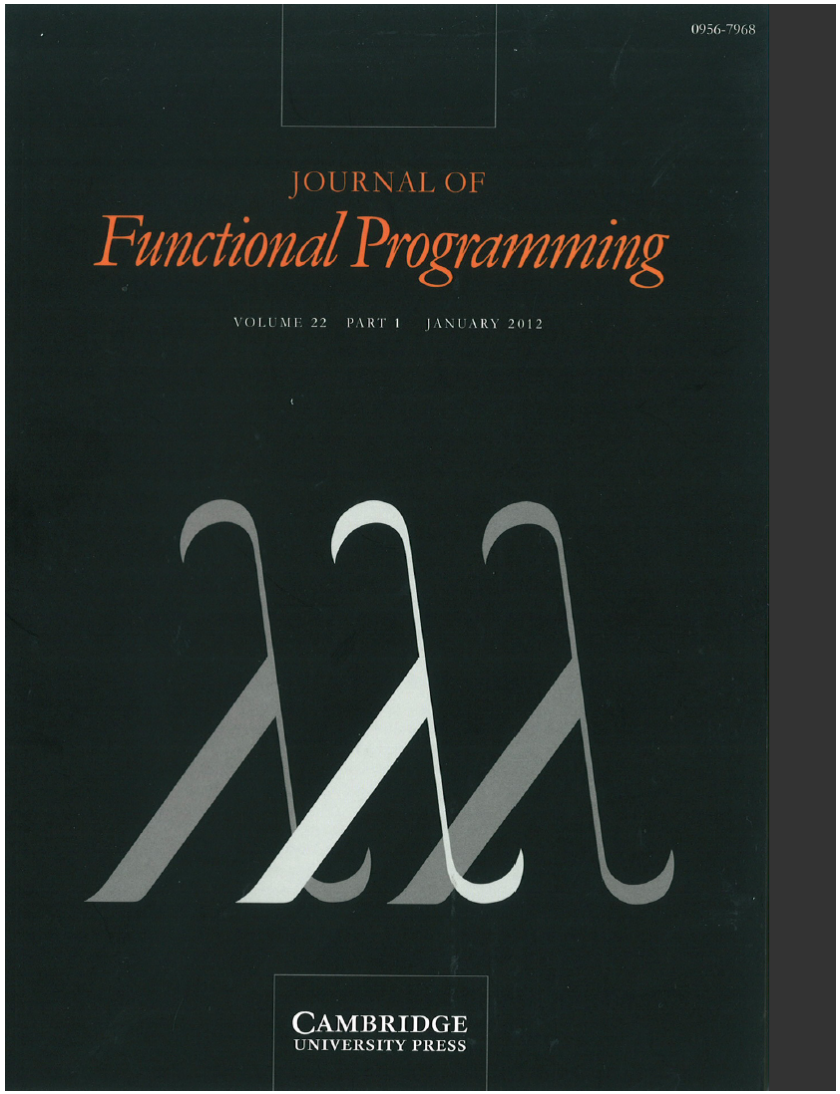
Church's Coincidences

Philip Wadler
University of Edinburgh

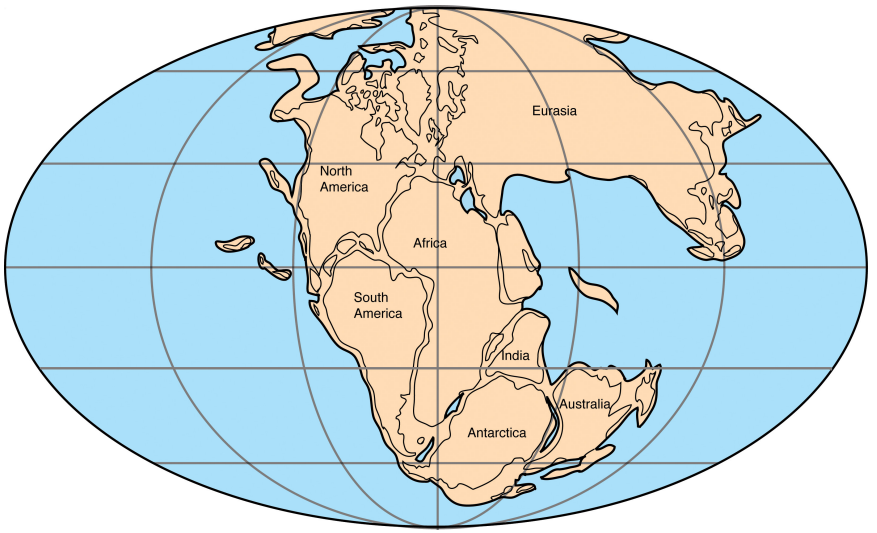
Midlands Graduate School, Leicester
8 April 2013

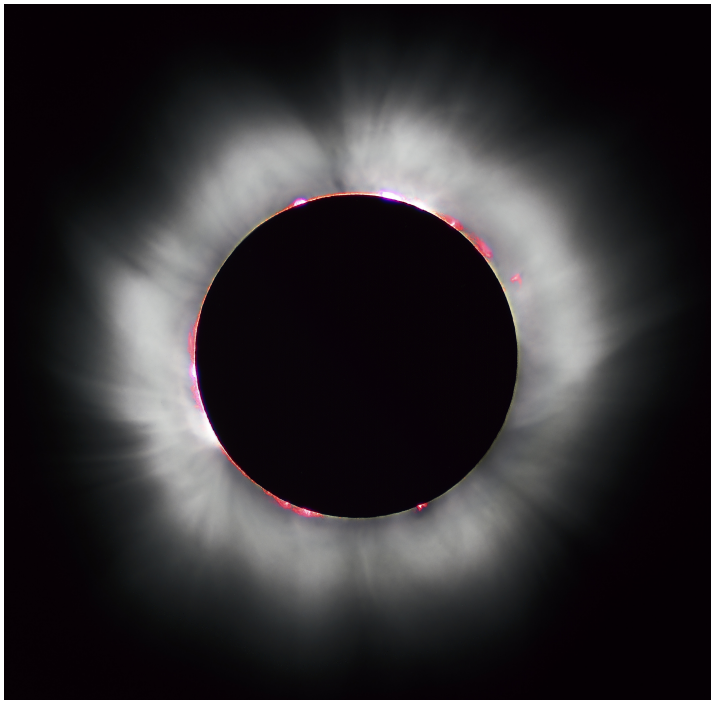

0956-7968
JOURNAL OF
Functional Programming
VOLUME 22 PART 1 JANUARY 2012
CAMBRIDGE
UNIVERSITY PRESS

Part I

About Coincidences

Two Kinds of Coincidence





Part II

The First Coincidence: Effective Computability

Effective Computability

- Alonzo Church: Lambda calculus

  An unsolvable problem of elementary number theory

  (Abstract) *Bulletin the American Mathematical Society*, May 1935

- Stephen C. Kleene: Recursive functions

  General recursive functions of natural numbers

  (Abstract) *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

David Hilbert (1930) — An Address

Königsberg, 8 September 1930

Society of German Scientists and Physicians

"We must know! We will know!"

Kurt Gödel (1906–1978)

Kurt Gödel (1930) — Incompleteness

Königsberg, 7 September 1930

Society of German Scientists and Physicians

ON FORMALLY UNDECIDABLE PROPOSITIONS OF *PRINCIPIA MATHEMATICA* AND RELATED SYSTEMS I[1]

(1931)

42. $Ax(x) \equiv Z\text{-}Ax(x) \vee A\text{-}Ax(x) \vee L_1\text{-}Ax(x) \vee L_2\text{-}Ax(x) \vee R\text{-}Ax(x) \vee M\text{-}Ax(x)$,
x is an AXIOM.

