

# Linguagens e Ambientes de Programação

## (Aula Teórica 1)

**LEI - Licenciatura em Engenharia Informática**

**João Costa Seco ([joao.seco@fct.unl.pt](mailto:joao.seco@fct.unl.pt))**

# Linguagens de Programação

# Alonzo Church (1903 - 1995)

- A linguagem de programação fundamental
- O sistema formal cálculo Lambda

$$E ::= x \mid \lambda x. E \mid E E$$

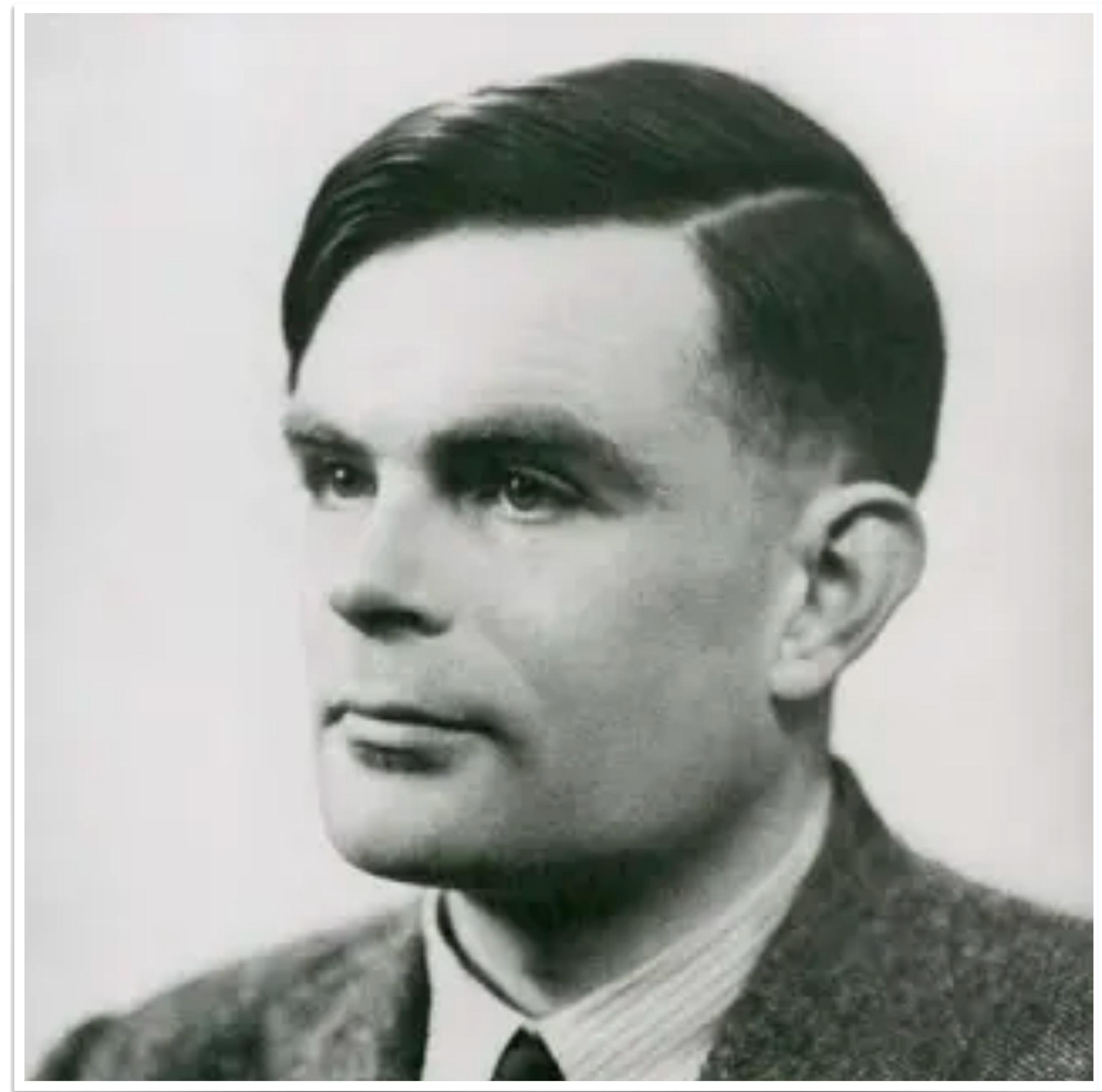
$$(\lambda x. E) E' \longrightarrow E\{E'/x\} \quad \frac{E \longrightarrow E''}{E E' \longrightarrow E'' E'}$$

- Provou a impossibilidade de resolver o *Entscheidungsproblem* (problema de decisão)



# Alan Turing (1912 - 1954)

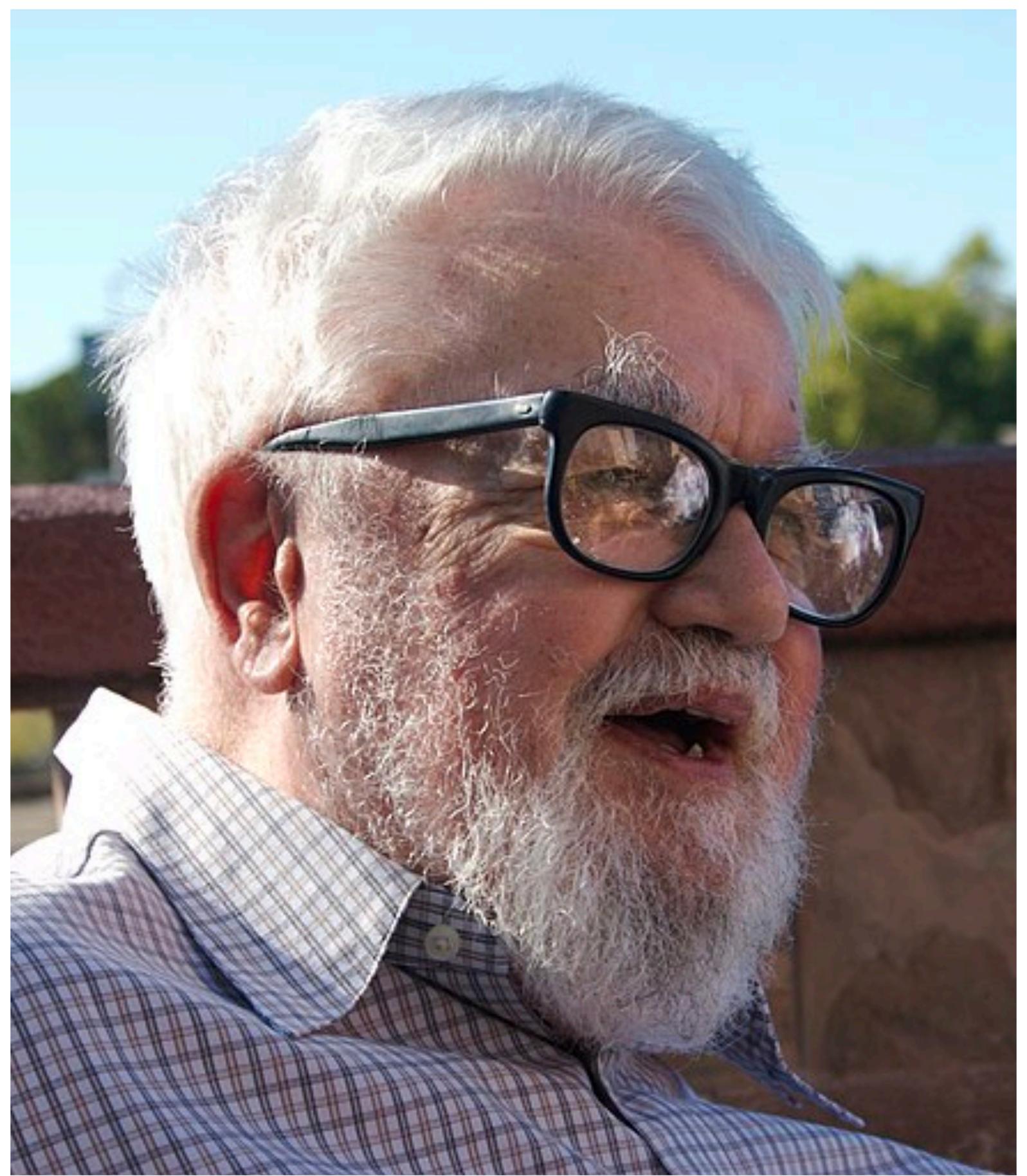
- A máquina de Turing
- o primeiro computador (WWII)
- A indecidibilidade do “halting problem”.
- Prova que o cálculo lambda é Turing complete.



