Alternative Logics: Constructive Logic

CS 81

February 7, 2011

Fallacies

This dog is a father

Fallacies

This dog is a father

This dog is mine

Fallacies

This dog is a father

This dog is mine

This dog is my father

Fallacies

$$\frac{\text{Prof. Ran is Prof. Libeskind-Hadas} \quad \text{This is Prof. Libeskind-Hadas}}{\text{This is Prof. Ran}}$$

Fallacies

$$\frac{\text{Prof. Ran is Prof. Libeskind-Hadas} \quad \text{This is Prof. Libeskind-Hadas}}{\text{This is Prof. Ran}} \qquad \frac{\text{Man is Animal} \quad \text{Fluffy is Animal}}{\text{Fluffy is Man}}$$

Fallacies

The steelworkers were unionized

Fallacies

The steelworkers were unionized

They had no missing electrons

Ambiguous Language

John saw a picture of the prettiest girl he had ever seen hanging on a locker door.

Logics We've Seen So Far

1. Classical Propositional Logic
2. Classical Predicate Logic

For each, you've seen many sets of rules, *all proving the same theorems*. What about other choices?

- ✓ Same logical language but different theorems?
- ✓ New operators?

Selective Biography of Modern Logic

Aristotle

Boole

Frege

Russell

Hilbert

Gödel

Brouwer

Heyting

Kolmogorov

The BHK Interpretation of Logic

What is a proof?

The BHK Interpretation of Logic

What is a proof?

- ✓ A proof of $p \wedge q$ consists of a proof of p and a proof of q.

The BHK Interpretation of Logic

What is a proof?

- ✓ A proof of $p \wedge q$ consists of a proof of p and a proof of q.
- ✓ A proof of $p \Rightarrow q$ is a method of converting proofs of p into proofs of q.

The BHK Interpretation of Logic

What is a proof?

- ✓ A proof of $p \wedge q$ consists of a proof of p and a proof of q.
- ✓ A proof of $p \Rightarrow q$ is a method of converting proofs of p into proofs of q.
- ✓ A proof of $\forall x.\, p(x)$ is a method that when given any x produces a proof of $p(x)$.

The BHK Interpretation of Logic

What is a proof?

- ✓ A proof of $p \wedge q$ consists of a proof of p and a proof of q.
- ✓ A proof of $p \Rightarrow q$ is a method of converting proofs of p into proofs of q.
- ✓ A proof of $\forall x.\, p(x)$ is a method that when given any x produces a proof of $p(x)$.
- ✓ A proof of $\neg p$ is a method of converting proofs of p into a proof of $\bot$.

The BHK Interpretation of Logic

What is a proof?

- ✓ A proof of $p \wedge q$ consists of a proof of p and a proof of q.
- ✓ A proof of $p \Rightarrow q$ is a method of converting proofs of p into proofs of q.
- ✓ A proof of $\forall x.\, p(x)$ is a method that when given any x produces a proof of $p(x)$.
- ✓ A proof of $\neg p$ is a method of converting proofs of p into a proof of $\bot$.
- ✓ A proof of $p \vee q$ is either a proof of p or a proof of q.

The BHK Interpretation of Logic

What is a proof?

- ✓ A proof of $p \wedge q$ consists of a proof of p and a proof of q.
- ✓ A proof of $p \Rightarrow q$ is a method of converting proofs of p into proofs of q.
- ✓ A proof of $\forall x.\, p(x)$ is a method that when given any x produces a proof of $p(x)$.
- ✓ A proof of $\neg p$ is a method of converting proofs of p into a proof of $\bot$.
- ✓ A proof of $p \vee q$ is either a proof of p or a proof of q.
- ✓ A proof of $\exists x.\, p(x)$ is a value x and a proof that $p(x)$ holds.

The BHK Interpretation of Logic

What is a proof?

- ✓ A proof of $p \wedge q$ consists of a proof of p and a proof of q.
- ✓ A proof of $p \Rightarrow q$ is a method of converting proofs of p into proofs of q.
- ✓ A proof of $\forall x.\, p(x)$ is a method that when given any x produces a proof of $p(x)$.
- ✓ A proof of $\neg p$ is a method of converting proofs of p into a proof of $\bot$.
- ✓ A proof of $p \vee q$ is either a proof of p or a proof of q.
- ✓ A proof of $\exists x.\, p(x)$ is a value x and a proof that $p(x)$ holds.
- ✓ There are no proofs of $\bot$.

BHK Examples

Which are constructively plausible?