43. $Fl(x, y, z) \equiv y = z \operatorname{Imp} x \vee (Ev)[v \leqq x \,\&\, \operatorname{Var}(v) \,\&\, x = v \operatorname{Gen} y]$,
x is an IMMEDIATE CONSEQUENCE of y and z.

44. $Bw(x) \equiv (n)\{0 < n \leqq l(x) \to Ax(n\ Gl\ x) \vee (Ep, q)[0 < p, q < n \,\&\, Fl(n\ Gl\ x, p\ Gl\ x, q\ Gl\ x)]\} \,\&\, l(x) > 0$,
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. $x\ B\ y \equiv Bw(x) \,\&\, [l(x)]\ Gl\ x = y$,
x is a PROOF of the FORMULA y.

46. $\operatorname{Bew}(x) \equiv (Ey)y\ B\ x$,
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.)

Alonzo Church (1903–1995)

Alonzo Church (1932) — λ-calculus

$$f(x) = x^2 + x + 42$$

$$\Downarrow$$

$$f = \lambda x.\, x^2 + x + 42$$

$$\forall x.A = \forall\, (\lambda x.A)$$

Alonzo Church (1932) — λ-calculus

Then	Now
x	x
$\lambda x[N]$	$(\lambda x.N)$
$\{L\}(M)$	$(L\,M)$

Alonzo Church (1932) — Lambda Calculus

A SET OF POSTULATES FOR THE FOUNDATION OF LOGIC.[1]

By Alonzo Church.[2]

. . .

application must be held irrelevant. There may, indeed, be other applications of the system than its use as a logic.

. . .

An occurrence of a variable **x** in a given formula is called an occurrence of **x** as a *bound variable* in the given formula if it is an occurrence of **x** in a part of the formula of the form λ**x**[**M**]; that is, if there is a formula **M** such that λ**x**[**M**] occurs in the given formula and the occurrence of **x** in question is an occurrence in λ**x**[**M**]. All other occurrences of a variable in a formula are called occurrences as a *free variable*.

A formula is said to be *well-formed* if it is a variable, or if it is one of the symbols Π, Σ, &, $\sim$, ι, A, or if it is obtainable from these symbols by repeated combinations of them of one of the forms {**M**} (**N**) and λ**x**[**M**], where **x** is any variable and **M** and **N** are symbols or formulas which are being combined. This is a definition by induction. It implies the following rules: (1) a variable is well-formed (2) Π, Σ, &, $\sim$, ι, and A are well-formed (3) if **M** and **N** are well-formed then {**M**} (**N**) is well-formed (4) if **x** is a variable and **M** is well-formed then λ**x**[**M**] is well-formed.

Alonzo Church (1936) — Undecidability

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

The purpose of the present paper is to propose a definition of effective calculability [3] 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.

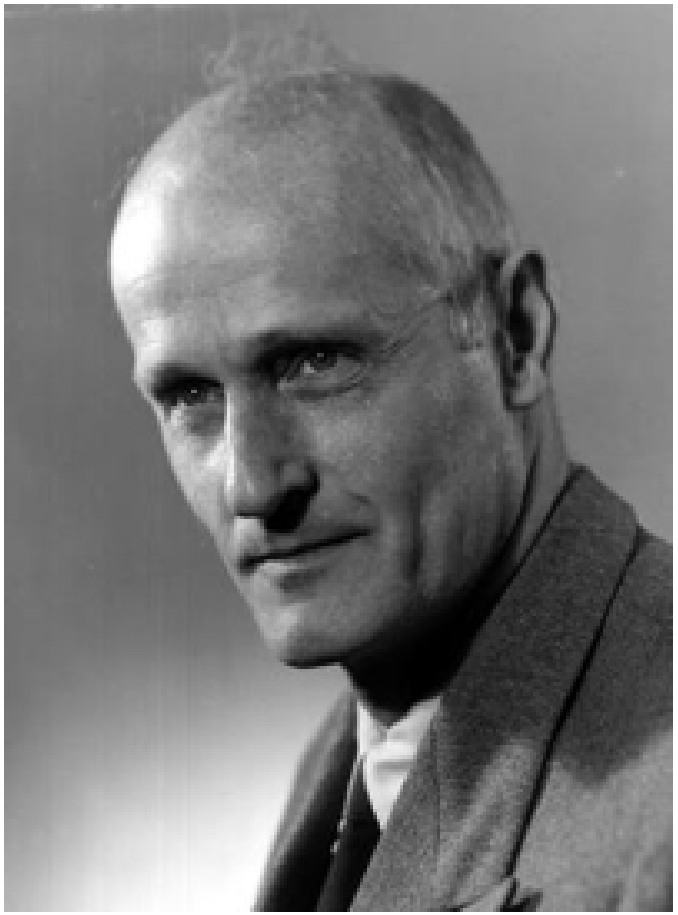
. . .

We introduce at once the following infinite list of abbreviations,

$$1 \rightarrow \lambda ab \cdot a(b),$$
$$2 \rightarrow \lambda ab \cdot a(a(b)),$$
$$3 \rightarrow \lambda ab \cdot a(a(a(b))),$$

and so on, each positive integer in Arabic notation standing for a formula of the form $\lambda ab \cdot a(a(\cdot \cdot \cdot a(b) \cdot \cdot \cdot))$.

Stephen Kleene (1909–1994)



Stephen Kleene (1932) — Predecessor

In 1932, soon after returning to Church's identification, one day late in January or early in February while in a dentists office, it came to me that I could λ-define the predecessor function by

. . .

And a λ-formula G is easily constructed to perform the following operation on any $\underline{n}$-tuple:

$$(\underline{n}_1, \underline{n}_2, \underline{n}_3)$$

$$(\underline{n}_2, \underline{n}_3, \underline{S}(\underline{n}_3)).$$

So if A is (1,1,1), then $\lambda\underline{n}.\underline{n}(G,A)$ λ-defines the sequence of number-triples

(3) (1,1,2), (1,2,3), (2,3,4), (3,4,5), ...

It is then easy by a λ-formula H to erase all but the first number of each triple so as to obtain

(4) 1, 1, 2, 3, ... ,

which is the sequence of values of the predecessor

. . .

abbreviate it "$\underline{P}$". When I brought this result to Church, he told me that he had just about convinced himself that there is no λ-definition of the predecessor function.

Stephen Kleene (1936) — Recursive Functions

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

(2) $$\varphi(0, x_2, \ldots, x_n) = \psi(x_2, \ldots, x_n)$$
$$\varphi(y+1, x_2, \ldots, x_n) = \chi(y, \varphi(y, x_2, \ldots, x_n), x_2, \ldots, x_n),$$

where $\theta, \chi_1, \ldots, \chi_m, \psi, \chi$ are given functions of natural numbers, are examples of the definition of a function φ 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)

Alan Turing (1936) — Turing Machine

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.

3. *Examples of computing machines.*

I. A machine can be constructed to compute the sequence 010101 .... The machine is to have the four m-configurations "$\mathfrak{b}$", "$\mathfrak{c}$", "$\mathfrak{e}$", "$\mathfrak{f}$" and is capable of printing "0" and "1". The behaviour of the machine is described in the following table in which "R" means "the machine moves so that it scans the square immediately on the right of the one it was scanning previously". Similarly for "L". "E" means "the scanned symbol is erased" and "P" stands for "prints". This table (and all succeeding tables of the same kind) is to be understood to mean that for a configuration described in the first two columns the operations in the third column are carried out successively, and the machine then goes over into the m-configuration described in the last column. When the second column is left blank, it is understood that the behaviour of the third and fourth columns applies for any symbol and for no symbol. The machine starts in the m-configuration $\mathfrak{b}$ with a blank tape.

