Syntax of LISP 2 Tokens

ABSTRACT

defines the syntax of LISP 2 at the token level. Tokens are parsed by a finite state machine and then used to construct source language or S-expressions.
FOREWORD

LISP 2 is a joint development of SDC and III. The idea for LISP 2 as a language combining the properties of an algebraic language like ALGOL and the list-processing language LISP was conceived by M. Levin of MIT. Development of the concepts of LISP 2 was carried forth in a series of conferences held at MIT and Stanford University. Contributions in concepts and detail were made by Prof. John McCarthy of Stanford University, Prof. Marvin Minsky of MIT, and the LISP 2 project team consisting of M. Levin, L. Hawkinson, R. Saunders and P. Abrahams of III, and S. Kameny, C. Weissman, E. Book, Donna Firth, J. Barnett and V. Schorre of SDC.

For the implementation of LISP 2, it was decided to define a standard, computer-independent, LISP-like intermediate language and to define the LISP 2 source language in terms of its translation into the intermediate language.

This document describes the syntax of tokens.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>5</td>
</tr>
<tr>
<td>2. Characters</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Character Classes.</td>
<td>5</td>
</tr>
<tr>
<td>3. Notation</td>
<td>5</td>
</tr>
<tr>
<td>4. Special Characters</td>
<td>6</td>
</tr>
<tr>
<td>5. Character Mapping.</td>
<td>6</td>
</tr>
<tr>
<td>6. Tokens</td>
<td>7</td>
</tr>
<tr>
<td>6.1 Basic Alphabet</td>
<td>7</td>
</tr>
<tr>
<td>6.2 Syntax of Tokens</td>
<td>8</td>
</tr>
<tr>
<td>7. S-expressions at the Token Level</td>
<td>10</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

Tokens are the units from which S-expressions and source language are constructed. A token has no internal structure as do atoms or lists; it exists momentarily when the finite state machine stops; further existence depends upon the use made of it by the S-expression reader, the syntax translator, or the token reader.

2. CHARACTERS

The LISP 2 character set is the 128 characters of the revised ASCII standard. Character mapping (Section 5) is also available so that non-graphic characters may be entered, or the limitations of certain input devices circumvented. All characters shown in token syntax are assumed to be the result of character mapping which takes precedence when used.

2.1 CHARACTER CLASSES

Tokens are formed by combining character classes rather than individual characters. All 128 characters belong to the class named character with subclasses as in Section 6.1. A basic alphabet of 58 graphic characters, a space (\,), and 6 non-graphic characters is used in this document. The class assignment of the remaining 63 characters is implementation-dependent. Class membership can be dynamically changed within the LISP 2 system when a user so desires. The finite state machine which parses tokens is also changeable when languages other than source language or S-expressions are being read, although a change may not be required.

If the lower case letters are assigned to the class letter, the question of the equivalence of such sequences as 'BEGIN' and 'begin' arises. The answer to this question will depend on implementation.

3. NOTATION

The symbols | { } *, the use of italics, and the form of syntax equations conform to the usage in TM-2710/220/01, LISP 2 Intermediate Language. The characters shown in the basic alphabet stand for themselves (in the ASCII scheme) except for + \ which are assigned for each implementation, and the non-graphic characters which are written as \ for space, CR for carriage return, NUL for null, etc.

Other symbols used are the superscript $^0$, which means that the entity so designated is not a part of the token which is formed, the negation sign $\sim$, and superscripts referring to footnotes which are not part of the syntax equations themselves. In all token syntax equations in Sections 4 through 6, space is explicitly indicated in the equations. In Section 7, spaces are implicit in the definition of expressions as token sequences.

4. **SPECIAL CHARACTERS**

The class \texttt{escape:character} has one member which will usually be \% although this is changeable as are all other class assignments. The use of \texttt{escape:character} has the highest precedence in token parsing. At present it is used for creation of unusual identifiers, character mapping, remarks, and hyphenators. The hyphenator is actually a special case of character mapping. The syntax of hyphenator is given with the basic alphabet because it maps onto the single character \texttt{NUL}.

\texttt{NUL} and \texttt{hyphenator} constitute the \texttt{null:class} which is a character class completely invisible in token parsing except when preceded by a prime in string context. Outside of this special context the following is always true:

\begin{equation}
\text{character null:class} \equiv \text{character character}
\end{equation}

The occurrence of \texttt{null:class} is not shown in token syntax; it may occur at any point in a character sequence with an effect as above.

