Propositions as Types

Philip Wadler
University of Edinburgh
Part I

Computability
Euclid (325–265 BCE)
Al Khwarizmi (780–850 CE)
Algorithm, formalised

Alonzo Church: *Lambda calculus*
An unsolvable problem of elementary number theory, *Bulletin the American Mathematical Society*, May 1935

Kurt Gödel: *Recursive functions*

Alan M. Turing: *Turing machines*
On computable numbers, with an application to the *Entscheidungsproblem*, *Proceedings of the London Mathematical Society*, received 25 May 1936
David Hilbert (1862-1943)
David Hilbert (1928)
Entscheidungsproblem
Kurt Gödel (1906-1978)
42. $Ax(x) \equiv Z-Ax(x) \lor A-Ax(x) \lor L_1-Ax(x) \lor L_2-Ax(x) \lor R-Ax(x) \lor M-Ax(x)$,
x is an axiom.

43. $Fl(x, y, z) \equiv y = z \text{ Imp } x \lor (\text{Ev})[v \leq x \& \text{Var}(v) \& x = v \text{ Gen } y]$,
x is an immediate consequence of y and z.

44. $Bw(x) \equiv (n)[0 < n \leq l(x) \rightarrow Ax(n \text{ Gl } x) \lor (Ep, q)[0 < p, q < n \& Fl(n \text{ Gl } x, p \text{ Gl } x, q \text{ Gl } x)] \& l(x) > 0$,
x is a proof array (a finite sequence of formulas, each of which is either an axiom
or an immediate consequence of two of the preceding formulas.

45. $x \parallel y \equiv Bw(x) \& [l(x)] \text{ Gl } x = y$,
x is a proof of the formula y.

46. $Bew(x) \equiv (Ey)y B x$,
x is a provable formula. (Bew(x) is the only one of the notions 1–46 of which we
cannot assert that it is recursive.)

“This statement is not provable”
Alonzo Church (1903–1995)
AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.\textsuperscript{1}

By Alonzo Church.

The purpose of the present paper is to propose a definition of effective calculability\textsuperscript{8} which is thought to correspond satisfactorily to the somewhat vague intuitive notion in terms of which problems of this class are often stated, and to show, by means of an example, that not every problem of this class is solvable.

We introduce at once the following infinite list of abbreviations,

\[ 1 \to \lambda ab \cdot a(b), \]
\[ 2 \to \lambda ab \cdot a(a(b)), \]
\[ 3 \to \lambda ab \cdot a(a(a(b))), \]

and so on, each positive integer in Arabic notation standing for a formula of the form \( \lambda ab \cdot a(a(\cdots a(b)\cdots)) \).
Alonzo Church (1932)
\( \lambda \)-calculus

\[
L, M, N ::= \quad x
\]
\[
\quad | \quad (\lambda x. N)
\]
\[
\quad | \quad (L M)
\]
Kurt Gödel (1906-1978)
The substitution

1) \( \varphi(x_1, \ldots, x_n) = \theta(\chi_1(x_1, \ldots, x_n), \ldots, \chi_m(x_1, \ldots, x_n)) \),
and the ordinary recursion with respect to one variable

\[ \varphi(0, x_2, \ldots, x_n) = \psi(x_2, \ldots, x_n) \]

(2)

\[ \varphi(y + 1, x_2, \ldots, x_n) = \chi(y, \varphi(y, x_2, \ldots, x_n), x_2, \ldots, x_n), \]

where \( \theta, \chi_1, \ldots, \chi_m, \psi, \chi \) are given functions of natural numbers, are examples of the definition of a function \( \varphi \) by equations which provide a step by step process for computing the value \( \varphi(k_1, \ldots, k_n) \) for any given set \( k_1, \ldots, k_n \) of natural numbers. It is known that there are other definitions of this sort, e.g. certain recursions with respect to two or more variables simultaneously, which cannot be reduced to a succession of substitutions and ordinary recursions\(^2\)). Hence, a characterization of the notion of recursive definition in general, which would include all these cases, is desirable. A definition of general recursive function of natural numbers was suggested by Herbrand to Gödel, and was used by Gödel with an important modification in a series of lectures at Princeton in 1934. In this paper we offer several observations on general recursive functions, using essentially Gödel's form of the definition.
Alan Turing (1912-1954)
ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO
THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

[Received 28 May, 1936.—Read 12 November, 1936.]

The “computable” numbers may be described briefly as the real
numbers whose expressions as a decimal are calculable by finite means.