Configuration		*Behaviour*		
m-config.	*symbol*	*operations*	*final*	*m-config.*
$\mathfrak{b}$	None	$P0$, R		$\mathfrak{c}$
$\mathfrak{c}$	None	R		$\mathfrak{c}$
$\mathfrak{e}$	None	$P1$, R		$\mathfrak{f}$
$\mathfrak{f}$	None	R		$\mathfrak{b}$

Robin Gandy (1988) — Turing's Theorem

(2) By considering the limitations of our sensory and mental apparatus, Turing arrives at the following restrictions on the actions of the computor.[24]

(i) There is a fixed upper bound to the number of distinct symbols which can be written in a cell.

(ii) There is a fixed upper bound to the number of contiguous cells whose contents the computor can take in ('at a glance', one might say) when he is deciding, at a given stage, what to do. Turing shows by an example that for a normal human being—the reader—this bound, for a linear arrangement, is less than 15.

(iii) At each step the computor may alter the contents of only one cell, and there is a fixed upper bound to the distance the computor can move to get to this cell from the scanned cells; so we may as well suppose it is one of them.

(iv) There is a fixed upper bound to the distance through which the scan can be moved between steps. (Turing's argument is rather indirect). Moving the scan is part of the action.

(v) There is a fixed upper bound to the number of 'states of mind' of the computor; his state of mind, together with the contents of the scanned cells, determine the action he takes and his next state of mind. In place of a 'state of mind' Turing admits that the computor might leave an instruction as to how to continue (p. 253). Thus the computor must follow a fixed, finite set of instructions.

Theorem: *Any function which can be calculated by a human being can be computed by a Turing Machine.*

Alan Turing (1937) — Equivalence

COMPUTABILITY AND λ-DEFINABILITY

A. M. TURING

. . .

The identification of 'effectively calculable' functions with computable functions is possibly more convincing than an identification with the λ-definable or general recursive functions. For those who take this view the formal proof of equivalence provides a justification for Church's calculus, and allows the 'machines' which generate computable functions to be replaced by the more convenient λ-definitions.

Alan Turing (1946) — Automatic Computing Engine

PROPOSED ELECTRONIC CALCULATOR.

PART I.

Descriptive Account.

1. Introductory.

Calculating machinery in the past has been designed to carry out accurately and moderately quickly small parts of calculations which frequently recur. The four processes addition, subtraction, multiplication and division, together perhaps with sorting and interpolation, cover all that could be done until quite recently, if we except machines of the nature of the differential analyser and wind tunnels, etc. which operate by measurement rather than by calculation.

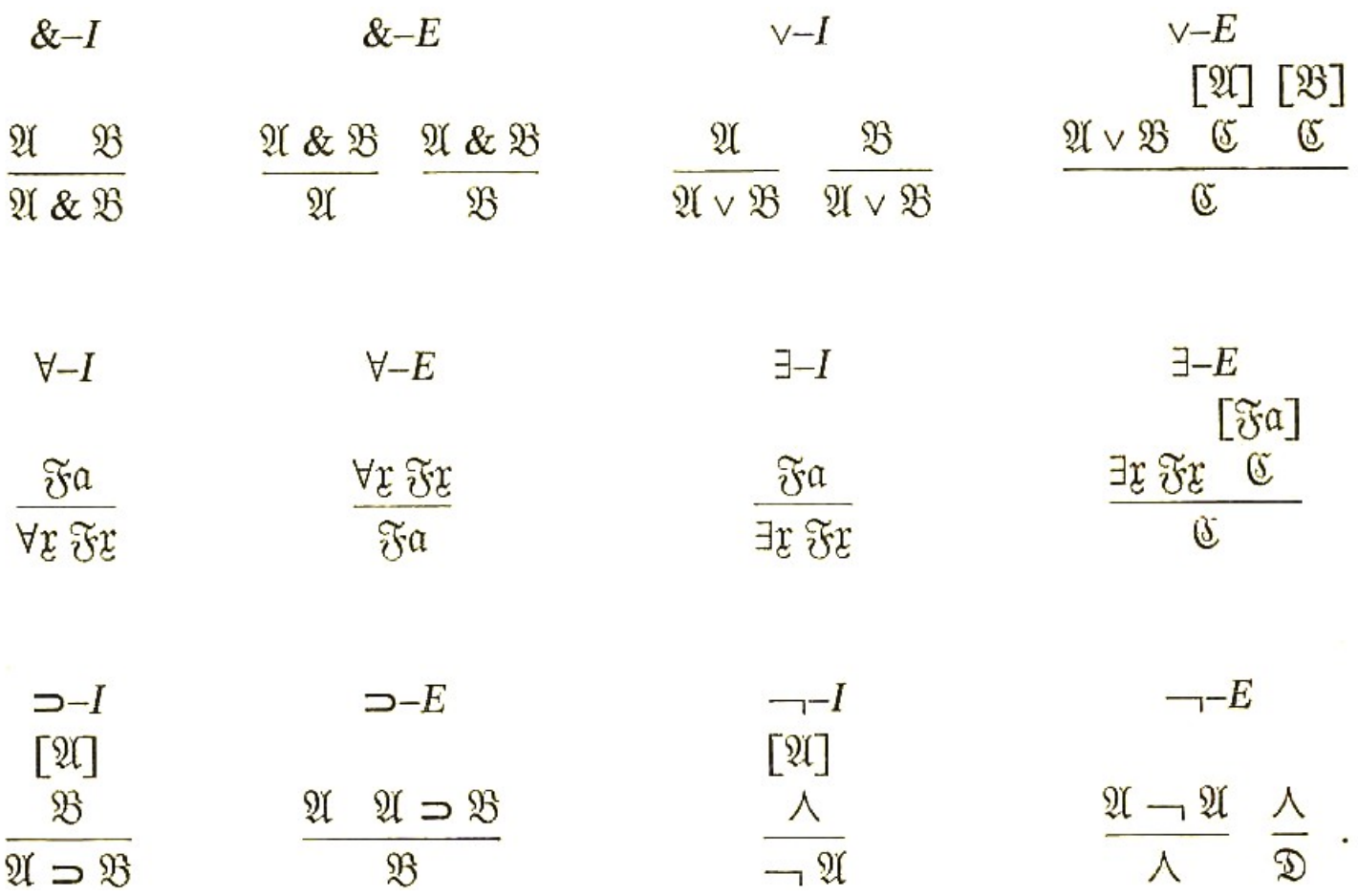
"Instruction tables will have to be made up by mathematicians with computing experience and perhaps a certain puzzle-solving ability. There need be no real danger of it ever becoming a drudge, for any processes that are quite mechanical may be turned over to the machine itself."

