λ-Calculus: Then & Now

Dana S. Scott

University Professor Emeritus
Carnegie Mellon University
Visiting Scholar
University of California, Berkeley

dana.scott@cs.cmu.edu

TURING CENTENNIAL CELEBRATION
Princeton University, May 10-12, 2012

ACM TURING CENTENARY CELEBRATION
San Francisco, June 15-16, 2012

UC BERKELEY LOGIC COLLOQUIUM
Berkeley, August 24, 2012

Notes derived from the slides presented at the conferences.
A brief amount of text has been added for continuity.
The author would be happy to hear reactions and suggestions.
Version of 25 August 2012

Symbols of Princeton

Traditional

From the Graduate Alumni
(to encourage ecology)

The λ**-calculus** was begun at Princeton, and the purpose of this report is to show how it has been **recycled** every decade after the 1930s in **new** and **useful** ways.

WARNING: We cannot give here a complete history of Mathematical Logic and related areas. The present report may even have too much detail. But it is hoped readers might be encouraged to look further.

A Quick Look Back to Beginnings

1870s

Begriffsschrift	Frege (1879)

1880s

What are numbers?	Dedekind (1888)
Number-theoretic axioms	Peano (1889)

1890s

Vorlesungen über die Algebra der Logik	Schröder (1890–1905)
Grundgesetze der Arithmetik	Frege (1893-1903)
Formulario Mathematico	Peano (1895-1901)
Grundlagen der Geometrie	Hilbert (1899)

1900s

Diophantine problem	Hilbert (1900)
Russell's Paradox	Russell (1901)
Principles of Mathematics	Russell (1903)
Richard's Paradox	Richard (1905)
Theory of Types	Russell (1908)

1910s

Principia Mathematica	Whitehead-Russell (1910-12-13)
Calculus of relatives	Löwenheim (1915)

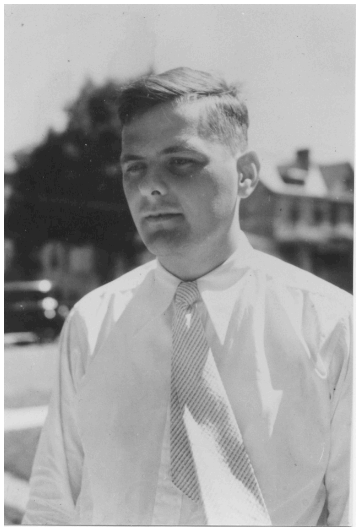
WW I

1920s

Löwenheim-Skolem Theorem	Skolem (1920)
Propositional calculus completeness	Post (1921)
Monadic predicate calculus decidable	Behmann (1922)
Abstract proof rules	Hertz (1922)
Primitive recursive arithmetic	Skolem (1923)
Combinators	Schönfinkel (1924)
Function-based set theory	von Neumann (1925)
“Conceptual” undecidability	Finsler (1926)
Epsilon operator	Hilbert-Bernays (1927)
Combinators (again)	Curry (1927)
Ackermann function	Ackermann (1928)
Entscheidungsproblem	Hilbert-Ackermann (1928)
Abriss der Logistik & simple type theory	Carnap (1929)

It was very reasonable for Hilbert and Ackermann to emphasize the Decision Problem, as special cases had been solved.

Church vs. Turing



Alonzo Church
Born: 14 June 1903 in Washington, D.C., USA.
Died: 11 Aug 1995 in Hudson, Ohio, USA.
Ph.D.: Princeton University, 1927, USA

Alan Turing
Born: 23 June 1912, Maida Vale, London, UK.
Died: 7 June 1954, Wilmslow, Cheshire, UK.
Ph.D.: Princeton University, 1938, USA.

Alonzo Church, *"An Unsolvable Problem in Elementary Number Theory,"* American J. of Mathematics, vol. 5 (1936), pp. 345-363.

Alonzo Church, *"A Note on the Entscheidungsproblem,"* J. of Symbolic Logic, vol. 1 (1936) pp. 40-41. Correction: *ibid*, pp. 101-102.

Alan Turing,*"On Computable Numbers with an Application to the Entscheidungsproblem,"* Proc. of the London Math. Soc., vol. 42 (1936), pp. 230-267. Correction: vol. 43 (1937), pp. 544-546.

Alan Turing,*"Computability and λ -definability,"* J. Symbolic Logic, vol. 2 (1937), pp. 153-163.

The work of Church and Turing in 1936 was done independently.

Three Pioneers

Haskell Brooks Curry

Born: 12 Sept 1900 in Millis, MA, USA.
Died: 1 Sept 1982 in State College, PA, USA.
Ph.D.: Göttingen Universität, 1930, Germany.
Thesis: Grundlagen der kombinatorischen Logik

Stephen Cole Kleene

Born: 5 Jan 1909 in Hartford, CN, USA.
Died: 25 Jan 1994 in Madison, WI, USA.
Ph.D.: Princeton University, 1934, USA.
Thesis: A Theory of Positive Integers in Formal Logic

J. Barkley Rosser

Born: 6 Dec 1907 in Jacksonville, FL, USA.
Died: 5 Sept 1989 in Madison, WI, USA.
Ph.D.: Princeton University, 1934, USA.
Thesis: A Mathematical Logic without Variables

It seems, sadly, that Alan Turing never had a chance to meet these people or Kurt Gödel.

A Very Busy Decade

1930s

Combinatory logic	Curry (1930-32)
Herbrand's Theorem	Herbrand (1930)
Completeness proof	Gödel (1930)
Partial consistency proof	Herbrand (1931)
Incompleteness	Gödel (1931)
Untyped λ-calculus	Church (1932-33-41)
Studies of primitive recursion	Péter (1932-36)
Non-standard models	Skolem (1933)
Functionality in Combinatory Logic	Curry (1934)
Grundlagen der Mathematik	Hilbert-Bernays (1934-39)
Natural deduction	Gentzen (1934)
Number-theoretic consistency & ε_0 *-induction*	Gentzen (1934)
Inconsistency of Church's System	Kleene-Rosser (1936)
Confluence theorem	Church-Rosser (1936)
Finite combinatory processes	Post (1936)
Turing machines	Turing (1936-37)
Recursive undecidability	Church-Turing (1936)
General recursive functions	Kleene (1936)
Further completeness proofs	Maltsev (1936)
Improving incompleteness theorems	Rosser (1936)
Fixed-point combinator	Turing (1937)
Computability and λ-definability	Turing (1937)

Starting out with Gödel and ending up with Turing,
it would take a long time to comprehend
and apply all the developments
in this period.

