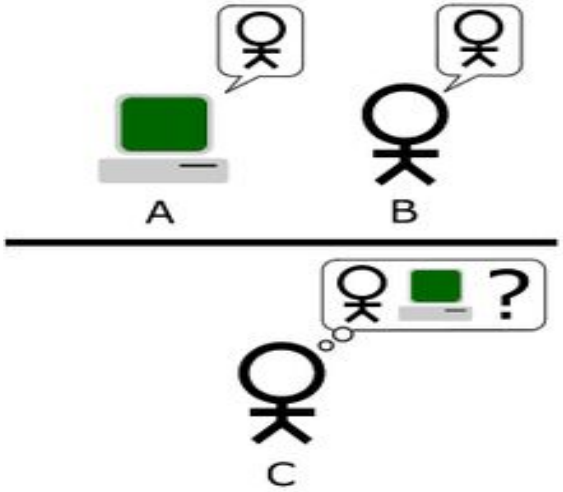
Логическое программирование

Искусственный интеллект

- Тест Тьюринга.



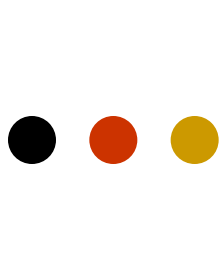
- Проблемы:
  - неоднозначность человеческого языка;
  - при коммуникации мы полагаемся на картину мира, которая у нас в голове (**common knowledge**).

Тест Тьюринга

Эмпирический тест, идея которого была предложена *Аланом Тьюрингом* в статье «Вычислительные машины и разум» (англ. *Computing Machinery and Intelligence*, 1950 г., журнал *«Mind»*). Тьюринг задался целью определить, может ли машина мыслить.

Стандартная интерпретация теста : «*Человек взаимодействует с одним компьютером и одним человеком. На основании ответов на вопросы он должен определить, с кем он разговаривает: с человеком или компьютерной программой. Задача компьютерной программы — ввести человека в заблуждение, заставив сделать неверный выбор*».

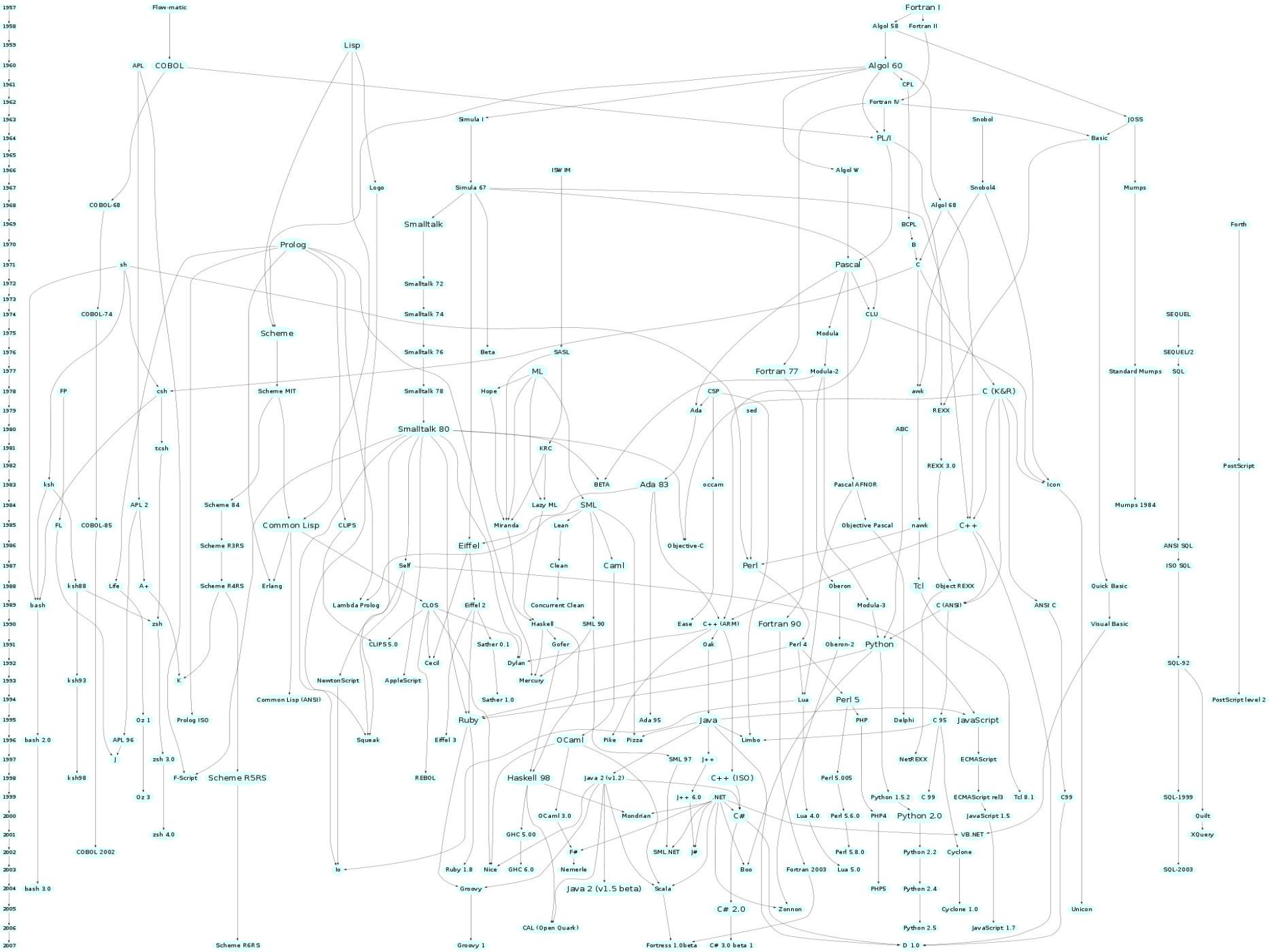
Все участники теста не видят друг друга. Если судья не может сказать определенно, кто из собеседников является человеком, то считается, что машина прошла тест. Чтобы протестировать именно интеллект машины, а не её возможность распознавать устную речь, беседа ведется в режиме «только текст», например, с помощью клавиатуры и экрана (компьютера-посредника). Переписка должна производиться через контролируемые промежутки времени, чтобы судья не мог делать заключения, исходя из скорости ответов. Во времена Тьюринга компьютеры реагировали медленнее человека. Сейчас это правило необходимо, потому что они реагируют гораздо быстрее, чем человек.