- ✓ $p \to p$
- ✓ $p \to (q \to p)$
- ✓ $p \to \neg\neg p$
- ✓ $\neg\neg p \to p$
- ✓ $p \vee \neg p$
- ✓ $\neg(p \wedge \neg p)$
- ✓ $(\neg p \vee q) \to (p \to q)$
- ✓ $(p \to q) \to (\neg p \vee q)$
- ✓ $(\exists x.\, P(x)) \to \neg(\forall x.\, \neg P(x))$
- ✓ $\neg(\forall x.\, \neg P(x)) \to (\exists x.\, P(x))$

- ✓ A proof of $p \wedge q$ consists of a proof of p and a proof of q.
- ✓ A proof of $p \Rightarrow q$ is a method of converting proofs of p into proofs of q.
- ✓ A proof of $\forall x.\, p(x)$ is a method that when given any x produces a proof of $p(x)$.
- ✓ A proof of $\neg p$ is a method of converting proofs of p into a proof of $\bot$.
- ✓ A proof of $p \vee q$ is either a proof of p or a proof of q.
- ✓ A proof of $\exists x.\, p(x)$ is a value x and a proof that $p(x)$ holds.
- ✓ There are no proofs of $\bot$.

Constructive Logic

The BHK interpretation leads to constructive logic (a.k.a. intuitionistic logic). Essentially, classical logic without the law of the excluded middle.

$$\overline{P \vee \neg P}$$

As a consequence, we no longer have

$$\frac{\neg\neg P}{P} \qquad \frac{\begin{array}{c}\neg P \\ \vdots \\ \bot\end{array}}{P}$$

The Canonical **Nonconstructive** Proof

Theorem

There exist irrational numbers r and s such that r^s is rational.

The Canonical **Nonconstructive** Proof

Theorem

There exist irrational numbers r and s such that r^s is rational.

Proof.

We know that $\sqrt{2}$ is irrational.

Consider $\sqrt{2}^{\sqrt{2}}$. Either this is rational or it is not.

The Canonical **Nonconstructive** Proof

Theorem

There exist irrational numbers r and s such that r^s is rational.

Proof.

We know that $\sqrt{2}$ is irrational.

Consider $\sqrt{2}^{\sqrt{2}}$. Either this is rational or it is not. LEM!

The Canonical **Nonconstructive** Proof

Theorem

There exist irrational numbers r and s such that r^s is rational.

Proof.

We know that $\sqrt{2}$ is irrational.

Consider $\sqrt{2}^{\sqrt{2}}$. Either this is rational or it is not. LEM!

- ✓ If it's rational, we're done.

The Canonical **Nonconstructive** Proof

Theorem

There exist irrational numbers r and s such that r^s is rational.

Proof.

We know that $\sqrt{2}$ is irrational.

Consider $\sqrt{2}^{\sqrt{2}}$. Either this is rational or it is not. LEM!

- ✓ If it's rational, we're done.
- ✓ If not, then $(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\sqrt{2}} = \sqrt{2}^2 = 2$.

The Canonical **Nonconstructive** Proof

Theorem

There exist irrational numbers r and s such that r^s is rational.

Proof.

We know that $\sqrt{2}$ is irrational.

Consider $\sqrt{2}^{\sqrt{2}}$. Either this is rational or it is not. LEM!

- ✓ If it's rational, we're done.
- ✓ If not, then $(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\sqrt{2}} = \sqrt{2}^2 = 2$.

□

A Constructive Proof

Theorem

There exist irrational numbers r and s such that r^s is rational.

A Constructive Proof

Theorem

There exist irrational numbers r and s such that r^s is rational.

Proof.

We know that $a := \sqrt{2}$ is irrational.

A similar argument shows that $b := \log_2 9$ is too.

Then $a^b =$

A Constructive Proof

Theorem

There exist irrational numbers r and s such that r^s is rational.

Proof.

We know that $a := \sqrt{2}$ is irrational.

A similar argument shows that $b := \log_2 9$ is too.

Then $a^b = 3$. □

Constructive Reasoning

- ✓ If you want to prove $\exists x.p(x)$, you have to show how it can be found.

Constructive Reasoning

- ✓ If you want to prove $\exists x.p(x)$, you have to show how it can be found.
- ✓ If you want to prove $p \vee q$, you have to show how to figure out which of the two holds.

Constructive Reasoning

- ✓ If you want to prove $\exists x.p(x)$, you have to show how it can be found.
- ✓ If you want to prove $p \vee q$, you have to show how to figure out which of the two holds.
- ✓ If you want to prove $\forall x.\ \exists y.\ R(x, y)$, you have to provide a method of turning x's into suitable y's.