What is the λ-Calculus?

The calculus gives rules for the ***explicit definition*** of functions; however, the ***type-free*** version also permits ***recursion*** and ***self-replication.***

α-conversion

λX.[...X...] = λY.[...Y...]

β-conversion

(λX.[...X...])(T) = [...T...]

η-conversion

λX.F(X) = F

Church's original system (1932) also had rules for ***logic***, but that was the system Kleene-Rosser (1936) proved ***inconsistent!***

The names of the rules are due to Curry. The last rule fails in many interpretations, and special efforts are needed to make it valid.

Does λ-Calculus have Models?

Yes! There ***is*** a calculus for ***enumeration operators!***
First we need some simple definitions on integers and sets of integers:

$$(n,m) = 2^n(2m+1)$$

$$\mathbf{set}(0) = \varnothing$$

$$\mathbf{set}((n,m)) = \mathbf{set}(n) \cup \{m\}$$

$$X^* = \{n \mid \mathbf{set}(n) \subseteq X\}$$

Application

$$F(X) = \{m \mid \exists n \in X^*.(n,m) \in F\}$$

Abstraction

$$\lambda X.[\ldots X\ldots] =$$

$$\{0\} \cup \{(n,m) \mid m \in [\ldots\mathbf{set}(n)\ldots]\}$$

Every set of integers can be used as an enumeration operator. The operator is computable if the set is r.e. Many compound contexts do define enumeration operators.

The Connection to Computability

Church Numerals

$$\underline{\mathbf{0}} = \lambda F.\lambda X.X$$

$$\underline{\mathbf{n+1}} = \lambda F.\lambda X.F(\underline{\mathbf{n}}(F)(X))$$

$$\underline{\mathbf{n+m}} = \lambda F.\lambda X.\underline{\mathbf{n}}(F)(\underline{\mathbf{m}}(F)(X))$$

$$\underline{\mathbf{n\times m}} = \lambda F.\underline{\mathbf{n}}(\underline{\mathbf{m}}(F))$$

$$\underline{\mathbf{m^n}} = \underline{\mathbf{n}}(\underline{\mathbf{m}})$$

$\underline{\mathbf{n-1}}$ = [*a little harder*]

Fixed-Point Combinator

$$\mathbf{Y} = \lambda F.(\lambda X.F(X(X)))(\lambda X.F(X(X)))$$

$$\mathbf{Y}(F) = F(\mathbf{Y}(F))$$

Theorem. For every ***partial recursive function*** g(n), there is a ***constant λ-term*** **G** such that

$\mathbf{G}(\underline{n}) = \underline{g(n)}$, for all n.

Kleene and **Turing** independently proved this in different ways.

In the **model**, G denotes an r.e. set.

Some λ-Definitions

pair = λX.λY.λF.F(X)(Y)

fst = λP.P(λX.λY.X)

snd = λP.P(λX.λY.Y)

succ = λN.λF.λX.F(N(F)(X))

shft = λS.λP. **pair**(S(**fst**(P)))(**fst**(P))

pred = λN. **snd**(N(**shft**(**succ**))(**pair**(<u>**0**</u>)(<u>**0**</u>)))

Kleene's "trick" here is to introduce **pairs** as a **data structure**, and then apply iteration to get a **sequence** of pairs.

test = λN.λU.λV.**snd**(N(**shft**(λX.X))(**pair**(V)(U)))

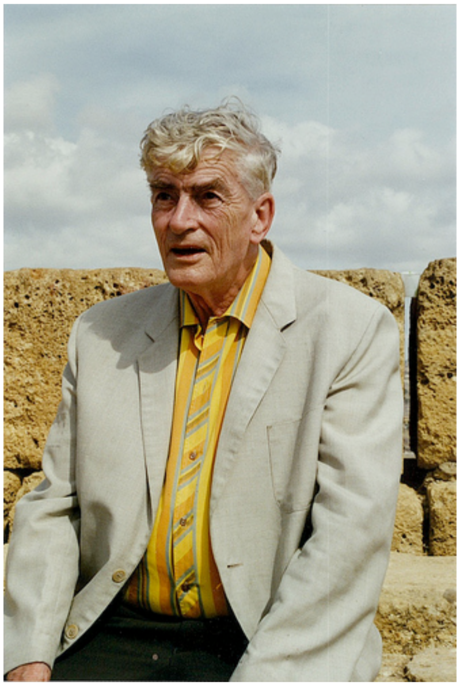
mult = λN.λM.λF.N(M(F))

fact = λN.**test**(N)(<u>**1**</u>)(**mult**(N)(**fact**(**pred**(N))))

fact = **Y**(λF.λN.**test**(N)(<u>**1**</u>)(**mult**(N)(F(**pred**(N)))))

The factorial function must be the most **overdefined** function in the history of mankind!

Turing's Only Student



Robin Oliver Gandy

Born: 23 September 1919, Peppard, Oxon., UK.
Died: 20 November 1995, Oxford, UK.
Ph.D.: Cambridge, 1953.
Thesis: *On axiomatic systems in Mathematics and theories in Physics.*
Supervisor: Alan Turing.
Reader: Oxford University, Wolfson College, 1969-1986.
Students: 26 and 126 descendants.

Another pioneer, **Gandy**, later became a key contributor to the development of **Recursive Function Theory**.

It is interesting to note that both the teams of **Myhill** and **Shepherdson** and, later, **Friedberg** and **Rogers** **defined** enumeration operators without seeing they had **models** for the λ-calculus.

