Propositions as Types

Aaron McClellan and Dr. Philip Wadler

February 28, 2023

Intro

Goals

State propositions as types

Motivate, introduce theory of computation

Introduce theory of logic

Introduce natural deduction

Introduce lambda calculus as a synonym of logic

Goals

$$\left.\begin{array}{l}\text{State propositions as types}\\ \text{Motivate, introduce theory of computation}\\ \text{Introduce theory of logic}\end{array}\right\}\text{Part 1}$$

$$\left.\begin{array}{l}\text{Introduce natural deduction}\\ \text{Introduce lambda calculus as a synonym of logic}\end{array}\right\}\text{Part 2}$$

Part 1

Propositions as types

Propositions $\longleftrightarrow$ types

Propositions as types

Propositions $\longleftrightarrow$ types

Proofs $\longleftrightarrow$ programs

Propositions as types

Propositions $\longleftrightarrow$ types

Proofs $\longleftrightarrow$ programs

Simplification of proofs $\longleftrightarrow$ evaluation of programs

Entscheidungsproblem

David Hilbert in 1930

Hilbert's Entscheidungsproblem

An "effectively calculable" algorithm to prove all statements

Entscheidungsproblem

prove all statements

Entscheidungsproblem

all statements

Kurt Gödel

“effectively calculable”

Hilbert was vague and imprecise

“effectively calculable”

Hilbert was vague and imprecise

Church’s lambda calculus in 1936

Gödel’s recursive functions in 1936

Turing’s Turing machines in 1937

Lambda calculus

Inductive construction of primitives ("λ-definable")

Evaluation represented as substitution rules ("normalization " to a "normal form")

Lambda calculus

Inductive construction of primitives (“λ-definable”)

Evaluation represented as substitution rules (“normalization ” to a “normal form”)

Showed problems with no λ-definable solution i.e. uncalculable/undecidable problems

Proposed λ-definable as definition for “effectively calculable”

Proposal *eventually* accepted

Lambda calculus

Inductive construction of primitives (“λ-definable”)

Evaluation represented as substitution rules (“normalization ” to a “normal form”)

Showed problems with no λ-definable solution i.e. uncalculable/undecidable problems

Proposed λ-definable as definition for “effectively calculable”

Proposal *eventually* accepted

Types introduced to guarantee termination

The theory of proof

Gentzen introduced natural deduction

Logical rules come in pairs: introduction and elimination

Introduce $A \wedge B$ with proofs of A and B, eliminate by concluding A or B

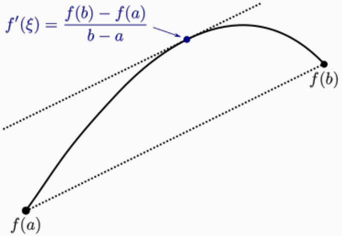
Rules to simplify proofs $\implies$ consistency

Subformula principle: proofs contain only what is necessary (not “roundabout”)

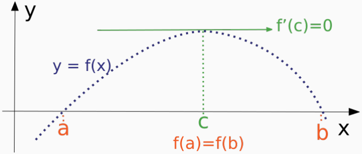
$$sfp(A \wedge B) = \{A \wedge B\} \cup sfp(A) \cup sfp(B)$$

Used a roundabout proof for SFP in natural deduction

Roundabout proofs?



Mean value theorem



Rolle’s theorem

Intuitionistic logic

Introduced by Brouwer in early 1900s

Generalization of classical logic

Predates ZFC...

...but not Russell's paradox

Disagreement about the nature of infinity

Intuitionistic logic

Proof of $A \lor B$ must show *which* of A or B

Law of excluded middle ($A \lor \neg A$) doesn't hold generally

Proof by negation holds, proof by contradiction doesn't generally

$$(A \implies \text{false}) \implies \neg A \qquad (\neg A \implies \text{false}) \implies A$$

Why?

Simplifies teaching

Easier to discern types

Allows immediate program generation

Introduces other systems of logic

Why?