# John McCarthy (1927 - 2011)

---

- A linguagem LISP (1960) e *garbage collection*
- LISP = LISt Processor
- Turing award 1971
- LISP já incluía :
  - Estruturas de dados em árvore
  - Garbage collection
  - Tipificação dinâmica
  - Funções de ordem superior
  - Recursão
  - Um REPL



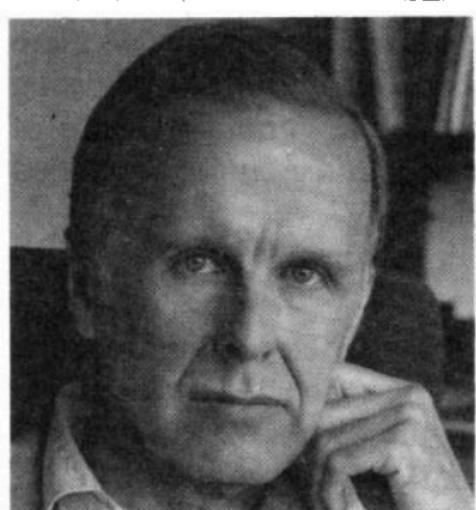
# John Backus (1924 - 2007)

- Fortran (1966), BNF e FP
- Comité do ALGOL 58, e ALGOL 60
- Turing award 1977
- FP (Point Free Style) - 1977



## Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs

John Backus  
IBM Research Laboratory, San Jose



Conventional programming languages are growing ever more enormous, but not stronger. Inherent defects at the most basic level cause them to be both fat and weak: their primitive word-at-a-time style of programming inherited from their common ancestor—the von Neumann computer, their close coupling of semantics to state transitions, their division of programming into a world of expressions and a world of statements, their inability to effectively use powerful combining forms for building new programs from existing ones, and their lack of useful mathematical properties for reasoning about programs.

O estilo Point Free não usa os nomes dos parâmetros e define funções só por composição.

f = (+ 1)

sum = foldr (+) 0

f x = x + 1

sum' xs = foldr (+) 0 xs

# Peter Landin (1930 - 2009)

- Máquina SECD para linguagens funcionais
  - Stack, Environment, Control, Dump
  - Registros de pilhas e uma lista associativa (Environment)
  - Instruções: nil, ldc, ld, sel, join, ldf, ap, ret, dum, rap



## The Next 700 Programming Languages

P. J. Landin

Univac Division of Sperry Rand Corp., New York, New York

"... today . . . 1,700 special programming languages used to 'communicate' in over 700 application areas."—*Computer Software Issues*, an American Mathematical Association Prospectus, July 1965.

A family of unimplemented computing languages is described that is intended to span differences of application area by a unified framework. This framework dictates the rules about the uses of user-coined names, and the conventions about characterizing functional relationships. Within this frame-

differences in the set of things provided by the library or operating system. Perhaps had ALGOL 60 been launched as a family instead of proclaimed as a language, it would have fielded some of the less relevant criticisms of its deficiencies.

# Edsger W. Dijkstra (1930 - 2002)

- Turing award 1972
- Definiu a noção de *structured programming*
  - Blocos de programa aninhados
  - Sem saltos
- Implementou o primeiro compilador de ALGOL

## Edgar Dijkstra: Go To Statement Considered Harmful

### Go To Statement Considered Harmful

Key Words and Phrases: go to statement, jump instruction, branch instruction, conditional clause, alternative clause, repetitive clause, program intelligibility, program sequencing