Constructive Reasoning

- ✓ If you want to prove $\exists x.p(x)$, you have to show how it can be found.
- ✓ If you want to prove $p \vee q$, you have to show how to figure out which of the two holds.
- ✓ If you want to prove $\forall x.\ \exists y.\ R(x, y)$, you have to provide a method of turning x's into suitable y's.

- ✓ Some predicates (called the *decidable* predicates) might satisfy

$$\forall x.\ P(x) \vee \neg P(x)$$

but we don't *assume* that every predicate is decidable.

Constructive Math = Math Definitions + Constructive Reasoning

Constructive Math = Math Definitions + Constructive Reasoning

There are at least three "versions" of Constructive Mathematics!

- ✓ Bishop-style: classical mathematics without excluded middle
  - Anything you prove is classically true.
  - Not every classical theorem is provable.
  - Proofs can be often harder, but yield "stronger" results.

Constructive Math = Math Definitions + Constructive Reasoning

There are at least three "versions" of Constructive Mathematics!

- ✓ Bishop-style: classical mathematics without excluded middle
  - Anything you prove is classically true.
  - Not every classical theorem is provable.
  - Proofs can be often harder, but yield "stronger" results.
- ✓ Brouwer-style (Bishop + Extra Axioms)
  - E.g., any function $\mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ is continuous.
- ✓ Russian-style (Bishop + Other Axioms)
  - E.g., every function $\mathbb{N} \to \mathbb{N}$ is computable by a Turing Machine.

Constructive Math Consequences

- ✓ Can't assume all properties are decidable

Constructive Math Consequences

- ✓ Can't assume all properties are decidable
  - Greater-than-zero on the integers is decidable

Constructive Math Consequences

- ✓ Can't assume all properties are decidable
  - Greater-than-zero on the integers is decidable
  - Greater-than-zero on the reals is not.

Constructive Math Consequences

- ✓ Can't assume all properties are decidable
  - Greater-than-zero on the integers is decidable
  - Greater-than-zero on the reals is not.
- ✓ Classical definitions often "split" into inequivalent definitions. For example:

  Or:

Constructive Math Consequences

- ✓ Can't assume all properties are decidable
  - Greater-than-zero on the integers is decidable
  - Greater-than-zero on the reals is not.
- ✓ Classical definitions often "split" into inequivalent definitions. For example:
  - $\neg(x = y)$ (i.e., $(x = y) \to \bot$)

  Or:

Constructive Math Consequences

- ✓ Can't assume all properties are decidable
  - Greater-than-zero on the integers is decidable
  - Greater-than-zero on the reals is not.
- ✓ Classical definitions often "split" into inequivalent definitions. For example:
  - $\neg(x = y)$ (i.e., $(x = y) \to \bot$)
  - $x \# y$ (i.e., $\exists n. \quad |x - y| > \frac{1}{n}$)

  Or:

Constructive Math Consequences

- ✓ Can't assume all properties are decidable
  - Greater-than-zero on the integers is decidable
  - Greater-than-zero on the reals is not.
- ✓ Classical definitions often "split" into inequivalent definitions.
  For example:
  - $\neg(x = y)$ (i.e., $(x = y) \to \bot$)
  - $x \# y$ (i.e., $\exists n. \quad |x - y| > \frac{1}{n}$)

  Or:
  - A set S is *finite* if there is a bijective function from $\{1, \ldots, n\} \to S$.

Constructive Math Consequences

- ✓ Can't assume all properties are decidable
  - Greater-than-zero on the integers is decidable
  - Greater-than-zero on the reals is not.
- ✓ Classical definitions often "split" into inequivalent definitions.
  For example:
  - $\neg(x = y)$ (i.e., $(x = y) \to \bot$)
  - $x \# y$ (i.e., $\exists n. \quad |x - y| > \frac{1}{n}$)

  Or:
  - A set S is *finite* if there is a bijective function from $\{1, \ldots, n\} \to S$.
  - A set S is *finitely enumerable* if there is a surjective (onto) function from $\{1, \ldots, n\} \to S$.

Constructive Math Consequences

- ✓ Can't assume all properties are decidable
  - Greater-than-zero on the integers is decidable
  - Greater-than-zero on the reals is not.
- ✓ Classical definitions often "split" into inequivalent definitions.
  For example:
  - $\neg(x = y)$ (i.e., $(x = y) \to \bot$)
  - $x \# y$ (i.e., $\exists n. \quad |x - y| > \frac{1}{n}$)