Church–Turing Thesis

accepted with the help of Kleene after Turing explained his machines.

Effectively computable functions of natural numbers can be identified with those definable by:

- **λ-calculus**
- **Herbrand-Gödel equations**
- **Partial-recursive schemata**
- **Turing-Post machine programs**

If Gödel had stayed in Princeton, and

If Church and Kleene had argued better for data structures in the λ-calculus,

Then surely Gödel would have accepted λ-calculus as a foundation much earlier.

Note that Kleene proved the equivalence with Herbrand-Gödel computability **before** Turing's work.

Kleene's Complaint

I myself, perhaps unduly influenced by rather chilly receptions from audiences around **1933-35** to disquisitions on λ-definability, chose, after ***general recursiveness*** had appeared, to put my work in that format. I did later publish one paper **1962** on λ-definability in higher recursion theory.

I thought general recursiveness came the closest to ***traditional mathematics***. It spoke in a language familiar to mathematicians, extending the theory of ***special recursiveness***, which derived from formulations of Dedekind and Peano in the mainstream of mathematics.

I cannot complain about my audiences after **1935**, although whether the improvement came from switching I do not know. In retrospect, I now feel it was too bad I did not keep active in λ-definability as well. So I am glad that interest in λ-definability has revived, as illustrated by Dana Scott's **1963** communication.

Were the truth to be known, Kleene **translated** much of what he had done in λ-calculus into working with integers. Indeed, the **application operation** {e}(n) defines a **partial combinatory algebra** with many properties similar to the work of Curry and Rosser.

What is the Entscheidungsproblem?

To determine whether a formula of the ***first-order*** predicate calculus is ***provable*** or not.

Church's Solution

Theorem. Only a finite number of axioms are needed to define a ***non-recursive*** set of integers.

R.M.Robinson's Arithmetic

(1) $\forall x \forall y \, [\, x = y \iff Sx = Sy \,]$

(2) $\forall x \, [\, x = 0 \iff \neg \exists y.\ x = Sy \,]$

(3) $\forall x \forall y \, [\, (x + 0) = x \;\&\; (x + Sy) = S(x + y) \,]$

(4) $\forall x \forall y \, [\, (x \times 0) = 0 \;\&\; (x \times Sy) = ((x \times y) + x) \,]$

After the solution of Hilbert's 10th Problem, the applicability of this theory became even easier.

Turing's Solution

Theorem. Only a finite number of axioms are needed to define the ***Universal Turing Machine***.

Minskyizing the UTS

Starting with **Claude Shannon** in 1956, many people — often in competition with **Marvin Minsky** — proposed **very small UTMs** (but their operation requires extensive coding of **patterns**). But, **axiomatically**, they do not require as many axioms as Turing did.

Post-Markov's Solution

The basic idea of Post (1943) was that a **logistic system** is simply a set of rules specifying how to **change** one string of symbols (**antecedent**) into another string of symbols (**consequent**). This leads to:

The Word Problem for Semigroups

(1) $\forall x \forall y \, [\, x\, 1 = x = 1\, x \,]$

(2) $\forall x \ \forall y \ \forall z \, [\, x\,(y\, z) = (x\, y)\, z \,]$

Problem: Determine the provability of

$$A_0 = B_0 \ \& \ A_1 = B_1 \ \& \ \ldots \ \& \ A_{n-1} = B_{n-1} \Longrightarrow A_n = B_n \ .$$

Schönfinkel-Curry's Solution

Schönfinkel in 1924 and then Curry in 1929, both at Göttingen, began the study of **combinators**, which were quickly connected with Church's λ**-calculus** of 1932.

From them — with hindsight — we get:

Another Undecidable Theory

(1) $\forall x\ \forall y\ [\ \mathbf{K}(x)(y) = x\ ]$

(2) $\forall x\ \forall y\ \forall z\ [\ \mathbf{S}(x)(y)(z) = x(z)(y(z))\ ]$

(3) $\neg\ \mathbf{K} = \mathbf{S}$

Problem: Determine the provability of T = **0**.

The only problem with this theory is that you either need **models** or something like the

Church-Rosser Theorem

to know it is **consistent**. A weaker theory of **deterministic reduction** can be given a fairly short axiomatization and then be proved consistent by much simpler means.

What's Happened Since the 1930s?

The 1940s

Simple type theory & λ-calculus	Church (1940)
Primitive recursive functionals	Gödel (1941-58)
WW II	
Recursive hierarchies	Kleene (1943)
Theory of categories	Eilenberg-Mac Lane (1945)
New completeness proofs	Henkin (1949-50)

The 1950s

Computing and Intelligence	Turing (1950)
Rethinking combinators	Rosenbloom (1950)
IAS Computer (MANIAC)	von Neumann (1951)
Introduction to Metamathematics	Kleene (1952)
IBM 701	Thomas Watson, Jr. (1952)
Arithmetical predicates	Kleene (1955)
FORTRAN	Backus et al. (1956-57)
ALGOL 58	Bauer et al. (1958)
LISP	McCarthy (1958)
Combinatory Logic. Volume I.	Curry-Feys-Craig (1958)
Adjoint functors	Kan (1958)
Recursive functionals & quantifiers, I.&II.	Kleene (1959-63)
Countable functionals	Kleene-Kreisel (1959)

McCarthy, LISP, & λ-Calculus

LISP History according to McCarthy's memory in 1978. Presented at the ACM SIGPLAN History of Programming Languages Conference, June 1-3, 1978. It was pubished in **History of Programming Languages**, edited by Richard Wexelblat, Academic Press 1981. ***Two quotations:***

I spent the summer of 1958 at the IBM Information Research Department at the invitation of Nathaniel Rochester and chose differentiating algebraic expressions as a sample problem. It led to the following innovations beyond the FORTRAN List Processing Language:

• • • •

(c) To use functions as arguments, one needs a notation for functions, and it seemed natural to use the λ-notation of Church (1941). I didn't understand the rest of his book, so I wasn't tempted to try to implement his more general mechanism for defining functions. Church used higher-order functionals instead of using conditional expressions. Conditional expressions are much more readily implemented on computers.