Part III

The Second Coincidence: Propositions as Types

Gerhard Gentzen (1909–1945)

Gerhard Gentzen (1935) — Natural Deduction



Gerhard Gentzen (1935) — Natural Deduction

$$\dfrac{\begin{array}{c}[A]^x\\ \vdots\\ B\end{array}}{A \supset B}\ \supset\text{-I}^x \qquad \dfrac{A \supset B \qquad A}{B}\ \supset\text{-E}$$

$$\dfrac{A \qquad B}{A \,\&\, B}\ \&\text{-I} \qquad \dfrac{A \,\&\, B}{A}\ \&\text{-E}_0 \qquad \dfrac{A \,\&\, B}{B}\ \&\text{-E}_1$$

Gerhard Gentzen (1935) — Natural Deduction

$$\dfrac{\dfrac{\begin{array}{c}[A]^x\\ \vdots\\ B\end{array}}{A \supset B}\,\supset\text{-I}^x \qquad \begin{array}{c}\vdots\\ A\end{array}}{B}\,\supset\text{-E} \quad \Rightarrow \quad \begin{array}{c}\vdots\\ A\\ \vdots\\ B\end{array}$$

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ A\end{array} \qquad \begin{array}{c}\vdots\\ B\end{array}}{A \,\&\, B}\,\&\text{-I}}{A}\,\&\text{-E}_0 \quad \Rightarrow \quad \begin{array}{c}\vdots\\ A\end{array}$$

A proof

$$\dfrac{\dfrac{\dfrac{[B \mathbin{\&} A]^z}{A}\,\&\text{-E}_1 \qquad \dfrac{[B \mathbin{\&} A]^z}{B}\,\&\text{-E}_0}{A \mathbin{\&} B}\,\&\text{-I}}{(B \mathbin{\&} A) \supset (A \mathbin{\&} B)}\,\supset\text{-I}^z$$

Simplifying a proof

$$\dfrac{\dfrac{\dfrac{\dfrac{[B \mathbin{\&} A]^z}{A}\,\&\text{-E}_1 \qquad \dfrac{[B \mathbin{\&} A]^z}{B}\,\&\text{-E}_0}{A \mathbin{\&} B}\,\&\text{-I}}{(B \mathbin{\&} A) \supset (A \mathbin{\&} B)}\,\supset\text{-I}^z \qquad \dfrac{[B]^y \quad [A]^x}{B \mathbin{\&} A}\,\&\text{-I}}{A \mathbin{\&} B}\,\supset\text{-E}$$

Simplifying a proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[B \,\&\, A]^z}{A}\,\&\text{-E}_1 \qquad \dfrac{[B \,\&\, A]^z}{B}\,\&\text{-E}_0}{A \,\&\, B}\,\&\text{-I}}{(B \,\&\, A) \supset (A \,\&\, B)}\,\supset\text{-I}^z \qquad \dfrac{[B]^y \quad [A]^x}{B \,\&\, A}\,\&\text{-I}}{A \,\&\, B}\,\supset\text{-E}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\dfrac{[B]^y \quad [A]^x}{B \,\&\, A}\,\&\text{-I}}{A}\,\&\text{-E}_1 \qquad \dfrac{\dfrac{[B]^y \quad [A]^x}{B \,\&\, A}\,\&\text{-I}}{B}\,\&\text{-E}_0}{A \,\&\, B}\,\&\text{-I}$$

Simplifying a proof

$$\dfrac{\dfrac{\dfrac{\dfrac{[B \mathbin{\&} A]^z}{A}\,\&\text{-E}_1 \qquad \dfrac{[B \mathbin{\&} A]^z}{B}\,\&\text{-E}_0}{A \mathbin{\&} B}\,\&\text{-I}}{(B \mathbin{\&} A) \supset (A \mathbin{\&} B)}\,\supset\text{-I}^z \qquad \dfrac{[B]^y \quad [A]^x}{B \mathbin{\&} A}\,\&\text{-I}}{A \mathbin{\&} B}\,\supset\text{-E}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\dfrac{[B]^y \quad [A]^x}{B \mathbin{\&} A}\,\&\text{-I}}{A}\,\&\text{-E}_1 \qquad \dfrac{\dfrac{[B]^y \quad [A]^x}{B \mathbin{\&} A}\,\&\text{-I}}{B}\,\&\text{-E}_0}{A \mathbin{\&} B}\,\&\text{-I}$$

$$\Downarrow$$

$$\dfrac{[A]^x \quad [B]^y}{A \mathbin{\&} B}\,\&\text{-I}$$

Alonzo Church (1903–1995)

Alonzo Church (1940) — Typed λ-calculus

$$\frac{\begin{array}{c}[x : A]^x \\ \vdots \\ N : B\end{array}}{\lambda x.\, N : A \supset B}\ \supset\text{-I}^x \qquad \frac{L : A \supset B \qquad M : A}{L\, M : B}\ \supset\text{-E}$$

$$\frac{M : A \qquad N : B}{\langle M, N\rangle : A \,\&\, B}\ \&\text{-I} \qquad \frac{L : A \,\&\, B}{L_0 : A}\ \&\text{-E}_0 \qquad \frac{L : A \,\&\, B}{L_1 : B}\ \&\text{-E}_1$$

Alonzo Church (1940) — Typed λ-calculus

$$\dfrac{\dfrac{\begin{array}{c}[x : A]^x \\ \vdots \\ N : B\end{array}}{\lambda x.\, N : A \supset B}\supset\text{-I}^x \qquad \begin{array}{c}\vdots \\ M : A\end{array}}{(\lambda x.\, N)\, M : B}\supset\text{-E} \quad \Rightarrow \quad \begin{array}{c}\vdots \\ M : A \\ \vdots \\ N\{M/x\} : B\end{array}$$

$$\dfrac{\dfrac{\begin{array}{c}\vdots \\ M : A\end{array} \qquad \begin{array}{c}\vdots \\ N : B\end{array}}{\langle M, N\rangle : A \,\&\, B}\,\&\text{-I}}{\langle M, N\rangle_0 : A}\,\&\text{-E}_0 \quad \Rightarrow \quad \begin{array}{c}\vdots \\ M : A\end{array}$$

Alan Turing (1942)

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 theory of types' (J. Symbolic Logic 5 (1940), pp. 56-68).

A program

$$\dfrac{\dfrac{\dfrac{[z : B \mathbin{\&} A]^z}{z_1 : A}\,\&\text{-E}_1 \qquad \dfrac{[z : B \mathbin{\&} A]^z}{z_0 : B}\,\&\text{-E}_0}{\langle z_1, z_0 \rangle : A \mathbin{\&} B}\,\&\text{-I}}{\lambda z.\, \langle z_1, z_0 \rangle : (B \mathbin{\&} A) \supset (A \mathbin{\&} B)}\,\supset\text{-I}^z$$

Evaluating a program

$$\dfrac{\dfrac{\dfrac{\dfrac{[z : B \mathbin{\&} A]^z}{z_1 : A}\,\&\text{-E}_1 \qquad \dfrac{[z : B \mathbin{\&} A]^z}{z_0 : B}\,\&\text{-E}_0}{\langle z_1, z_0\rangle : A \mathbin{\&} B}\,\&\text{-I}}{\lambda z.\,\langle z_1, z_0\rangle : (B \mathbin{\&} A) \supset (A \mathbin{\&} B)}\,\supset\text{-I}^z \qquad \dfrac{[y : B]^y \quad [x : A]^x}{\langle y, x\rangle : B \mathbin{\&} A}\,\&\text{-I}}{(\lambda z.\,\langle z_1, z_0\rangle)\,\langle y, x\rangle : A \mathbin{\&} B}\,\supset\text{-E}$$