  Or:
  - A set S is *finite* if there is a bijective function from $\{1, \ldots, n\} \to S$.
  - A set S is *finitely enumerable* if there is a surjective (onto) function from $\{1, \ldots, n\} \to S$.
- ✓ Some seemingly obvious things might *not* be provable

$$[-1, 1] \quad = \quad [-1, 0] \cup [0, 1]$$

Constructive Math Consequences

- ✓ Can't assume all properties are decidable
  - Greater-than-zero on the integers is decidable
  - Greater-than-zero on the reals is not.
- ✓ Classical definitions often "split" into inequivalent definitions.
  For example:
  - $\neg(x = y)$ (i.e., $(x = y) \to \bot$)
  - $x \# y$ (i.e., $\exists n. \quad |x - y| > \frac{1}{n}$)

  Or:
  - A set S is *finite* if there is a bijective function from $\{1, \ldots, n\} \to S$.
  - A set S is *finitely enumerable* if there is a surjective (onto) function from $\{1, \ldots, n\} \to S$.
- ✓ Some seemingly obvious things might *not* be provable

$$[-1, 1] \quad = \quad [-1, 0] \cup [0, 1]$$

- ✓ Bishop: can't show whether non-continuous functions on $\mathbb{R}$ exist.

Constructive Math Consequences

- ✓ Can't assume all properties are decidable
  - Greater-than-zero on the integers is decidable
  - Greater-than-zero on the reals is not.
- ✓ Classical definitions often "split" into inequivalent definitions.
  For example:
  - $\neg(x = y)$ (i.e., $(x = y) \to \bot$)
  - $x \# y$ (i.e., $\exists n. \quad |x - y| > \frac{1}{n}$)

  Or:
  - A set S is *finite* if there is a bijective function from $\{1, \ldots, n\} \to S$.
  - A set S is *finitely enumerable* if there is a surjective (onto) function from $\{1, \ldots, n\} \to S$.
- ✓ Some seemingly obvious things might *not* be provable

$$[-1, 1] \quad = \quad [-1, 0] \cup [0, 1]$$

- ✓ Bishop: can't show whether non-continuous functions on $\mathbb{R}$ exist.
- ✓ Brouwer/Russian: Every function on $\mathbb{R}$ is provably continuous!

Constructive Math Consequences

- ✓ Can't assume all properties are decidable
  - Greater-than-zero on the integers is decidable
  - Greater-than-zero on the reals is not.
- ✓ Classical definitions often "split" into inequivalent definitions.
  For example:
  - $\neg(x = y)$ (i.e., $(x = y) \to \bot$)
  - $x \# y$ (i.e., $\exists n. \quad |x - y| > \frac{1}{n}$)

  Or:
  - A set S is *finite* if there is a bijective function from $\{1, \ldots, n\} \to S$.
  - A set S is *finitely enumerable* if there is a surjective (onto) function from $\{1, \ldots, n\} \to S$.
- ✓ Some seemingly obvious things might *not* be provable

$$[-1, 1] \quad = \quad [-1, 0] \cup [0, 1]$$

- ✓ Bishop: can't show whether non-continuous functions on $\mathbb{R}$ exist.
- ✓ Brouwer/Russian: Every function on $\mathbb{R}$ is provably continuous!
- ✓ Nonsense? No, we have *computable* mathematics.

Computable Mathematics

Traditional CS is closely connected to *discrete* mathematics

Data is finite (though arbitrarily large)

- ✓ Lists
- ✓ Trees
- ✓ Hash tables
- ✓ Bigints (arbitrarily large integers)

Computable Mathematics

Traditional CS is closely connected to *discrete* mathematics

Data is finite (though arbitrarily large)

- ✓ Lists
- ✓ Trees
- ✓ Hash tables
- ✓ Bigints (arbitrarily large integers)

But what if you want to compute with

- ✓ Real numbers
- ✓ Infinite sequences and limits
- ✓ Integration and differentiation
- ✓ Arbitrary continuous functions
- ✓ Arbitrary open (or closed, or compact) subsets of the plane
- ✓ Differential equations

Some Topics of Computable Mathematics

Some Topics of Computable Mathematics

1. How can a computer represent
   - "Real" real numbers
   - "Real" complex numbers?
   - Infinite sequences of reals?
   - Differentiable functions?
   - Arbitrary open (or closed, or compact) subsets of the plane?
   - ...

Some Topics of Computable Mathematics

1. How can a computer represent
   - "Real" real numbers
   - "Real" complex numbers?
   - Infinite sequences of reals?
   - Differentiable functions?
   - Arbitrary open (or closed, or compact) subsets of the plane?
   - ...

2. What operations are computable?
   - Multiplication of reals?
   - Square root of a complex?
   - Finding a zero of a function?
   - Integral of a function over a closed interval?
   - Intersection of two open sets?
   - ...

