Algorithm 4.19: Eliminating left recursion.

INPUT: Grammar G with no cycles or ϵ-productions.

OUTPUT: An equivalent grammar with no left recursion.

METHOD: Apply the algorithm in Fig. 4.11 to G. Note that the resulting non-left-recursive grammar may have ϵ-productions. □

```
1)  arrange the nonterminals in some order A_1, A_2, ..., A_n.
2)  for ( each i from 1 to n ) {
3)          for ( each j from 1 to i − 1 ) {
4)                  replace each production of the form A_i → A_j γ by the
                        productions A_i → δ_1 γ | δ_2 γ | ··· | δ_k γ, where
                        A_j → δ_1 | δ_2 | ··· | δ_k are all current A_j-productions
5)          }
6)          eliminate the immediate left recursion among the A_i-productions
7)  }
```

Figure 4.11: Algorithm to eliminate left recursion from a grammar

$$
\begin{aligned}
S &\rightarrow A\ a \ \mid\ b \\
A &\rightarrow A\ c \ \mid\ S\ d \ \mid\ \epsilon
\end{aligned}
\tag{4.18}
$$

Example 4.20: Let us apply Algorithm 4.19 to the grammar (4.18). Technically, the algorithm is not guaranteed to work, because of the ϵ-production, but in this case, the production $A \to \epsilon$ turns out to be harmless.

We order the nonterminals S, A. There is no immediate left recursion among the S-productions, so nothing happens during the outer loop for $i = 1$. For $i = 2$, we substitute for S in $A \to S\ d$ to obtain the following A-productions.

$$A \to A\ c \ \mid\ A\ a\ d \ \mid\ b\ d \ \mid\ \epsilon$$

Eliminating the immediate left recursion among these A-productions yields the following grammar.

$$\begin{aligned} S &\to A\ a \ \mid\ b \\ A &\to b\ d\ A' \ \mid\ A' \\ A' &\to c\ A' \ \mid\ a\ d\ A' \ \mid\ \epsilon \end{aligned}$$