Simplifies teaching

Easier to discern types

Allows immediate program generation

Introduces other systems of logic 👀

Extensions to propositions as types

Extensions to propositions as types

$\forall$ and $\exists$ quantifiers $\longleftrightarrow$ dependent types

Classical logic $\longleftrightarrow$ continuation-passing style

Second-order logic $\longleftrightarrow$ second-order lambda calculus

Higher-order logic $\longleftrightarrow$ depends (Isabelle/HOL)

Extensions to propositions as types

Modal logic ("necessarily true" + "sometimes true") $\longleftrightarrow$ monads (distributed computation, Haskell)

Temporal logic ("holds now", "will hold eventually", "will hold soon") $\longleftrightarrow$ partial evaluation (computer engineering)

Linear logic (use assumptions exactly once) $\longleftrightarrow$ linear types, session types (C++, quantum computing, communication)

Extensions to propositions as types

Cartesian closed category $\longleftrightarrow$ simply-typed lambda calculus (category theory, quantum physics)

Extensions to propositions as types

Cartesian closed category $\longleftrightarrow$ simply-typed lambda calculus (category theory, quantum physics)

Topology $\longleftrightarrow$ homotopy type theory

Part 2

Natural deduction

Colors

Logical formulas ($A \wedge B$) will always blue...

...unless I'm using color to group things together

Intuitive reading

$$\frac{\textit{premise}1 \quad \textit{premise}2 \quad \textit{premise}3}{\textit{conclusion}}\ \text{rule or logic}$$

Intuitive reading

$$\frac{\textit{premise1} \qquad \textit{premise2} \qquad \textit{premise3}}{\textit{conclusion}}\ \text{rule or logic}$$

If the premises hold then we can conclude that the conclusion holds by some rule or logic.

& rules

$$\frac{A \qquad B}{A \wedge B}\ \wedge\text{-I} \qquad\qquad \frac{A \wedge B}{A}\ \wedge\text{-E}_1 \qquad\qquad \frac{A \wedge B}{B}\ \wedge\text{-E}_2$$

$\Longrightarrow$ rules

$$\dfrac{\begin{array}{c}[A]^x \\ \vdots \\ B\end{array}}{A \Longrightarrow B}\ \text{Imp-I}^x$$

$$\dfrac{A \Longrightarrow B \qquad A}{B}\ \text{Imp-E}$$

Intuitive reading

$$\begin{array}{c}[A]^x \\ \vdots \\ B\end{array}$$

The assumption of A leads to a proof of B.

Example proof

Let's prove $B \land A \implies A \land B$.

Example proof

Let's prove $B \wedge A \implies A \wedge B$.

$$\dfrac{\dfrac{\dfrac{[B \wedge A]^z}{A}\ \wedge\text{-E}_2 \qquad \dfrac{[B \wedge A]^z}{B}\ \wedge\text{-E}_1}{A \wedge B}\ \wedge\text{-I}}{B \wedge A \implies A \wedge B}\ \text{Imp-I}^z$$

Simplification rules

$$\dfrac{\dfrac{\begin{matrix}\vdots \\ P\end{matrix} \qquad \begin{matrix}\vdots \\ Q\end{matrix}}{P \wedge Q}\;\wedge\text{-I}}{P}\;\wedge\text{-E}_1 \qquad \text{simplifies to} \qquad \begin{matrix}\vdots \\ P\end{matrix}$$

$$\dfrac{\dfrac{\begin{matrix}[P]^x \\ \vdots \\ Q\end{matrix}}{P \implies Q}\;\text{Imp-I}^x \qquad \begin{matrix}\vdots \\ P\end{matrix}}{Q}\;\text{Imp-E} \qquad \text{simplifies to} \qquad \begin{matrix}\vdots \\ P \\ \vdots \\ Q\end{matrix}$$