Some Topics of Computable Mathematics

1. How can a computer represent
   - "Real" real numbers
   - "Real" complex numbers?
   - Infinite sequences of reals?
   - Differentiable functions?
   - Arbitrary open (or closed, or compact) subsets of the plane?
   - ...

2. What operations are computable?
   - Multiplication of reals?
   - Square root of a complex?
   - Finding a zero of a function?
   - Integral of a function over a closed interval?
   - Intersection of two open sets?
   - ...

3. Can we do these efficiently (at least in practice)?

Problems with Floating-Point: Limited Range

Floating-point numbers are a *finite* subset of the *rationals*, i.e,

$$\pm(2^{24}+m)2^n \qquad \begin{cases} 0 \leq m < 2^{24} \\ -140 \leq n \leq 103 \end{cases}$$

Problems with Floating-Point: Limited Range

Floating-point numbers are a *finite* subset of the *rationals*, i.e,

$$\pm(2^{24}+m)2^n \qquad \begin{cases} 0 \leq m < 2^{24} \\ -140 \leq n \leq 103 \end{cases}$$

$$340282346638528859811704183484516925440$$
$$340282326356119256160033759537265639424$$
$$\vdots$$
$$1/2^{126}$$
$$+0$$
$$-0$$
$$1/2^{126}$$
$$\vdots$$
$$-340282326356119256160033759537265639424$$
$$-340282346638528859811704183484516925440$$

Problems with Floating-Point: Approximate Answers

Round-off and cancellation matter!

```
//  1/1 + 1/2 + ... + 1/(N-1)
float sum1 = 0.0;
for (int n = 1; n < N; ++n)
   sum1 += 1.0 / float(n);

//  1/(N-1) + ... + 1/2 + 1/1
float sum2 = 0.0;
for (int n = N-1; n > 0; --n)
   sum2 += 1.0 / float(n);
```

Problems with Floating-Point: Approximate Answers

Round-off and cancellation matter!

```
//  1/1 + 1/2 + ... + 1/(N-1)
float sum1 = 0.0;
for (int n = 1; n < N; ++n)
   sum1 += 1.0 / float(n);

//  1/(N-1) + ... + 1/2 + 1/1
float sum2 = 0.0;
for (int n = N-1; n > 0; --n)
   sum2 += 1.0 / float(n);
```

When N = 10^8, sum1 = 15.4037 and sum2 = 18.8079

Representing Real Numbers

What *does* it mean for a computer to "have" a *real* number?

Representing Real Numbers

What *does* it mean for a computer to "have" a *real* number?

What is a reasonable representation for a "real" *real* number?

Representing Real Numbers

What *does* it mean for a computer to "have" a *real* number?

What is a reasonable representation for a "real" *real* number?

First idea: infinite sequence of decimal digits

$$\begin{aligned} 1/2 &= [5, 0, 0, 0, 0, 0, 0, \ldots] \\ 1/30 &= [0, 3, 3, 3, 3, 3, 3, 3, \ldots] \end{aligned}$$

Representing Real Numbers

What *does* it mean for a computer to "have" a *real* number?

What is a reasonable representation for a "real" *real* number?

First idea: infinite sequence of decimal digits

$$\begin{aligned} 1/2 &= [5, 0, 0, 0, 0, 0, 0, \ldots] \\ 1/30 &= [0, 3, 3, 3, 3, 3, 3, 3, \ldots] \end{aligned}$$

In practice, we'd have to produce digits *on demand*, e.g.,

$$\texttt{real} \equiv \texttt{nat} \to \{0, \ldots, 9\}$$

Critique

The operation "Divide by 10" is computable! (How?)

$$1, 4, 1, 5, 9, 2, 6, 5, 3, 5, \ldots$$

Critique

The operation "Divide by 10" is computable! (How?)

$$1, 4, 1, 5, 9, 2, 6, 5, 3, 5, \ldots$$

The operation "Multiply by 3" is not!

0, 3, 1	× 3 =	0, 9, 3
0, 3, 3, 1	× 3 =	0, 9, 9, 3
0, 3, 3, 3, 1	× 3 =	0, 9, 9, 9, 3
0, 3, 9	× 3 =	1, 1, 7
0, 3, 3, 9	× 3 =	1, 0, 1, 7
0, 3, 3, 3, 9	× 3 =	1, 0, 0, 1, 7

Critique

The operation "Divide by 10" is computable! (How?)

$$1, 4, 1, 5, 9, 2, 6, 5, 3, 5, \ldots$$

The operation "Multiply by 3" is not!

