Lexis

/Reference manual

1. Contents of this page

2. Introduction

The lexis defines the translation from sequences of UCS characters to tokens (or lexemes) of the syntax, such as *NAME* and *NUMERAL*. It is performed in two phases: the context-free phase, then the context-sensitive phase.

The context-free lexis is formalized below using syntactic metalanguage.

3. ISO Standard lexis

3.1. Formal definition of context-free lexis

This formal definition is public domain material, and appears as it appears in ISO/IEC 13568:2002 (the Z standard).

$$TOKENSTREAM = \{SPACE\}, \{TOKEN, \{SPACE\}\};$$

$$\begin{aligned} TOKEN = {} & DECORWORD \mid NUMERAL \mid STROKE \\ \mid {} & (-tok \mid) - tok \mid [-tok \mid] - tok \mid \{-tok \mid\} - tok \mid \langle\!|\mid\!\rangle \mid \langle\langle\mid\rangle\rangle \\ \mid {} & ZED \mid AX \mid GENAX \mid SCH \mid GENSCH \mid END \mid NL \\ ; & \end{aligned}$$

$$DECORWORD = WORD, \{STROKE\};$$

$$\begin{aligned} WORD = {} & WORDPART, \{WORDPART\} \\ \mid {} & (LETTER \mid (DIGIT - - - DECIMAL)), ALPHASTR, \{WORDPART\} \\ \mid {} & SYMBOL, SYMBOLSTR, \{WORDPART\} \\ ; & \end{aligned}$$

$$WORDPART = WORDGLUE, (ALPHASTR \mid SYMBOLSTR);$$

$$ALPHASTR = \{LETTER \mid DIGIT\};$$

$$SYMBOLSTR = \{SYMBOL\};$$

$$
\begin{array}{lcl}
NUMERAL & = & NUMERAL, DECIMAL \\
& | & DECIMAL \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
STROKE & = & STROKECHAR \\
& | & '\searrow', DECIMAL, '\nwarrow' \\
& ; &
\end{array}
$$

$$
\begin{array}{lcl}
(-tok & = & '('; \\
) - tok & = & ')'; \\
[-tok & = & '['; \\
] - tok & = & ']'; \\
\{-tok & = & '\{'; \\
\} - tok & = & '\}'; \\
\langle\!| & = & '\langle\!|'; \\
|\!\rangle & = & '|\!\rangle'; \\
\langle\!\langle & = & '\langle\!\langle'; \\
\rangle\!\rangle & = & '\rangle\!\rangle';
\end{array}
$$

$$
\begin{array}{lcl}
ZED & = & ZEDCHAR; \\
AX & = & AXCHAR; \\
SCH & = & SCHCHAR; \\
GENAX & = & GENAXCHAR; \\
GENSCH & = & GENSCHCHAR; \\
END & = & ENDCHAR; \\
NL & = & NLCHAR;
\end{array}
$$

3.2. Additional lexical rules

The above formal definition contains an ambiguity regarding individually subscripted decimal digits at the end of a *WORD*. These are lexed as *STROKE*s, not as glued *WORDPART*s. Multiple digit subscripts at the end of a *WORD* can be lexed only as glued *WORDPART*s, yet look the same, and so use of them is deprecated.

3.3. Notes on context-free lexis

SPACE characters are necessary between neighbouring characters when lexing gives two tokens and would otherwise give a single token. The situations where *SPACE* is needed are: between two neighbouring *WORD*s (otherwise a single *WORD*); between two neighbouring *NUMERAL*s (otherwise a single *NUMERAL*); between a *WORD* and a *NUMERAL* (otherwise a single *WORD*); and between a *DECORWORD* and a *STROKE* (otherwise a single *DECORWORD*). *SPACE* characters are never needed around the brackets enumerated above, as those are *SPECIAL* characters that cannot appear in larger tokens.

The lexis of *WORD* tokens allows alphanumeric and symbolic parts to be glued together within a *WORD*, while neighbouring letters and symbols (with no intervening *WORDGLUE* or *SPACE*) are lexed as separate tokens.

More discussion of these issues may be found in [**?**].

3.4. Informal definition of context-sensitive lexis

All *SPACE*s are elided from the *TOKENSTREAM*.

