Propositions as Types

Philip Wadler
University of Edinburgh

Types and Semantics for Programming Languages
Part I

Computability
Euclid (325–265 BCE) / Al Khwarizmi (780–850)
Effective Computability

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

- **Kurt Gödel**: Recursive functions
  Stephen Kleene, General recursive functions of natural numbers
  *Bulletin the American Mathematical Society*, July 1935

- **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)
Kurt Gödel (1930) — Incompleteness

ON FORMALLY UNDECIDABLE PROPOSITIONS OF PRINCIPIA MATHEMATICA AND RELATED SYSTEMS I
(1931)

42. $Ax(x) \equiv Z\cdot Ax(x) \lor A\cdot Ax(x) \lor L_1\cdot Ax(x) \lor L_2\cdot Ax(x) \lor R\cdot Ax(x) \lor M\cdot Ax(x)$,
x is an axiom.

43. $Fl(x, y, z) \equiv y = z \land x \vdash (Ev[v \leq x \land Var(v) \land x = v \iff Gen y]$, 
x is an immediate consequence of y and z.

44. $Bw(x) \equiv (n)[0 < n \leq l(x) \rightarrow Ax(n \cdot Gl x) \lor (Ep, q)[0 < p, q < n \land 
\quad Fl(n \cdot Gl x, p \cdot Gl x, q \cdot Gl x)] \land l(x) > 0$,
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. $x B y \equiv Bw(x) \land [l(x)] \cdot Gl x = y$,
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.¹

By Alonzo Church.

The purpose of the present paper is to propose a definition of effective calculability² 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 → λab · a(b),
2 → λab · a(a(b)),
3 → λab · a(a(a(b))),

and so on, each positive integer in Arabic notation standing for a formula of the form λab · a(a(⋯ a(b) ⋯)).
Alonzo Church (1932) — $\lambda$-calculus

$L, M, N ::= x$

| $(\lambda x. N)$
| $(L M)$
Kurt Gödel (1906–1978)
General recursive functions of natural numbers\(^1\).

Von

S. C. Kleene in Madison (Wis., U.S.A.).

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

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

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

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 $\pi$, $e$, 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

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

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

\[ \vee-1 \]
\[ \frac{A}{A \vee B} \]

\[ \vee-E \]
\[ \frac{[A] \quad [B]}{C \quad C} \]

\[ \forall-I \]
\[ \frac{\exists x}{\forall x \, \exists x} \]

\[ \exists-E \]
\[ \frac{\exists x \, \exists x}{C} \]

\[ \Rightarrow-I \]
\[ \frac{[A]}{A \Rightarrow B} \]

\[ \Rightarrow-E \]
\[ \frac{A \Rightarrow B \quad B}{C} \]

\[ \land-1 \]
\[ \frac{A \quad B}{A \land B} \]

\[ \land-E \]
\[ \frac{A \quad \land B}{A} \]

\[ \neg-1 \]
\[ \frac{[A]}{\neg A} \]

\[ \neg-E \]
\[ \frac{A \quad \neg A}{\bot} \]
Gerhard Gentzen (1935) — Natural Deduction

\[
\begin{align*}
[A]^x \\
\vdots \\
B \\
\hline
A \supset B
\end{align*}
\]

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

\[
\begin{align*}
A & \quad B \\
\hline
A \& B
\end{align*}
\]

\[
\begin{align*}
A \& B \\
\hline
A
\end{align*}
\]

\[
\begin{align*}
A \& B \\
\hline
B
\end{align*}
\]
A proof

\[ [B \& A] \quad \text{\&-E}_1 \quad [B \& A] \quad \text{\&-E}_0 \]

\[
\frac{\begin{array}{l}
A \\
B
\end{array}}{A \& B \quad \text{\&-I}}
\]

