Suggested BNF Syntax for Template Table for ISO/IEC 14651 Extended BNF (ISO/IEC 14977:1996)

98-08-19

Here is a version of the syntax in "SC22/WG20 N 578R, July 20, 1998" rewritten in Extended BNF (ISO/IEC 14977:1996) by Roger Scowen and modified by Ken Whistler.

Ken Whistler writes:

My thanks to Roger Scowen for reviewing the BNF that I had proposed for the common template table in 14651.

The following section is the revised BNF following the Extended BNF of 14977. This is converted by Roger Scowen from the homebrew BNF I wrote in WG20 N 578R. Below, I have resolved a number of the questions posed by Roger, have corrected a few entries, and have posted my own further comments, labeled "kww".

The general problem I have with the Extended BNF syntactic metalanguage as presented here is that the defined symbols are not connected with underscores (e.g. "simple symbol", not "simple_symbol"). This has the apparent benefit of making the BNF rules a little easier on the eye, but has a proliferating problem in that when all the underscores are taken out of the well-formedness rules, it is much harder to distinguish between ordinary words of text and formal constructs defined by the BNF syntax. The formal constructs stand out much better when using underscores.

--Ken

1. Extended BNF Syntax Rules

The syntax below conforms to the Extended BNF syntactic metalanguage defined in ISO/IEC 14977:1996.

```
character
   = (* any member of the repertoire of the encoded character set in use
*) ;
line delimiter
   = (* end-of-line in the text conventions in use *) ;
digit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;
hexdigit = 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | digit ;
id start
   = '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'
   | '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'
   | digit ;
id part = id start | '-' | '_' | '+' ;

comment char = '%' ;
space = ' ' ;
string = { character } ;
four digit hex string = 4 * hexdigit ;
```

```
comment = comment char, { character } ;
identifier = id start, { id part } ;
simple symbol = '<', identifier, '>' ;
hex symbol ::= '<@', four digit hex string, '>' ;
ucs symbol ::= '<U', four digit hex string, '>' ;
symbol ::= simple symbol | ucs symbol | hex symbol ;

(* kww: There was a mistake in the following rule, omitting the
   option of a symbol group consisting just of a symbol, not quoted.
   I'm not sure what the 14977 precedence rules are for "|" and ",",
   so I'm not sure whether the following rule is correct, either. *)

symbol group = symbol | '"', symbol, { symbol }, '"' ;
level token = symbol group | 'IGNORE' ;
delimited level token = level token, ';' ;
multiple level token = {delimited level token}, level token ;

(* kww: To simplify the problem of matching the intended table
   syntax, I've modified the line completion to optionally allow
   a space before a comment at the end of a line, and removed
   the terminal space for the definition of the symbol weight
   entry. *)

line completion = [ [ space ] comment ], line delimiter ;
symbol list item = simple symbol | ucs symbol ;
symbol list item range
   = symbol list item, '..', symbol list item ;
symbol list element
   = symbol list item range | symbol list item ;
symbol definition = symbol list element ;
symbol weight entry
   = symbol list item, space, multiple level token,
     line completion ;

(* kww: Removed the optional space after the semicolon here,
   for consistency. *)

delimited symbol list element
   = symbol list element, ';' ;
symbol list
   = { delimited symbol list element }, symbol list element ;
section identifier = identifier ;
section definition simple entry
   = 'section', space, section identifier, line completion ;
section definition list entry
   = 'section', space, section identifier, space,
     symbol list, line completion ;
section definition entry
   = section definition simple entry
   | section definition list entry ;
target symbol = symbol ;
reorder after entry
   = 'reorder after', space, target symbol, line completion ;
reorder end entry = 'reorder end', line completion ;
reorder section after entry
   = 'reorder section after', space, section identifier,
     space, target symbol, line completion ;
direction = 'forward' | 'backward' ;
delimited direction = direction, ';' ;
multiple level direction = { delimited direction }, direction ;
```

```
order start entry
   = 'order start', space, identifier, ';',
     multiple level direction, space, ',position', line completion ;
order end entry = 'order end', line completion ;
simple line
   = [ symbol definition | symbol weight entry ], line completion ;
tailoring line
   =  [ section definition entry
   | reorder after entry
   | reorder end entry
   | reorder section after entry
   | order start entry
   | order end entry ], line completion ;
table line = simple line | tailoring line ;
section =  [ ordered set of simple lines ] (* See IF1 below *) ;
untailored template table = { simple line } ;
tailored table = { table line } ;
weight table = untailored template table | tailored table ;
```