Proof simplification example

$$\dfrac{\dfrac{\dfrac{\dfrac{[B \wedge A]^z}{A}\ \wedge\text{-E}_2 \qquad \dfrac{[B \wedge A]^z}{B}\ \wedge\text{-E}_1}{A \wedge B}\ \wedge\text{-I}}{(B \wedge A) \implies (A \wedge B)}\ \text{Imp-I}^z \qquad \dfrac{B \qquad A}{B \wedge A}\ \wedge\text{-I}}{A \wedge B}\ \text{Imp-E}$$

Left is the proof from earlier

Right is the machinery to use the implication

Proof simplification example

$$\dfrac{\dfrac{\dfrac{[B \wedge A]^z}{A}\ \wedge\text{-E}_2 \qquad \dfrac{[B \wedge A]^z}{B}\ \wedge\text{-E}_1}{A \wedge B}\ \wedge\text{-I}}{(B \wedge A) \implies (A \wedge B)}\ \text{Imp-I}^z \qquad \dfrac{B \qquad A}{B \wedge A}\ \wedge\text{-I}$$

$$\dfrac{(B \wedge A) \implies (A \wedge B) \qquad B \wedge A}{A \wedge B}\ \text{Imp-E}$$

simplifies to

$$\dfrac{\dfrac{\dfrac{B \qquad A}{B \wedge A}\ \wedge\text{-I}}{A}\ \wedge\text{-E}_2 \qquad \dfrac{\dfrac{B \qquad A}{B \wedge A}\ \wedge\text{-I}}{B}\ \wedge\text{-E}_1}{A \wedge B}\ \wedge\text{-I}$$

Simplification rules

$$\dfrac{\dfrac{\begin{array}{cc}\vdots & \vdots\\ P & Q\end{array}}{P \land Q}\;\land\text{-I}}{P}\;\land\text{-E}_1 \qquad \text{simplifies to} \qquad \begin{array}{c}\vdots\\ P\end{array}$$

$$\dfrac{\dfrac{\begin{array}{c}[P]^x\\ \vdots\\ Q\end{array}}{P \implies Q}\;\text{Imp-I}^x \qquad \begin{array}{c}\vdots\\ P\end{array}}{Q}\;\text{Imp-E} \qquad \text{simplifies to} \qquad \begin{array}{c}\vdots\\ P\\ \vdots\\ Q\end{array}$$

Proof simplification example

$$\dfrac{\dfrac{\dfrac{B \quad A}{B \wedge A}\,\wedge\text{-I}}{A}\,\wedge\text{-E}_2 \qquad \dfrac{\dfrac{B \quad A}{B \wedge A}\,\wedge\text{-I}}{B}\,\wedge\text{-E}_1}{A \wedge B}\,\wedge\text{-I}$$

simplifies to

$$\dfrac{A \quad B}{A \wedge B}\,\wedge\text{-I}$$

Lambda calculus

Untyped lambda calculus

Let L, M, N be lambda terms and x, y, z be variables

$$\Lambda := x \mid \lambda x.M \mid MN$$

A system of functions (and later types); no sets

Intuitive reading

$$\Lambda := x \mid \lambda x.M \mid MN$$

Variables are placeholders for lambda terms

Abstractions are functions with a variable as input
$f(x) = M$ or `func f(x) { return M(); }`

Applications start computation in M with N as input
$M(N)$

Evaluation is applying all terms of the form $(\lambda x.M)N$

Exercise

What does $\lambda x.x$ do?

Exercise

What does $\lambda x.x$ do? Identity function

What does $(\lambda x.xx)(\lambda x.xx)$ do?

Exercise

What does $\lambda x.x$ do? Identity function

What does $(\lambda x.xx)(\lambda x.xx)$ do?

Hint: Let $M = \lambda x.xx$ and $N = \lambda x.xx$

Exercise

What does $\lambda x.x$ do? Identity function

What does $(\lambda x.xx)(\lambda x.xx)$ do?

Hint: Let $M = \lambda x.xx$ and $N = \lambda x.xx$

Hint: $MN = M(\lambda x.xx)$

Exercise

What does $\lambda x.x$ do? Identity function

What does $(\lambda x.xx)(\lambda x.xx)$ do?

