History of the Lisp Language

The following information is derived from the history section of dpANS Common Lisp. Lisp is a family of languages with a long history. Early key ideas in Lisp were developed by John McCarthy during the 1956 Dartmouth Summer Research Project on Artificial Intelligence. McCarthy's motivation was to develop an algebraic list processing language for artificial intelligence work. Implementation efforts for early dialects of Lisp were undertaken on the IBM 704, the IBM 7090, the Digital Equipment Corporation (DEC) PDP−1, the DEC PDP−6, and the PDP−10. The primary dialect of Lisp between 1960 and 1965 was Lisp 1.5. By the early 1970's there were two predominant dialects of Lisp, both arising from these early efforts: MacLisp and Interlisp. For further information about very early Lisp dialects, see *The Anatomy of Lisp* or *Lisp 1.5 Programmer's Manual*.

MacLisp improved on the Lisp 1.5 notion of special variables and error handling. MacLisp also introduced the concept of functions that could take a variable number of arguments, macros, arrays, non−local dynamic exits, fast arithmetic, the first good Lisp compiler, and an emphasis on execution speed. For further information about Maclisp, see *Maclisp Reference Manual, Revision 0* or *The Revised Maclisp Manual*.

Interlisp introduced many ideas into Lisp programming environments and methodology. One of the Interlisp ideas that influenced Common Lisp was an iteration construct implemented by Warren Teitelman that inspired the **loop** macro used both on the Lisp Machines and in MacLisp, and now in Common Lisp. For further information about Interlisp, see *Interlisp Reference Manual*.

Although the first implementations of Lisp were on the IBM 704 and the IBM 7090, later work focussed on the DEC PDP−6 and, later, PDP−10 computers, the latter being the mainstay of Lisp and artificial intelligence work at such places as Massachusetts Institute of Technology (MIT), Stanford University, and Carnegie Mellon University (CMU) from the mid−1960's through much of the 1970's. The PDP−10 computer and its predecessor the PDP−6 computer were, by design, especially well−suited to Lisp because they had 36−bit words and 18−bit addresses. This architecture allowed a cons cell to be stored in one word; single instructions could extract the car and cdr parts. The PDP−6 and PDP−10 had fast, powerful stack instructions that enabled fast function calling. But the limitations of the PDP−10 were evident by 1973: it supported a small number of researchers using Lisp, and the small, 18−bit address space (262,144 36−bit words) limited the size of a single program. One response to the address space problem was the Lisp Machine, a special−purpose computer designed to run Lisp programs. The other response was to use general−purpose computers with address spaces larger than 18 bits, such as the DEC VAX and the S−1 Mark IIA. For further information about S−1 Common Lisp, see ''S−1 Common Lisp Implementation.''

The Lisp machine concept was developed in the late 1960's. In the early 1970's, Peter Deutsch, working with Daniel Bobrow, implemented a Lisp on the Alto, a single−user minicomputer, using microcode to interpret a byte−code implementation language. Shortly thereafter, Richard Greenblatt began work on a different hardware and instruction set design at MIT. Although the Alto was not a total success as a Lisp machine, a dialect of Interlisp known as Interlisp−D became available on the D−series machines manufactured by Xerox−−−the Dorado, Dandelion, Dandetiger, and Dove (or Daybreak). An upward−compatible extension of MacLisp called Lisp Machine Lisp became available on the early MIT Lisp Machines. Commercial Lisp machines from Xerox, Lisp Machines (LMI), and Symbolics were on the market by 1981. For further information about Lisp Machine Lisp, see *Lisp Machine Manual*.

During the late 1970's, Lisp Machine Lisp began to expand towards a much fuller language. Sophisticated lambda lists, **setf**, multiple values, and structures like those in Common Lisp are the results of early experimentation with programming styles by the Lisp Machine group. Jonl White and others migrated these features to MacLisp. Around 1980, Scott Fahlman and others at CMU began work on a Lisp to run on the Scientific Personal Integrated Computing Environment (SPICE) workstation. One of the goals of the project was to design a simpler dialect than Lisp Machine Lisp.

The Macsyma group at MIT began a project during the late 1970's called the New Implementation of Lisp (NIL) for the VAX, which was headed by White. One of the stated goals of the NIL project was to fix many of the historic, but annoying, problems with Lisp while retaining significant compatibility with MacLisp. At about the same time, a research group at Stanford University and Lawrence Livermore National Laboratory headed by

Richard P. Gabriel began the design of a Lisp to run on the S−1 Mark IIA supercomputer. S−1 Lisp, never completely functional, was the test bed for adapting advanced compiler techniques to Lisp implementation. Eventually the S−1 and NIL groups collaborated. For further information about the NIL project, see ‘‘NIL−−−A Perspective.’’

