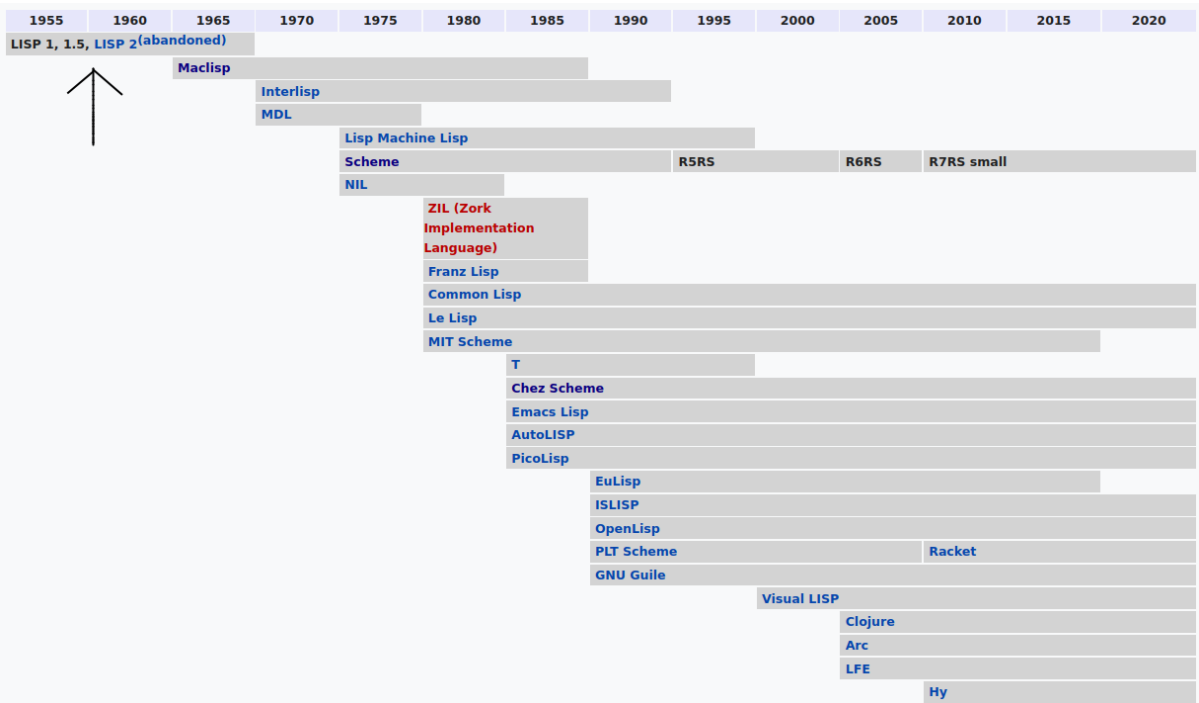
A TALE OF LISP

FROM LISP TO CLOJURE (AND BACK)

JUXT CONF 2022 - HORWOOD HOUSE

=> EARLY LISP 1955-1960



ONCE UPON A TIME, IN 1950

- John McCarthy (1927-2011)
- Ph.D. mathematics from Princeton 1951
- Then assistant professor at Dartmouth, 1955
- Soon after researcher at MIT, 1956

WHAT IS INTELLIGENCE?

- Hot topic in the ’50
- Shannon’s first chess program 1949
- Alan Turing’s Computing Machinery and Intelligence 1950
- Reasoning automation with the Logic Theorist 1955

“ARTIFICIAL INTELLIGENCE” STUDY PROPOSAL

A Proposal for the

DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE

June 17 - Aug. 16

We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

GOALS

“ The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it.
[…] It therefore seems to be desiderable to attempt to construct an artificial language which a computer can be programmed to use on problems and self-reference.