In §§ 9, 10 I give some arguments with the intention of showing that the
computable numbers include all numbers which could naturally be
regarded as computable. In particular, I show that certain large classes
of numbers are computable. They include, for instance, the real parts of
all algebraic numbers, the real parts of the zeros of the Bessel functions,
the numbers π, e, etc. The computable numbers do not, however, include
all definable numbers, and an example is given of a definable number
which is not computable.
Is Mathematics Invented or Discovered?
Part II

Propositions as Types
Gerhard Gentzen (1909-1945)
Gerhard Gentzen (1935)

Natural Deduction

\[
\begin{array}{cccc}
& \&-I & \&-E & \lor-I & \lor-E \\
\hline
A & B & A & B & A & B & C & C \\
\hline
\forall-I & \forall-E & \exists-I & \exists-E \\
\exists a & \forall x \exists x & \exists a & \forall x \exists x & C \\
\hline
\Rightarrow-I & \Rightarrow-E & \neg I & \neg E \\
A & \Rightarrow A & [A] & A & \Rightarrow A & \land & \neg A & \land & \neg A & \land & \neg A & D \\
\hline
\end{array}
\]
Gerhard Genesen (1935)
Natural Deduction

\[
\begin{align*}
[A]^x \\
\vdots \\
B \\
\hline \\
A \supset B & \quad \quad \quad A \supset B \quad A \quad \supset\text{-E} \\
& \quad \quad \quad A \quad \supset\text{-E}
\end{align*}
\]

\[
\begin{align*}
A & B \quad \&\text{-I} \\
A \& B \quad A \quad \&\text{-E}_1 \\
A \& B \quad B \quad \&\text{-E}_2
\end{align*}
\]
A proof

\[
\begin{align*}
[B \land A]^\_z \\
\_2 &-E \quad & [B \land A]^\_z \\
\_1 &-E \\
A &-I \\
B &-I \\
A \land B &-I \\
(B \land A) \supset (A \land B) &-I
\end{align*}
\]
Simplifying proofs

\[ [A]^x \]
\[ \vdots \]
\[ B \]
\[
\begin{array}{c}
A \supset B \\
\hline
\end{array}
\]
\(-\text{I}^x\)
\[ \vdots \]
\[ A \]
\[ \vdash-E \]
Simplifying proofs

\[
\begin{array}{c}
\vdots \\
A & B \\
\hline
A \& B \\
A \\
\end{array}
\]

\&-I

\&-E_1
Simplifying a proof

\[
\frac{[B \& A]^z}{A \quad \&-E_2} \quad \frac{[B \& A]^z}{B \quad \&-E_1}
\]

\[
\frac{A \& B}{\therefore \quad \&-I}
\]

\[
\frac{(B \& A) \supset (A \& B)}{A \& B \quad \&-I^z}
\]

\[
\frac{B \quad A}{B \& A \quad \&-I}
\]

\[
\frac{B \& A}{A \& B \quad \therefore \&-E}
\]
Simplifying a proof

\[ \therefore B \quad \therefore A \]
\[ \frac{B \land A}{B \land A} \quad \&-I \]
\[ \frac{A}{A} \quad \&-E_2 \]
\[ \therefore B \quad \therefore A \]
\[ \frac{B \land A}{B} \quad \&-I \]
\[ \frac{A \land B}{B} \quad \&-E_1 \]
\[ \quad \&-I \]
Alonzo Church (1903–1995)
Alonzo Church (1932)  
Typed λ-calculus

\[ [x : A]^x \quad \vdash \quad \lambda x. N : A \supset B \]

\[
\begin{array}{c}
L : A \supset B \\
M : A
\end{array}
\quad \Gamma \text{-E}

\[
\begin{array}{c}
M : A \\
N : B
\end{array}
\quad \&-I

\[
\begin{array}{c}
L : A \& B \\
\pi_1 L : A \\
\pi_2 L : B
\end{array}
\quad \&-E_1 \quad \&-E_2
\]
A program

\[
\begin{align*}
&\frac{[z : B \& A]^z}{\pi_2 z : A} \quad \&-E_2 \quad \frac{[z : B \& A]^z}{\pi_1 z : B} \quad \&-E_1 \\
&\quad \frac{(\pi_2 z, \pi_1 z) : A \& B}{\&-I} \\
&\lambda z. (\pi_2 z, \pi_1 z) : (B \& A) \supset (A \& B) \quad \supset-I^z
\end{align*}
\]
Evaluating programs

\[
\begin{aligned}
\vdash \text{[} z : A \text{]} \vdash \\
\vdots \\
N : B \\
\lambda z. N : A \\ B \\
\vdash \text{[} \vdash \text{]} \\
M : A \\
(\lambda z. N) M : B \\
\end{aligned}
\]
Evaluating programs

\[ \begin{align*}
\vdots & \quad \vdots \\
M : A & \quad N : B \\
(M, N) : A \& B & \quad \text{&-I} \\
\pi_1(M, N) : A & \quad \text{&-E_1}
\end{align*} \]
AN EARLY PROOF OF NORMALIZATION
BY A.M. TURING