Hint: Let $M = \lambda x.xx$ and $N = \lambda x.xx$

Hint: $MN = M(\lambda x.xx)$

$(\lambda x.xx)(\lambda x.xx)$ expands to itself $\implies$ infinite loop!

Types

Solution to nontermination

Let A, B be types

Let $A \to B$ be function types

Types

Solution to nontermination

Let A, B be types

Let $A \to B$ be function types

Function types $\implies$ product types $A \times B$

Intuitive reading

$$A \to B$$
$$f : A \to B$$

$$A \times B$$
$$(a \in A, b \in B)$$

Typed lambda calculus

Append a type to every lambda term

$M : A$, read M has type A

$\times$ rules

$$\frac{M : A \qquad N : B}{\langle M, N \rangle : A \times B} \times\text{-I}$$

$$\frac{L : A \times B}{\mathrm{fst}\ L : A} \times\text{-E}_1 \qquad \frac{L : A \times B}{\mathrm{sec}\ L : B} \times\text{-E}_2$$

$\to$ rules

$$\dfrac{\begin{array}{c}[x : A]^x \\ \vdots \\ N : B\end{array}}{\lambda x.N : A \to B} \to\text{-I}^x \qquad \dfrac{L : A \to B \quad M : A}{LM : B} \to\text{-E}$$

Intuitive reading

$$\begin{array}{c}[x : A]^{x} \\ \vdots \\ N : B\end{array}$$

A variable x of type A is used to construct a term N of type B.

Example program

Let's create a program of type $B \times A \rightarrow A \times B$.

Example program

Let's create a program of type $B \times A \to A \times B$.

$$\dfrac{\dfrac{\dfrac{[z : B \times A]^z}{\mathrm{sec}\ z : A}\ \times\text{-E}_2 \qquad \dfrac{[z : B \times A]^z}{\mathrm{fst}\ z : B}\ \times\text{-E}_1}{\langle \mathrm{sec}\ z, \mathrm{fst}\ z \rangle : A \times B}\ \times\text{-I}}{\lambda z.\langle \mathrm{sec}\ z, \mathrm{fst}\ z \rangle : B \times A \to A \times B}\ \to\text{-I}^z$$

Evaluation rules

$$\dfrac{\dfrac{\begin{matrix}\vdots \\ M : P\end{matrix} \qquad \begin{matrix}\vdots \\ N : Q\end{matrix}}{\langle M, N\rangle : P \times Q}\times\text{-I}}{\mathsf{fst}\ \langle M, N\rangle : P}\times\text{-E}_1 \qquad \text{evaluates to} \qquad \begin{matrix}\vdots \\ M : P\end{matrix}$$

$$\dfrac{\dfrac{\begin{matrix}[x : P]^x \\ \vdots \\ N : Q\end{matrix}}{\lambda x.N : P \to Q}\to\text{-I}^x \qquad \begin{matrix}\vdots \\ M : P\end{matrix}}{(\lambda x.N)M : Q}\to\text{-E} \qquad \text{evaluates to} \qquad \begin{matrix}\vdots \\ M : P \\ \vdots \\ N[M/x] : Q\end{matrix}$$

Intuitive reading

$$N[M/x]$$

N with M replacing x.

Program evaluation example

$$
\dfrac{\dfrac{\dfrac{\dfrac{[B \times A]^z}{\mathsf{sec}\ z : A}\ \times\text{-E}_2 \qquad \dfrac{[B \times A]^z}{\mathsf{fst}\ z : B}\ \times\text{-E}_1}{\langle \mathsf{sec}\ z, \mathsf{fst}\ z \rangle : A \times B}\ \times\text{-I}}{\lambda z.\langle \mathsf{sec}\ z, \mathsf{fst}\ z \rangle : B \times A \to A \times B}\ \to\text{-I}^z \qquad \dfrac{y : B \qquad x : A}{\langle y, x \rangle : B \times A}\ \times\text{-I}}{(\lambda z.\langle \mathsf{sec}\ z, \mathsf{fst}\ z \rangle)\langle y, x \rangle : A \times B}\ \to\text{-E}
$$