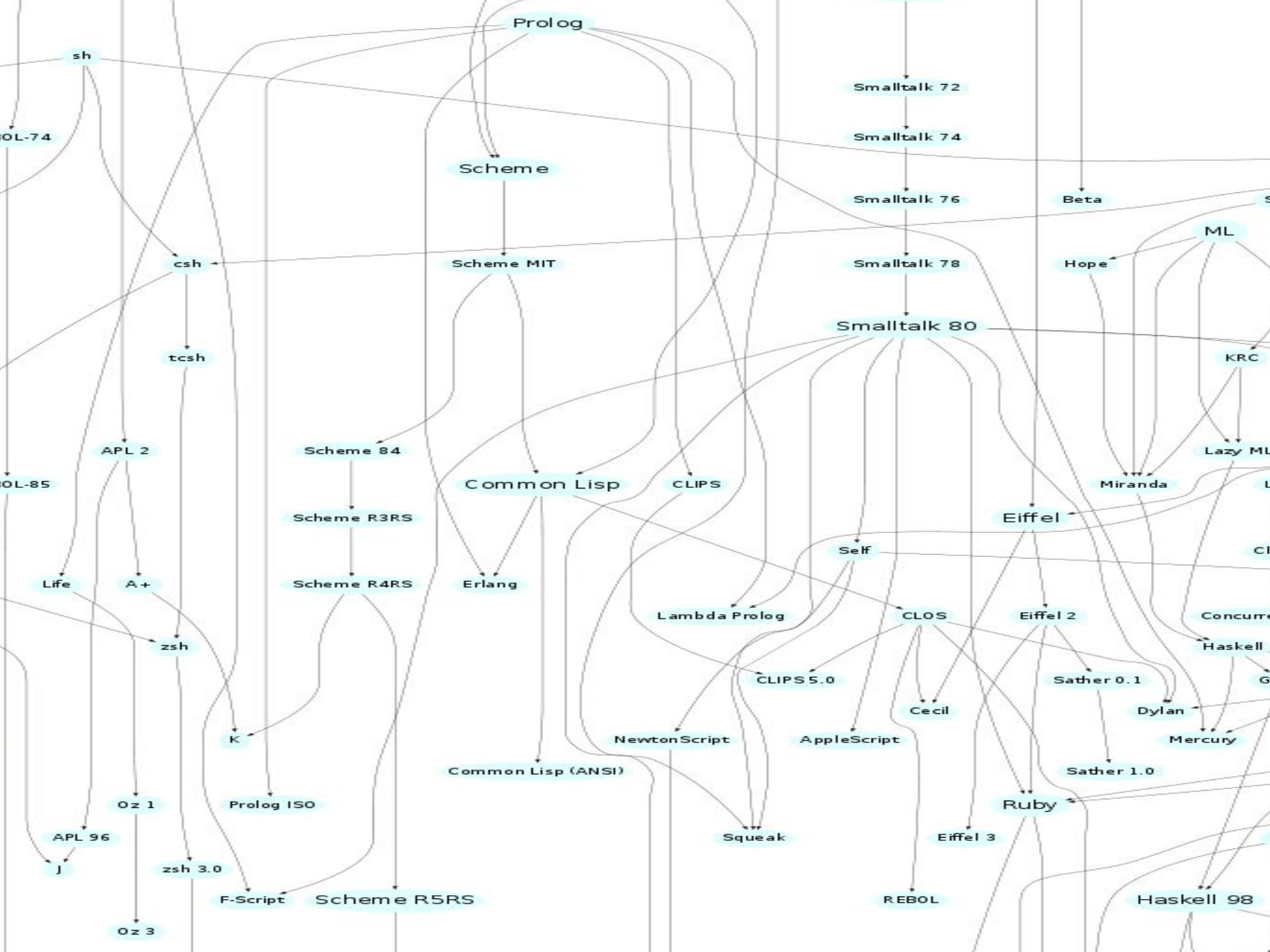


Языки программирования

- Assembler (x86, …)
- C, C++, C#, Java
- Pascal
- …

- LISP, FP, ML, Haskell, OCaml, F#, …
- Prolog, Mercury, Datalog, …

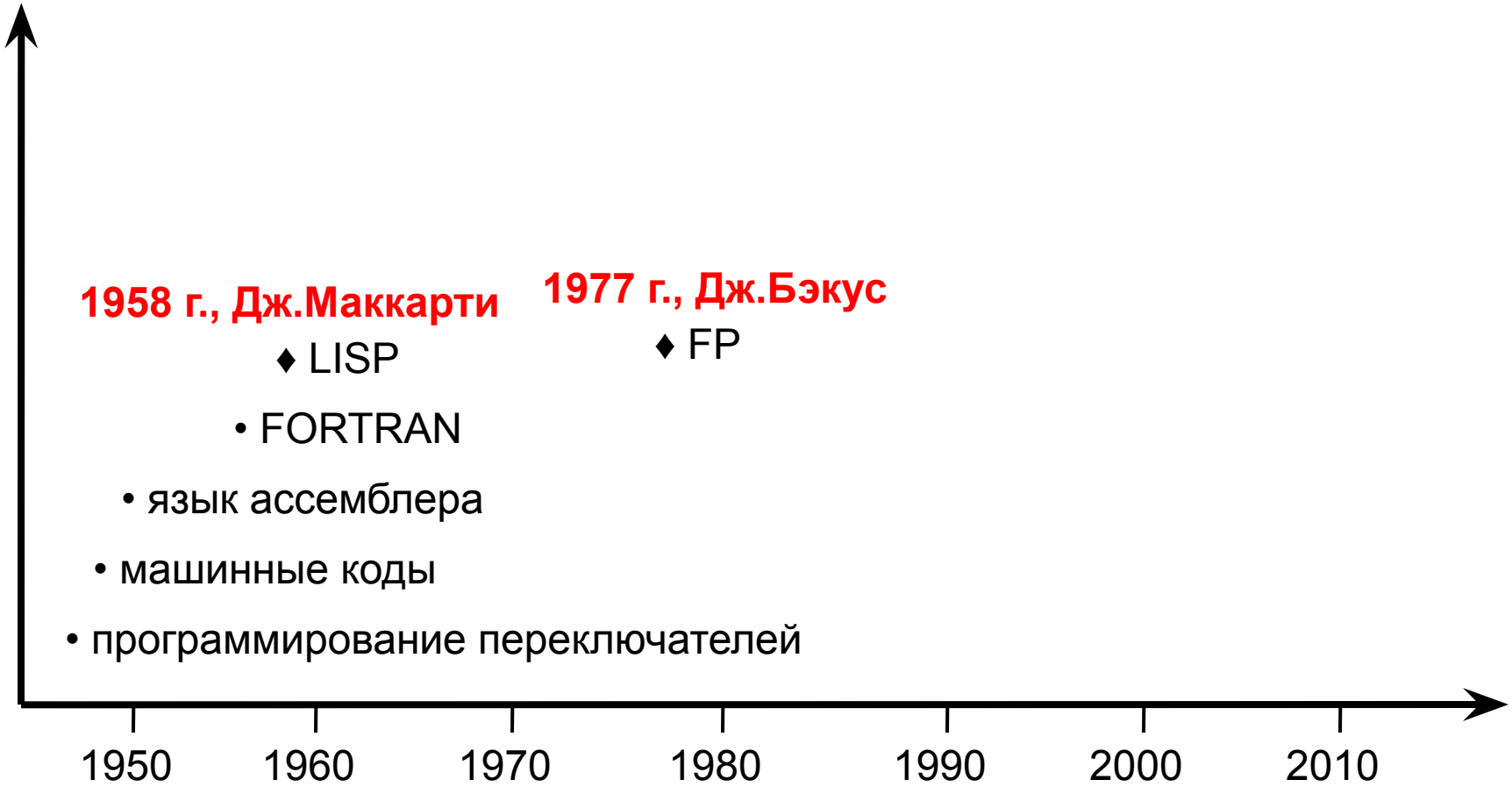

1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
Flow-matic
Fortran I
Algol 58
Fortran II
Lisp
APL
COBOL
Algol 60
CPL
Fortran IV
Simula I
Snobol
JOSS
PL/I
Basic
Algol W
ISWIM
Logo
Simula 67
Snobol4
Mumps
Algol 68
COBOL-68
Smalltalk
BCPL
B
C
Forth
Prolog
sh
Pascal
Smalltalk 72
Smalltalk 74
CLU
SEQUEL
COBOL-74
Scheme
Modula
Smalltalk 76
Beta
SASL
SEQUEL/2
Standard Mumps
SQL
ML
Fortran 77
Modula-2
FP
csh
Scheme MIT
Smalltalk 78
Hope
CSP
awk
C (K&R)