Evaluating a program

$$\dfrac{\dfrac{\dfrac{\dfrac{[z : B \mathbin{\&} A]^z}{z_1 : A}\,\&\text{-E}_1 \qquad \dfrac{[z : B \mathbin{\&} A]^z}{z_0 : B}\,\&\text{-E}_0}{\langle z_1, z_0\rangle : A \mathbin{\&} B}\,\&\text{-I}}{\lambda z.\,\langle z_1, z_0\rangle : (B \mathbin{\&} A) \supset (A \mathbin{\&} B)}\,\supset\text{-I}^z \qquad \dfrac{[y : B]^y \quad [x : A]^x}{\langle y, x\rangle : B \mathbin{\&} A}\,\&\text{-I}}{(\lambda z.\,\langle z_1, z_0\rangle)\,\langle y, x\rangle : A \mathbin{\&} B}\,\supset\text{-E}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\dfrac{[y : B]^y \quad [x : A]^x}{\langle y, x\rangle : B \mathbin{\&} A}\,\&\text{-I}}{\langle y, x\rangle_1 : A}\,\&\text{-E}_1 \qquad \dfrac{\dfrac{[y : B]^y \quad [x : A]^x}{\langle y, x\rangle : B \mathbin{\&} A}\,\&\text{-I}}{\langle y, x\rangle_0 : B}\,\&\text{-E}_0}{\langle \langle y, x\rangle_1, \langle y, x\rangle_0\rangle : A \mathbin{\&} B}\,\&\text{-I}$$

Evaluating a program

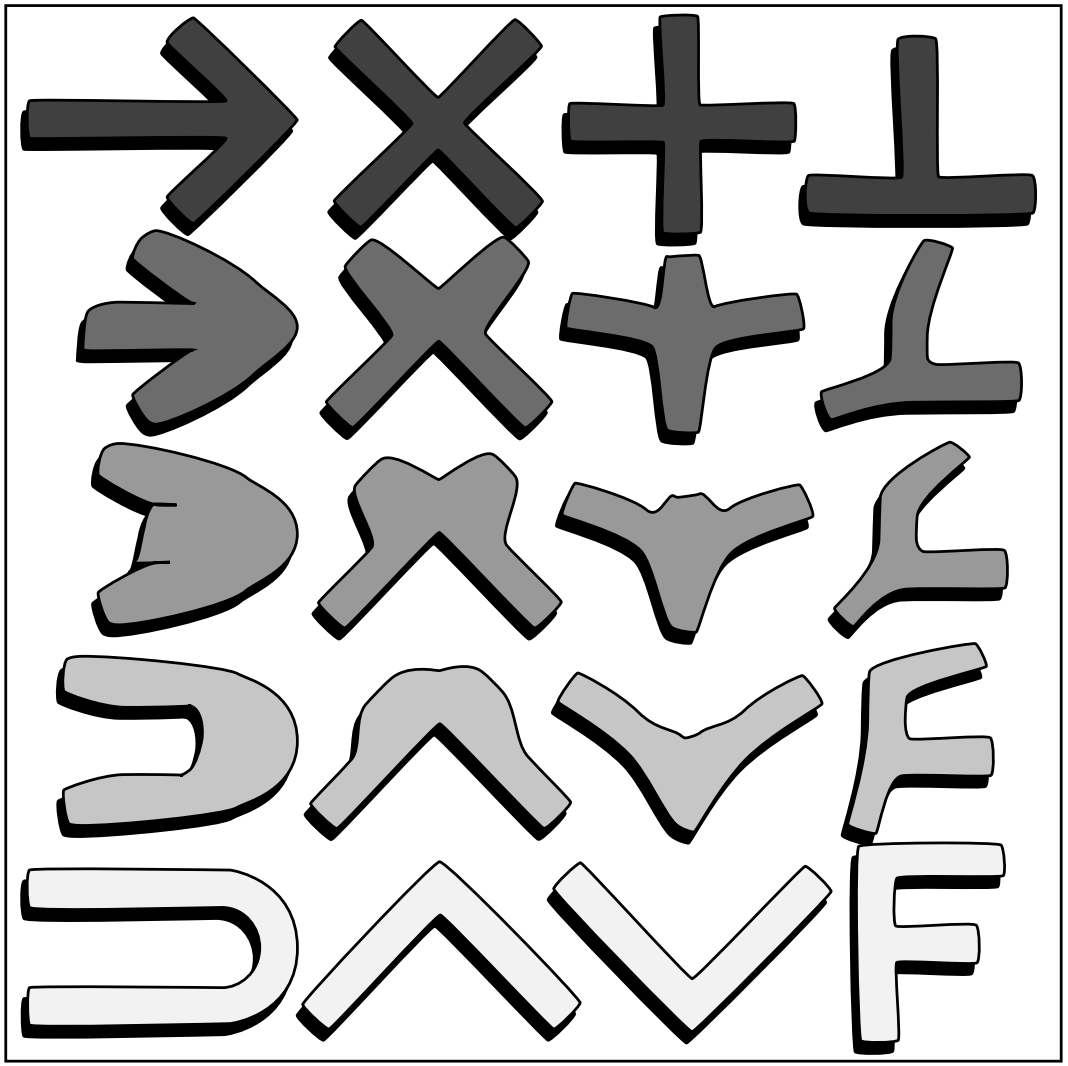
$$\dfrac{\dfrac{\dfrac{\dfrac{[z : B \,\&\, A]^z}{z_1 : A}\,\&\text{-E}_1 \qquad \dfrac{[z : B \,\&\, A]^z}{z_0 : B}\,\&\text{-E}_0}{\langle z_1, z_0\rangle : A \,\&\, B}\,\&\text{-I}}{\lambda z.\,\langle z_1, z_0\rangle : (B \,\&\, A) \supset (A \,\&\, B)}\,\supset\text{-I}^z \qquad \dfrac{[y : B]^y \quad [x : A]^x}{\langle y, x\rangle : B \,\&\, A}\,\&\text{-I}}{(\lambda z.\,\langle z_1, z_0\rangle)\,\langle y, x\rangle : A \,\&\, B}\,\supset\text{-E}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\dfrac{[y : B]^y \quad [x : A]^x}{\langle y, x\rangle : B \,\&\, A}\,\&\text{-I}}{\langle y, x\rangle_1 : A}\,\&\text{-E}_1 \qquad \dfrac{\dfrac{[y : B]^y \quad [x : A]^x}{\langle y, x\rangle : B \,\&\, A}\,\&\text{-I}}{\langle y, x\rangle_0 : B}\,\&\text{-E}_0}{\langle\langle y, x\rangle_1, \langle y, x\rangle_0\rangle : A \,\&\, B}\,\&\text{-I}$$

$$\Downarrow$$

$$\dfrac{[x : A]^x \quad [y : B]^y}{\langle x, y\rangle : A \,\&\, B}\,\&\text{-I}$$



Haskell Curry (1900–1982) / William Howard (1926–)

