Is Constructive Logic relevant for Computer Science?

Thorsten Altenkirch
University of Nottingham

Birth of Modern Mathematics

Birth of Modern Mathematics

Isaac Newton (1642 - 1727)

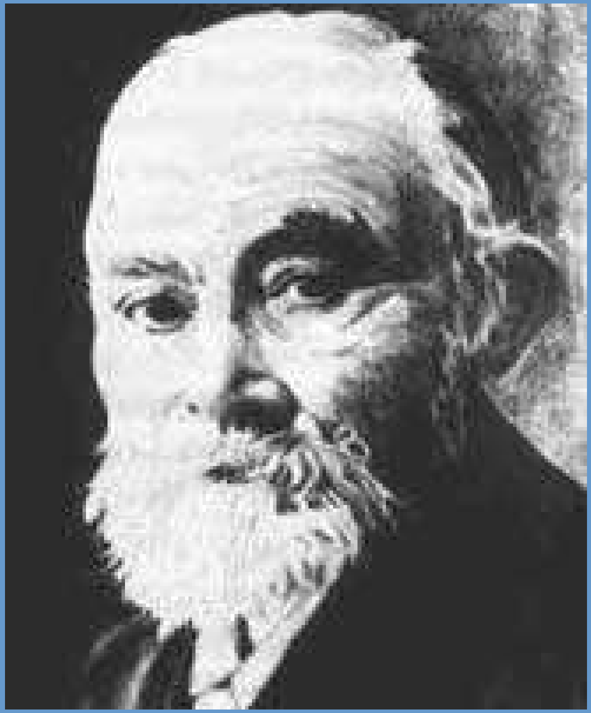
Birth of Modern Mathematics

Isaac Newton (1642 - 1727)

1687: Philosophiae Naturalis Principia Mathematica

19/20th century: Foundations?

19/20th century: Foundations?



Frege (1848-1925)

Russell (1872-1970)

≈ 1925: ZF set theory

Zermelo (1871-1953)

Fraenkel (1891-1965)

≈ 1925: ZF set theory

Zermelo (1871-1953)

Fraenkel (1891-1965)

End of story ?

Mathematics is *universal*

The foundations which are good for mathematical reasoning within natural sciences are equally useful in Computer Science.

Constructivism?

Constructivism?

- Computer Science focusses on *constructive solutions* to problems.

Constructivism?

- Computer Science focusses on *constructive solutions* to problems.
- Classical Mathematics is based on the *platonic* idea of truth.

Constructivism?

- Computer Science focusses on *constructive solutions* to problems.
- Classical Mathematics is based on the *platonic* idea of truth.
- Constructive Mathematics is based on the notion of *evidence* or proof.

BHK: Programs are evidence

BHK: Programs are evidence


Brouwer (1881-1966)


Heyting (1898-1980)


Kolmogorov (1903-1987)

$A \wedge (B \vee C) \implies (A \wedge B) \vee (A \wedge C)$, classically

$A \wedge (B \vee C) \implies (A \wedge B) \vee (A \wedge C)$, classically

A	B	C	$l = A \wedge (B \vee C)$	$r = A \wedge B \vee A \wedge C$	$l \implies r$
F	F	F	F	F	T
F	F	T	F	F	T
F	T	F	F	F	T
F	T	T	F	F	T
T	F	F	F	F	T
T	F	T	T	T	T
T	T	F	T	T	T
T	T	T	T	T	T

$A \wedge (B \vee C) \implies (A \wedge B) \vee (A \wedge C)$, classically

A	B	C	$l = A \wedge (B \vee C)$	$r = A \wedge B \vee A \wedge C$	$l \implies r$
F	F	F	F	F	T
F	F	T	F	F	T
F	T	F	F	F	T
F	T	T	F	F	T
T	F	F	F	F	T
T	F	T	T	T	T
T	T	F	T	T	T
T	T	T	T	T	T

- The same truth table shows that $A \wedge (B \vee C) \iff (A \wedge B) \vee (A \wedge C)$

BHK semantics

BHK semantics

- Evidence for $A \wedge B$ is given by pairs:
  $\textbf{type}\ a \wedge b = (a, b)$

BHK semantics

- Evidence for $A \wedge B$ is given by pairs:

  $$\textbf{type}\ a \wedge b = (a, b)$$

- Evidence for $A \vee B$ is tagged evidence for A or evidence for B.