Those *NL* tokens that appear within a formula (rather than between formulae) are elided, i.e. those preceded immediately by a prefix or infix notation, or followed immediately by a postfix or infix notation.

Each *DECORWORD* token is replaced by either a keyword token (if its spelling, including strokes, is that of a keyword), an operator token (if its spelling, excluding strokes, is that of an operator word), or the token *NAME* otherwise.

The keywords and the operator tokens, classified for the elision of *NL* tokens, are as follows.

Infix: else function generic leftassoc parents relation rightassoc section then
$-tok$ $\langle\!\langle$ $\rangle\!\rangle$ *ampersand* $\vdash?$ $,,$ $\land$ $\lor$ $\Rightarrow$ $\Leftrightarrow$ $\times$ $/$ $=$
$-tok$ $\in$ $==$ $:$ $;$ $-tok$ $.$ $\bullet$ $\setminus$ $\upharpoonright$ $\frac{o}{9}$ $\gg$ *I* *IP* *EL* *ELP* *ERE* *EREP* *ES* *SS*

Prefix: if let pre $[-tok$ $\_$ $\neg$ $\forall$ $\exists$ $\exists_1$ $\mathbb{P}$ $(-tok$ $\{-tok$ $\{\!|$ λ μ θ *PRE* *PREF*

Postfix: $]-tok$ $)-tok$ $\}-tok$ $|\!\}$ *POST* *POSTP* *ER* *ERP* *SR* *SRP*

Nofix: *false* *true* *NAME* *NUMERAL* *STROKE*

4. CADiZ-specific lexis

CADiℤ redefines *NUMERAL* to permit floating-point numerals, formally defined as follows.

$$NUMERAL \quad = \quad DECIMAL, \{DECIMAL\}, ['.', DECIMAL, \{DECIMAL\}], ['E', ['+' \mid '$$

There is no limit on the size of a numeral. The . and *E* characters in this formalisation are taken to be part of the *NUMERAL* only if no space precedes them. Examples of numerals accepted by cadiz include `42`, `3.1415926`, `1E3` and `2.5E-16`. The following are not recognised as numerals: `-42`, `3.`, `.1415926` and `1e3`.

In the *TOKEN* rule, CADiℤ adds *STRING* as another alternative, whose syntax is based on that of string literals in the C programming language, formally defined as follows.

$$STRING \quad = \quad '"', \{STRCHAR\}, '"';$$

$$\begin{aligned} STRCHAR \quad = \quad & '\backslash', ESCAPE \\ \mid \quad & ?\ any\ UCS\ character\ other\ than\ '\backslash',\ '"'\ and\ NLCHAR\ ? \\ ; & \end{aligned}$$

$$
\begin{array}{lcl}
ESCAPE & = & 'n' \\
& | & 't' \\
& | & 'b' \qquad (* \\
& | & 'r' \\
& | & 'f' \qquad (* \\
& | & '"' \\
& | & '\backslash' \qquad (* \\
& | & NLCHAR \qquad (* \text{ continuati} \\
& | & OCTAL \mid OCTAL, OCTAL \mid OCTAL, OCTAL, OCTAL \qquad (* \text{ base} \\
& | & 'u', HEX, HEX, HEX, HEX \qquad (* \text{ encodin} \\
& | & 'U', HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX \qquad (* \text{ UC} \\
& ; &
\end{array}
$$

$$OCTAL \quad = \quad '0' \mid '1' \mid '2' \mid '3' \mid '4' \mid '5' \mid '6' \mid '7';$$

$$HEX \quad = \quad '0' \mid '1' \mid '2' \mid '3' \mid '4' \mid '5' \mid '6' \mid '7' \mid '8' \mid '9' \mid 'a' \mid 'b' \mid 'c' \mid 'd' \mid 'e' \mid 'f' \mid 'A' \mid 'B$$

There is no limit on the length of a string. The character '"' remains in the *SYMBOL* class and can be used in a *WORDPART*, but not as the first character in a *WORD*.

The class of infix tokens is extended with *undecor*, $\veebar$, $\oplus$ and $\dagger$.

The class of prefix tokens is extended with *post*.

The class of nofix tokens is extended with *STRING*.

The uses of these additional tokens are documented in extensions. To check that a Z specification uses only ISO Standard notations, invoke cadiz with the `-ws`

option.

IT 28-Jan-2002