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

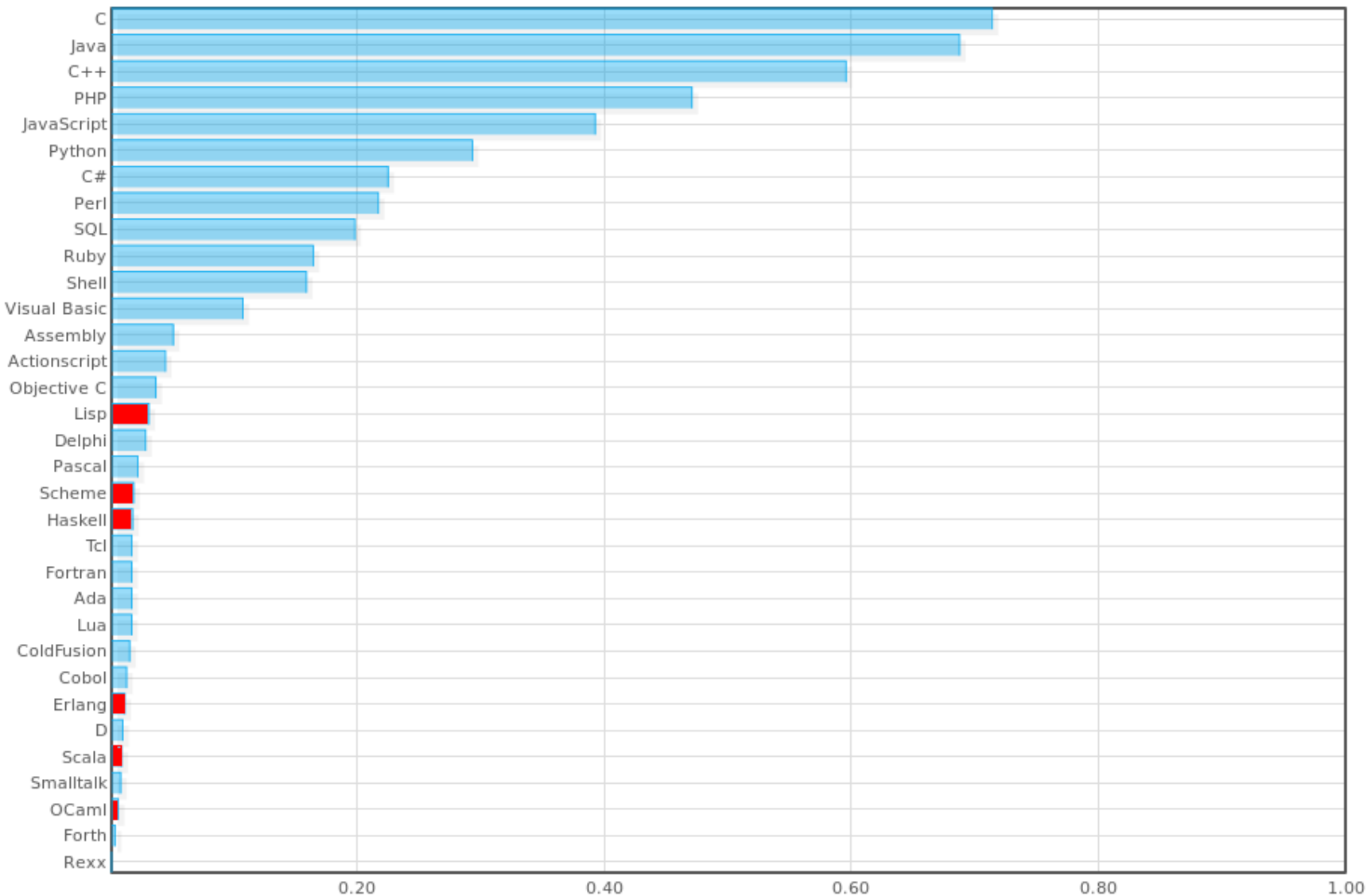
Curry-Howard correspondence

Natural Deduction	$\leftrightarrow$	Typed Lambda Calculus
Gentzen (1935)		Church (1940)
Type Schemes	$\leftrightarrow$	ML Type System
Hindley (1969)		Milner (1975)
System F	$\leftrightarrow$	Polymorphic Lambda Calculus
Girard (1972)		Reynolds (1974)
Modal Logic	$\leftrightarrow$	Monads (state, exceptions)
Lewis (1910)		Kleisli (1965), Moggi (1987)
Classical-Intuitionistic Embedding	$\leftrightarrow$	Continuation Passing Style
Gödel (1933)		Reynolds (1972)

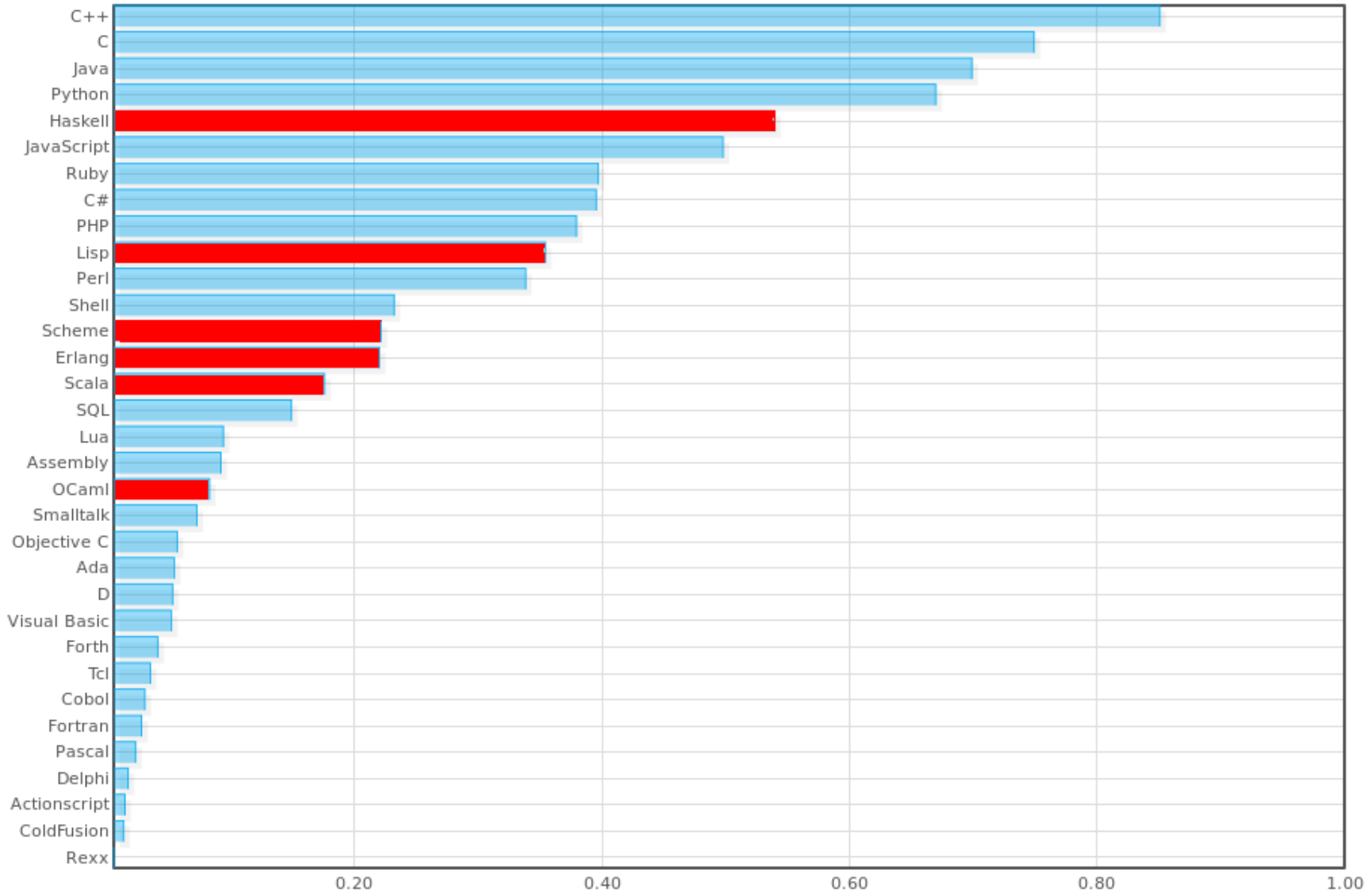
Functional Languages

- 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)

