A Forth Implementation of LISP

Tom Hand

Department of Computer Science
Florida Institute of Technology
Melbourne, Florida

This paper describes a Forth implementation of the language LISP. After discussing various aspects of this implementation, a table of the implemented functions is presented.

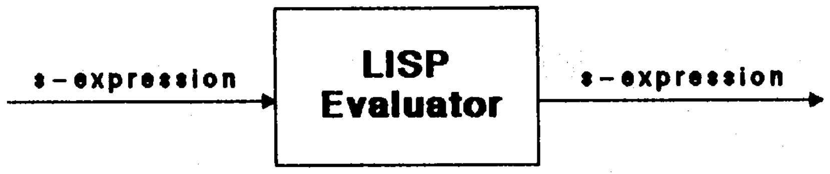
INTRODUCTION

The LISP language was invented by John McCarthy of MIT in 1957. LISP is the most popular language of Artificial Intelligence in the United States. It is the second oldest high-level language in current use.

This implementation is based on Franz LISP and was totally written in TTForth on a Motorla 68000 microprocessor. TTForth is a token-threaded Forth that was written by H. S. Lee at Florida Institute of Technology.

IMPLEMENTATION

As in Forth, both data and programs are treated in the same way by LISP. The basic data structure in LISP is a list. LISP contains an evaluation process that translates one s-expression (short for symbolic expression) into another s-expression.



For simplicity, this implementation requires that tokens in the LISP source code be enclosed within blanks.

The primitive data objects of LISP are called atoms; they are either numbers or symbols. The basic data structures are linked lists and property lists. Atoms and lists are handled differently. Atoms have structure that is quite similar to Forth; they are words but they are not directly accessible by the inner interpreter. The structure of an atom is illustrated by the following figure:

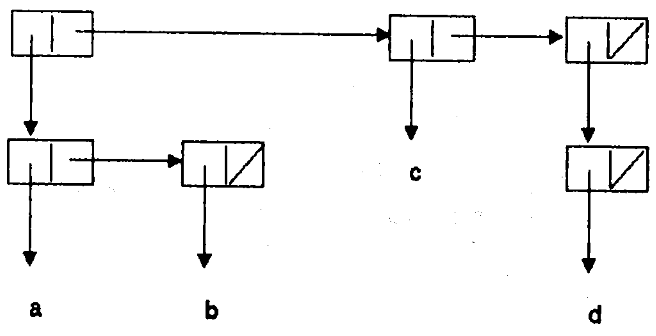
link field
name field
code field
type field
value field
property field
function definition field

Type checking is performed at run-time because LISP requires dynamic type checking. Any number of properties can be assigned to a symbol with these properties having arbitrary types.

Lists are represented as binary trees. For example, the list

((a b) c (d))

is represented as the following binary tree:



It is always the case that the left node points to the current element of the list while the second node points to the remainder of the list.

Standard LISP functions have been implemented as Forth words. On the other hand, user defined functions are represented as binary trees in a manner similar to that done by most LISPs. Forward references are handled by maintaining a list of undefined symbols along with the addresses that correspond to each such symbol.

As mentioned earlier, the basic operation in LISP is called evaluation. For a list to be evaluated, its first argument must be a function. If such a function is detected, the remaining arguments of the list are each evaluated and finally the function itself is evaluated.

Because data structures are dynamically created and deleted, tree nodes require continual allocation and deallocation. Efficient memory allocation requires that garbage collection be performed to reclaim any unused nodes. A first fit algorithm decides which memory is allocated from the free pool. Garbage collection is performed as data structures change dynammically.

INSTALLED LISP FUNCTIONS

A reasonable subset of LISP functions from Franz LISP have been implemented in the current prototype. They are listed below:

List Functions:

car	cdr	last
nthelem	nthcdr	cons
append	list	nconc
rplaca	rplacd	assoc
length	reverse	

Comparator Functions:

and	or	not
atom	null	numberp
symbolp	listp	eq
equal	neg	nequal
greaterp	lessp	zerop
minusp	plusp	evenp
oddp	=	<=
>=		

Data Structure Functions:

setq	putprop	defprop
get	plist	

Evaluation Control Functions:

quote	eval	funcal

Arithmetic Functions:

+	–	*
/	1+	1–
mod	abs	

Input/Output Functions:

print	read	load
msg		

Function Definition Functions:

defun	def	getd
putd		

Flow of Control Functions:

cond	if	while
prog	go	return

CONCLUSIONS

This version of LISP was successfully implemented as an initial prototype. Currently, two other enhanced versions are under development that take advantage of Forth's extensibility.

REFERENCES

1. Brodie, L., Starting Forth, Prentice–Hall, 1981.

2. Lee, H., TTFORTH: A Token Threaded Forth, Florida Institute of Technology, 1985.

3. McCarthy, J., LISP 1.5 Programmer's Manual, MIT Press, 1965.

4. Wilensky, R., LISPcraft, W. W. Norton & Company, 1985.

5. Winston & Horn, LISP, Addison Wesley, 1984.

6. Yin, Q., FLISP: A LISP Embedded in the Forth Environment, Florida Institute of Technology, 1986.