}$.
- For any $x : \mathsf{R}$ s.t. $(f\ x)\downarrow$:
- get nondeterministically $\phi : \mathsf{Q}^{\mathsf{N}}$ such that $\delta\phi = x$ (by α)
- obtain $n : \mathsf{N}$ such that $\bar{\phi}_n = \bar{\psi}_n$ implies $(f\ (\hat{\psi}))\downarrow$ (by μ)
- claim this n works:
- For any $y : \mathsf{R}$ s.t. $|x - y| \leq 2^{-n}$:
  there **classically exists** $\psi : \mathsf{Q}^{\mathsf{N}}$ s.t. $\bar{\phi}_n = \bar{\psi}_n$ and $\delta\psi = y$.
- Hence, $\neg\neg(f\ (\delta\psi))\downarrow$ and indeed $(f\ y)\downarrow$.

and the modulus is continuous on Q

So What?

- Continuity is used to make $X \to \mathsf{S}$ indeed the space of opens.
- From opens, define various classes of hyperspaces: closed, overt, compact, overt-compact, located, ….
- For overt-compactness, to cover an open $X \to \mathsf{S}$, it is required the obtained continuity is continuous; when modulus is not continuous, it can fail to cover a compact interval
- E.g., in our system, we can define fractals as the (Hausdorff-) limit and extract certified drawings:



Conclusion

In this talk, the followings were presented:

- A constructive dependent type theory as a language of exact real number computations (cAERN project)
- Recent formalization of continuity principle and hyperspace computations

Future work includes:

- Other applications e.g., extending the ODE solving:

  SP and Holger Thies: **A Coq Formalization of Taylor Models and Power Series for Solving Ordinary Differential Equations**. *ITP 2024*

- Formalizing and verifying program extraction using meta-level programming and reasoning using e.g. `MetaCoq`
- Relating ours to classical formalizations e.g. `mathcomp-analysis` (transfer principle, type-theoretic generalization of the double-negation translation)

$\Rightarrow$ Classical reasoning (of computational content) in cAERN can be done relying on those rich libraries

% Thank you for your attention!