Language Popularity Index



Language Popularity Index—Discussion



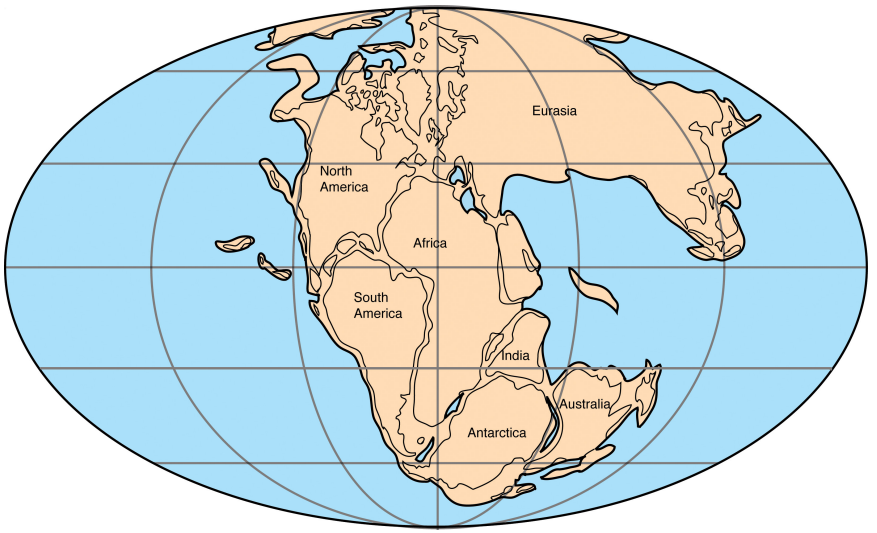
Proof systems

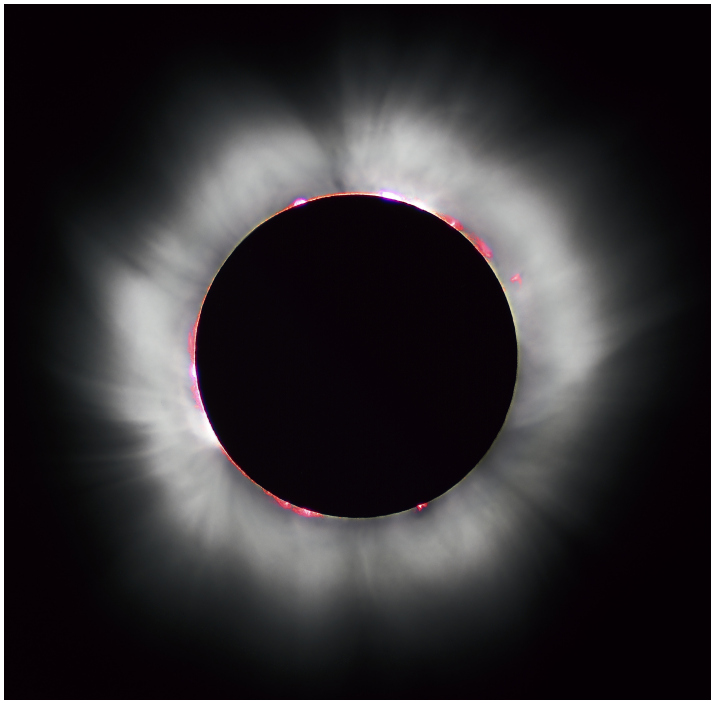
- Automath (de Bruijn, 1970)
- Type Theory (Martin Löf, 1975)
- Mizar (Trybulec, 1975)
- ML/LCF (Milner, Gordon, and Wadsworth, 1979)
- NuPrl (Constable, 1985)
- HOL (Gordon and Melham, 1988)
- Coq (Huet and Coquand, 1988)
- Isabelle (Paulson, 1993)
- Epigram (McBride and McKinna, 2004)
- Agda (Norell,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