The first effort towards Lisp standardization was made in 1969, when Anthony Hearn and Martin Griss at the University of Utah defined Standard Lisp−−−a subset of Lisp 1.5 and other dialects−−−to transport REDUCE, a symbolic algebra system. During the 1970's, the Utah group implemented first a retargetable optimizing compiler for Standard Lisp, and then an extended implementation known as Portable Standard Lisp (PSL). By the mid 1980's, PSL ran on about a dozen kinds of computers. For further information about Standard Lisp, see ‘‘Standard LISP Report.’’

PSL and Franz Lisp−−−a MacLisp−like dialect for Unix machines−−−were the first examples of widely available Lisp dialects on multiple hardware platforms.

One of the most important developments in Lisp occurred during the second half of the 1970's: Scheme. Scheme, designed by Gerald J. Sussman and Guy L. Steele Jr., is a simple dialect of Lisp whose design brought to Lisp some of the ideas from programming language semantics developed in the 1960's. Sussman was one of the prime innovators behind many other advances in Lisp technology from the late 1960's through the 1970's. The major contributions of Scheme were lexical scoping, lexical closures, first−class continuations, and simplified syntax (no separation of value cells and function cells). Some of these contributions made a large impact on the design of Common Lisp. For further information about Scheme, see *IEEE Standard for the Scheme Programming Language* or ‘‘Revised^4 Report on the Algorithmic Language Scheme.’’

In the late 1970's object−oriented programming concepts started to make a strong impact on Lisp. At MIT, certain ideas from Smalltalk made their way into several widely used programming systems. Flavors, an object−oriented programming system with multiple inheritance, was developed at MIT for the Lisp machine community by Howard Cannon and others. At Xerox, the experience with Smalltalk and Knowledge Representation Language (KRL) led to the development of Lisp Object Oriented Programming System (LOOPS) and later Common LOOPS. For further information on Smalltalk, see *Smalltalk−80: The Language and its Implementation*. For further information on Flavors, see ‘‘Flavors: A Non−Hierarchical Approach to Object−Oriented Programming.’’

These systems influenced the design of the Common Lisp Object System (CLOS). CLOS was developed specifically for X3J13's standardization effort, and was separately written up in ‘‘Common Lisp Object System Specification.’’ However, minor details of its design have changed slightly since that publication, and that paper should not be taken as an authoritative reference to the semantics of the Common Lisp Object System.

In 1980 Symbolics and LMI were developing Lisp Machine Lisp; stock−hardware implementation groups were developing NIL, Franz Lisp, and PSL; Xerox was developing Interlisp; and the SPICE project at CMU was developing a MacLisp−like dialect of Lisp called SpiceLisp.

In April 1981, after a DARPA−sponsored meeting concerning the splintered Lisp community, Symbolics, the SPICE project, the NIL project, and the S−1 Lisp project joined together to define Common Lisp. Initially spearheaded by White and Gabriel, the driving force behind this grassroots effort was provided by Fahlman, Daniel Weinreb, David Moon, Steele, and Gabriel. Common Lisp was designed as a description of a family of languages. The primary influences on Common Lisp were Lisp Machine Lisp, MacLisp, NIL, S−1 Lisp, Spice Lisp, and Scheme. *Common Lisp: The Language* is a description of that design. Its semantics were intentionally underspecified in places where it was felt that a tight specification would overly constrain Common Lisp research and use.

In 1986 X3J13 was formed as a technical working group to produce a draft for an ANSI Common Lisp standard. Because of the acceptance of Common Lisp, the goals of this group differed from those of the original designers. These new goals included stricter standardization for portability, an object−oriented programming system, a condition system, iteration facilities, and a way to handle large character sets. To accommodate those goals, a new language specification was developed.

[Note: Gabriel and Steele's "The Evolution of Lisp" 1993 ACM History of Programming Languages conference, is also available − miller]

This page was extracted from the Common Lisp specification by:

kmp@harlequin.com (Kent M Pitman)
Ongoing maintenance of this page is being done by:
miller@cs.rochester.edu (Brad Miller)
Last Update: 22 June 1994 / Brad Miller