• • • •

Logical completeness required that the notation used to express functions used as functional arguments be extended to provide for recursive functions, and the LABEL notation was invented by Nathaniel Rochester for that purpose. D. M. R. Park pointed out that LABEL was logically unnecessary since the result could be achieved using only λ — by a construction analogous to Church's Y-operator, albeit in a more complicated way.

Other key McCarthy publications:

Recursive Functions of Symbolic Expressions and their Computation by Machine (Part I). The original paper on LISP from **CACM**, April 1960. Part II, which never appeared, was to have had some Lisp programs for algebraic computation.

A Basis for a Mathematical Theory of Computation, first given in 1961, was published by North-Holland in 1963 in **Computer Programming and Formal Systems**, edited by P. Braffort and D. Hirschberg.

Towards a Mathematical Science of Computation, IFIPS 1962 extends the results of the previous paper. Perhaps the first mention and use of ***abstract syntax***.

Correctness of a Compiler for Arithmetic Expressions with James Painter. May have been the first proof of ***correctness of a compiler***. Abstract syntax and Lisp-style recursive definitions kept the paper short.

An HTML site concerning Lisp history can be found at:

http://www8.informatik.uni-erlangen.de/html/lisp-enter.html

The 1960s

Recursive procedures	Dijkstra (1960)
ALGOL 60	Backus et al. (1960)
Elementary formal systems	Smullyan (1961)
Grothendieck topologies	M.Artin (1962)
Higher-type λ-definability	Kleene (1962)
Grothendieck topoi	Grothendieck et al. SGA 4 (1963-64-72)
CPL	Strachey, et al. (1963)
Functorial semantics	Lawvere (1963)
Continuations (1)	van Wijngaarden (1964)
Adjoint functors & triples	Eilenberg-Moore (1965)
•Cartesian closed categories•	Eilenberg-Kelly (1966)
ISWIM *& SECD machine*	Landin (1966)
CUCH *& combinator programming*	Böhm (1966)
New foundations of recursion theory	Platek (1966)
Normalization Theorem	Tait (1967)
AUTOMATH *& dependent types*	de Bruijn (1967)
Finite-type computable functionals	Gandy (1967)
ALGOL 68	van Wijngaarden (1968)
Normal-form discrimination	Böhm (1968)
Category of sets	Lawvere (1969)
Typed domain logic	Scott (1969-93)
Domain-theoretic λ-models	Scott (1969)
Formulae-as-types	Howard (1969 -1980)
Adjointness in foundations	Lawvere (1969)

Theorem. The category of $\mathbf{T_0}$-topological spaces and continuous functions is ***not*** cartesian closed.

Theorem. The category of $\mathbf{T_0}$-topological spaces ***with*** an equivalence relation and continuous functions ***respecting*** equivalence ***is*** cartesian closed.

Cartesian closed categories give us the algebraic version of typed λ-calculus.

The 1970s

Continuations (2)	Mazurkiewicz (1970)
Continuations (3)	F. Lockwood Morris (1970)
Continuations (4)	Wadsworth (1970)
Categorical logic	Joyal (1970+)
Elementary topoi	Lawvere-Tierney (1970)
Denotational semantics	Scott-Strachey (1970)
Coherence in closed categories	Kelly (1971)
Quantifiers and sheaves	Lawvere (1971)
Martin-Löf type theory	Martin-Löf (1971)
Logic for Computable Functions	Milner (1972)
System F, Fω	Girard (1974)
From sheaves to logic	Reyes (1974)
Polymorphic λ-calculus	Reynolds (1974)
Call-by-name, call-by-value	Plotkin (1975)
Modeling Processes	Milner (1975)
Scheme	Sussman-Steele (1975-80)
Functional programming & FP	Backus (1977)
First-order categorical logic	Makkai-Reyes (1977)
Edinburgh LCF	Milner et al. (1978)
Let-polymorphic type inference	Milner (1978)
Intersection types	Coppo-Dezani (1978)
ML	Milner et al. (1979)
**-Autonomous categories*	Barr (1979)
Sheaves and logic	Fourman-Scott (1979)

This decade saw the importance of constructive logic, the applications to language design and semantics, and the connections to category theory become much clearer.

The 1980s

Frege structures	Aczel (1980)
HOPE	Burstall et al. (1980)
The Lambda Calculus Book	Barendregt (1981-84)
Structural Operational Semantics	Plotkin (1981)
Effective Topos	Hyland (1982)
Dependent types & modularity	Burstall-Lampson (1984)
Locally CCC & type theory	Seely (1984)
Calculus of Constructions	Coquand-Huet (1985)
Bounded quantification	Cardelli-Wegner (1985)
NUPRL	Constable et al. (1986)
Higher-order categorical logic	Lambek-P.J.Scott (1986)
Cambridge LCF	Paulson (1987)
Linear logic	Girard et al. (1987-89)
HOL	Gordon (1988)
FORSYTHE	Reynolds (1988)
Proofs and Types	Girard et al. (1989)
Integrating logical & categorical types	Gray (1989)
Computational λ-calculus & monads	Moggi (1989)

Type theory, resource logic, and computer-assisted theorem proving finally became practical during these years.

The 1990s

HASKELL	Hudak-Hughes-Peyton Jones-Wadler (1990)
Higher-type recursion theory	Sacks (1990)
STANDARD ML	Milner, et al. (1990-97)
Lazy λ-calculus	Abramsky (1990)
Higher-order subtyping	Cardelli-Longo (1991)
Categories, Types and Structure	Asperti-Longo (1991)
STANDARD ML of NJ	MacQueen-Appel (1991-98)
QUEST	Cardelli (1991)
Edinburgh LF	Harper, et al. (1992)
Pi-Calculus	Milner-Parrow-Walker (1992)
Categorical combinators	Curien (1993)
Translucent types & modular	Harper-Lillibridge (1994)
Full abstraction for PCF	Hyland-Ong/Abramsky, et al. (1995)
Algebraic set theory	Joyal-Moerdijk (1995)
Object Calculus	Abadi-Cardelli (1996)
Typed intermediate languages	Tarditi, Morrisett, et al. (1996)
Proof-carrying code	Necula-Lee (1996)
Computability and totality in domains	Berger (1997)
Typed assembly language	Morrisett, et al. (1998)
Type theory via exact categories	Birkedal, et al. (1998)
Categorification	Baez (1998)

Abstract ideas now found many applications in language implementation and in compiling.

