PLOT(n, s, m) ::=
PLOT(n_1, s_1, m)
& PLOT(n_2, m+RMI(n_1, n_2))
& ...
& PLOT(n_k, s_k, m+RMI(n_1 n_2 ... n_{k-1}, n_k))
& NEWL(s_{i1}) & NEWL(s_{i2}) & ... & NEWL(s_{ip})

where RMI(n_1 ... n_{j-1}, n_j) is the ruling margin increment for the n_j as shown in Figure 2 for that production rule and only the nonterminals n_{i1}, n_{i2} ... n_{ip} has the \n to the left of the reference vertical.

Thus, the above PLOT(*repeat statement*, *rpstst*, m) would be **true** for $m=0$, for example, if *st*1 and *st*2 were empty strings, *w*1 and *w*2 were equal to \n and *exp* did not contain \n.

Definition of ISAT

ISAT(s, m) ::= **true** iff either $s = \%\ *\,|\,\backslash n\,|\,\backslash b\ **\ m\,|\,c\,|\,y$, for some string y and non-white character c, or s does not contain \n.

If s does have a \n, then ISAT(s, m) will be true iff the left-most non-white character of s is exactly m blanks away from the preceding \n.

Definition of NEWL

NEWL(s) ::= **true** iff $s = \%\ *\,|\,\backslash n\,|\,x$, for some string x.

That is, NEWL(s) is true iff s has a \n preceding which there are no non-white characters.

4.2. The indented file

We say that a string s is lexically equivalent to t if both produce the same sequence of tokens. More formally, s and t are lexically equivalent if by replacing the inter-token white space by a single blank, and by deleting any white space prefix/suffix, if any, the resulting strings $s1$ and $t1$ become equal. (See also the next section.)

Given a file FI (the input) an indenting program should produce file FU such that

1. for each i, $1 \leqslant i \leqslant$ number of lines in FI, there exists a u, $1 \leqslant u \leqslant$ number of lines in FU, such that FI[1 .. i] and FU[1 .. u] are lexically equivalent where F[1..n] stands for the first n lines of file F,
2. PLOT(nt, \n|FU, 0) = **true**, and
3. no file with fewer lines than are in FU satisfies the above,

whenever FI is a sentence corresponding to a non-terminal nt of Pascal grammar. This is the specification of indenting programs that appeals to us. Part (3) guarantees that input lines are not split up unnecessarily. In part (2), a \n is prefixed to FU so as to treat the end of line character as a 'new line' character. Without this \n, a NEWL predicate might be false even though the first token of the very first line is at the correct margin. Note that the behaviour of the indenting program is unspecified when FI does not contain a legal construct of Pascal. Note also that part (1) of the specification implies that FU will have at least as many lines as in FI. It also rules out recombination of input lines and then splitting them up into output lines.