  $$\textbf{data}\ a \vee b = \mathit{Inl}\ a \mid \mathit{Inr}\ b$$

BHK semantics

- Evidence for $A \wedge B$ is given by pairs:

  $$\textbf{type}\ a \wedge b = (a, b)$$

- Evidence for $A \vee B$ is tagged evidence for A or evidence for B.

  $$\textbf{data}\ a \vee b = \mathit{Inl}\ \ a \mid \mathit{Inr}\ \ b$$

- Evidence for $A \implies B$ is a program constructing evidence for B from evidence for A.

  $$\textbf{type}\ a \implies b = a \rightarrow b$$

$A \wedge (B \vee C) \implies (A \wedge B) \vee (A \wedge C)$, constructively

$A \wedge (B \vee C) \implies (A \wedge B) \vee (A \wedge C)$, constructively

$$f :: a \wedge (b \vee c) \rightarrow (a \wedge b) \vee (a \wedge c)$$

$$f\ (a, Inl\ b) = Inl\ (a, b)$$

$$f\ (a, Inr\ c) = Inr\ (a, c)$$

$A \wedge (B \vee C) \implies (A \wedge B) \vee (A \wedge C)$, constructively

$$f :: a \wedge (b \vee c) \rightarrow (a \wedge b) \vee (a \wedge c)$$

$$f\ (a, Inl\ b) = Inl\ (a, b)$$
$$f\ (a, Inr\ c) = Inr\ (a, c)$$

- The program is invertible, because the right hand sides are patterns.

$A \wedge (B \vee C) \implies (A \wedge B) \vee (A \wedge C)$, constructively

$$f :: a \wedge (b \vee c) \rightarrow (a \wedge b) \vee (a \wedge c)$$
$$f\ (a, \mathit{Inl}\ b) = \mathit{Inl}\ (a, b)$$
$$f\ (a, \mathit{Inr}\ c) = \mathit{Inr}\ (a, c)$$

- The program is invertible, because the right hand sides are patterns.
- This shows that the types are *isomorphic*.

Predicate logic

Predicate logic

- Evidence for $\forall x : S . P\, x$ is a function f which assigns to each $s : S$ evidence for $P\, s$.

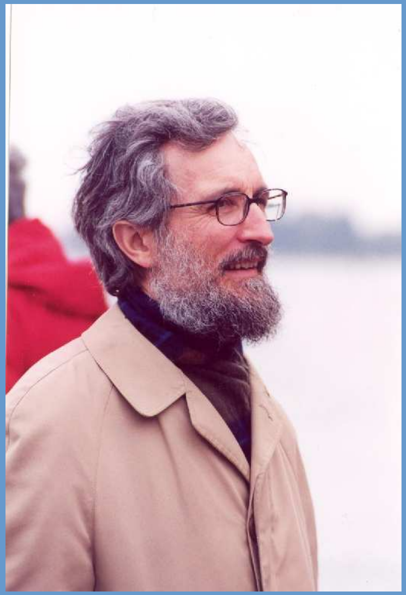
Predicate logic

- Evidence for $\forall x : S. P\, x$ is a function f which assigns to each $s : S$ evidence for $P\, s$.
- Evidence for $\exists x : S. P\, x$ is a pair (s, p) where $s : S$ and $p : P\, s$.

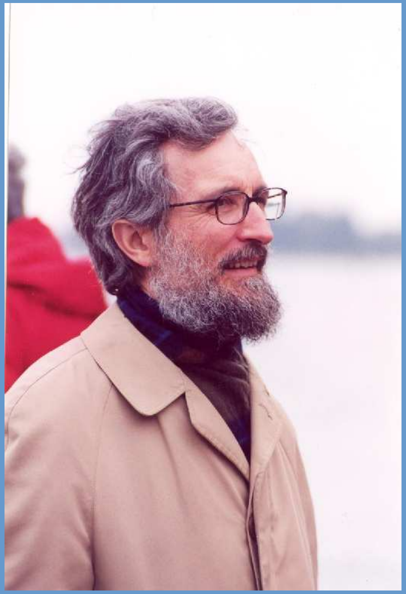
Predicate logic

- Evidence for $\forall x : S.P\,x$ is a function f which assigns to each $s : S$ evidence for $P\,s$.
- Evidence for $\exists x : S.P\,x$ is a pair (s, p) where $s : S$ and $p : P\,s$.
- We need *dependent types*!