5. **CHARACTER MAPPING**

The meaning of \texttt{escape:character} is always governed by the following character. When the \texttt{escape:character} itself is intended it is followed by an I. This is the identity mapping and is the only way that the \texttt{escape:character} can mean itself. The characters I, \#, R, G, \&, ;, CR, US, RS, and C have special meaning in token syntax when they follow the \texttt{escape:character}. The use of \texttt{escape:character} C is a general form of character mapping, as follows:

\begin{equation}
\text{cardinal} = \text{unsigned:integer} \mid \text{unsigned:octal} \quad \text{(See Section 6.2)}
\end{equation}

\begin{equation}
\text{character} = \text{escape:character} \; C \; \text{cardinal}.
\end{equation}

The \texttt{character} resulting from this mapping is the one whose numeric code is the same as the \texttt{cardinal}. For example, \%\texttt{C101Q} means A. Character mapping is recursive, consequently \%\texttt{C103Q.101Q} also means A. In this latter example the \texttt{character} following the \% is C and not another \% because of the precedence of character mapping previously mentioned.
The use of ¥, R, G, #, ;, CR, US, and RS following the escape:character is given in Sections 6.1 and 6.2; the meaning of any character other than these mentioned will depend on implementation.

The use of the characters DEL (delete) and BS (backspace) may cause a form of character mapping, but these also are implementation dependent.

6. **TOKENS**

All tokens but one are explicitly defined below. The one exception is **unrecognizable**. This is defined by default to mean any character sequence which does not satisfy one of the syntax equations for the other tokens. Examples of unrecognizable are:

```
#A#FS  1.E10A  5E6.  #('FS;1ZEM
```

6.1 **BASIC ALPHABET**

\[\begin{align*}
ec &= E \\
qc &= Q \\
rc &= R \\
letter &= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z \\
octal:digit &= 0|1|2|3|4|5|6|7 \\
digit &= octal:digit|8|9 \\
mark &= :|\|<|>|=|(||\] |+$|+|\cdot |¥ \\
p:mark &= (|[|,|$,|%,|+-|.|¥ \\
ordinary &= letter|digit|mark|p:mark \\
prime &= ' \\
fence &= # \\
semi:colon &= ;
\end{align*}\]
6.2 SYNTAX OF TOKENS

The effect of prime character is to enter the character in the token and to discard the prime. This character may be any character at all, including boundary, data:separator, and null:class. This is the only place in which null:class is meaningful.
unsigned:integer = (decimal ec decimal|decimal)n:delimiter
octal:spelling = octal:digit octal:digit* qc
unsigned:octal = {octal:spelling decimal|octal:spelling}n:delimiter
exponent = ec{plmn decimal|decimal}
mantissa = decimal period decimal|decimal period|period decimal
unsigned:real = {mantissa exponent|mantissa}n:delimiter
sign = plmn(period|digit)
spacer = space space*|boundary
dot = period {~alpha}


1A remark may occur in SL or IL wherever a spacer may be used. In IL commas are not optional. See Section 7.

2The definitions of dot and dotted:literal prevent the character period from being a dotted:literal.
7. S-EXPRESSIONS AT THE TOKEN LEVEL

Occurrences of words such as FUNCTION, REAL, etc. denote the token that was a literal with the same character representation.

\[\text{octal} = \text{sign unsigned:octal|unsigned:octal}\]
\[\text{integer} = \text{sign unsigned:integer|unsigned:integer}\]
\[\text{real} = \text{sign unsigned:real|unsigned:real}\]
\[\text{number} = \text{octal|integer|real}\]
\[\text{spaces} = \text{spacer|remark}\]
\[\text{empty} = \text{spaces}\]
\[\text{false} = \text{FALSE|NIL|lpar rpar}\]
\[\text{boolean} = \text{TRUE|false}\]
\[\text{string} = \text{string:spelling}\]
\[\text{identifier} = \text{literal2|dotted:literal|string:name|operator|gen:spelling|u:mark}\]
\[\text{f:name} = \text{lpar identifier dot identifier rpar}\]
\[\text{value:type} = \text{literal}\]
\[\text{f:type} = \text{literal}\]
\[\text{a:type} = \text{f:type|lpar f:type \{LOC\VALUE\} rpar}\]
\[\text{i:type} = \text{lpar f:type \{LOC\VALUE|empty\} INDEF rpar}\]
\[\text{functional:constant} = \text{lbrac "FUNCTION f:name lpar value:type a:type" \{i:type|empty\} rpar rbrac}\]
\[\text{numeric:row} = \text{lbrac \{number number* | numeric:row numeric:row* \} rbrac}\]
\[\text{real:array} = \text{lbrac REAL \{number* | numeric:row* \} rbrac}\]

1 empty means either a sequence of spaces or nothing. It has no semantic effect on the s:expression. By the definition of sign in Section 6.2, no spacer or remark can occur between a sign and an unsigned number; in all other s:expressions which are token sequences empty may occur between tokens.