The New Millennium

Predicative topos	Moerdijk-Palmgren (2000)
Sketches of an Elephant	Johnstone (2002+)
Differential λ-calculus	Ehrhard/Regnier (2003)
Modular Structural Operational Semantics	Mosses (2004)
A λ-calculus for real analysis	Taylor (2005+)
Homotopy type theory	Awodey-Warren (2006)
Univalence axiom	Voevodsky (2006+)
The safe λ-calculus	Ong, et al. (2007)
Higher topos theory	Lurie (2009)

Univalent Foundations Program @ IAS
Voevodsky, et al. (2012-13)

In the natural world, convergent evolution can give creatures analogous structures — even though they cannot mate. But, in the intellectual world, analogous structures can be taken advantage of through interfertilization of areas and in finding new applications.

And that we have seen happen with the λ-calculus many, many times over the years.

A Closing Thought from Robert Harper

For me, I think it is important to stress the ***overwhelming influence*** of the λ-calculus among all other models of computation:

- It codifies not only computation, but also the basic principles of ***human reason*** (natural deduction).
- Moreover, it was ***born fully formed***, and is directly and immediately relevant to this day, rather than something that collects dust on the shelf.

Admittedly Turing's model had the advantage of being ***explicitly psychologically motivated***, but on the other hand Church focused on one of the greatest achievements of the human mind, ***the concept of a variable*** (= reasoning under hypotheses). Church saw that this was central, and time has born out the significance of his insight.

By contrast, no one cares one bit about the ***details*** of a Turing Machine; for, it fails to address the central issue of ***modularity*** (logical consequence), which is so important in programming and reasoning. And it does not extend to ***higher-order computation*** in anything like a natural or smooth way.

LAMBDA CONQUERS ALL!

Perhaps my good friend and colleague has spoken a little too strongly here, as Turing Machines have had many applications, say in Complexity Theory. But the study of Programming Languages does not seem to need them today.

A Selective Bibliography

*A very helpful review of the subject of the λ-calculus is in the first reference, and the memoirs by Alonzo Church's two early students are also useful in checking history. The thesis by Rod Adams gives a very careful survey of early literature. A somewhat revisionist view of the history of recursive function theory with many helpful references is found in the Soare paper. Jones and Simonsen fill out ideas related to machine structure. The whole Royal Society volume is devoted to **The Turing Legacy.** And Plotkin also recently wrote on operational semantics. The older collection edited by Rolf Herken, **The Universal Turing Machine: A Half-Century Survey**, has many, many excellent historical discussions by Kleene, Gandy, Davis, Feferman, and others. The papers of Davis and Sieg give very detailed historical reviews of the early 1930s. The recent conference **Church's Thesis After 70 Years** (Olszewski, et al. eds. 2006) has many interesting discussions.*

F. Cardone and J.R. Hindley. Lambda-Calculus and Combinators in the 20th Century. In: Volume 5, pp. 723-818, of ***Handbook of the History of Logic***, Dov M. Gabbay and John Woods eds., North-Holland/Elsevier Science, 2009.

S. C. Kleene. Origins of recursive function theory. ***Annals of the History of Computing***, vol. 3 (1981), pp. 52–67.

J. B. Rosser. Highlights of the history of the lambda calculus. ***Annals of the History of Computing***, vol. 6 (1984), pp. 337–349.

R. Adams. ***An Early History of Recursive Functions and Computability from Gödel to Turing.*** 1983 Ph.D. Thesis. Reprinted by Docent Press, 2011.

R.I. Soare. Formalism and intuition in computability. ***Phil. Trans. Royal Soc. A,*** vol. 370 (2012), pp. 3277–3304.

N.D. Jones and J.G. Simonsen. Programs = data = first-class citizens in a computational world. ***Phil. Trans. Royal Soc. A,*** vol. 370 (2012), pp. 3305-3318.

G.D. Plotkin. The origins of structural operational semantics. ***Journal of Logic and Algebraic Programming***, vol. 60 (2004), pp. 3-15.

M. Davis. Why Gödel Didn't Have Church's Thesis, ***Information and Control***, vol.54 (1982), pp. 3-24.

W. Sieg. Step by Recursive Step: Church's Analysis of Effective Calculability. ***Bull. of Symbolic Logic***, vol. 3 (1997), pp. 154-180. (Reprinted in Olszewski, et al. 2006.)

*What follows is a listing of **books**. Ph.D. theses and conference proceedings have been excluded, for the most part, as well as very elementary text books. A comprehensive survey is impossible, but the current list has tried to indicate some of the history and development of the **intertwining strands** of λ-calculus, logic, recursive-function theory, category theory, and programming-language semantics.*

M. Abadi and L. Cardelli. ***A Theory of Objects.*** Springer, 1996.

J. Adamek, H. Herrlich and G. E. Strecker. ***Abstract and Concrete Categories: The Joy of Cats.*** Dover Pub., 2009.

L. Allison. ***A Practical Introduction to Denotational Semantics.*** Cambridge Univ. Press, 1987.

R. Amadio and P.-L. Curien. ***Domains and Lambda-Calculi***, volume 46 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998.

C.A. Anderson and M. Zelëny, eds. ***Logic, Meaning and Computation : Essays in Memory of Alonzo Church.*** Springer, 2001.

A.W. Appel. ***Compiling with Continuations.*** Cambridge Univ. Press, 2007.