Ada
sed
REXX
Smalltalk 80
ABC
tcsh
KRC
REXX 3.0
PostScript
ksh
BETA
Ada 83
occam
Pascal Distributed
Icon
APL 2
Scheme 84
Lazy ML
SML
Mumps 1984
FL
COBOL-85
Common Lisp
CLIPS
Miranda
Lean
Objective Pascal
nawk
C++
Eiffel
Objective-C
ANSI SQL
Self
Clean
Caml
Perl
ISO SQL
ksh88
Life
A+
Scheme R3RS
Erlang
Oberon
Tcl
Object REXX
Quick Basic
bash
Lambda Prolog
CLOS
Eiffel 2
Concurrent Clean
Modula-3
C (ANSI)
ANSI C
zsh
Haskell
SML 90
Ease
C++ (ARM)
Fortran 90
Visual Basic
CLIPS 5.0
Sather 0.1
Gofer
Oak
Perl 4
Oberon-2
Python
ksh93
K
Scheme R4RS
Cecil
Dylan
SQL-92
NewtonScript
AppleScript
Mercury
Common Lisp (ANSI)
Sather 1.0
Lua
Perl 5
PostScript level 2
Oz 1
Prolog ISO
Ruby
Ada 95
Java
PHP
Delphi
C 95
JavaScript
bash 2.0
APL 96
Squeak
Eiffel 3
OCaml
Pike
Pizza
Limbo
J
SML 97
J++
NetREXX
ECMAScript
ksh98
Scheme R5RS
REBOL
Haskell 98
Java 2 (v1.2)
C++ (ISO)
Perl 5.005
Oz 3
zsh 3.0
F-Script
J++ 6.0
.NET
Python 1.5.2
C 99
ECMAScript rel3
Tcl 8.1
SQL-1999
Quilt
OCaml 3.0
Mondrian
C#
Lua 4.0
Perl 5.6.0
PHP4
Python 2.0
JavaScript 1.5
zsh 4.0
GHC 5.00
VB.NET
XQuery
COBOL 2002
F#
SML.NET
J#
Perl 5.8.0
Python 2.2
Cyclone
Ruby 1.8
Nice
GHC 6.0
Nemerle
Boo
Fortran 2003
Lua 5.0
SQL-2003
bash 3.0
Groovy
Java 2 (v1.5 beta)
Scala
PHP5
Python 2.4
C# 2.0
Zonnon
Cyclone 1.0
Unicon
CAL (Open Quark)
Python 2.5
JavaScript 1.7
Fortress 1.0beta
C# 3.0 beta 1
D 1.0
Groovy 1
Scheme R6RS