0, 3, 1	× 3 =	0, 9, 3
0, 3, 3, 1	× 3 =	0, 9, 9, 3
0, 3, 3, 3, 1	× 3 =	0, 9, 9, 9, 3
0, 3, 9	× 3 =	1, 1, 7
0, 3, 3, 9	× 3 =	1, 0, 1, 7
0, 3, 3, 3, 9	× 3 =	1, 0, 0, 1, 7

Suppose you check 100 digits and see 0, 3, 3, 3, 3, ...
What's the *first* digit of the output?

Better Representations of Reals

As just a few examples among many

- ✓ Infinite sequences of *signed* digits (e.g., -9 to 9)
- ✓ Rapidly converging Cauchy Sequences:

$$\{\, a \in \mathbb{Q}^{\mathbb{N}} \mid \forall 0 \leq i < j.\, |a_i - a_j| < 2^i \,\}$$

(even better: use dyadic rationals!)

- ✓ Converging infinite sequences of open intervals (with rational endpoints)

Key Results

Given "good" representations of reals:

1. The basic operations ($+, -, \times, \div$) are computable.

Key Results

Given "good" representations of reals:

1. The basic operations ($+, -, \times, \div$) are computable.
2. So are square-root, exponential, sine, ...

Key Results

Given "good" representations of reals:

1. The basic operations ($+$, $-$, $\times$, $\div$) are computable.
2. So are square-root, exponential, sine, ...
3. Comparisons ($=$, $<$, $\leq$, etc.) on $\mathbb{R}$ are *not* computable.

Key Results

Given "good" representations of reals:

1. The basic operations ($+$, $-$, $\times$, $\div$) are computable.
2. So are square-root, exponential, sine, ...
3. Comparisons ($=$, $<$, $\leq$, etc.) on $\mathbb{R}$ are *not* computable.
4. Every computable function $\mathbb{R} \to \mathbb{R}$ is *continuous*

Key Results

Given "good" representations of reals:

1. The basic operations ($+$, $-$, $\times$, $\div$) are computable.
2. So are square-root, exponential, sine, ...
3. Comparisons ($=$, $<$, $\leq$, etc.) on $\mathbb{R}$ are *not* computable.
4. Every computable function $\mathbb{R} \to \mathbb{R}$ is *continuous*
5. The floor function $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{N}$ is *not* computable.

Key Results

Given "good" representations of reals:

1. The basic operations ($+$, $-$, $\times$, $\div$) are computable.
2. So are square-root, exponential, sine, ...
3. Comparisons ($=$, $<$, $\leq$, etc.) on $\mathbb{R}$ are *not* computable.
4. Every computable function $\mathbb{R} \to \mathbb{R}$ is *continuous*
5. The floor function $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{N}$ is *not* computable.
6. Every computable function $\mathbb{R} \to \mathbb{N}$ is constant.

Key Results

Given "good" representations of reals:

1. The basic operations ($+$, $-$, $\times$, $\div$) are computable.
2. So are square-root, exponential, sine, ...
3. Comparisons ($=$, $<$, $\leq$, etc.) on $\mathbb{R}$ are *not* computable.
4. Every computable function $\mathbb{R} \to \mathbb{R}$ is *continuous*
5. The floor function $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{N}$ is *not* computable.
6. Every computable function $\mathbb{R} \to \mathbb{N}$ is constant.
7. There is no computable operator that finds a zero of every continuous $f : [0, 1] \to \mathbb{R}$ with $f(0) < 0 < f(1)$.

Key Results

Given "good" representations of reals:

1. The basic operations ($+$, $-$, $\times$, $\div$) are computable.
2. So are square-root, exponential, sine, ...
3. Comparisons ($=$, $<$, $\leq$, etc.) on $\mathbb{R}$ are *not* computable.
4. Every computable function $\mathbb{R} \to \mathbb{R}$ is *continuous*
5. The floor function $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{N}$ is *not* computable.
6. Every computable function $\mathbb{R} \to \mathbb{N}$ is constant.
7. There is no computable operator that finds a zero of every continuous $f : [0, 1] \to \mathbb{R}$ with $f(0) < 0 < f(1)$.
8. Integrals on finite intervals are generally computable; derivatives of computable functions may not be.

Key Results

Given "good" representations of reals:

1. The basic operations ($+$, $-$, $\times$, $\div$) are computable.
2. So are square-root, exponential, sine, ...
3. Comparisons ($=$, $<$, $\leq$, etc.) on $\mathbb{R}$ are *not* computable.
4. Every computable function $\mathbb{R} \to \mathbb{R}$ is *continuous*
5. The floor function $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{N}$ is *not* computable.
6. Every computable function $\mathbb{R} \to \mathbb{N}$ is constant.
7. There is no computable operator that finds a zero of every continuous $f : [0, 1] \to \mathbb{R}$ with $f(0) < 0 < f(1)$.
8. Integrals on finite intervals are generally computable; derivatives of computable functions may not be.

