mentioned before, syntax validation is not in the domain of indenting programs we are considering.

Cases 2, 3, 4 and 5 would be simpler if Pascal had a different syntax. The special token DECL indicates that declarations (of labels, constants, types, variables and procedure/functions) are due next. If the last token of T is LPAREN, which can arise in a syntactically correct program only inside the parameter list, the tokens VAR, PROCEDURE and FUNCTION have no effect. The declarations end when a BEGIN is encountered; this is shown in case 5. Case 4 arises because FORWARD and EXTERN are not reserved words. They have the special meaning only when they appear immediately following the procedure headline.

Case 7 arises because of variant records with tag fields. In our specification COLON indents and it is, in this case, terminated by the OF.

5.3. Line splitting

Each split up part of a line is called a *segment*. As we shall see, there is a one-to-one correspondence between input segments and output lines. These two are in fact identical but for the prefix and suffix white spaces.

The function FIRSTSEG maps non-white prefixes of a line to its first segment, using the sets LO, and LC. The function SEGSEQ maps arbitrary strings to segment sequences. The set LO contains all (line opening) tokens whose corresponding words should always appear as the first non-white string in an output line. Similarly, the set LC contains all tokens which always close an output line but allow any immediately following comments. Thus the occurrence of a token from LO in the middle of an input line will split it just to the left of the token. The sets LO, LC are chosen to match the specifications of Figure 2.

LO ::= {PROCEDURE, FUNCTION, PROGRAM, LABEL, CONST, TYPE, VAR,
WHILE, REPEAT, UNTIL, IF, ELSE, CASE, GOTO}
LC ::= {SEMICOLON}

Intuitively, the segmentation of strings as produced by SEGSEQ can be explained as follows. Place imaginary markers as follows: (1) before the very first and after the very last characters of the string, (2) to the immediate right of every $\backslash n$, (3) to the immediate left of a token belonging to LO, and (4) to the immediate right of a token belonging to LC but skipping over comments following it. The strings thus enclosed between pairs of consecutive markers are segments. The functions FIRSTSEG and SEGSEQ imitate this process in a non-operational way.

Recall that we denote by 0ss, the empty sequence of segments, and by ! concatenation of segment sequences.

Definition of FIRSTSEG

Let w be a prefix of a line, and let Q = LEX(T, w).

1. Suppose lc ∘ COMBGN ∘ ORDINARY* is a suffix of T, where lc stands for a token from LC. Then FIRSTSEG(T, w) ::= w, if Q does not contain COMEND; otherwise FIRSTSEG(T, w) ::= x, where x is the longest prefix of w such that LEX(T, x) = ORDINARY* ∘ COMEND ∘ (COMBGN ∘ ORDINARY* ∘ COMEND)*.
2. Suppose lc ∘ COMBGN ∘ ORDINARY* is not a suffix of T. Then let x be the longest prefix of w such that LEX(T, x) does not contain (i) any token from LO except as its first token, or (ii) the subsequence lc ∘ (COMBGN ∘ ORDINARY* ∘ COMEND)* ∘ q, where $q \neq$ COMBGN. Then FIRSTSEG(T, w) = x.