DESIRED FEATURES

- Abstracted away from hardware constraints
- Not much larger than the sentence it describes
- Allowing representable logical rules
- Enabling symbolic computations (not just numbers)

REALITY CHECK: PROGRAMMING IN 1958

- Fortran: symbolic, arithmetic
- IPL: list based, low level
- F-LPL: Fortan-compiled for list processing
- Algol: McCarthy tried to influence unsuccessfully

Too restrictive, in one way or another

THE PERFECT STORM

- September 1958 funding was granted
- Creation of a new MIT AI Lab with the following
  - 2 rooms
  - 2 programmers, 6 students
  - 1 secretary, 1 typewriter
  - Some limited CPU time

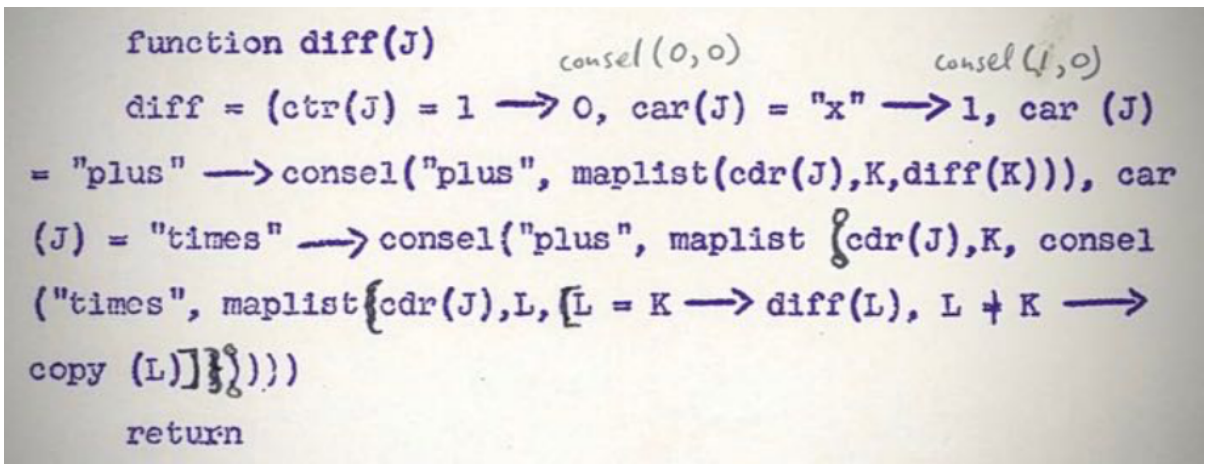
REALITY CHECK: IBM 704

CAR, CDR AND FRIENDS

Part	Name	Bits	Fn
w	whole-word	0-35	cwr
p	prefix	0-1-2	cpr
i	indicator	1-2	cir
s	sign-bit	0	csr
d	decrement	3-17	cdr
t	tag	18-20	ctr
a	address	21-35	car

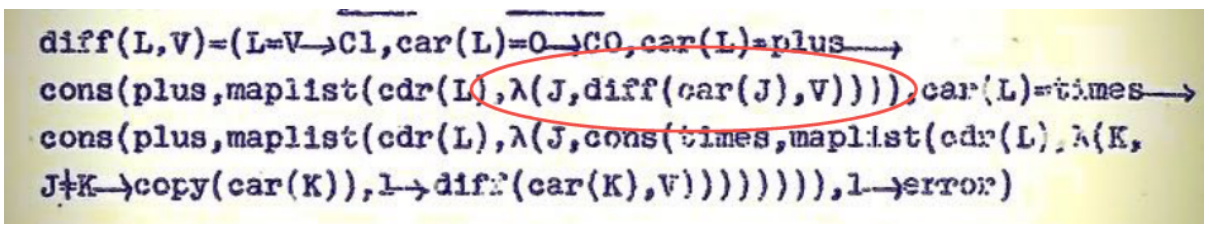
C (content of the) **R** (egistry)

EARLY ATTEMPTS



- APL-ish single liner
- Algol like, not homoiconic
- “K”: a dummy var to iterate memory addresses

LAMBDA TO THE RESCUE



- Inspiration coming from Church's Lambda Calculus
- **λ** now allow `maplist` to declare the meaning of "J"
- No mutation, `maplist` gets fresh elements from storage

Still not very lispy!