2 The character representation is not TRUE, FALSE or NIL.
integer:array = ibrac INTEGER (number* | numeric:row* ) rbrac
octal:array = ibrac OCTAL (number* | numeric:row* ) rbrac
numeric:array = real:array|integer:array|octal:array

boolean:exp = s:expression

boolean:row = ibrac (boolean:exp boolean:exp* | boolean:row boolean:row* ) rbrac

boolean:array = ibrac BOOLEAN (boolean:exp* | boolean:row* ) rbrac

symbol:element = boolean|number|string|identifier|list|
dot{array|functional:constant}

symbol:row = ibrac {symbol:element symbol:element* |
symbol:row symbol:row* } rbrac

symbol:array = ibrac SYMBOL {symbol:element* | symbol:row* } rbrac

functional:row = ibrac {functional:constant functional:constant* |
functional:row functional:row* } rbrac

functional:array = ibrac FUNCTIONAL {functional:constant* | 
functional:row* } rbrac

array = boolean:array|numeric:array|symbol:array|functional:array

simple:datum = boolean|number|string|array|functional:constant

atom = simple:datum|identifier

list = lpar s:expression s:expression* {dot s:expression|empty} rpar

s:expression = atom|list

---

1This notation is used for arrays or functional constants that are elements of the symbol:array and not for sub-elements. For example:

- [SYMBOL A (B C) .[REAL 1.0 2.0]]
- [SYMBOL [A (B C) .[INTEGER 1 2]] [(x) #(# (E . F))]

require the notation but

- [SYMBOL A (B [REAL 1.0 2.0])] does not. In the last example use of dot would make the element into a dotted pair.

2The rows of a multi-dimensional array must have the same number of elements.
### Distribution

<table>
<thead>
<tr>
<th>Name</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>D. Barancik</td>
<td>2105</td>
</tr>
<tr>
<td>J. Barnett</td>
<td>2025</td>
</tr>
<tr>
<td>R. Herman</td>
<td>4317</td>
</tr>
<tr>
<td>E. Hook</td>
<td>2332</td>
</tr>
<tr>
<td>S. Rosak</td>
<td>2013</td>
</tr>
<tr>
<td>T. Burger</td>
<td>9919</td>
</tr>
<tr>
<td>D. Drucy</td>
<td>2105</td>
</tr>
<tr>
<td>Marsha Drapkin(2)</td>
<td>9723</td>
</tr>
<tr>
<td>S. Feinzold</td>
<td>9525</td>
</tr>
<tr>
<td>D. Firth</td>
<td>2310</td>
</tr>
<tr>
<td>M. Hirward</td>
<td>2042</td>
</tr>
<tr>
<td>H. Howell</td>
<td>9912</td>
</tr>
<tr>
<td>A. Irvine</td>
<td>9627</td>
</tr>
<tr>
<td>E. Jacobs</td>
<td>2344</td>
</tr>
<tr>
<td>B. Jones</td>
<td>7231</td>
</tr>
<tr>
<td>S. Kameny (50)</td>
<td>2009</td>
</tr>
<tr>
<td>R. Long</td>
<td>9913</td>
</tr>
<tr>
<td>R. Martin</td>
<td>2226</td>
</tr>
<tr>
<td>E. Myer</td>
<td>2227</td>
</tr>
<tr>
<td>M. Perstein</td>
<td>2344</td>
</tr>
<tr>
<td>D. Perry</td>
<td>2042</td>
</tr>
<tr>
<td>V. Schorre</td>
<td>2330</td>
</tr>
<tr>
<td>J. Schwartz</td>
<td>7173</td>
</tr>
<tr>
<td>J. Simmons</td>
<td>9439</td>
</tr>
<tr>
<td>L. Steffrud</td>
<td>9734</td>
</tr>
<tr>
<td>V. Vorhaus</td>
<td>2213</td>
</tr>
<tr>
<td>Y. Weissman(10)</td>
<td>2214</td>
</tr>
<tr>
<td>H. Wolfson</td>
<td>2368</td>
</tr>
</tbody>
</table>

- L. Hawkins(III) 9720
- D. Crandel (III) 9721
- D. Aschultz (III) 9721
- B. Saunders (III) 9721
- P. Stygar (III) 9721