Lisp: Background and History

CSE 410/510 ETH: Interactive Programming Environments

Ethan Blanton

Department of Computer Science and Engineering

University at Buffalo

Lisp Meets World

John McCarthy introduced Lisp in 1960. [6]

It had been under development since 1958.

This Lisp looked a whole lot like modern Lisp!

It introduces:

- CAR, CDR, CONS
- ATOM
- QUOTE
- EVAL, APPLY
- COND
- …

Architectural Influence

The first Lisp was on the IBM 704.

Lawrence Livermore National Laboratory

This (surprisingly?) had a large influence on its vocabulary.

The IBM 704

The IBM 704 was a vacuum tube computer.

It had 15 bits of 36-bit words in core memory.

Two values would fit in a word: an offset and a base address. [1]

Instructions called these the decrement and the address.

- CAR: Contents of Address part of Register
- CDR: Contents of Decrement part of Register

[1] Not quite, but close enough for now.

Lisp 1.5

Lisp 1.5 was released in 1962.[2]

It was the first Lisp available widely outside of MIT.

It would ultimately run on several machines and architectures.

Lisp 1.5 programs were punch card programs on mainframes.

The real influence of Lisp 1.5 was perhaps its programmer's Manual.

[2] ish?

Maxwell's Equations of Software

The Lisp 1.5 manual introduced the metacircular evaluator.

This is an implementation of Lisp in Lisp in half a page of text.

Alan Kay says [3]:

> *Yes, that was the big revelation to me[—]when I finally understood [that half a page of Lisp] was Lisp in itself. These were "Maxwell's Equations of Software!" This is the whole world of programming in a few lines that I can put my hand over.*

This one concept has inspired and re-inspired languages and programmers.

Lisp on the PDP-1

L. Peter Deutsch implemented Lisp on the PDP-1 in 1963. [5]

It was the first interactive lisp.

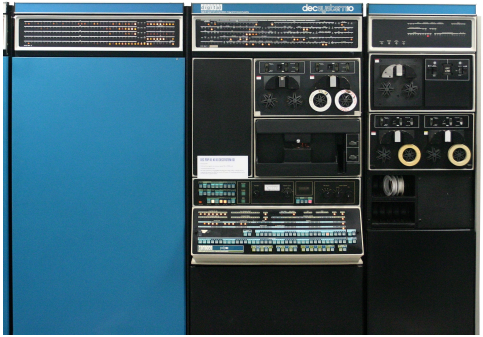
> *It is convenient to define functions, test them, and re-edit them without ever leaving the LISP interpreter. [...] [Deutsch] implemented the first interactive LISP [...], but the PDP-1 had too small a memory for serious symbolic computation.*

PDP-1 Lisp introduced the read-eval-print loop (REPL). [2]

It went on to be hugely influential.

Lisp on the PDP-6/10

In the mid-1960s, Lisp moved to the PDP-10.



by Gah4, https://creativecommons.org/licenses/by-sa/4.0/deed.en

Interactive Lisps

Two influential, interactive Lisps were developed on the PDP-10:

- BBN Lisp (which became Interlisp)
- MacLisp (which became Lisp Machine Lisp)

These Lisps provided:

- Both interpreted and compiled Lisp
- Garbage collection
- Interactive editing of Lisp code
- Interactive debugging of Lisp programs

Divergence

Prior to MacLisp and BBN Lisp, most Lisps were compatible.

These Lisps caused significant divergence. [7]

BBN Lisp begat Interlisp.

MacLisp begat Lisp Machine Lisp.

Common Lisp later tried to unify the various Lisps.

MacLisp

MacLisp is probably the most influential Lisp-2.[3]

It was from Project MAC at MIT.

It was ultimately ported to many systems.

Multics MacLisp greatly expanded memory allowance of Lisp.
(The PDP-10 allowed only 18 bits of 36-bit words per program.)

The first Lisp Emacs was implemented in Multics MacLisp.

[3] More on Lisp-1 vs Lisp-2 later!

Namespaces

Lisp 1.5 had two namespaces: code and data.

A symbol could be represent both a function and a value.

This has several implications:

- Simplified naming
- Additional syntax for higher-order functions

(Its implementation also had incidental benefits.)

Lisp-1 vs Lisp-2

In 1975, Sussman and Steele introduced Scheme. [8]

One of the primary distinctions of Scheme was:

> *LAMBDA expressions need not be QUOTEd, FUNCTIONed, or *FUNCTIONed when passed as arguments or returned as values; they will evaluate to closures of themselves.*

This introduced a Lisp with a single namespace, or Lisp-1.
(the -1 and -2 refer to the number of namespaces)

Scheme

Scheme tried to be closer to the lambda calculus.

Lambdas or named functions can be passed just like values.

Any symbol can have arguments applied like a function.

The type system will sort out violations.

Iteration is expressed as application of functions!

Iteration in Scheme

```
(define (factorial n)
  (let do-factorial ((n n) (c 1))
    (if (= n 0)
        c
        (do-factorial (- n 1) (* c n)))))
```

(The original syntax was somewhat more cumbersome.)

