The research reported in this paper was sponsored by the Advanced Research Projects Agency Information Processing Techniques Office and was monitored by the Electronic Systems Division, Air Force Systems Command under contract F1962867C0004, Information Processing Techniques, with the System Development Corporation.

TECH MEMO *a working paper*

System Development Corporation / 2500 Colorado Avenue / Santa Monica, California 90406
Information International Inc. / 11161 Pico Boulevard / Los Angeles, California 90064

TM-3417/001/00

AUTHOR R Wills
R. Wills

TECHNICAL Jeff Barnett
J. Barnett

RELEASE [illegible]
C. Weissman, S.D.C.
[illegible]
D. Anschultz, I.I.I.
for J. I. Schwartz

DATE 4/26/67 PAGE 1 OF 6 PAGES

(Page 2 is blank)

LISP 2 Document Conventions

ABSTRACT

This document describes conventions employed in a series of documents which specify the LISP 2 language and processor for the IBM S/360 computer. Included in this document are rules for writing syntax equations for the LISP 2 language.

1. INTRODUCTION

Specification of the LISP 2 language and system for an IBM S/360 computer has been a joint effort of S.D.C. and I.I.I. A large number of system analysts, designers and programmers were involved in this effort over a long period of time. The documents resulting from this work (which were published as separate volumes of TM-3417) define and describe the language and processor required to implement LISP 2 on the 360.

Due to the large number of people involved in this project, and the inherent complexity of the LISP 2 system as well as the complexity of the machine for which it was designed, some of the specification documents are at variance concerning the specific techniques for implementing certain system functions. There is also some variation among documents in regard to symbology, particularly that employed in syntax equations. However, each document in the series is internally consistent and contains explanations of any special symbols or any deviations from conventional LISP documentation practices.

2. SYNTAX EQUATION CONVENTIONS

Within documents in this series, LISP 2 syntax is specified in terms of two languages: a kernel language and a transformed language. Both the kernel language and the transformed language are in turn subdivided into Source Language (SL) and Intermediate Language (IL). The language specification consists of the syntax and semantics of the kernel language together with a set of syntactic transformations. A given sequence of characters is a legitimate LISP 2 program if it can be obtained from a kernel program through the application of transformation rules. The semantics of the transformed program are the same as the semantics of the kernel program from which it was derived. The syntax of the kernel language is much simpler than the syntax of the transformed language, which is never given explicitly.

A transformation rule consists of two sequences of meta-syntactic symbols: an input pattern and an output pattern. The rule is applied by matching the input pattern to a segment of a program, transforming the segment according to the rule, and replacing the original segment by the transformed segment. Transformation rules are of two kinds: optional and required. An optional rule may or may not be applied if it is applicable; a required rule must be applied whenever it is applicable.

Transformation rules are intermixed with syntax equations in this document series. If a transformation rule is in an SL group, then it is applied to SL programs; if it is in an IL group, then it is applied to IL programs. Optional transformation rules are indicated in the syntax equations by $\textcircled{\small OT}$; required transformation rules are indicated by $\textcircled{\small RT}$. When a program segment matches a metasyntactic symbol in the input pattern, then it also matches any occurrence of the corresponding metasyntactic symbol in the output pattern. If a metasyntactic symbol appears more than once in the input pattern, then subscripting may be used to distinguish the occurrences of it. If the input pattern is followed by "C" and a metasyntactic symbol or a choice of metasyntactic symbols, then the input pattern must be part of the syntactic construct specified to the right of the "C". The notation "$\overline{\{\alpha\}}$" means "anything other than an α". The symbol $\Longrightarrow$ indicates "is replaced by".

The syntax equations for SL and IL are distinguished by the appearance of $\boxed{\text{SL}}$, $\boxed{\text{IL}}$, and $\boxed{\begin{matrix}\text{SL}\\\text{IL}\end{matrix}}$ at the beginning of a group of equations. When one of these symbols appears, it applies until the occurrence of the next one. $\boxed{\begin{matrix}\text{SL}\\\text{IL}\end{matrix}}$ indicates that the following group is part of both SL and IL. The same metasyntactic symbols are used in both SL and IL syntax equations, and corresponding metasyntactic symbols have the same semantics. When a $\boxed{\text{T}}$ appears, it indicates that the remaining equations in the group are token equations. In token equations, successive syntactic entities are assumed to be written without intervening blanks. The equations for SL and IL are all built up from token equations. In the kernel language it is assumed that tokens are separated by a single blank; transformation rules are used in order to insert additional blanks and remove redundant blanks. Spaces within syntax equations are purely for the sake of legibility and have no effect on the meaning of the equation.