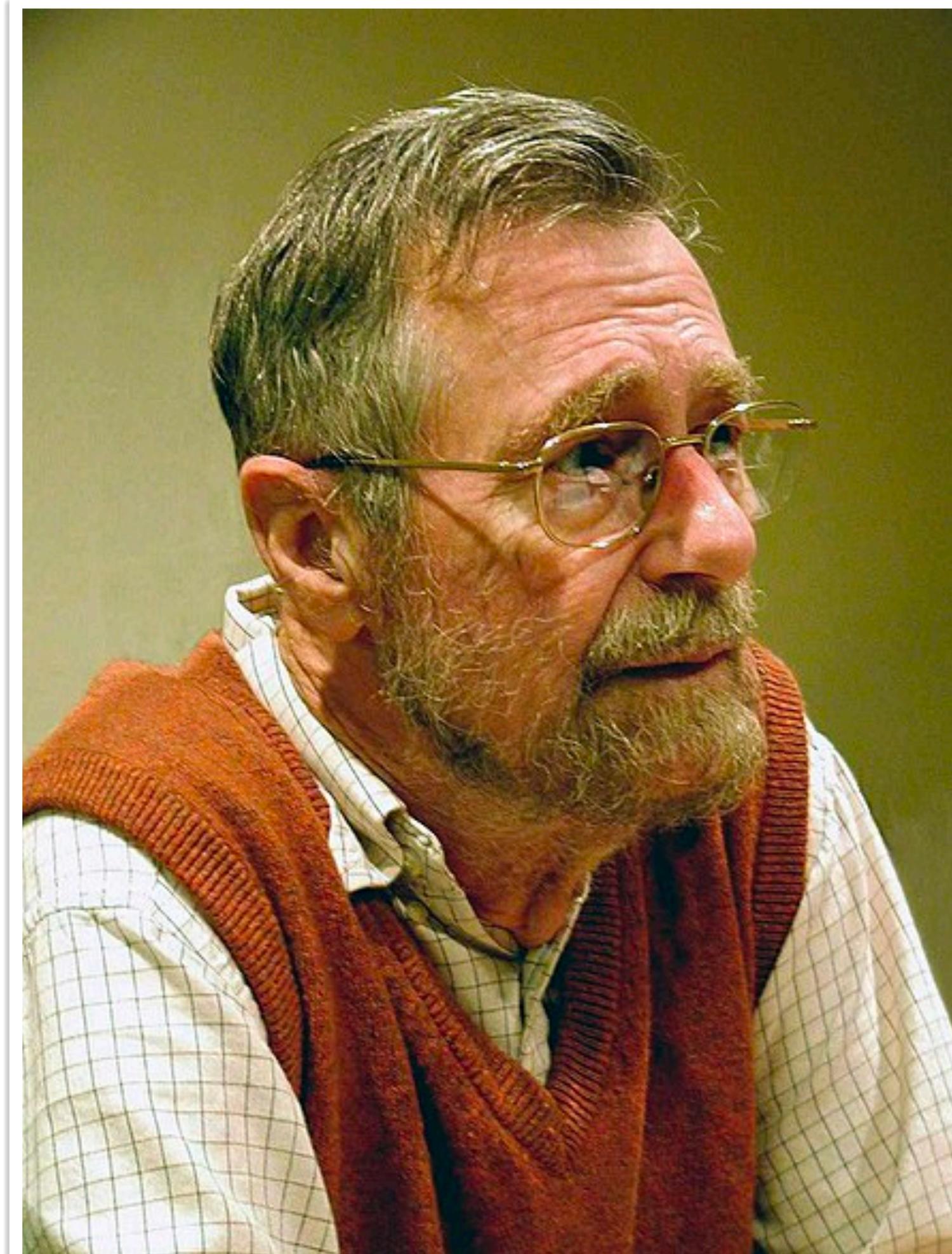
CR Categories: 4.22, 5.23, 5.24

#### EDITOR:

For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of **go to** statements in the programs they produce. More recently I discovered why the use of the **go to** statement has such

dynamic progress is only characterized when we also give to which call of the procedure we refer. With the inclusion of procedures we can characterize the progress of the process via a sequence of textual indices, the length of this sequence being equal to the dynamic depth of procedure calling.

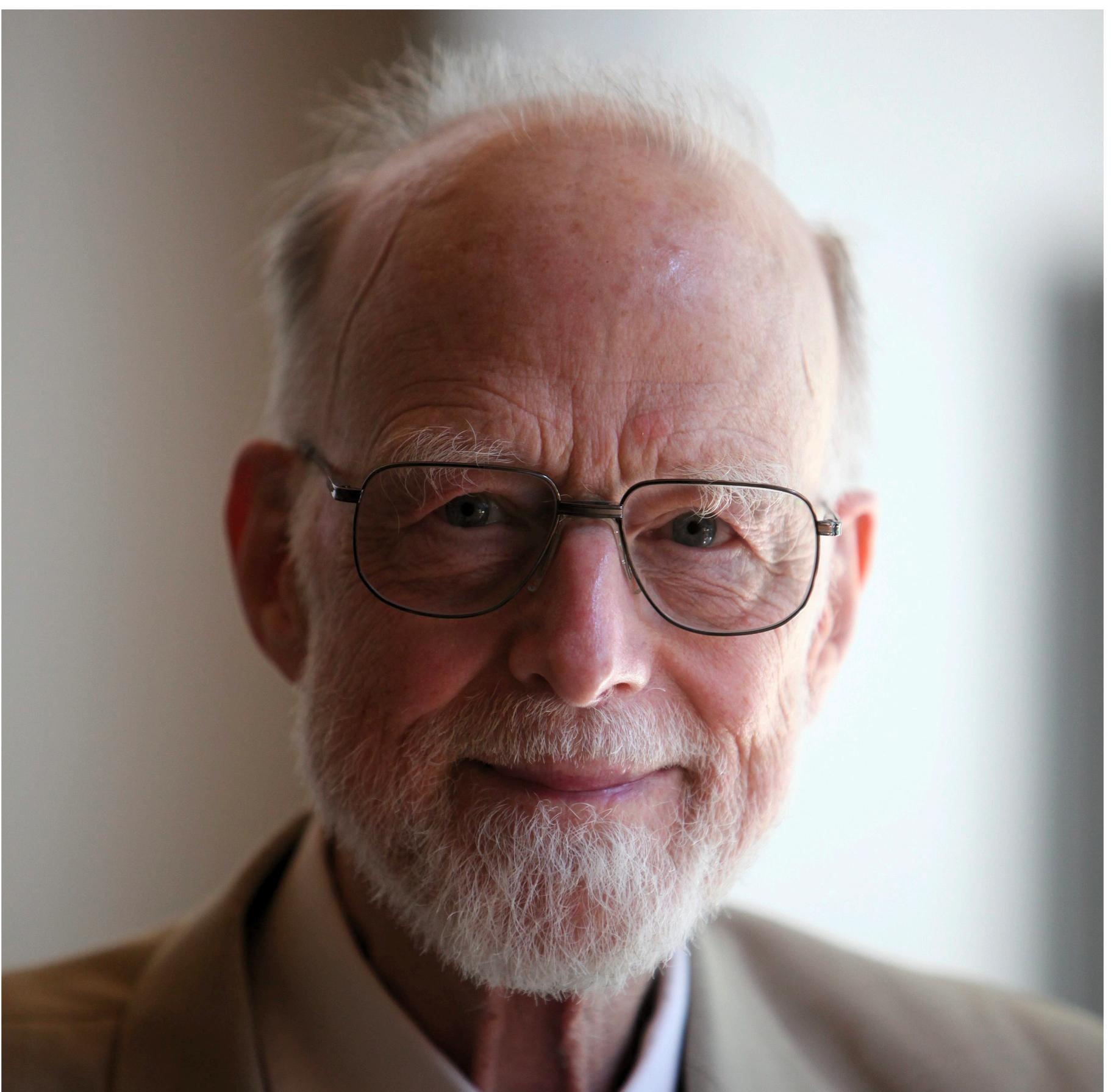
Let us now consider repetition clauses (like, **while** *B* **repeat** *A* or **repeat** *A* **until** *B*). Logically speaking, such clauses are now superfluous, because we can express repetition with the aid of recursive procedures. For reasons of realism I don't wish to exclude them: on the one hand, repetition clauses can be implemented quite comfortably with present day finite equipment; on



# Tony Hoare (1934)

---

- Turing Award 1980
- Contribuições em:
  - Linguagens de Programação (ALGOL)
  - Algoritmos (Qsort)
  - Sistemas operativos (Monitors)
  - Verificação formal de programas (HL)
  - programação concorrente (CSP)
- “Null References: The Billion Dollar Mistake”