TURNING POINT: S-EXPRESSIONS

- Challenge: demonstrate Turing completeness
- Proof: feed Lisp to another Lisp
- Alternative syntax required for easier “punching”
- Extend `apply` to receive Lisp forms as lists
- `eval` would need to interpret 6-7 special forms

AND THEN STEVE RUSSELL CAME AROUND

- *Steve:* “Why don’t we write this apply-eval down to assembly?”
- *McCarthy:* “ho, ho, you’re confusing theory with practice… this `eval` is intended for reading not for computing.”

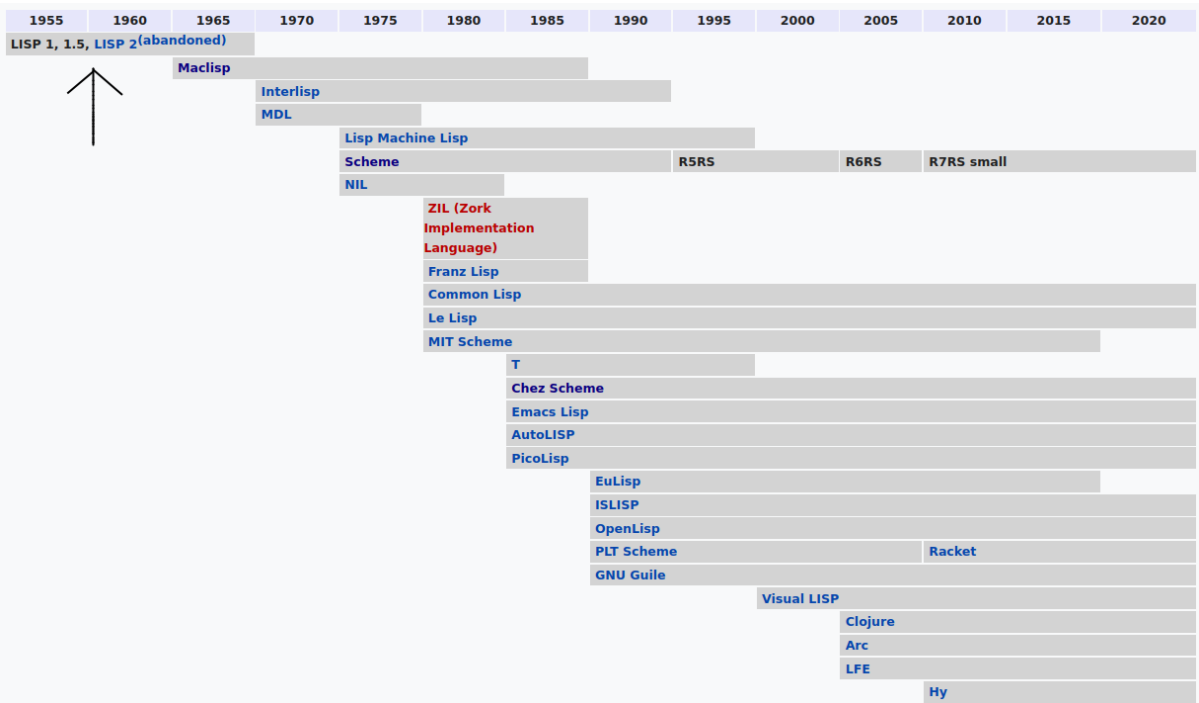
EVAL <=> APPLY

LISP 1 WAS BORN

LESSON LEARNED

- Well defined goals
- Baby steps, small increments, refined versions
- If the alternative doesn’t work, make your own
- Constrained resources => more time to think
- Team interplay
- Ultimately, no fear!

=> FROM LISP TO COMMON LISP 1960-1990



AFTER LISP 1

- It spread rapidly to a variety of machines
- Lisp 1.5 came out with minor improvements, 1962
- PDP-1 assembly porting by Peter Deutch, 1963
- Any AI lab with a Lisp wizard down the hall…
- We'll go fast and touch a few

MACLISP 1966-1990

- Targeting PDP-6-10, a production grade Lisp
- Lisp implementation of Project MAC (time-sharing)
- Consolidation of implementation techniques
- Small core written in Assembly, Lisp libraries on top

INTERLISP 1968-1990

- Born as BBN Lisp, by Bobrow and al.
- Emphasis on user interaction
- IDE, debuggers, auto-completion, spell checking
- Infinite login sessions: moving away from files
- Ended up at Xerox, inspiring Smalltalk
- Microcoded for the Alto by Deutch (see next)