Propositions = Types

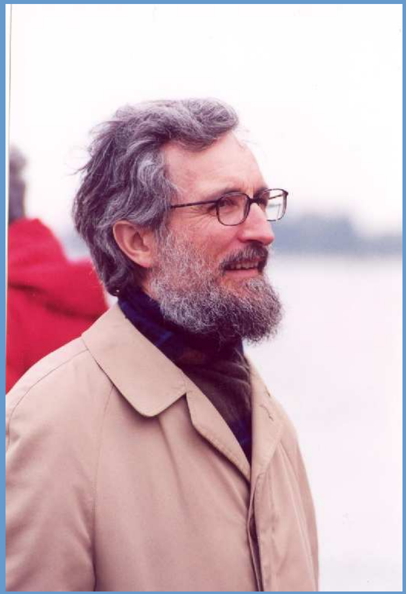


Propositions = Types



Per Martin-Löf

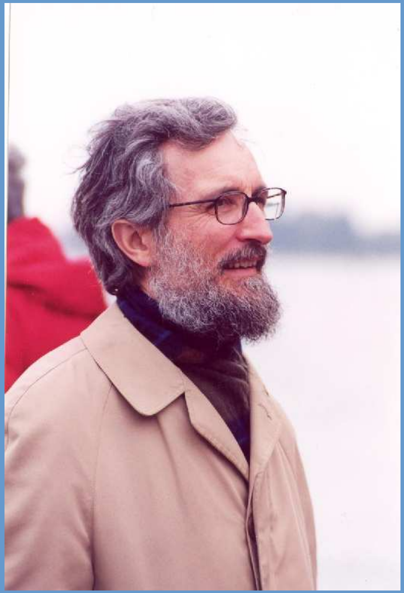
Propositions = Types



Per Martin-Löf

- Martin-Löf Type Theory

Propositions = Types



Per Martin-Löf

- Martin-Löf Type Theory
- Implementations: NuPRL, LEGO, ALF, COQ, AGDA, Epigram . . .

$A \vee \neg A$

$A \vee \neg A$

- We cannot prove $A \vee \neg A$, where $\neg A = A \implies \emptyset$, for an undecided proposition A.

$A \vee \neg A$

- We cannot prove $A \vee \neg A$, where $\neg A = A \implies \emptyset$, for an undecided proposition A.
- $\forall n : \mathrm{Nat}.\mathrm{Prime}\, n \vee \neg \mathrm{Prime}\, n$

$A \vee \neg A$

- We cannot prove $A \vee \neg A$, where $\neg A = A \implies \emptyset$, for an undecided proposition A.
- $\forall n : \mathrm{Nat}.\mathrm{Prime}\, n \vee \neg\mathrm{Prime}\, n$
  is provable, i.e. Prime is *decidable*.

$A \vee \neg A$

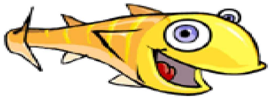
- We cannot prove $A \vee \neg A$, where $\neg A = A \implies \emptyset$, for an undecided proposition A.
- $\forall n : \mathrm{Nat}.\mathrm{Prime}\, n \vee \neg\mathrm{Prime}\, n$
  is provable, i.e. Prime is *decidable*.
- Indeed, the proof is the program which decides Prime.

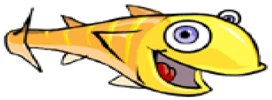
$A \vee \neg A$

- We cannot prove $A \vee \neg A$, where $\neg A = A \implies \emptyset$, for an undecided proposition A.
- $\forall n : \mathrm{Nat}.\mathrm{Prime}\, n \vee \neg \mathrm{Prime}\, n$
  is provable, i.e. Prime is *decidable*.
- Indeed, the proof is the program which decides Prime.
- $\forall n : \mathrm{Nat}.\mathrm{Halt}\, n \vee \neg \mathrm{Halt}\, n$

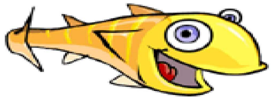
$A \vee \neg A$

- We cannot prove $A \vee \neg A$, where $\neg A = A \implies \emptyset$, for an undecided proposition A.
- $\forall n : \mathrm{Nat}.\mathrm{Prime}\, n \vee \neg \mathrm{Prime}\, n$
  is provable, i.e. Prime is *decidable*.
- Indeed, the proof is the program which decides Prime.
- $\forall n : \mathrm{Nat}.\mathrm{Halt}\, n \vee \neg \mathrm{Halt}\, n$
  is not provable, because Halt is *undecidable*.