Dedicating Hardware

The late 1970s saw a dearth of promising Lisp platforms. [7]

The problem was hardware suitability: memory and speed.

By 1980, dedicated Lisp machines were becoming available.

These machines typically used microcode to assist important operations.

The MIT Lineage

MIT built a prototype machine called CONS, followed by CADR.

The CADR was a local success, running Lisp Machine Lisp.

Lisp Machine LISP was similar to MacLisp. [7]

Two companies were formed to commercialize the CADR:

- Symbolics
- Lisp Machines, Inc. (LMI)

Symbolics Legacy

The Symbolics machine had modifiers like Meta and Super

Outside of MIT

Xerox PARC experimented with Lisp on the Alto.

This led to the “D-machines”: Dolphin, Dorado, Dandelion, …

These machines ran a re-implementation of Interlisp. [1]

Texas Instruments shipped the Explorer.

The Explorer ran a version of the CADR system from LMI.

Standardization

The fragmentation of MacLisp led to a desire to standardize.

Common Lisp intended to provide [4]:

- Commonality (among implementations)
- Portability (between implementations)
- Consistency (between interpretation and compilation)
- Expressiveness
- Compatibility (with Lisp Machine Lisp, MacLisp, and Interlisp)
- Efficiency
- Power
- Stability (from standard version to standard version)

Stagnation

Common Lisp was standardized by ANSI in 1994.

It was envisioned that it would evolve over time. [4]

In practice, it has never been updated.

However, Lisp is very high level and the standard forward-thinking.

It has aged reasonably well.

Implementations

There are many implementations of Common Lisp.

Complete, free implementations include:

- Steel Bank Common Lisp (SBCL)
- Embeddable Common Lisp (ECL)
- GNU Common Lisp (GCL)
- Armed Bear Common Lisp (ABCL)
- CMU Common Lisp (CMUCL)

Integrations and Extensions

Common Lisp has a large ecosystem, including:

- Package management (Quicklisp, Roswell, ASDF)
- Graphics toolkits (McCLIM, Qt, Gtk+)
- Editor integrations (SLIME, Lem, not-Emacs editors (?))
- Web services (Hutchentoot)

Many programming paradigms are built-in or readily available:

- Functional programming
- Object orientation (the Common Lisp Object System)
- Actor models, channels, promises, …

Trailblazing and Influence

Lisp lays claim to a number of “firsts” in computing.

- The REPL (as READ-EVAL-PRINT cycle) [2]
- First compiler written in the target language [5]
- First program written on the PDP-6 [7]
- Undo and Redo [9]
- Structural Editing [9]
- …

Modern Lisps

There are notable modern Lisps.

Clojure uses the JVM and integrates easily with Java libraries.

Racket is a scheme popular in PL research and education.

Hy compiles to Python IL and integrates with Python libraries.

Fennel compiles to Lua and runs on the Lua VM.

Lisp Systems

There are several newer Lisp operating systems.

Mezzano is a 64-bit Common Lisp OS.

ChrysaLisp is a VM-based Lisp Machine OS.

uLisp runs on microcontrollers and embedded systems.

Emacs

GNU Emacs is, of course, basically a Lisp Machine.

Emacs Lisp emulates many aspects of MacLisp.

Lisp applications can be written in Emacs.

Modern Emacs provides:

- Interpreted, bytecode, or native code Lisp execution
- Sockets, HTTP, other network protocols
- sqlite3 database integration
- A vast package ecosystem

References I

Optional Readings

[1] Richard R. Burton et al. “Overview and Status of DoradoLisp”. In: *Proceedings of the 1980 ACM Conference on LISP and Functional Programming*. Association for Computing Machinery, 1980, pp. 243–247.

[2] L. Peter Deutsch and Edmund C. Berkeley. *The LISP Implementation for the PDP-1 Computer*. Mar. 1964.

[3] Stuart Feldman. “A Conversation with Alan Kay. Big talk with the creator of Smalltalk—and much more”. In: *Queue* 2.9 (Dec. 2004), pp. 20–30.

[4] Guy L. Steele Jr. *Common Lisp: The Language*. Second. Digital Press, 1990.

[5] John McCarthy. “History of Lisp”. In: *ACM SIGPLAN Notices* 13.8 (Aug. 1978), pp. 217–223.

[6] John McCarthy. “Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I”. In: *Communications of the ACM* 3.4 (Apr. 1960), pp. 184–195.

References II

[7] Guy L. Steele and Richard P. Gabriel. “The Evolution of Lisp”. In: *History of Programming Languages II*. Association for Computing Machinery, 1996, pp. 233–330.

[8] Gerald Jay Sussman and Guy Lewis Steele Jr. *Scheme. An Interpreter for Extended Lambda Calculus*. Tech. rep. AI Memo 349. AI Laboratory of MIT, Dec. 1975.

[9] Warren Teitelman. “PILOT: A Step toward Man-Computer Synthesis”. PhD thesis. Massachusetts Institute of Technology, Sept. 1966.

License

To retrieve a copy of this material, or related materials, see `https://www.cse.buffalo.edu/~eblanton/`.