A.W. Appel, ed. ***Alan Turing's Systems of Logic: The Princeton Thesis.*** Princeton Univ. Press, 2012.

A. Asperti and G. Longo. ***Categories, Types and Structures: An Introduction to Category Theory for the Working Computer Scientist.*** MIT Press, 1991.

S. Awodey. ***Category Theory.*** Oxford Univ. Press, 2010.

C. Badesa. ***The Birth of Model Theory: Löwenheim's Theorem in the Frame of the Theory of Relatives.*** Princeton Univ. Press, 2004.

H. P. Barendregt. ***The Lambda Calculus, its Syntax and Semantics***. North-Holland, 1981. 2nd (revised) ed. 1984.

J.L. Bell. ***Toposes and Local Set Theories: An Introduction.*** Dover Pub., 2008.

J. van Benthem. ***Language in Action: Categories, Lambdas, and Dynamic Logic.*** MIT Press, 1995.

T.J. Bergin and R.G. Gibson, eds. ***History of Programming Languages.*** ACM Press and Addison-Wesley, 1996.

K. Bimbó. ***Combinatory Logic: Pure, Applied, Typed.*** CRC Press, 2012.

G.S. Boolos, J.P. Burgess, and R.C. Jeffery. ***Computability and Logic.*** 5th ed., Cambridge Univ. Press, 2007.

G. Brady. ***From Peirce to Skolem: A Neglected Chapter in the History of Logic.*** North Holland, 2000.

K.B. Bruce. ***Foundations of Object-Oriented Languages: Types and Semantics.*** MIT Press, 2002.

W.H. Burge. ***Recursive Programming Techniques.*** Addison-Wesley, 1975.

G. Castagna. ***Object Oriented Programming: A Unified Foundation.*** Birkhäuser, 1997.

A. Church. ***The Calculi of Lambda Conversion***. Princeton University Press, 1941. Reprinted 1951 and 2000.

P. Cockshott, L.M. Mackenzie and G. Michaelson. ***Computation and its Limits.*** Oxford Univ. Press, 2012.

R. Constable, *et al*. ***Implementing Mathematics with The Nuprl Proof Development System.*** CreateSpace, reprint 2012.

S.B. Cooper. ***Computability Theory.*** CRC Press, 2003.

S.B. Cooper and J. van Leeuwen, eds. ***Alan Turing: His Work and Impact.*** Elsevier Science, 2012.

G. Cousineau and M. Mauny. ***The Functional Approach to Programming.*** (K. Callaway, trans.), Cambridge Univ. Press, 1998.

R.L. Crole. ***Categories for Types.*** Cambridge Univ. Press, 1994.

P.-L. Curien. ***Categorical Combinators, Sequential Algorithms and Functional Programming***. Pitman (UK) and Wiley (USA), 1986. 2nd edn., Birkhäuser, 1993.

H. B. Curry and R. Feys. ***Combinatory Logic, Volume I.*** North-Holland, 1958. (3rd edn. 1974).

H. B. Curry, J. R. Hindley, and J. P. Seldin. ***Combinatory Logic, Volume II.*** North-Holland, 1972.

M. Davis. ***Computability and Unsolvability.*** Dover Pub., 1982.

M. Davis, R. Sigal, E.J. Weyuker. ***Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science.*** 2nd ed., Morgan Kaufmann, 1994.

M. Davis, ed. ***The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions.*** Dover Pub., 2004.

M. Davis. ***The Universal Computer: The Road from Leibniz to Turing.*** CRC Press, 2012.

M. Dezani, S. R. della Rocca, and M. Venturini Zilli, eds. ***A Collection of Contributions in Honour of Corrado Böhm***. Elsevier, 1993.

R. DiCosmo. ***Isomorphisms of Types: from Delta-Calculus to Information Retrieval and Language Design.*** Birkhäuser, 1994.

K. Doets and J. van Eijck. ***The Haskell Road to Logic, Maths and Programming.*** College Publications, 2004.

G. Dowek and J-J. Lévy. ***Introduction to the Theory of Programming Languages.*** Springer, 2010.

R.K. Dybvig. ***The Scheme Programming Language.*** MIT Press, 2009.

G. Dyson. ***Turing's Cathedral: The Origins of the Digital Universe.*** Pantheon Books, 2012.

E. Engeler, et al. ***The Combinatory Programme.*** Birkhäuser, 1994.

T. Ehrhard, et al. ***Linear Logic in Computer Science.*** Cambridge Univ. Press, 2004.

S. Feferman. ***In the Light of Logic.*** Oxford Univ. Press, 1998.

M.P. Fiore. ***Axiomatic Domain Theory in Categories of Partial Maps.*** Cambridge Univ. Press, 2004.

P. Fletcher. ***Truth, Proof and Infinity: A Theory of Constructive Reasoning.*** Springer, 2010.

M.P. Fourman, P.T. Johnstone and A.M. Pitts, eds. ***Applications of Categories in Computer Science: Proceedings of the London Mathematical Society Symposium, Durham 1991.*** Cambridge Univ. Press, 1992.

T. Franzén. ***Gödel's Theorem: An Incomplete Guide to Its Use and Abuse.*** A K Peters/CRC Press, 2005.

D.P. Friedman and M. Felleisen. ***The Little Schemer.*** MIT Press, 1995.D.P. Friedman and M. Felleisen. ***The Seasoned Schemer.*** MIT Press, 1995.

D.P. Friedman, W.E. Byrd and O. Kiselyov. ***The Reasoned Schemer.*** MIT Press, 2005.

D.P. Friedman and M. Wand. ***Essentials of Programming Languages.*** MIT Press, 2008.

G. Gierz, *et al.* ***Continuous Lattices and Domains.*** Cambridge Univ. Press, 2003.

J.-Y. Girard, Y. Lafont, and P. Taylor. ***Proofs and Types***. Cambridge Univ. Press, 1989.

R. Goldblatt. ***Topoi: The Categorial Analysis of Logic.*** Dover Pub, 2006.

M.J.C. Gordon. ***The Denotational Description of Programming Languages.*** Springer, 1979.