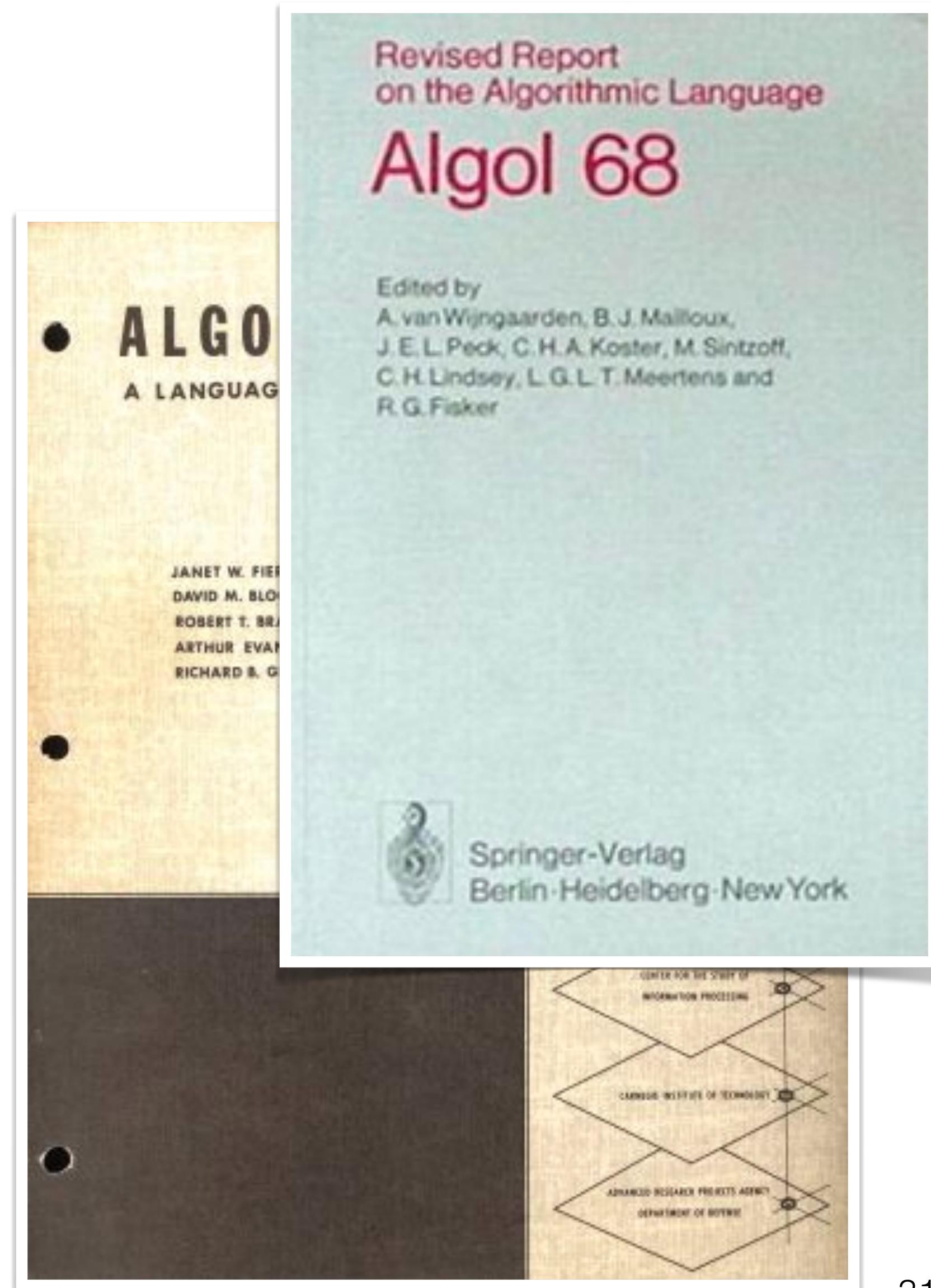


# The ALGOL Family

- Linguagem Imperativa concorrente com múltiplos paradigmas.
- Tipificação forte
- Objectos
- Null references



Top: John McCarthy, Fritz Bauer, Joe Wegstein. Bottom: John Backus, Peter Naur, Alan Perlis



# Robin Milner (1934-2010)

- Turing award 1991
- Contribuições:
  - LCF (Theorem Prover)
  - ML (Meta Language), a general-purpose FPL,
  - Polymorphic Hindley–Milner type system (type inference)
  - CCS & Pi-Calculus



[https://en.wikipedia.org/wiki/Robin\\_Milner](https://en.wikipedia.org/wiki/Robin_Milner)