Prolog
sh
Smalltalk 72
Smalltalk 74
Scheme
Smalltalk 76
Beta
csh
Scheme MIT
Smalltalk 78
ML
Hope
Smalltalk 80
tcsh
KRC
APL 2
Scheme 84
Common Lisp
CLIPS
Miranda
Lazy ML
Scheme R3RS
Eiffel
Self
Life
A+
Scheme R4RS
Erlang
Lambda Prolog
CLOS
Eiffel 2
Haskell
zsh
CLIPS 5.0
Sather 0.1
Cecil
Dylan
K
NewtonScript
AppleScript
Mercury
Common Lisp (ANSI)
Sather 1.0
Oz 1
Prolog ISO
Ruby
APL 96
Squeak
Eiffel 3
J
zsh 3.0
F-Script
Scheme R5RS
REBOL
Haskell 98
Oz 3

История программирования

- Первый язык программирования высокого уровня – ФОРТРАН – был создан Дж.Бэкусом, чтобы математики могли программировать на уровне формул.

Программирование для математиков



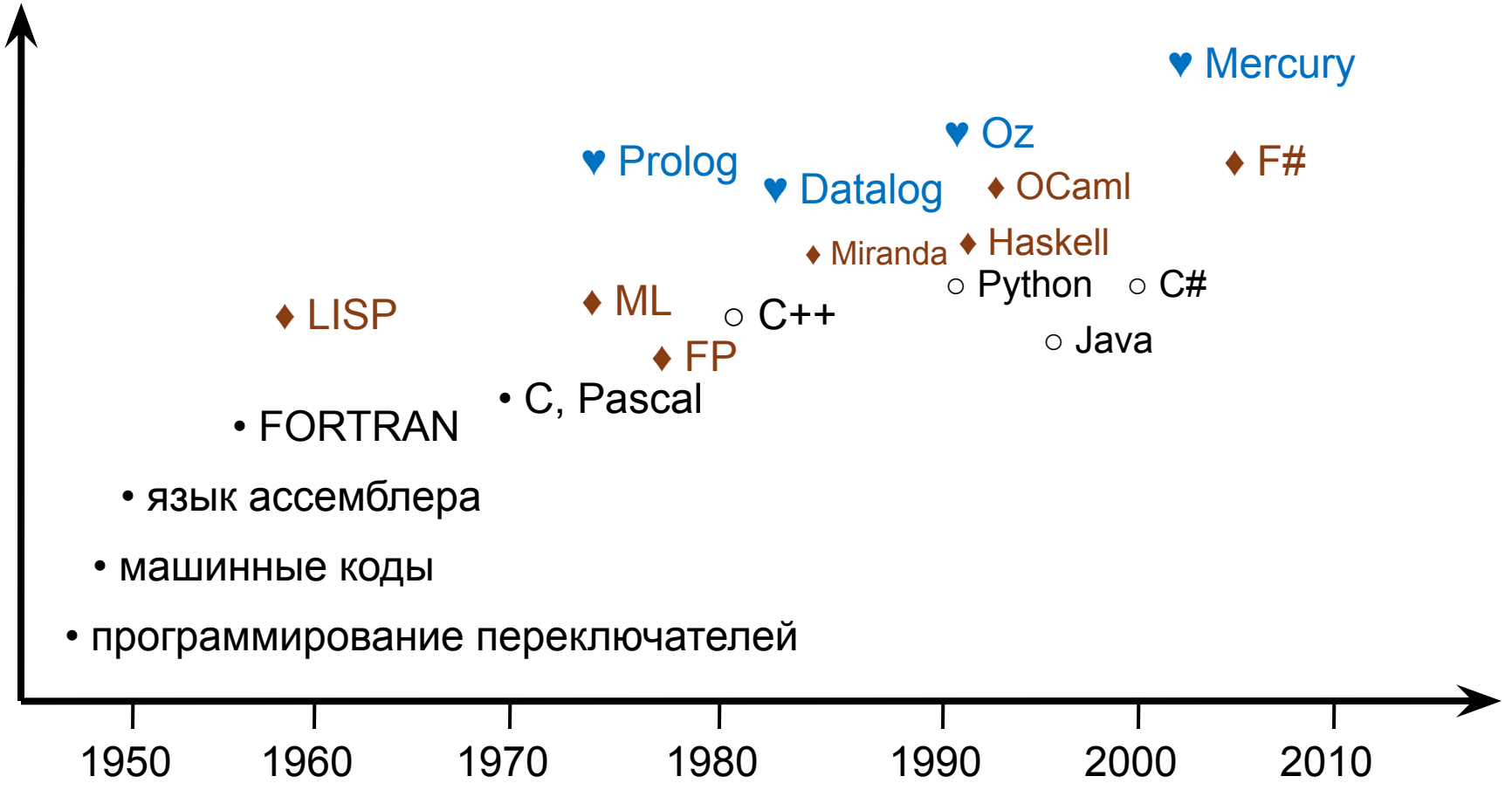
- Позже Дж.Бэкус пошел дальше и предложил язык FP, в котором формулы более соответствовали математическому понятию функции