A.D. Gordon. ***Functional Programming and Input/Output.*** Cambridge Univ. Press, 1994.

J.G. Granström. ***Treatise on Intuitionistic Type Theory.*** Springer, 2011.

I. Grattan-Guinness. ***The Search for Mathematical Roots 1870–1940.*** Princeton University Press, 2000.

C. A. Gunter. ***Semantics of Programming Languages: Structures and Techniques.*** MIT Press,1992.

C.A. Gunter and J.C. Mitchell, eds. ***Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design.*** MIT Press, 1994.

L. Haaparanta. ***The Development of Modern Logic.*** Oxford Univ. Press, 2009.

C. Hankin. ***Lambda Calculi: A Guide for Computer Scientists.*** Clarendon Press, 1994.

M. Hansen and H. Rischel. ***Introduction to Programming using SML.*** Addison Wesley, 1999.

R. Harper. ***Practical Foundations for Programming Languages.*** Cambridge Univ. Press, forthcoming 2013?

J. van Heijenoort. ***From Frege to Gödel: A Source Book in Mathematical Logic, 1879 – 1931.***
Harvard Univ. Press, 1967.

P. Henderson. ***Functional Programming: Application and Implementation***. Prentice Hall, 1980.

R. Herken. ***The Universal Turing Machine: A Half-Century Survey***, Springer-Verlag 1988 (2nd ed., 1995).

J. R. Hindley and J. P. Seldin, eds. ***To H. B. Curry, Essays on Combinatory Logic, Lambda Calculus and Formalism***. Academic Press, 1980.

J. R. Hindley and J. P. Seldin. ***Introduction to Combinators and Lambda-Calculus***. Cambridge Univ. Press, 1986.

J. R. Hindley, B. Lercher, and J. P. Seldin. ***Introduction to Combinatory Logic***. Cambridge Univ. Press, 1972.

J. R. Hindley. ***Basic Simple Type Theory.*** Cambridge Univ. Press, 1997.

G. Huet, ed. ***Logical Foundations of Functional Programming***. Addison-Wesley, 1990.

G. Huet and G. Plotkin, eds. ***Logical Frameworks.*** Cambridge Univ. Press, 1991.

M. Huth and M. Ryan. ***Logic in Computer Science: Modelling and Reasoning about Systems.*** Cambridge Univ. Press, 2004.

B. Jacobs. ***Categorical Logic and Type Theory***. Elsevier Science, 2001.

M.P. Jones. ***Qualified Types: Theory and Practice.*** Cambridge Univ. Press, 2003.

P.T. Johnstone. ***Sketches of an Elephant: A Topos Theory Compendium, vols. 1 and 2.*** Oxford Univ. Press, 2002. (Vol. 3 in preparation.)

F.D. Kamareddine, T. Laan, R. Nederpelt. ***A Modern Perspective on Type Theory: From its Origins until Today.*** Springer, 2004.

S.C. Kleene. ***Introduction to Metamathematics.*** North-Holland, 1952. (Reprinted 1964. Reprinted with an introduction by M. Beeson, Ishi Press, 2009.)

J.-L. Krivine. ***Lambda-Calculus, Types and Models***, Ellis-Horwood (USA) and Prentice-Hall (UK), 1993.

R. Krömer. ***Tool and Object: A History and Philosophy of Category Theory.*** Birkhäuser, 2007.

J. Lambek and P. J. Scott. ***Introduction to Higher-Order Categorical Logic.*** Cambridge Univ. Press, 1988.

F.W. Lawvere and S.H. Schanuel. ***Conceptual Mathematics: A First Introduction to Categories.*** Cambridge Univ. Press, 2009.

J. van Leeuwen, ed. ***Formal Models and Semantics.*** Elsevier Science, 1992.

H. Lewis and C.H. Papadimitriou. ***Elements of the Theory of Computation.*** 2nd ed., Prentice-Hall, 1997.

Z. Luo. ***Computation and Reasoning: A Type Theory for Computer Science.*** Oxford Univ. Press, 1994.

J. Lurie. ***Higher Topos Theory.*** Princeton Univ. Press, 2009.

S. Mac Lane. ***Mathematics: Form and Function.*** Springer, 1985.

S. Mac Lane. ***Categories for the Working Mathematician.*** Springer, 2nd ed., 1998.

P. Mancosu. ***The Adventure of Reason. Interplay Between Philosophy of Mathematics and Mathematical Logic, 1900-1940.*** Oxford Univ. Press, 2010.

J.-P. Marquis. ***From a Geometrical Point of View: A Study of the History and Philosophy of Category Theory.*** Springer, 2008.

P. Martin-Löf. ***Intuitionistic Type Theory***. Studies in Proof Theory. Bibliopolis, Napoli, 1984. Notes by Giovanni Sambin of lectures given in Padova, June 1980.

R. Milne and Christopher Strachey. ***Theory of Programming Language Semantics. Parts A & B in Two Volumes.*** Chapman & Hall, 1977.

R. Milner. ***A Calculus of Communicating Systems.*** LNCS, vol. 92, Springer-Verlag, 1980.

R. Milner. ***Communication and Concurrency.*** Prentice-Hall, 1989.

R. Milner and M. Tofte. ***Commentary on Standard ML.*** The MIT Press, 1991.

R. Milner, R. Harper, D. MacQueen, and M. Tofte. ***The Definition of Standard ML - Revised.*** MIT Press, 1997.

J.C. Mitchell. ***Foundations for Programming Languages.*** MIT Press, 1996.

J.C. Mitchell. ***Concepts in Programming Languages***. Cambridge Univ. Press, 2002.

R. P. Nederpelt, J. H. Geuvers, and R. C. de Vrijer, eds. ***Selected Papers on Automath***. Elsevier, 1994.

E.J. Neuhold, et al. eds. ***Formal Description of Programming Concepts.*** Springer, 1991.

H.R. Nielson and F. Nielson. ***Semantics with Applications: An Appetizer.*** Springer, 2007.