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

\[
\begin{align*}
[A]^x & \quad \cdot \\
\cdot & \quad \cdot \\
B & \quad \supset \text{-I}^x \\
A \supset B & \quad \cdot \\
\hline
A & \quad \cdot \\
\text{\supset-E} & \quad \Rightarrow \\
B & \quad \cdot \\
\cdot & \quad \cdot \\
A & \quad B \\
\hline
A \& B & \quad \&\text{-I} \\
\hline
A \& B & \quad \&\text{-E}_0 \\
\hline
A & \quad \Rightarrow
\end{align*}
\]
Simplifying a proof

\[
\frac{[B \& A]^z \&-E_1}{A} \quad \frac{[B \& A]^z \&-E_0}{B} \\
\frac{\frac{A \& B \&-I}{A \& B}}{(B \& A) \supset (A \& B)} \subseteq-I^z \quad \frac{[B]^y \& [A]^x \&-I}{B \& A} \\
\frac{A \& B}{A \& B} \subseteq-E
\]
Simplifying a proof

\[
\frac{[B \& A]^z}{A} \quad \&-E_1
\]
\[
\frac{[B \& A]^z}{B} \quad \&-E_0
\]
\[
\frac{A \& B}{B \& A} \quad \&-I
\]
\[
\frac{(B \& A) \supset (A \& B)}{(B \& A) \supset (A \& B)} \quad \supset-I^z
\]
\[
\frac{A \& B}{B \& A} \quad \supset-E
\]
\[
\frac{[B]^y \quad [A]^x}{B \& A} \quad \&-I
\]
\[
\frac{B \& A}{A} \quad \&-E_1
\]
\[
\frac{B \& A}{B} \quad \&-E_0
\]
\[
\frac{A \& B}{A \& B} \quad \&-I
\]
Simplifying a proof

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

\[
\frac{[B \& A] z}{B} \quad \&-E_0
\]

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

\[
\frac{A \& B}{B \& A} \quad \supset-E
\]

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

\[
\frac{[B] y \quad [A] x}{B \& A} \quad \&-E_1
\]

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

\[
\frac{B \& A}{B} \quad \&-E_0
\]

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

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

29
Alonzo Church (1903–1995)
Alonzo Church (1940) — Typed $\lambda$-calculus

\[
\begin{array}{c}
\frac{[x : A]_{\cdot} \cdot N : B}{\lambda x. N : A \supset B} \quad \supset \text{-I}^x
\\
\frac{L : A \supset B \quad M : A}{LM : B} \quad \supset \text{-E}
\end{array}
\]

\[
\begin{array}{c}
\frac{M : A \quad N : B}{(M, N) : A \& B} \quad \& \text{-I}
\\
\frac{L : A \& B}{fst \; L : A} \quad \& \text{-E}_0
\\
\frac{L : A \& B}{snd \; L : B} \quad \& \text{-E}_1
\end{array}
\]
A program

\[
\begin{align*}
[ z : B & A ]^z & \quad \&-E_1 \\
\text{snd } z : A & \quad \&-E_0 \\
\text{fst } z : B & \quad \&-I \\
\frac{(\text{snd } z, \text{fst } z) : A & B}{\lambda z. (\text{snd } z, \text{fst } z) : (B & A) \supset (A & B)} \quad \supset-I^z
\end{align*}
\]
Evaluating programs

\[
\begin{align*}
\begin{array}{c}
\left[ x : A \right]^{x} \\
\vdots
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
N : B \\
\vdash \text{-I}^{x}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\lambda \cdot x. N : A \supset B \\
\vdash \text{-E} \quad \Rightarrow
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
M : A \\
\vdash \text{-E} \quad \Rightarrow
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
N \{ M / x \} : B
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
M : A \\
\vdash \text{-E} \quad \Rightarrow
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
N : B
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\vdash \text{-I}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
(M, N) : A \& B
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\vdash \text{-E}_0 \quad \Rightarrow
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{fst} (M, N) : A
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
M : A
\end{array}
\end{align*}
\]

33
AN EARLY PROOF OF NORMALIZATION
BY A.M. TURING

R.O. Gandy

Mathematical Institute, 24-29 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)
typescript 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
type theory of types' (J. Symbolic Logic 5 (1940), pp. 56-68).
Church had previously been working...
Evaluating a program

\[
\begin{align*}
[z : B \land A]^z & \quad \land \text{-}E_1 \\
n \text{snd } z & : A \\
& \quad \land \text{-}I \\
\text{(snd } z, \text{fst } z) & : A \land B \\
\lambda z. (\text{snd } z, \text{fst } z) & : (B \land A) \supset (A \land B) \\
& \quad \supset \text{-}I^z \\
\end{align*}
\]

