TM- 2260/004/00

AUTHOR M. Levin & S. Kameny

TECHNICAL

RELEASE B. Barancik

for D. L. Drukey

DATE 8/23/65 PAGE 1 OF 4 PAGES

TECH MEMO *a working paper*

System Development Corporation / 2500 Colorado Ave. / Santa Monica, California

LISP II PROJECT

MEMO NO. 11

THE SYNTAX OF TOKENS

Abstract

This memo defines the syntax of LISP II at the token level. The parsing of tokens is the function of the finite state machine.

This memo defines the token syntax of LISP II. The LISP II reference language, publication language, and Q-32 hardware language are identical. If LISP II is to be expressed within the FORTRAN character set, a different hardware language would be needed. No hardware changes should ever go beyond the token level.

The LISP II language uses 58 printing characters, the space (printed as ␣ in this memo) and the carriage return (printed as cr in this memo). By using a subset of ASCII, we hope to avoid the need for different hardware languages at different installations. The final decision rests with the designers of hardware.

LISP II Character Set

26 letters	ABCDEFGHIJKLMNOPQRSTUVWXYZ
10 digits	␣0123456789
4	( ) [ ]
7	+ - * / \ ↑ ←
4	, ; . :
3	< > =
4	% # $ '
2	␣ cr
60	

Five ASCII characters (@, &, ?, ", and !) are not part of the LISP II language. Any installation may add new characters at the cost of creating a dialect that cannot be run at other installations. LISP II has provision to handle up to 255 characters without reprogramming.

letter = A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

octal-digit = 0|1|2|3|4|5|6|7

digit = octal-digit|8|9

string-character = letter|digit|(|)|[|]|+|-|*|/|\|↑|←|,|.|:|<|>|=|$|␣

(cr, ', #, %, and ; are not string-characters.)

token = token-atom|special-token

token-atom = Boolean|number|string|identifier

special-token = ,|;|:|⁄.⁄|<|>|=|/=|<=|>=|+|-|*|\|/|↑|←|-:|(|)|[|]|'|$

space = ⁄| (cr) |⁄ space| (cr) space|comment|comment space

comment = %⁄{string-character|#|'}*;|%⁄{string-character|#|/}*(cr)|%;|%(cr)

Boolean = TRUE|false

false = FALSE|NIL|()

The different ways of writing false are entirely equivalent.

number = integer|real

integer = octal|decimal

octal = sign octal-digit {octal-digit}* Q {unsigned-decimal|empty}

sign = ⁄+|-|empty

unsigned decimal = decimal-digit {decimal-digit}*

decimal = sign unsigned-decimal {empty|E|E unsigned-decimal}

real-object= unsigned-decimal .|.unsigned-decimal|unsigned-decimal . unsigned-decimal

real = sign real-object {empty|E scale}

scale = sign unsigned-decimal

string = #{string-character|;|''|'#|'%|'(cr)} *#

An unquoted carriage return inside a string will be ignored. An unquoted % is an error. An unquoted quote mark or fence is syntactically ambiguous. The meaning of a string is a substring containing all characters of the original in proper sequence except for the initial and terminal fences, and all quote marks that are not themselves quoted. (A quote mark is quoted if and only if it is an even number of places from the beginning of a consecutive sequence of quote marks.)

identifier = letter {letter|digit|.}*|% string
TRUE, FALSE and NIL are not identifiers.

The identifiers ABC, and %#ABC# are identical.

There is still another form for writing one character identifiers. It is legal only if the identifier is a datum or part of a datum. The identifier is simply preceded by a quote mark.

Thus '(is the same identifier as %#(#, but '(may only be used in internal language or in a quoted context in source language. The source language ''(will translate into (QUOTE '() which is identical to (QUOTE %#(#).

'(A B '() is identical to '(A B %# (#).

A, %#A#, and 'A are identical in a quoted context.

atom = token-atom|array

Arrays are atoms that are not tokens. They are parsed by the syntax translator. An array must be written with the same number of elements in each row, column, etc.

Examples: [INTEGER [0 1] [-1 0]]

[REAL 3.4 -6.E2]

[SYMBOL [U (V . W)][(X Y) 2.7]]

S-expression = atom| (S-expression $\{$S-expression$\}^*$ ↓ . ↓ S-expression)|
($\{$S-expression$\}^*$)

datum = 'S-expression|number|Boolean|string