XEROX ALTO

LISP MACHINES

- First serious proposal by Deutch 1973
- Microcoded for Lisp efficiency
- The first was the MIT Lisp machine of 1974
- Large bitmapped display inspired by the Alto
- Many industrial grade changes to Lisp

SYMBOLICS 3360

LISP MACHINES INNOVATIONS

- IDE-like environments, in the Interlisp spirit
- `defmacro` and syntax quoting
- `defstruct` record definition facility
- `format`
- Flavours object oriented system
- Moore's law made them obsolete in the early '90

SCHEME

- Created to study aspects of the actor model, 1975
- Lexical closures were unusual and avoided in Lisp
- First-class continuations was necessary for actors
- TCO always possible thanks to continuations
- `?` or `!` for predicates and side effects
- Minimalism and centralized specifications

LISP PROBLEMS

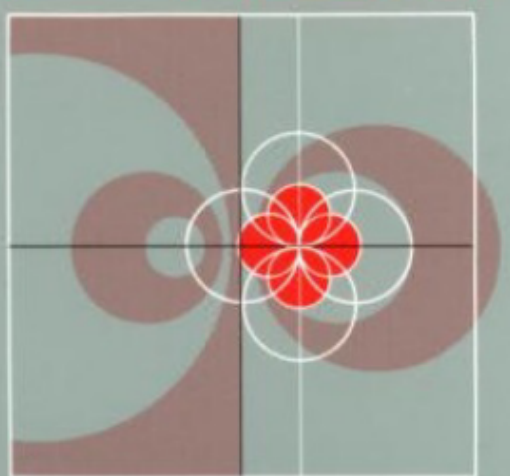
- Still a “kludge” allowing many styles
- Many dialects and diverging philosophies
- Multiple Lisp machines from different vendors
- A desperate need for standardization

COMMON LISP

- An initiative started in the early ’80
- Create a new Lisp with all the best features
- Initial discussions between MacLisp and Interlisp
- Priority given to Symbolics (largest money maker)
- A well delivered compromise, overall

GUY L. STEELE JR.