\[
\begin{align*}
[z : B \land A]^z & \quad \land \text{-}E_0 \\
\text{fst } z & : B \\
& \quad \land \text{-}I \\
\lambda z. (\text{snd } z, \text{fst } z) & : (B \land A) \supset (A \land B) \\
& \quad \supset \text{-}I^z \\
\end{align*}
\]

\[
\begin{align*}
[y : B]_y & \quad [x : A]_x \\
\text{(y, x)} & : B \land A \\
& \quad \land \text{-}I \\
& \quad \supset \text{-}E \\
\end{align*}
\]

\[
\begin{align*}
(\lambda z. (\text{snd } z, \text{fst } z)) (y, x) & : A \land B
\end{align*}
\]
Evaluating a program

\[ [z : B & A] \quad \&-E_1 \quad [z : B & A] \quad \&-E_0 \]

\[ \text{snd } z : A \quad \text{fst } z : B \quad \&-\text{I} \]

\[ (\text{snd } z, \text{fst } z) : A & B \quad \&-\text{I} \]

\[ \lambda z. (\text{snd } z, \text{fst } z) : (B & A) \supset (A & B) \quad \supset-\text{I}^z \]

\[ (\lambda z. (\text{snd } z, \text{fst } z))(y, x) : A & B \quad \supset-\text{E} \]

\[ \downarrow \]

\[ [y : B]^y \quad [x : A]^x \quad \&-\text{I} \]

\[ (y, x) : B & A \quad \&-\text{I} \]

\[ \text{snd} (y, x) : A \quad \&-E_1 \]

\[ \text{fst} (y, x) : B \quad \&-E_0 \]

\[ (\text{snd} (y, x), \text{fst} (y, x)) : A & B \quad \&-\text{I} \]
Evaluating a program

\[
\begin{align*}
[z : B \& A]^z &\quad \&-E_1 \\
\text{snd } z : A &\quad \text{fst } z : B \\
\hline
\text{(snd } z, \text{ fst } z) : A \& B &\quad \&-I
\end{align*}
\]

\[
\lambda z. (\text{snd } z, \text{ fst } z) : (B \& A) \supset (A \& B)
\]

\[
\begin{align*}
\text{(-I}^z &\quad [y : B]^y \\
\text{[x : A]^x} &\quad (y, x) : B \& A
\end{align*}
\]

\[
\downarrow
\]

\[
\begin{align*}
[y : B]^y \\
\text{x : A}^x &\quad \&-I
\end{align*}
\]

\[
\begin{align*}
\text{snd } (y, x) : A &\quad \text{fst } (y, x) : B \\
\hline
\text{(snd } y, x, \text{ fst } y, x) : A \& B &\quad \&-I
\end{align*}
\]

\[
\begin{align*}
[x : A]^x \\
y : B^y &\quad \&-I
\end{align*}
\]

\[
\begin{align*}
\text{(x, y) : A } \& B
\end{align*}
\]
The Curry-Howard homeomorphism
Haskell Curry (1900–1982) / William Howard (1926–)
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 worthwhile to publish them. They have been rearranged for easier reading, and some inessential corrections have been made.
Curry-Howard correspondence

propositions \textit{as} types

proofs \textit{as} programs

normalisation of proofs \textit{as} evaluation of programs
## Curry-Howard correspondence

<table>
<thead>
<tr>
<th>Natural Deduction</th>
<th>Typed Lambda Calculus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gentzen (1935)</td>
<td>Church (1940)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type Schemes</th>
<th>ML Type System</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>System F</th>
<th>Polymorphic Lambda Calculus</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Modal Logic</th>
<th>Monads (state, exceptions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lewis (1910)</td>
<td>Kleisli (1965), Moggi (1987)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classical-Intuitionistic Embedding</th>
<th>Continuation Passing Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gödel (1933)</td>
<td>Reynolds (1972)</td>
</tr>
</tbody>
</table>
Functional Languages

- **Lisp** (McCarthy, 1960)
- **Iswim** (Landin, 1966)
- **Scheme** (Steele and Sussman, 1975)
- **ML** (Milner, Gordon, Wadsworth, 1979)
- **Haskell** (Hudak, Peyton Jones, and Wadler, 1987)
- **O’Caml** (Leroy, 1996)
- **Erlang** (Armstrong, Virding, Williams, 1996)
- **Scala** (Odersky, 2004)
- **F#** (Syme, 2006)
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)
Part III

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