Part IV

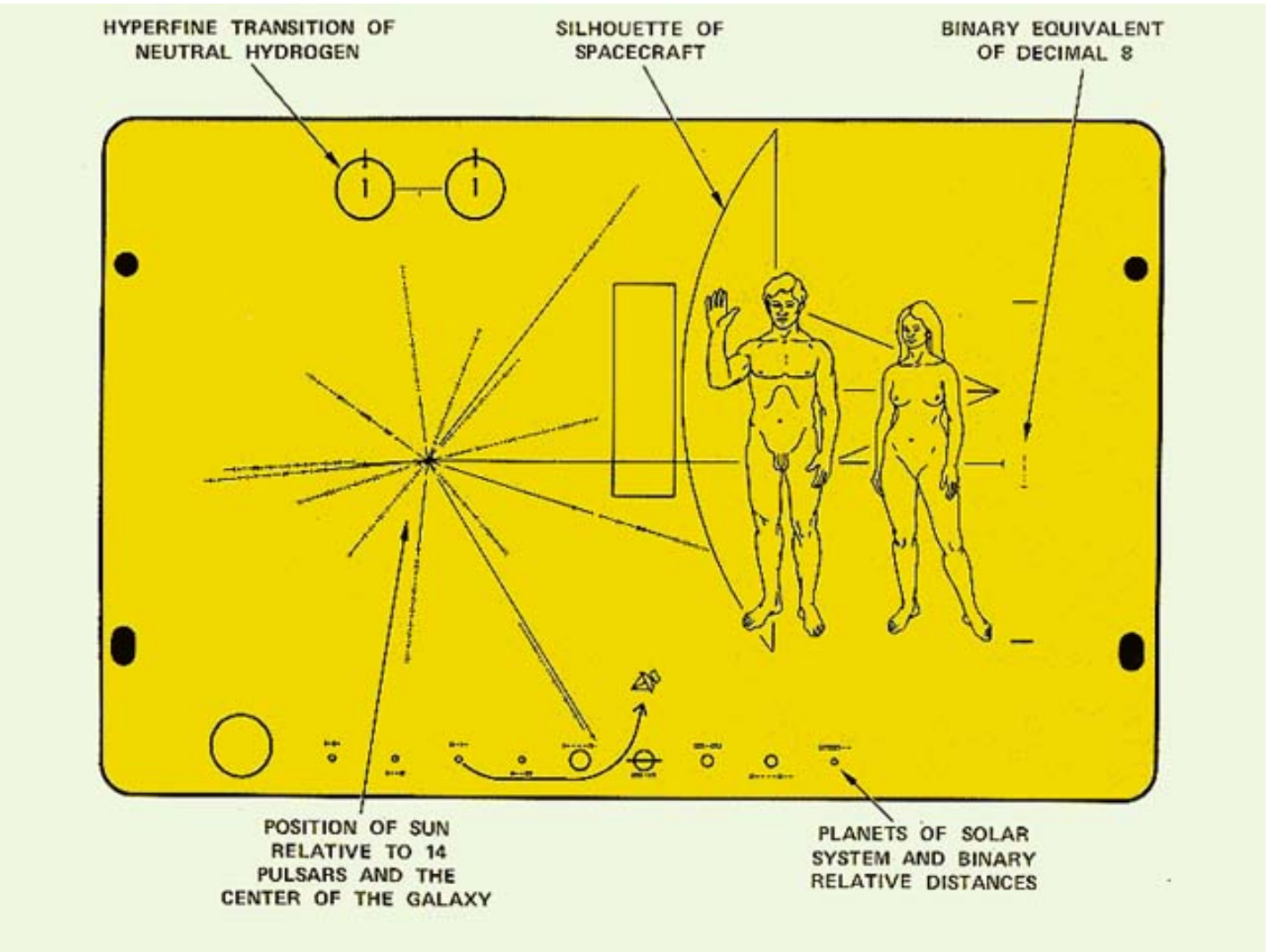
Conclusion:
Philosophy

Two Kinds of Coincidence

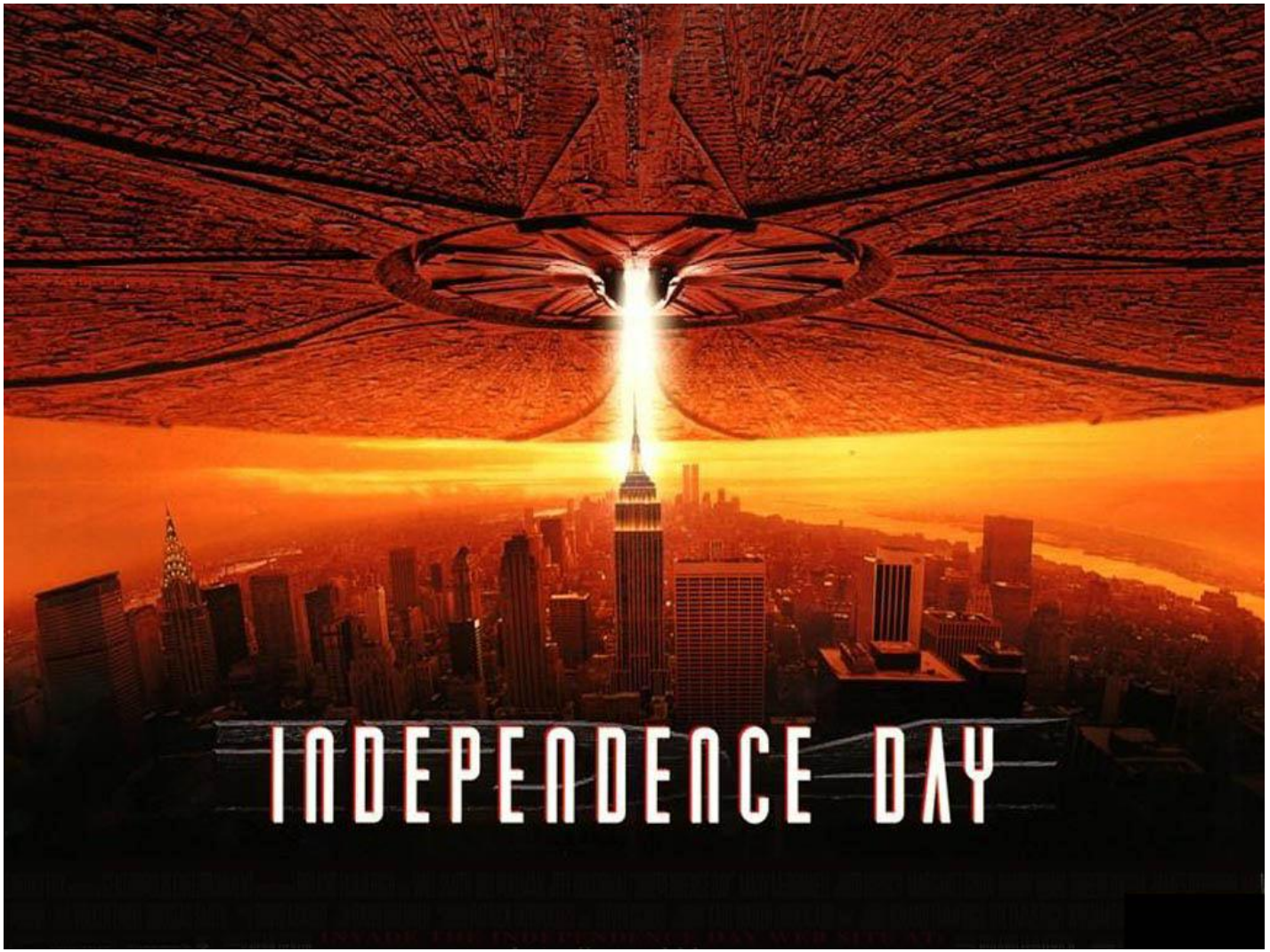




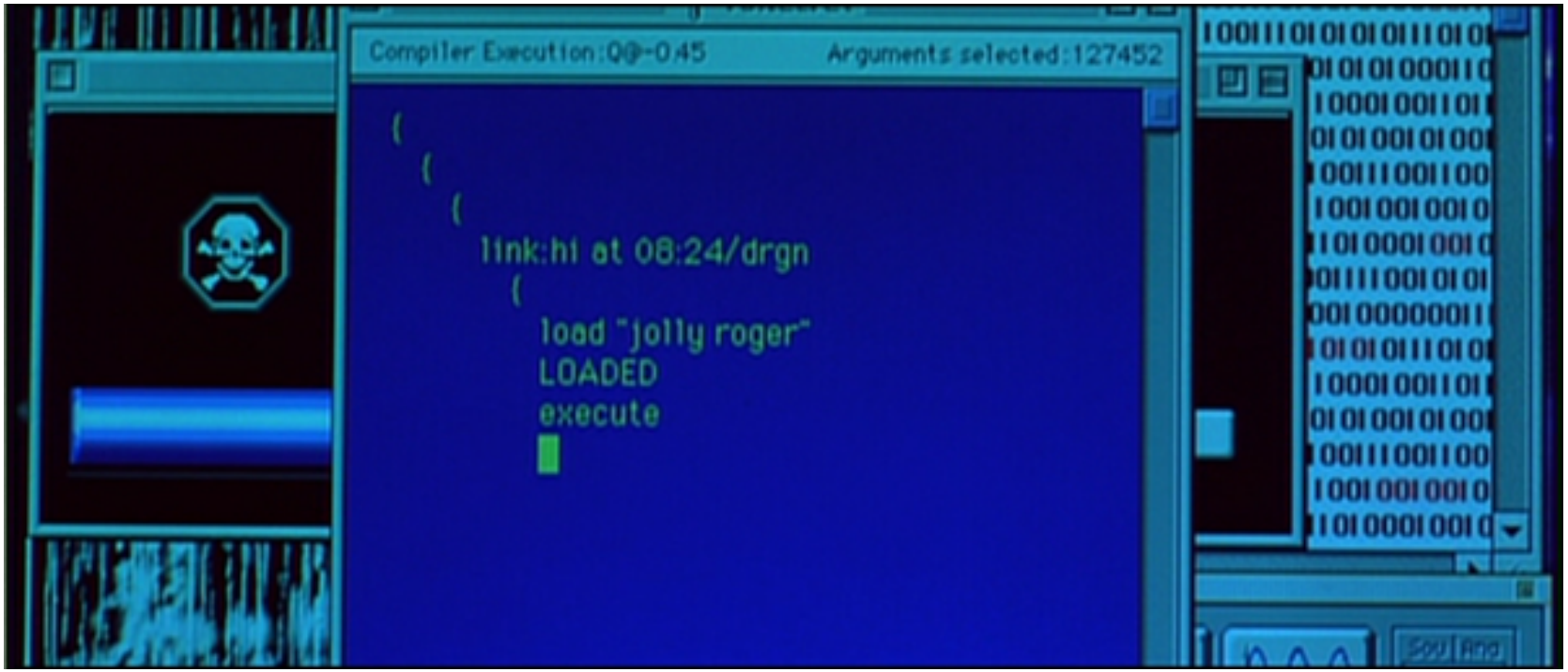
How to talk to aliens



Independence Day



A universal programming language?



Multiverses

Lambda is Omniversal