Key Results

Given "good" representations of reals:

1. The basic operations ($+$, $-$, $\times$, $\div$) are computable.
2. So are square-root, exponential, sine, ...
3. Comparisons ($=$, $<$, $\leq$, etc.) on $\mathbb{R}$ are *not* computable.
4. Every computable function $\mathbb{R} \to \mathbb{R}$ is *continuous*
5. The floor function $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{N}$ is *not* computable.
6. Every computable function $\mathbb{R} \to \mathbb{N}$ is constant.
7. There is no computable operator that finds a zero of every continuous $f : [0, 1] \to \mathbb{R}$ with $f(0) < 0 < f(1)$.
8. Integrals on finite intervals are generally computable; derivatives of computable functions may not be.

These are the same results that one can get by constructive reasoning.

Why Constructive Logic?

- ✓ *Every classical proof can be turned into a constructive proof*
  - Largely a matter of adding some ¬¬'s.

Why Constructive Logic?

- ✓ *Every* classical proof can be turned into a constructive proof
  - Largely a matter of adding some ¬¬'s.
  - E.g., a classical proof that a solution exists, might become a constructive proof that not all values are non-solutions.

Why Constructive Logic?

- ✓ *Every* classical proof can be turned into a constructive proof
  - Largely a matter of adding some ¬¬'s.
  - E.g., a classical proof that a solution exists, might become a constructive proof that not all values are non-solutions.

Why Constructive Logic?

- ✓ *Every classical proof can be turned into a constructive proof*
  - Largely a matter of adding some $\neg\neg$'s.
  - E.g., a classical proof that a solution exists, might become a constructive proof that not all values are non-solutions.

- ✓ When you're proving things about computation, it's much nicer to say

Theorem

Every function $f : [0, 1] \to \mathbb{R}$ *is continuous*

Why Constructive Logic?

- ✓ *Every* classical proof can be turned into a constructive proof
  - Largely a matter of adding some $\neg\neg$'s.
  - E.g., a classical proof that a solution exists, might become a constructive proof that not all values are non-solutions.

- ✓ When you're proving things about computation, it's much nicer to say

Theorem
Every function $f : [0, 1] \to \mathbb{R}$ *is continuous*

than to say

Theorem
Every computable function f *from the computable reals between* 0 *and* 1 *to the computable reals is continuous with a computable modulus of continuity.*

The RZ System

Program developed in collaboration with Andrej Bauer, University of Ljubljana

- ✓ Input: Specifications in constructive logic
  - Sets, functions, predicates exist
  - Certain (constructive) axioms hold
- ✓ Output: Interfaces describing code
  - Representations must exist
  - Functions operating on these representations must exist
  - They must satisfy certain (classical!) properties.

Upshot: A translation of constructive logic statements into normal programmer-speak.

Formal Basis: Realizability

Realizability dates back to Kleene, as an attempt to interpret constructive logic classically.

Embarrassingly short summary of a rich topic:

- ✓ A realizer of a mathematical object
  (e.g., set, vector, tuple, group, smooth manifold)
  is an implementation.
- ✓ A realizer of a mathematical proposition
  $\forall x \in \mathbb{R}.\, \exists y \in \mathbb{R}.\, x = y \times y$
  is its "constructive" content:

```
real_t sqrt(real_t x);
  // forall x:real_t,  x = sqrt(x) * sqrt(x)
```

Realizers are tedious to describe by hand. Hence, RZ.

RZ Example: Finitely enumerable sets

The axioms (not shown) imply a data structure for supporting:

- ✓ The empty set
- ✓ A way to add an element to a set
- ✓ A "fold" or "reduce" operation, for binary operators that are
  - Commutative (`f(x,y) = f(y,x)`)
  - Idempotent (`f(x,x) = x`)

RZ Example: Finitely enumerable sets

The axioms (not shown) imply a data structure for supporting:

- ✓ The empty set
- ✓ A way to add an element to a set
- ✓ A "fold" or "reduce" operation, for binary operators that are
  - Commutative (`f(x,y) = f(y,x)`)
  - Idempotent (`f(x,x) = x`)

Interpretation: "finite sets" of items without decidable equality.

RZ Example: Finitely enumerable sets

The axioms (not shown) imply a data structure for supporting:

- ✓ The empty set
- ✓ A way to add an element to a set
- ✓ A "fold" or "reduce" operation, for binary operators that are
  - Commutative (`f(x,y) = f(y,x)`)
  - Idempotent (`f(x,x) = x`)

Interpretation: "finite sets" of items without decidable equality.