Left is the program from earlier

Right is the machinery to run the function

Program evaluation example

$$\dfrac{\dfrac{\dfrac{[z : B \times A]^z}{\mathsf{sec}\ z : A}\ \times\text{-E}_2 \qquad \dfrac{[z : B \times A]^z}{\mathsf{fst}\ z : B}\ \times\text{-E}_1}{\langle \mathsf{sec}\ z, \mathsf{fst}\ z \rangle : A \times B}\ \times\text{-I}}{\lambda z.\langle \mathsf{sec}\ z, \mathsf{fst}\ z \rangle : B \times A \to A \times B}\ \to\text{-I}^z \qquad \dfrac{y : B \qquad x : A}{\langle y, x \rangle : B \times A}\ \times\text{-I}$$

$$\dfrac{\qquad\qquad}{(\lambda z.\langle \mathsf{sec}\ z, \mathsf{fst}\ z \rangle)\langle y, x \rangle : A \times B}\ \to\text{-E}$$

evaluates to

$$\dfrac{\dfrac{\dfrac{y : B \qquad x : A}{\langle y, x \rangle : B \times A}\ \times\text{-I}}{\mathsf{sec}\ \langle y, x \rangle : A}\ \times\text{-E}_2 \qquad \dfrac{\dfrac{y : B \qquad x : A}{\langle y, x \rangle : B \times A}\ \times\text{-I}}{\mathsf{fst}\ \langle y, x \rangle : A}\ \times\text{-E}_2}{\langle \mathsf{sec}\ \langle y, x \rangle, \mathsf{fst}\ \langle y, x \rangle \rangle : A \times B}\ \times\text{-I}$$

Program evaluation example

$$\dfrac{\dfrac{\dfrac{y : B \qquad x : A}{\langle y, x\rangle : B \times A}\ \times\text{-I}}{\mathsf{sec}\ \langle y, x\rangle : A}\ \times\text{-E}_2 \qquad \dfrac{\dfrac{y : B \qquad x : A}{\langle y, x\rangle : B \times A}\ \times\text{-I}}{\mathsf{fst}\ \langle y, x\rangle : A}\ \times\text{-E}_2}{\langle \mathsf{sec}\ \langle y, x\rangle, \mathsf{fst}\ \langle y, x\rangle\rangle : A \times B}\ \times\text{-I}$$

evaluates to

$$\dfrac{x : A \qquad y : B}{\langle x, y\rangle : A \times B}\ \times\text{-I}$$

Understanding?

Do you understand...

...the correspondence?

Do you understand...

...the correspondence?

...how natural deduction describes semantics?

...how program generation could work?

...why computer science would use this over other formulations?

Do you understand...

...the correspondence?

...how natural deduction describes semantics?

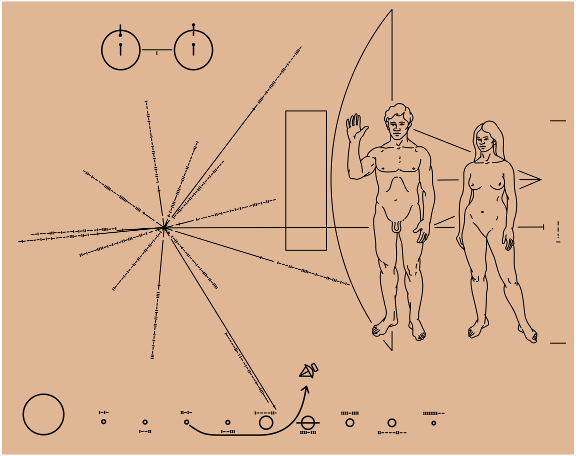
...how program generation could work?

...why computer science would use this over other formulations?

...more than that?

The punchline

The punchline



Plaque of the Pioneer

The punchline

It would be a mistake to characterize lambda calculus as universal, because calling it universal would be *too limiting*.