1954

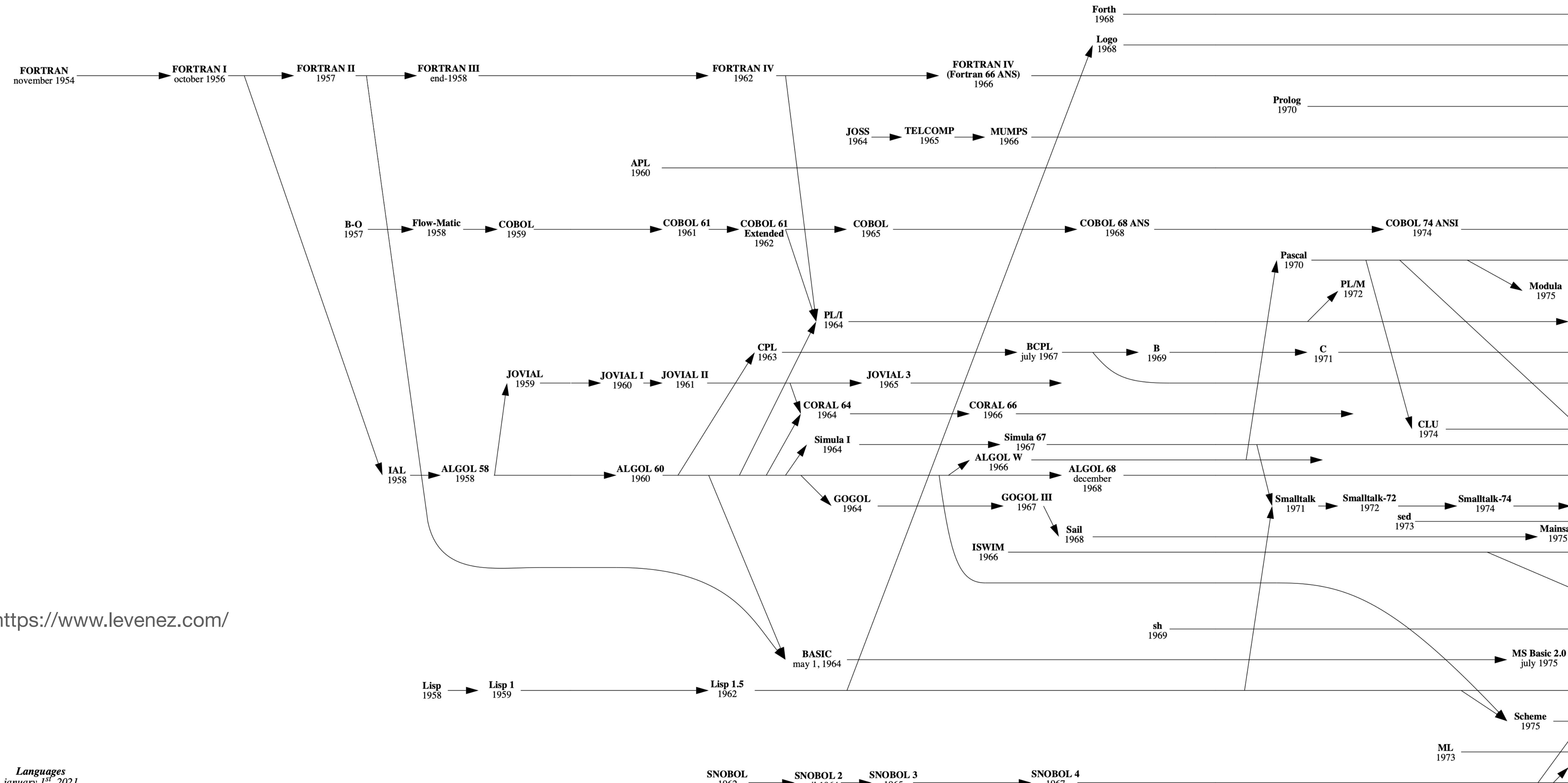
1957

1960

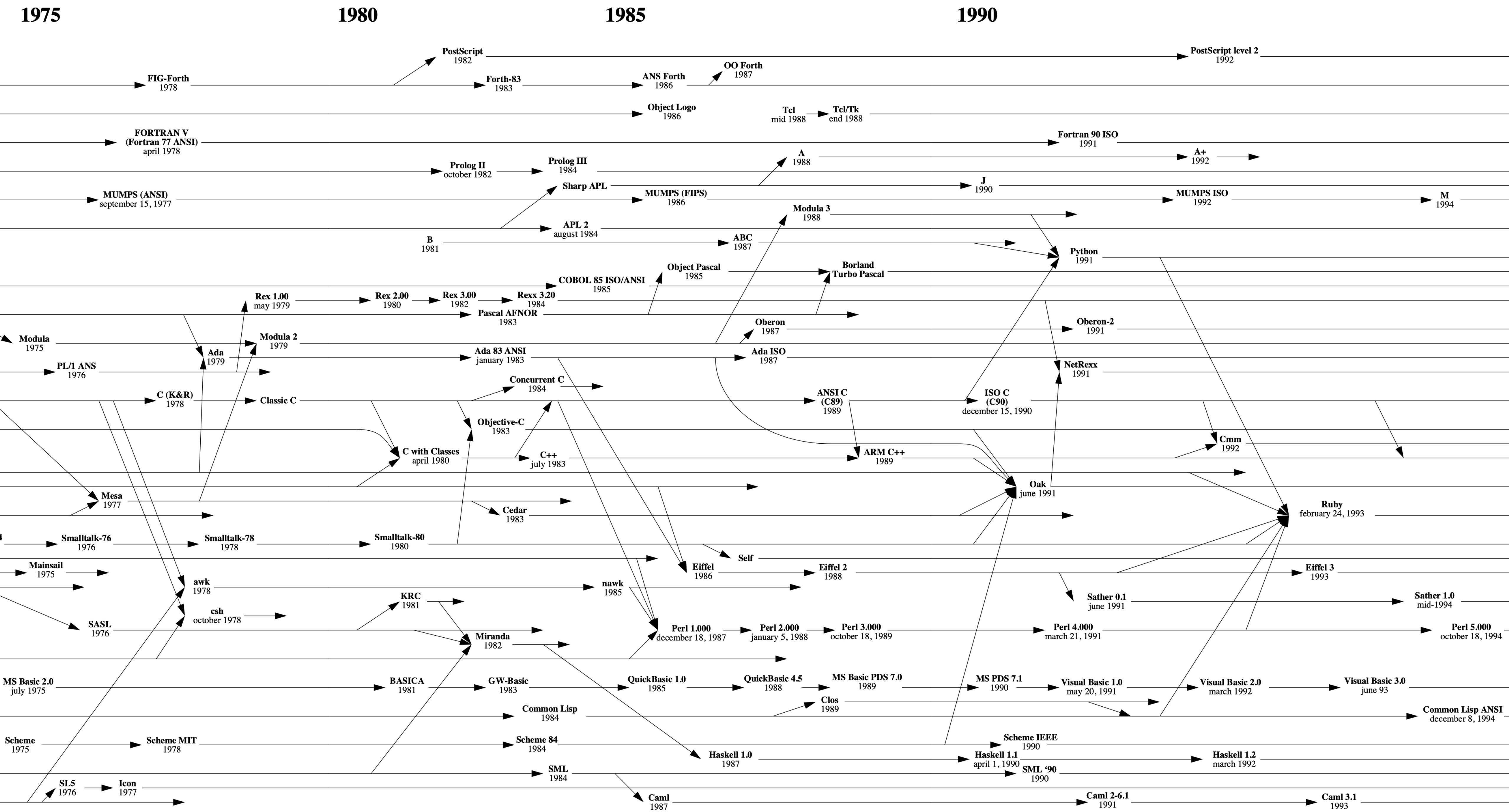
1965

1970

1975



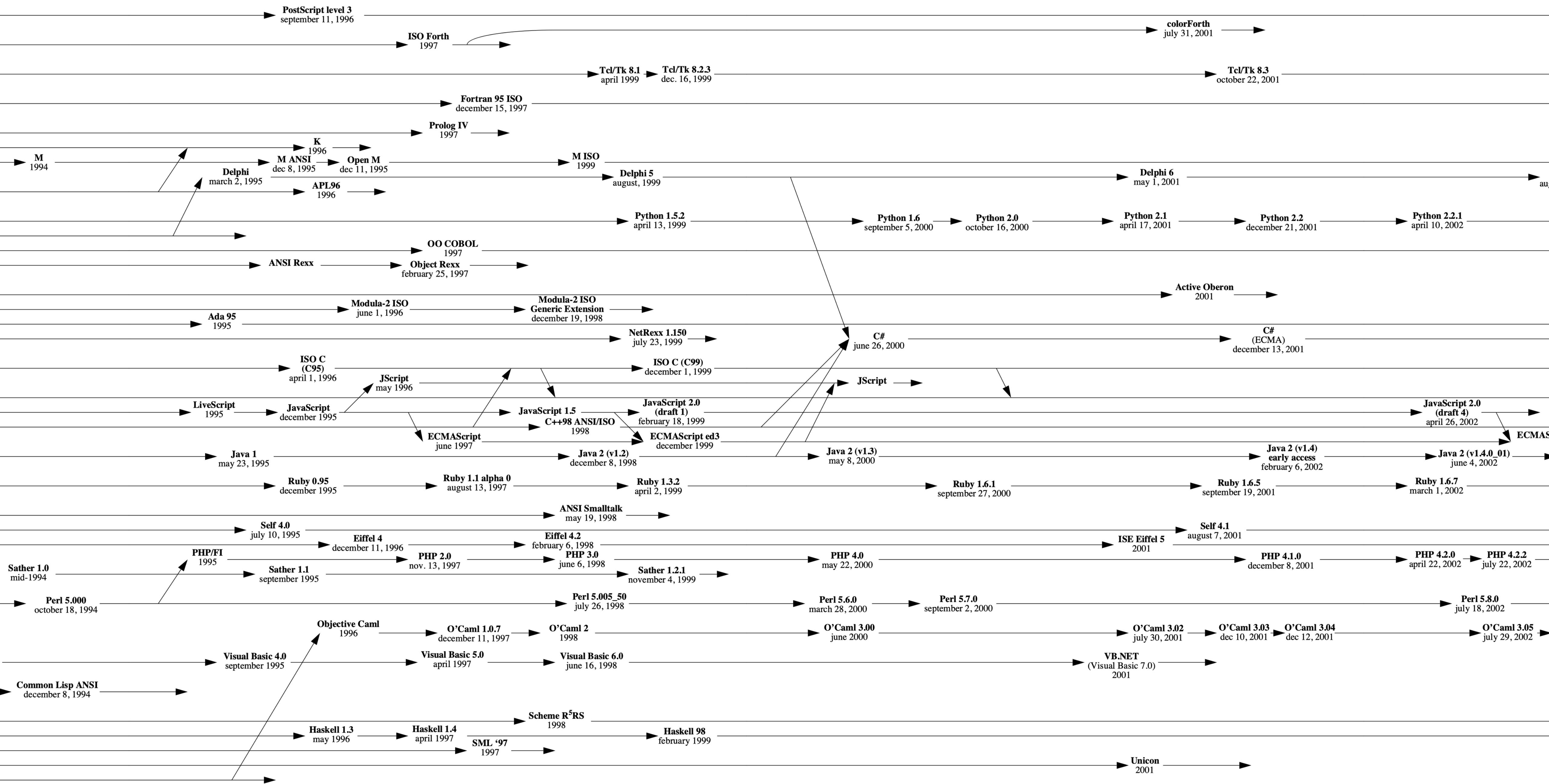
<https://www.levenez.com/>

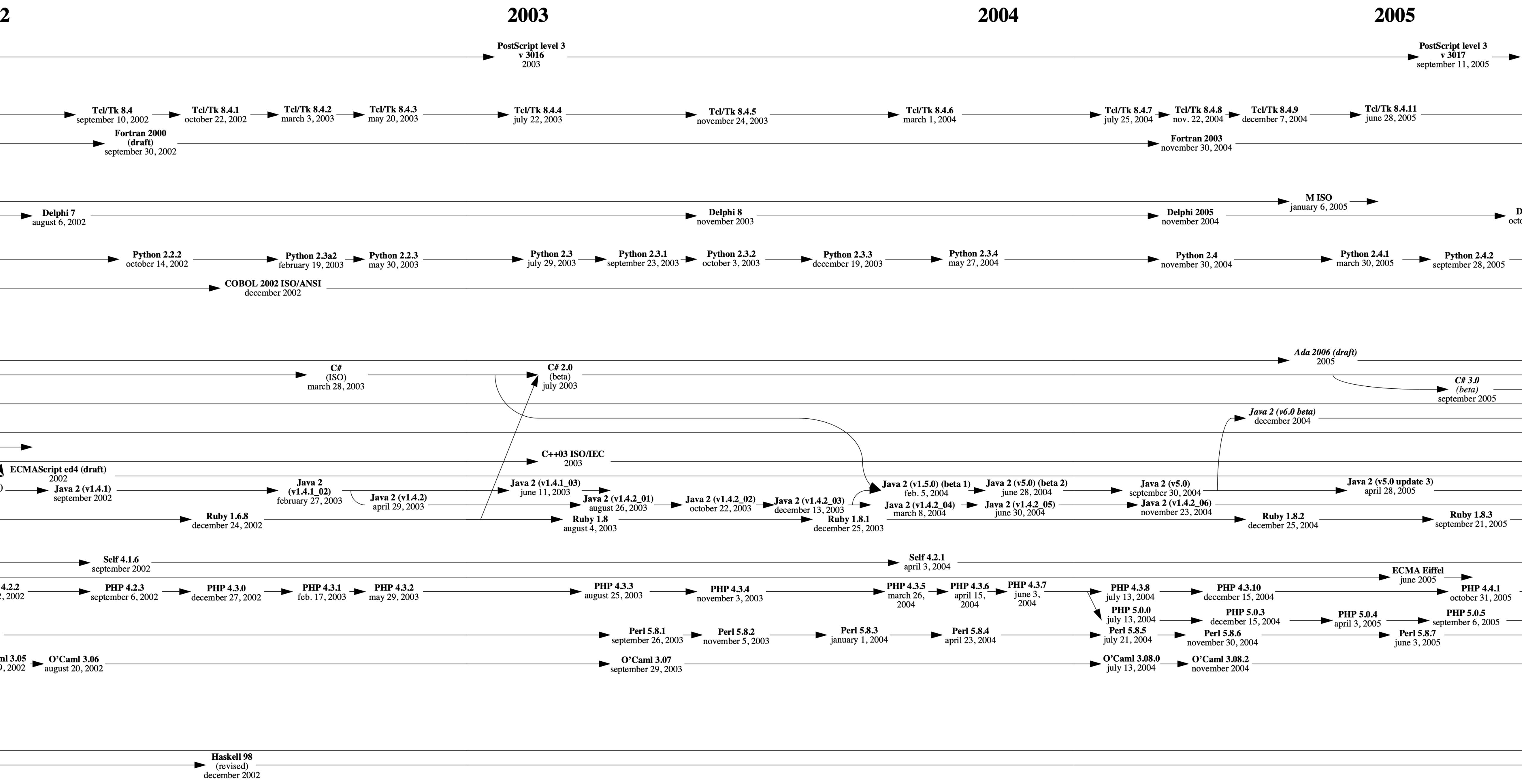


1995

2000

2002





2005

PostScript level 3  
v 3017  
september 11, 2005

2006

Tcl/Tk 8.4.11 june 28, 2005 → Tcl/Tk 8.4.12 december 6, 2005 → Tcl/Tk 8.4.13 april 19, 2006 → Tcl/Tk 8.4.14 october 19, 2006 → Tcl/Tk 8.4.15 may 25, 2007 → Tcl/Tk 8.5 december 20, 2007 → Tcl/Tk 8.5.5 october 15, 2008 → Tcl/Tk 8.5.6 january 2009 → Tcl/Tk 8.5.7 april 15, 2009

2007

Delphi 2006 october 30, 2005 → Delphi 2007 march 2007 → Python 3.0a2 december 7, 2007 → Delphi 2009 august 2008 → Python 3.0 december 3, 2008 → Python 3.0.1 february 13, 2009 → Python 3.1 june 27, 2009

aft)