COMMON LISP



THE LANGUAGE

SECOND EDITION

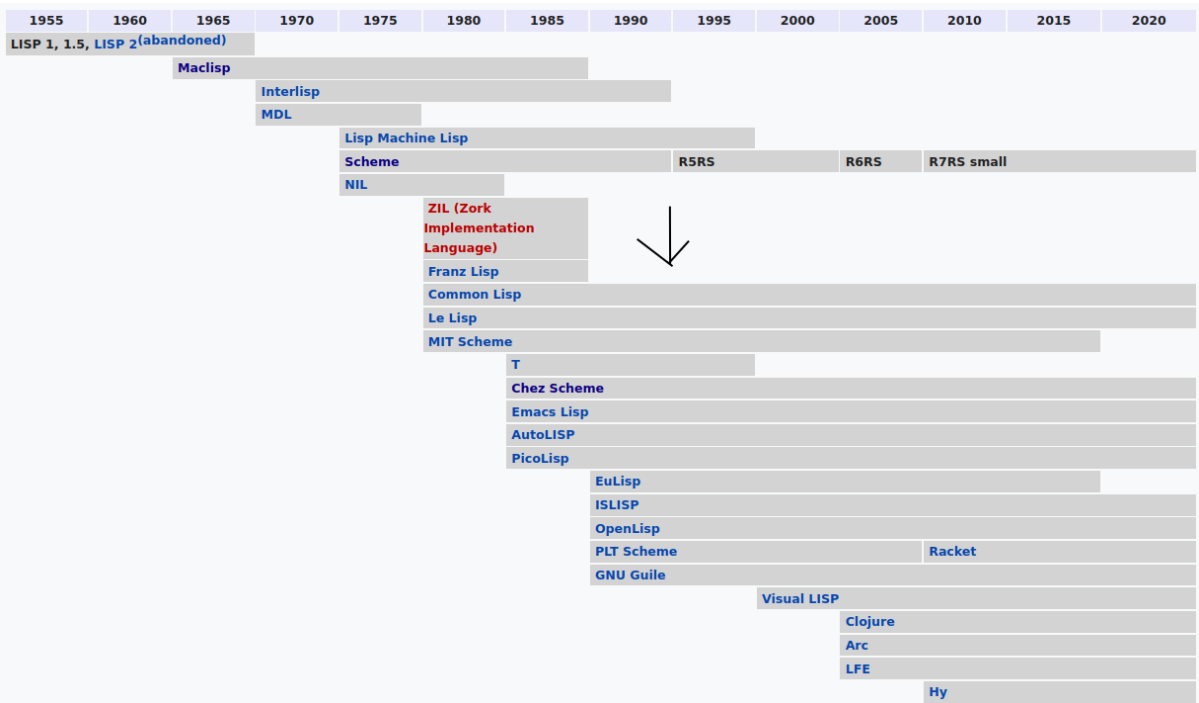
THE GOLDEN ERA 1984-1988

- AI was on the rise, Lisp was the language of AI
- Expert systems was the “killer app”
- The allure of AI attracted business and VCs
- Lisp was granted unprecedented popularity
- Lisp companies thrived thanks to quality and performance

LESSON LEARNED

- Importance of theoretical foundations
- Expressiveness freedom from low level details
- Flexible extensions similar to the core language
- Interactive and incremental nature
- Pivotal to fundamental innovation

=> FROM CL TO CLOJURE 1990 - 2007



THE AI WINTER, 1990+

- In typical bubble style, AI was over hyped
- Lisp machines power gradually eroded by Moore's law
- Large failures like the Fifth Generation System
- Self-Blaming Expert Systems as non-AI

TIMES ARE CHANGING

- Advent of the web: shift from personal to distributed
- Object Oriented Programming
- C++ (1985) Python (1991) Java (1995) JavaScript (1996)
- Software industry exponential growth
- But in general, a decrease in popularity for Lisp

WHAT MAKES A LANGUAGE POPULAR?

- free, efficient, expressive
- documented, beginner friendly, hackable
- scripting language for a popular OS or browser
- language distributed by large vendor
- Has a “killer app”
- Pivotal to fundamental innovation
- Batteries included (json, web, etc.)
- Backward compatible and stable
- Marketing and money

THE BUILD UP TO CLOJURE, 2003-2006

- Rich worked on a couple of large CL applications (2001)
- DotLisp (interpreted Lisp for the CLR), 2003
- jFli (embedded JVM in CL), 2004
- Foil (JVM to CL via IPC), 2005
- March 2006, Clojure started as a Lisp compiler

CLOJURE RATIONALE

- OOP makes things very complicated to reason about
- The JVM and the CLR are widely distributed
- Moore’s law plateau and multi-core CPUs
- In search for a new Lisp experience
- Immutability to tame concurrency

THE REST IS HISTORY!

- 2017, Conj Baltimore, 10th anniversary
- 2020, Clojure accepted at HOPL IV
- 2022, Milton Keynes 15th anniversary and counting!

IS CLOJURE POPULAR?

- ☑ free, efficient, expressive
- [?] documented, beginner friendly, hackable
- ☐ scripting language for a popular OS or browser
- ☐ language distributed by large vendor
- ☐ Has a “killer app”
- ☐ Pivotal to fundamental innovation
- ☐ Batteries included (json, web, etc.)
- ☑ Backward compatible and stable
- ☐ Marketing and money

DOES IT NEED TO BE POPULAR?

“ Popularity further separates good languages from bad ones, because feedback from real live users always leads to improvements. So whether or not a language has to be good to be popular, I think a language has to be popular to be good. And it has to stay popular to stay good.

Paul Graham, Hackers and Painters, 2010

IS THERE HOPE?

- Of course!
- Human beings are short lived
- Knowledge persists through books, media, etc.
- But humans forget, distort, make mistakes
- As a consequence, history repeats
- Learn the past to shape the future

WHAT CAN I DO?

- Tell the Lisp tale to your friends
- Talk at conferences and meetups
- Write blogs, libraries, books
- Let's "program" the next fundamental change in society
- Never stop learning!

~ fin ~