3. RULES FOR WRITING SYNTAX EQUATIONS

The following rules are intended for use in connection with the syntax equations that show how the complete language is built up from tokens. In these equations, tokens are the lowest-level entities that are considered. However, there are also equations for the syntax of tokens; these equations describe how tokens are built up from characters. At this level, special provisions need to be made to handle matters such as spacing, since the problems of delimitation are much more acute there.

Let α designate a metasyntactic symbol, e.g., <u>identifier</u>, or an explicit token, e.g., FOR. Then:

Rule 1: $\sigma_1 \sigma_2 \ldots \sigma_n$ designates the concatenation of $\sigma_1, \sigma_2, \ldots \sigma_n$ $(n \geq 1)$. Let α designate such a concatenation.

Rule 2: $\alpha_1|\alpha_2| \ldots |\alpha_n$ designates the alternation among $\alpha_1, \alpha_2, \ldots \alpha_n$ $(n \geq 1)$. Let β designate such an alternation.

Rule 3: The brackets { and }, without subscripts or superscripts, are used for grouping.

Rule 4: $[\beta] \equiv \beta|\text{empty}$

Rule 5: $\underline{[}$ designates the character [.
$\underline{]}$ designates the character].

Rule 6: $$\left\{\begin{array}{c}\beta_1\\ \beta_2\\ \cdot\\ \cdot\\ \cdot\\ \beta_n\end{array}\right\} \equiv \beta_1|\beta_2| \ldots |\beta_n$$

$$\left[\begin{array}{c}\beta_1\\ \beta_2\\ \cdot\\ \cdot\\ \cdot\\ \beta_n\end{array}\right] \equiv \beta_1|\beta_2| \ldots |\beta_n|\text{empty}$$

Rule 7: σ_* $\equiv$ 0 or more σ's
$\{\beta\}_*$ $\equiv$ 0 or more β's

Rule 8: σ_{*+1} $\equiv$ 1 or more σ's
$\{\beta\}_{*+1}$ $\equiv$ 1 or more β's

Rule 9: $\{\beta\}_*^\delta$ $\equiv$ 0 or more β's separated by δ (where δ is an explicit token)

Rule 10: $\{\beta\}_{*+1}^\delta$ $\equiv$ 1 or more β's separated by δ (where δ is an explicit token)

Rule 11: $\{\alpha_1 || \alpha_2 || \ldots || \alpha_n\} \equiv$ exactly one occurrence of each α_i in any order. Each α_i is either a σ or a $\{\beta\}$.

Rule 12: $\{\alpha_1 || \alpha_2 || \ldots || \alpha_n\}^{\delta} \equiv$ exactly one occurrence of each α_i, with the non-empty occurrences separated by δ.

Rule 13: The notation N1 preceding an equation designates footnote number 1; N2 designates footnote number 2, etc.

Rule 14: Terminal symbols in syntax equations are written with all capital Roman letters.

Rule 15: Non-primitive syntactic entities, e.g., those things that are on the left-hand side of syntax equations, are written in all lower-case Roman letters.

Rule 16: The name of a non-primitive syntactic entity is written without blanks. If it is composed of more than one word, hyphens are used to connect the words. If split over line boundaries, the hyphenator is not repeated on the second line.

Rule 17: In prose descriptions, the distinction between non-terminal syntactic entities and other uses of the same terms (e.g., "atom" as a LISP entity and "atom" as a chemical entity) is maintained by using simple English. If the distinction is not clear, it is spelled out with a few extra words.

Rule 18: Italic letters are not used in these documents. When a term is first mentioned in a prose description, it is sometimes underscored for emphasis.

Examples:

$\boxed{\text{SL}}$ section-list $\equiv$ $\{\text{section-name}\}^{,}_{*+1}$

$\boxed{\text{IL}}$ section-list $\equiv$ section-name$|$(section-name$_{*+1}$)

parameter-list $\equiv$ (parameter$_{*}$ indef-parameter)

$\boxed{\begin{array}{l}\text{SL}\\ \text{IL}\end{array}}$ attribute-list $\equiv$ $\{\text{type}||\text{storage-mode}||\text{reference-mode}\}^{,}$