C# 3.0 (beta) september 2005 → C# 2.0 november 2005 → Ada 2005 march 9, 2007 → C# 3.0 november 6, 2006 → C# 3.5 november 19, 2007 → Objective-C 2.0 august 7, 2006 → Java 6 december 11, 2006 → Java 6 update 2 july 5, 2007 → Java 6 update 7 july 11, 2008 → Java 6 update 11 december 2, 2008 → Java 6 update 14 june 10, 2009

Java 2 (v5.0 update 3) april 28, 2005 → Java 2 (v5.0 update 8) august 11, 2006 → Java 2 (v5.0 update 12) may 31, 2007 → Java 2 (v5.0 update 16) july 11, 2008 → Java 2 (v5.0 update 17) december 2, 2008 → Java 2 (v5.0 update 18) march 24, 2009 → Ruby 1.8.3 september 21, 2005 → Ruby 1.8.4 december 24, 2005 → Ruby 1.8.5 august 25, 2006 → Ruby 1.8.6 march 13, 2007 → Ruby 1.8.7 may 31, 2008 → Ruby 1.9.1 january 30, 2009

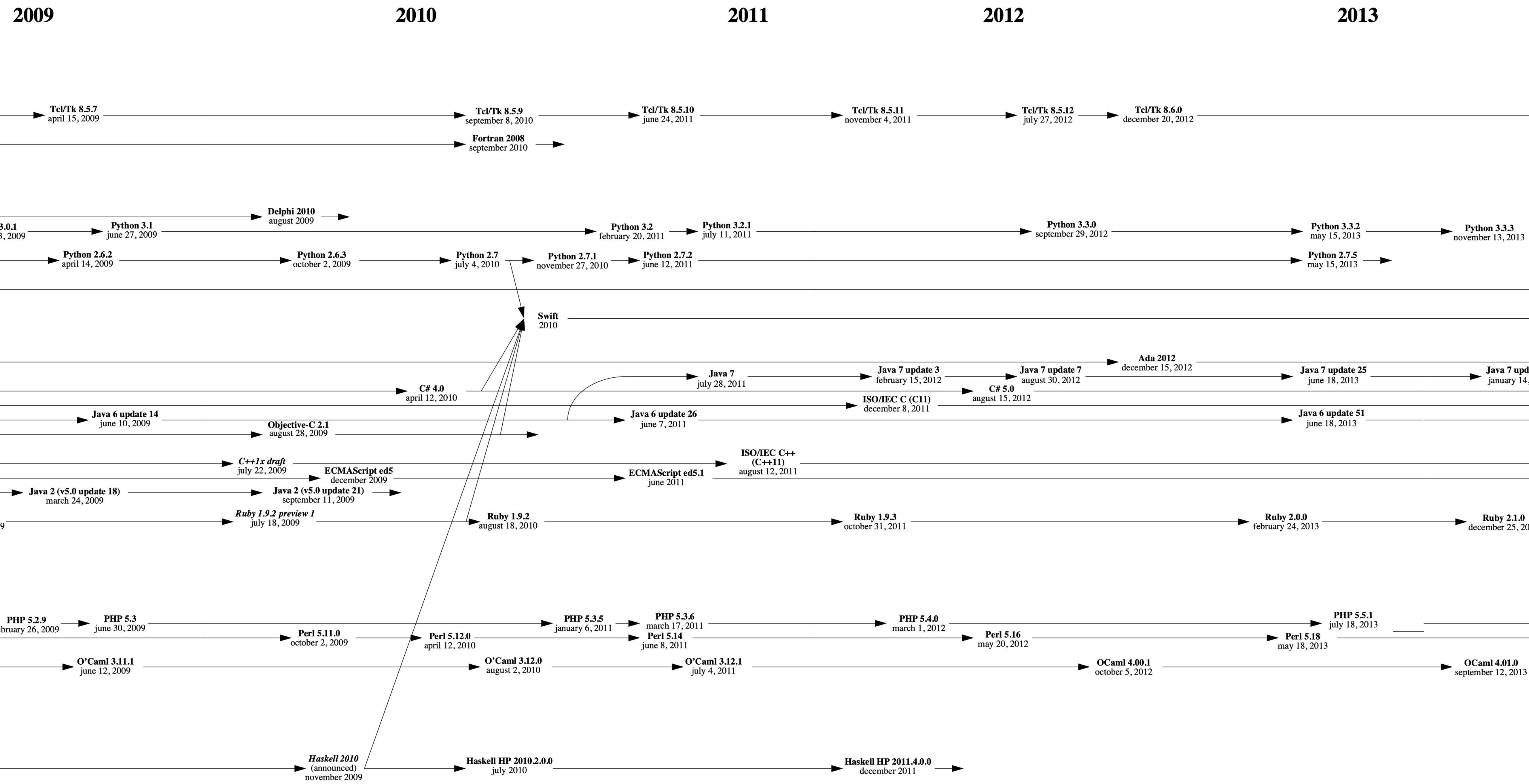
Self 4.3 june 30, 2006 → ECMA Eiffel june 2005 → PHP 4.4.1 october 31, 2005 → PHP 4.4.2 january 13, 2006 → PHP 4.4.4 august 17, 2006 → PHP 4.4.7 may 3, 2007 → PHP 4.4.8 january 3, 2008 → PHP 4.4.9 august 7, 2008 → PHP 5.0.4 july 3, 2005 → PHP 5.0.5 september 6, 2005 → PHP 5.1.0 november 24, 2005 → Perl 5.8.8 february 2, 2006 → O'Caml 3.09.2 april 14, 2006 → PHP 5.1.6 august 24, 2006 → PHP 5.2.0 november 2, 2006 → PHP 5.2.3 may 31, 2007 → PHP 5.2.4 august 30, 2007 → PHP 5.2.5 november 9, 2007 → Perl 5.10 december 18, 2007 → PHP 5.2.6 may 1, 2008 → PHP 5.2.7 december 4, 2008 → PHP 5.2.8 december 8, 2008 → PHP 5.2.9 february 26, 2009 → PHP 5.3 june 30, 2009 → O'Caml 3.10.0 may 16, 2007 → O'Caml 3.10.2 february 29, 2008 → O'Caml 3.11.0 december 4, 2008 → O'Caml 3.11.1 june 12, 2009