R.O. Gandy

Mathematical Institute, 24-25 St. Giles,
Oxford OX1 3LB, UK

Dedicated to H.B. Curry on the occasion of his 80th birthday

In the extract printed below, Turing shows that every
formula of Church's simple type theory has a normal form.
The extract is the first page of an unpublished (and incomplete)
typscript entitled 'Some theorems about Church's system'.
(Turing left his manuscripts to me; they are deposited in the
library of King's College, Cambridge). An account of this
system was published by Church in 'A formulation of the simple
theory of types' (J. Symbolic Logic 5 (1940), pp. 56-68).
Church had previously been told.
Evaluating a program

\[
\begin{array}{c}
\frac{z : B \& A}{\pi_2 z : A} & \&-E_2 & \frac{z : B \& A}{\pi_1 z : B} & \&-E_1 \\
\frac{\pi_2 z : A}{(\pi_2 z, \pi_1 z) : A \& B} & \&-I & \frac{\pi_1 z : B}{(\pi_2 z, \pi_1 z) : A \& B} & \&-I \\
\frac{\lambda z.(\pi_2 z, \pi_1 z) : (B \& A) \supset (A \& B)}{(\lambda z.(\pi_2 z, \pi_1 z))(N, M) : A \& B} & \supset-I^z & \frac{N : B}{(N, M) : B \& A} & \supset-I \\
\frac{M : A}{(N, M) : B \& A} & \supset-E
\end{array}
\]
Evaluating a program

\[
\begin{align*}
\vdots & \quad \vdots \\
N : B & \quad M : A \\
\therefore (N, M) : B \& A & \quad \&-E_2 \\
\pi_2(N, M) : A & \\
\pi_1(N, M) : B & \quad \&-I \\
(\pi_2(N, M), \pi_1(N, M)) : A \& B & \quad \&-I \\
\end{align*}
\]
The Curry-Howard homeomorphism
Haskell Curry (1900-1982)
William Howard (1926-)

[Image of Haskell Curry and William Howard]
William Howard (1980)

THE FORMULAE-AS-TYPES NOTION OF CONSTRUCTION

W. A. Howard

Department of Mathematics, University of Illinois at Chicago Circle, Chicago, Illinois 60680, U.S.A.

Dedicated to H. B. Curry on the occasion of his 80th birthday.

The following consists of notes which were privately circulated in 1969. Since they have been referred to a few times in the literature, it seems worth while to publish them. They have been rearranged for easier reading, and some inessential corrections have been made.
Curry-Howard correspondence

propositions as types
proofs as programs
normalisation of proofs as evaluation of programs
Curry-Howard correspondence

- Natural Deduction
  - Gentzen (1935)

- Typed Lambda Calculus
  - Church (1940)

- Type Schemes
  - Hindley (1969)

- ML Type System
  - Milner (1975)

- System F
  - Girard (1972)

- Polymorphic Lambda Calculus
  - Reynolds (1974)

- Modal Logic
  - Lewis (1910)

- Monads (state, exceptions)
  - Kleisli (1965), Moggi (1987)

- Classical-Intuitionistic Embedding
  - Gödel (1933)

- Continuation Passing Style
  - Reynolds (1972)

- Linear Logic
  - Girard (1987)

- Session Types
  - Honda (1993)
Functional Languages

- Lisp (McCarthy, 1960)
- Iswim (Landin, 1966)
- Scheme (Steele and Sussman, 1975)
- ML (Milner, Gordon, Wadsworth, 1979)
- Haskell (Hudak, Hughes, Peyton Jones, and Wadler, 1987)
- O’Caml (Leroy, 1996)
- Erlang (Armstrong, Virding, Williams, 1996)
- Scala (Odersky, 2004)
- F# (Syme, 2006)
- Clojure (Hickey, 2007)
- Elm (Czaplicki, 2012)
Proof Assistants

- Automath (de Bruijn, 1970)
- Type Theory (Martin Löf, 1975)
- Mizar (Trybulec, 1975)
- ML/LCF (Milner, Gordon, and Wadsworth, 1979)
- NuPrl (Constable, 1985)
- HOL (Gordon and Melham, 1988)
- Coq (Huet and Coquand, 1988)
- Isabelle (Paulson, 1993)
- Epigram (McBride and McKinna, 2004)
- Agda (Norell, 2005)
- Lean (de Moura, 2013)
Part III: Conclusion

Philosophy
Let’s talk to aliens!
Independence Day
A universal programming language?
Multiverses
Lambda is Omniversal