B. Nordström, K. Petersson, and J. M. Smith. ***Programming in Martin-Löf 's Type Theory***. Oxford Univ. Press, 1990.

P. O'Hearn and R. Tennent. ***Algol-like Languages.*** Birkhäuser, 1996.

C. Okasaki. ***Purely Functional Data Structures.*** Cambridge Univ. Press, 1998.
A. Olszewski, J. Wolenski, R. Janusz, eds. ***Church's Thesis after 70 Years***. Ontos-Verlag, 2006.

J. Van Oosten. ***Realizability: An Introduction to Its Categorical Side***. Elsevier Science, 2008.

L.C. Paulson. ***ML for the Working Programmer.*** Cambridge Univ. Press, 1996.

M.C. Pedicchio and W. Tholen, eds. ***Categorical Foundations: Special Topics in Order, Topology, Algebra, and Sheaf Theory.*** Cambridge Univ. Press, 2003.

C. Petzold. ***The Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine.*** Wiley, 2008.

F. Pfenning. ***Computation and Deduction.*** Cambridge Univ. Press, 2000.

B. C. Pierce. ***Basic Category Theory for Computer Scientists.*** MIT Press, 1991.

B. C. Pierce. ***Types and Programming Languages***. MIT Press, 2002.

B. C. Pierce. ***Advanced Topics in Types and Programming Languages***. MIT Press, 2004.

A.M. Pitts and P. Dybjer, eds. ***Semantics and Logics of Computation.*** Camb. Univ. Pr, 1997.

I. Poernomo, J.N. Crossley, and M. Wirsing. ***Adapting Proofs-as-Programs : The Curry-Howard Protocol.*** Springer, 2005.

D. Prawitz. ***Natural Deduction: A Proof-Theoretical Study.*** Dover Pub., 2006.

J.C. Reynolds. ***The Craft of Programming.*** Prentice-Hall, 1981.

J.C. Reynolds. ***Theories of Programming Languages.*** Cambridge Univ. Press, 2009.
G.E. Révész. ***Lambda-Claculus, Combinators, and Functional Programming.*** Cambridge Univ. Press, 1988.

S.R. della Rocca and L. Paolini. ***The Parametric Lambda Calculus: A Metamodel for Computation.*** Springer, 2004.

P. Rosenbloom. ***The Elements of Mathematical Logic***. Dover, 1950.

D. Sangiorgi and D. Walker. T***he π-calculus: a Theory of Mobile Processes.*** Cambridge Univ. Press, 2001

D.A. Schmidt. ***Denotational Semantics: A Methodology for Language Development.*** William C Brown Pub., 1988.

D.A. Schmidt. ***The Structure of Typed Programming Languages.*** MIT Press, 1994.

H. Simmons. ***Derivation and Computation: Taking the Curry-Howard Correspondence Seriously.*** Cambridge Univ. Press, 2000.

H. Simmons. ***An Introduction to Category Theory.*** Cambridge Univ. Press, 2011.

R.M. Smullyan, ***Theory of formal systems.*** Annals of Mathematics Studies, vol. 47, 1961

R.M. Smullyan. ***To Mock a Mockingbird: And Other Logic Puzzles Including an Amazing Adventure in Combinatory Logic.*** Alfred A. Knopf, 1985.

G. Sommaruga. ***History and Philosophy of Constructive Type Theory.*** Springer, 2000/2010.

G. Sommaruga, ed. ***Foundational Theories of Classical and Constructive Mathematics.*** Springer, 2011.

M.H. Sørensen and P. Urzyczyn. ***Lectures on the Curry-Howard Isomorphism.*** Elsevier Science, 2006.

S. Stenlund. ***Combinators, Lambda-Terms and Proof Theory***. D. Reidel, 1972.
J. Stoy. ***Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory***. MIT Press, 1977.

A. Tarski, A. Mostowski and R.M. Robinson. ***Undecidable Theories.*** Dover Pub., 2010.

R.D. Tennent. ***Principles of Programming Languages.*** Prentice Hall, 1981.

S. Thompson. ***Type Theory and Functional Programming.*** Addison-Wesley, 1991.

S. Thompson. ***Haskell: The Craft of Functional Programming.*** 3rd. ed., Addison-Wesley, 2011.
R. Turner. ***Constructive Foundations for Functional Languages.*** McGraw Hill, 1991.

R.F.C. Walters. ***Categories and Computer Science.*** Cambridge Univ. Press, 1992.

I. Watson. ***The Universal Machine: From the Dawn of Computing to Digital Consciousness.*** Copernicus Books, 2012.

P. Wegner. ***Programming Languages, Information Structures, and Machine Organization.*** McGraw-Hill, 1968.

G. Winskel. ***Formal Semantics of Programming Languages.*** MIT Press, 1993.

V.E. Wolfengagen. ***Categorical Abstract Machine: Introduction to Computations.*** Center JurInfoR, Moscow, 2nd. ed., 2002.

V.E. Wolfengagen. ***Combinatory Logic in Programming.*** Center JurInfoR, Moscow, 2nd. ed.,2003.

V.E. Wolfengagen. ***Methods and Means for Computations with Objects: Applicative Computational Systems.*** Center JurInfoR, Moscow, 2004.

G.Q. Zhang. ***Logic of Domains.*** Birkhäuser, 1991.

H. Zenil, ed. ***A Computable Universe: Understanding and Exploring Nature as Computation.*** World Scientific, 2012.

And, no, I have not read — or even seen — all these books!

Suggestions, corrections and additions would be appreciated, so please send e-mail to **dana.scott@cs.cmu.edu** *with the subject heading:*
Lambda calculus.

The question of finding the the most recent edition of a book is vexing, but Amazon.com was quite helpful. Bibliographies of several books and papers were "mined", and of course all these books themselves also give references to the ever more vast journal literature. There is also the problem — in outlining history — of comparing the date of ***discovery*** *to the date of* ***publication****.*
Perhaps there are many such confusions in this survey.