Как приблизиться к человеческому языку?

- Надо пытаться формализовать человеческий язык

- Основной инструмент формализации:
  - Формальные аксиоматические системы
  - Логика

Языки программирования



Декларативное программирование

При декларативном программировании (на некотором формальном языке) **описываем результат** (его свойства), а не способ его достижения.

Императивное программирование

- Императивное – говорим компьютеру, **как решать задачу** (что делать)
- Основной акцент – **манипулирование ячейками памяти**
  - Оператор присваивания
- **Явные операторы передачи управления**
  - Циклы, условный оператор

Практические преимущества

- Функциональные языки
  - Компактный синтаксис для списков, кортежей, вариантных типов
- Логические языки
  - Компактный синтаксис для списков, кортежей, вариантных типов
  - Возможность перебора и поиска различных решений, заложенная в язык

Что особенного?

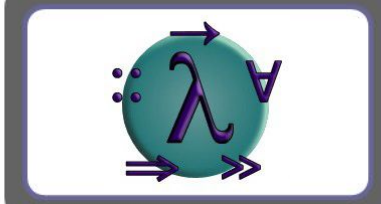
- Определения на логическом языке похожи на предложения математической логики
  - Логическое программирование имеет очень четкую математическую основу
  - Возможны рассуждения о программах: доказательство корректности, …
- **Отсутствует оператор присваивания**
  - Есть знак = , но он имеет другую семантику – унификация, связывание имен
  - Переменные связываются неявно, в процессе логического вывода
  - Будучи один раз связанным, имя может менять свое значение только в процессе пересмотра решения (возврата)

Парадигмы программирования

Шифрующие роторные машины. Слева направо: ENIGMA (Германия), SIGABA (США), PURPLE (Япония).

Императивное (алгоритмическое)

- Машина Тьюринга, Машина фон Неймана
- Pascal, C и т.д.



Аппликативное (функциональное)

- λ-исчисление, рекурсивные функции
- F#, LISP / Scheme, ML и друзья, Haskell

Декларативное (логическое)

- Логика предикатов 1-го порядка
- Prolog, Mercury, Oz, …

Ситуационное (продукционное)

- Нормальные алгоритмы Маркова
- Рефал

Объектное, компонентное, многоагентное (эмерджентое)

- Синергетика, теория сложных систем

Семантика языков

Императивные языки

- Оперируют состоянием памяти. Выполнение операторов изменяет состояние.

Функциональные языки

- Оперируют данными. Применение функции к аргументам изменяет данные.
- Подход, ориентированный на данные.

Логические языки

- Оперируют пространством поиска решений.
- Программа задаёт множество возможных переходов в пространстве поиска.

Мультипарадигмальные языки

- C# - императивный (ОО) + элементы функциональности
- F# - функциональный с элементами императивности
- Mercury – функционально-логический
- Oz
- Python
- …

Источники

- **Братко И.** Программирование на языке Пролог для искусственного интеллекта. пер. с англ. – М.: Мир, 1990.
- Bratko I. Programming in Prolog for Artificial Intelligence (3rd edition), Addison-Wesley Publishers, 2001.
- **Клоксин У., Меллиш К.** Программирование на языке Пролог. – М.: Мир, 1987.
- Хоггер К. Введение в логическое программирование: Пер. с англ. -М.: Мир, 1988.
- Набебин А.А. Логика и Пролог в дискретной математике. – М.: Изд-во МЭИ, 1996.
- Малпас Дж. Реляционный язык Пролог и его применение: Пер. с англ. -М.: Наука, 1990.
- Стерлинг Х., Шапиро Э. Искусство программирования на языке Пролог: Пер. с англ. - М.: Мир, 1990.