- ✓ Suppose you want to code up a "set" of exact real numbers

RZ Example: Finitely enumerable sets

The axioms (not shown) imply a data structure for supporting:

- ✓ The empty set
- ✓ A way to add an element to a set
- ✓ A "fold" or "reduce" operation, for binary operators that are
  - Commutative (`f(x,y) = f(y,x)`)
  - Idempotent (`f(x,x) = x`)

Interpretation: "finite sets" of items without decidable equality.

- ✓ Suppose you want to code up a "set" of exact real numbers
- ✓ Only possibility: unordered list, possibly with duplicates (why?)

RZ Example: Finitely enumerable sets

The axioms (not shown) imply a data structure for supporting:

- ✓ The empty set
- ✓ A way to add an element to a set
- ✓ A "fold" or "reduce" operation, for binary operators that are
  - Commutative (`f(x,y) = f(y,x)`)
  - Idempotent (`f(x,x) = x`)

Interpretation: "finite sets" of items without decidable equality.

- ✓ Suppose you want to code up a "set" of exact real numbers
- ✓ Only possibility: unordered list, possibly with duplicates (why?)
- ✓ Can't reliably compute the sum (not idempotent)

RZ Example: Finitely enumerable sets

The axioms (not shown) imply a data structure for supporting:

- ✓ The empty set
- ✓ A way to add an element to a set
- ✓ A "fold" or "reduce" operation, for binary operators that are
  - Commutative (`f(x,y) = f(y,x)`)
  - Idempotent (`f(x,x) = x`)

Interpretation: "finite sets" of items without decidable equality.

- ✓ Suppose you want to code up a "set" of exact real numbers
- ✓ Only possibility: unordered list, possibly with duplicates (why?)
- ✓ Can't reliably compute the sum (not idempotent)
- ✓ Can't reliably compute the size

RZ Example: Finitely enumerable sets

The axioms (not shown) imply a data structure for supporting:

- ✓ The empty set
- ✓ A way to add an element to a set
- ✓ A "fold" or "reduce" operation, for binary operators that are
  - Commutative (`f(x,y) = f(y,x)`)
  - Idempotent (`f(x,x) = x`)

Interpretation: "finite sets" of items without decidable equality.

- ✓ Suppose you want to code up a "set" of exact real numbers
- ✓ Only possibility: unordered list, possibly with duplicates (why?)
- ✓ Can't reliably compute the sum (not idempotent)
- ✓ Can't reliably compute the size
- ✓ Can't bound the size (as a deterministic function of the **`set`**)

RZ Example: Finitely enumerable sets

The axioms (not shown) imply a data structure for supporting:

- ✓ The empty set
- ✓ A way to add an element to a set
- ✓ A "fold" or "reduce" operation, for binary operators that are
  - Commutative (`f(x,y) = f(y,x)`)
  - Idempotent (`f(x,x) = x`)

Interpretation: "finite sets" of items without decidable equality.

- ✓ Suppose you want to code up a "set" of exact real numbers
- ✓ Only possibility: unordered list, possibly with duplicates (why?)
- ✓ Can't reliably compute the sum (not idempotent)
- ✓ Can't reliably compute the size
- ✓ Can't bound the size (as a deterministic function of the **`set`**)
- ✓ *Can* compute the `max` or `min`

RZ Example: Finitely enumerable sets

The axioms (not shown) imply a data structure for supporting:

- ✓ The empty set
- ✓ A way to add an element to a set
- ✓ A "fold" or "reduce" operation, for binary operators that are
  - Commutative (`f(x,y) = f(y,x)`)
  - Idempotent (`f(x,x) = x`)

Interpretation: "finite sets" of items without decidable equality.

- ✓ Suppose you want to code up a "set" of exact real numbers
- ✓ Only possibility: unordered list, possibly with duplicates (why?)
- ✓ Can't reliably compute the sum (not idempotent)
- ✓ Can't reliably compute the size
- ✓ Can't bound the size (as a deterministic function of the **`set`**)
- ✓ *Can* compute the `max` or `min`
- ✓ *Can* compute empty/non-empty

An Unrelated Application of Constructive Logic

Curry-Howard Isomorphism: The rules for type-checking are isomorphic to constructive proof rules.

$$\frac{a : T \qquad b : U}{(a, b) : T \times U} \qquad \frac{e : T \times U}{\mathsf{fst}(e) : T} \qquad \frac{e : T \times U}{\mathsf{snd}(e) : U}$$

$$\frac{P \qquad Q}{P \wedge Q} \qquad \frac{P \wedge Q}{P} \qquad \frac{P \wedge Q}{Q}$$

Practical application: Proofs-as-programs