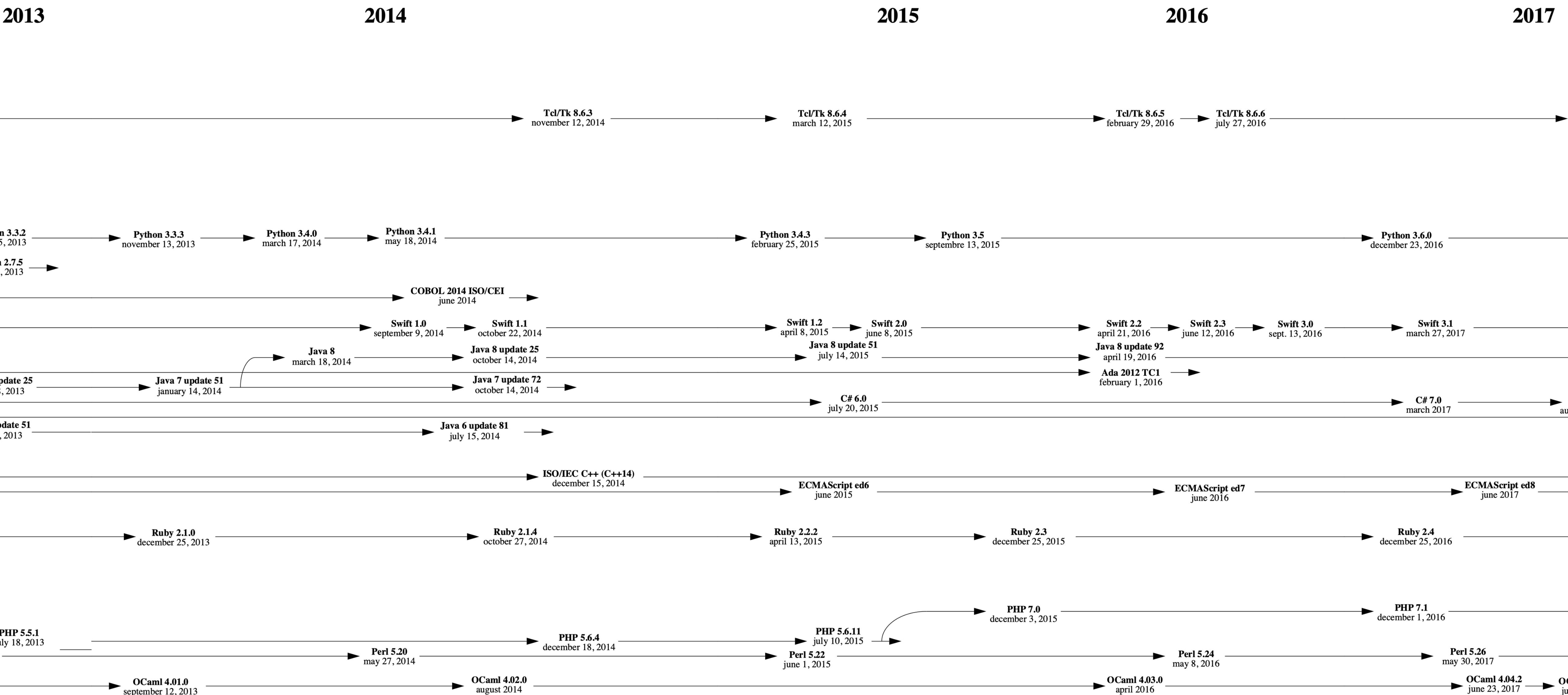
Scheme R<sup>6</sup>RS (draft)  
september 14, 2006

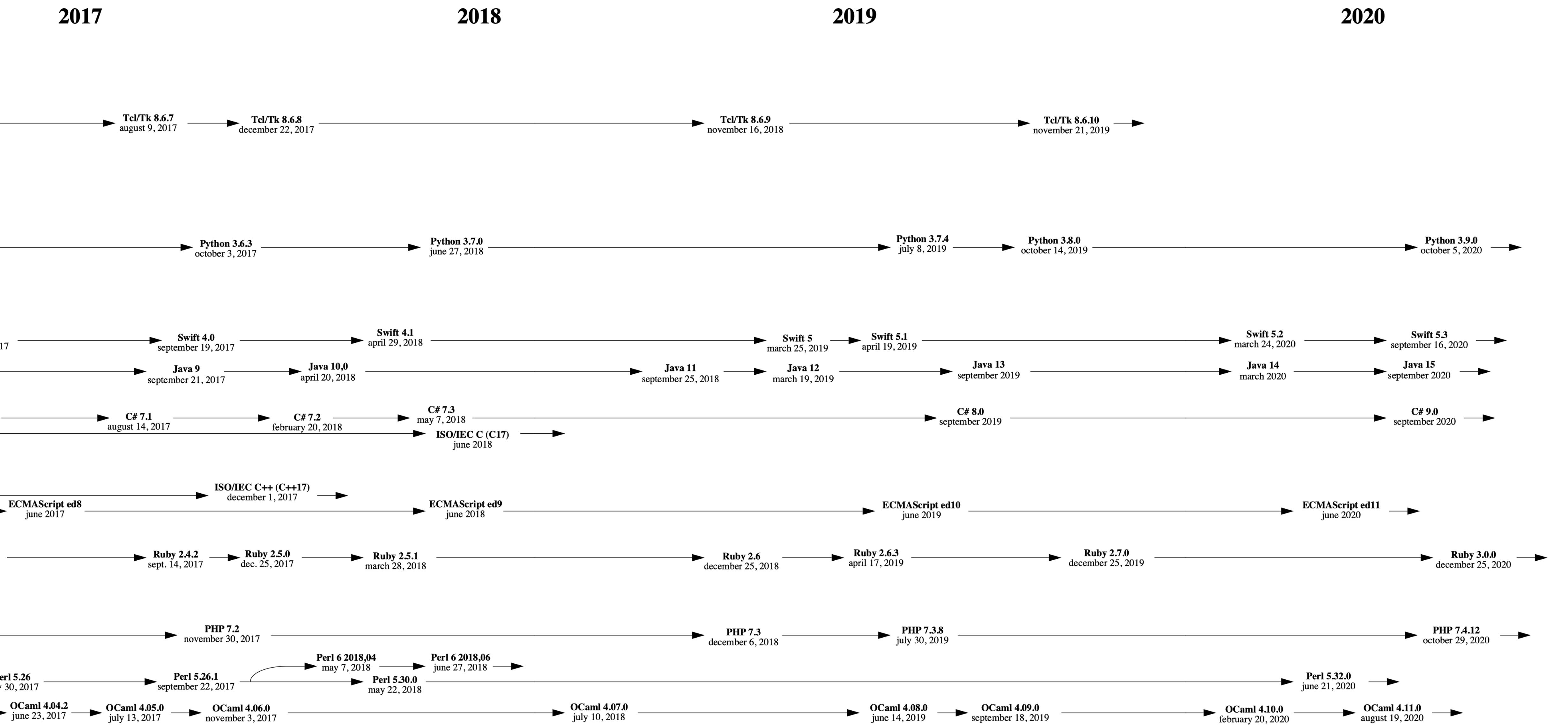
Scheme R<sup>6</sup>RS  
august 28, 2007

2008

2009







# Paradigma de programação

---

- Imperativo (How)
  - Procedimentos
  - Objetos (classes)
  - Variáveis de estado e campos de objetos
- Declarativo (What)
  - Funções
  - Predicados (logic)
  - Sinais (reactive)
  - Eventos
  - Variáveis (no sentido matemático)