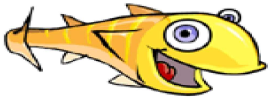
The classical Babelfish 

The classical Babelfish 

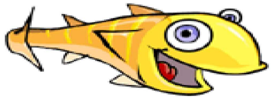
Classical reasoner says:	Babelfish translates to:

The classical Babelfish 

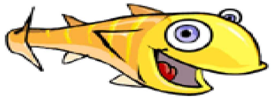
Classical reasoner says:	Babelfish translates to:
$A \vee B$	

The classical Babelfish 

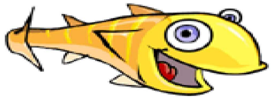
Classical reasoner says:	Babelfish translates to:
$A \vee B$	$\neg(\neg A \wedge \neg B)$

The classical Babelfish 

Classical reasoner says:	Babelfish translates to:
$A \vee B$	$\neg(\neg A \wedge \neg B)$
$\exists x : S.Px$	

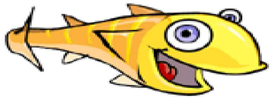
The classical Babelfish 

Classical reasoner says:	Babelfish translates to:
$A \vee B$	$\neg(\neg A \wedge \neg B)$
$\exists x : S.Px$	$\neg \forall x : S.\neg Px$

The classical Babelfish 

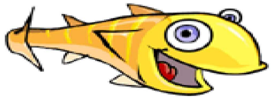
Classical reasoner says:	Babelfish translates to:
$A \vee B$	$\neg(\neg A \wedge \neg B)$
$\exists x : S.Px$	$\neg\forall x : S.\neg Px$

- *Negative translation*

The classical Babelfish 

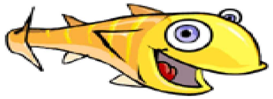
Classical reasoner says:	Babelfish translates to:
$A \vee B$	$\neg(\neg A \wedge \neg B)$
$\exists x : S.Px$	$\neg\forall x : S.\neg Px$

- *Negative translation*
- $A \vee \neg A$ is traslated to $\neg(\neg A \wedge \neg\neg A)$

The classical Babelfish 

Classical reasoner says:	Babelfish translates to:
$A \vee B$	$\neg(\neg A \wedge \neg B)$
$\exists x : S.Px$	$\neg \forall x : S.\neg Px$

- *Negative translation*
- $A \vee \neg A$ is traslated to $\neg(\neg A \wedge \neg\neg A)$ which is constructively provable.

The classical Babelfish 

Classical reasoner says:	Babelfish translates to:
$A \vee B$	$\neg(\neg A \wedge \neg B)$
$\exists x : S.Px$	$\neg\forall x : S.\neg Px$

- *Negative translation*
- $A \vee \neg A$ is traslated to $\neg(\neg A \wedge \neg\neg A)$ which is constructively provable.
- A classical reasoner is somebody who is unable to say anything positive.

The Axiom of Choice ?

The Axiom of Choice ?

- $$\frac{\forall x : S. \exists y : T. R\, x\, y}{\exists f : S \to T. \forall x : S. R\, x\, (f\, x)}\ \mathrm{AC}$$

The Axiom of Choice ?

- $$\frac{\forall x : S. \exists y : T. R\, x\, y}{\exists f : S \to T. \forall x : S. R\, x\, (f\, x)}\ \mathrm{AC}$$

  is provable constructively.

The Axiom of Choice ?

- $$\frac{\forall x : S.\exists y : T.R\,x\,y}{\exists f : S \to T.\forall x : S.R\,x\,(f\,x)}\,\mathrm{AC}$$

  is provable constructively.
- However, its negative translation:

  $$\frac{\forall x : S.\neg\forall y : T.\neg R\,x\,y}{\neg\forall f : S \to T.\neg\forall x : S.R\,x\,(f\,x)}\,\mathrm{CAC}$$

  is not.

The Axiom of Choice ?

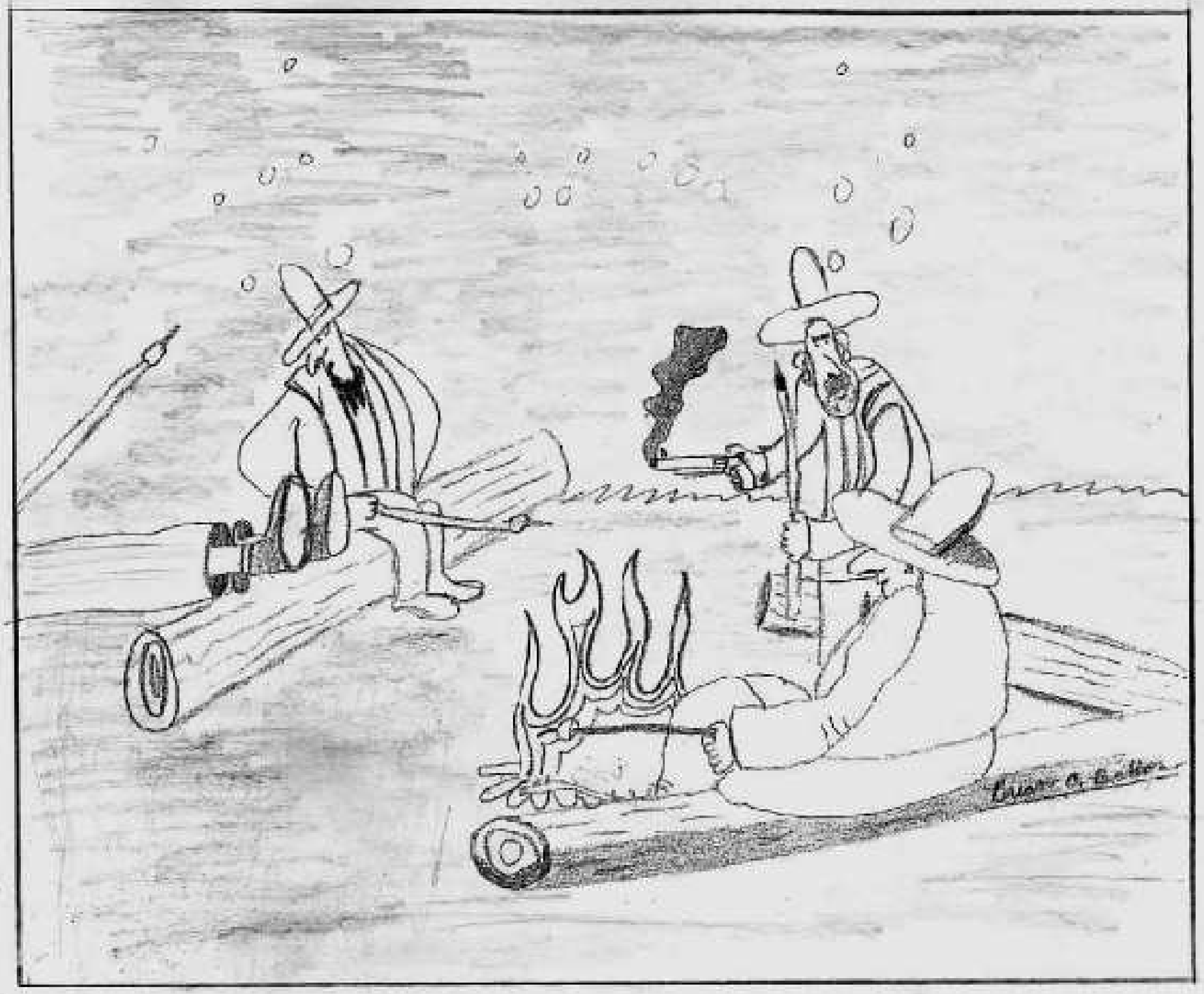
- $$\frac{\forall x : S.\exists y : T.R\,x\,y}{\exists f : S \to T.\forall x : S.R\,x\,(f\,x)}\,\mathrm{AC}$$

  is provable constructively.
- However, its negative translation:

  $$\frac{\forall x : S.\neg\forall y : T.\neg R\,x\,y}{\neg\forall f : S \to T.\neg\forall x : S.R\,x\,(f\,x)}\,\mathrm{CAC}$$

  is not.
- There is *empirical evidence* that CAC is consistent.

Summary



You guys are both my witnesses... He insinuated that
ZFC set theory is superior to Type Theory!