repeat
  begin
    let \( X \) be the top stack symbol and \( a \) the next input symbol;
    if \( X \) is a terminal or \( \$ \) then
      if \( X = a \) then
        pop \( X \) from the stack and remove \( a \) from the input
      else
        ERROR( )
    else /* \( X \) is a nonterminal */
      if \( M[X, a] = X \to Y_1 Y_2 \cdots Y_k \) then
        begin
          pop \( X \) from the stack;
          push \( Y_k, Y_{k-1}, \ldots, Y_1 \) onto the stack, \( Y_1 \) on top
        end
      else
        ERROR( )
  end
until \( X = \$ \) /* stack has emptied */

Fig. 5.23. Predictive parsing program.

<table>
<thead>
<tr>
<th>id</th>
<th>+</th>
<th>*</th>
<th>(</th>
<th>)</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>( E )</td>
<td>( E \to TE' )</td>
<td>( E \to TE' )</td>
<td>E' ( \to +TE' )</td>
<td>( E' \to \epsilon )</td>
<td>( E' \to \epsilon )</td>
</tr>
<tr>
<td>( E' )</td>
<td>( T \to FT' )</td>
<td>( T \to FT' )</td>
<td>( T' \to \epsilon )</td>
<td>( T' \to *FT' )</td>
<td>( T' \to \epsilon )</td>
</tr>
<tr>
<td>( T' )</td>
<td>( F \to \text{id} )</td>
<td>( F \to (E) )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5.24. Parsing table for grammar (5.9).

Define \( \text{FOLLOW}(A) \), for nonterminal \( A \), to be the set of terminals \( a \) that can appear immediately to the right of \( A \) in some sentential form, that is, \( S \Rightarrow \alpha Aa\beta \) for some \( \alpha \) and \( \beta \). If \( A \) can be the rightmost symbol in some sentential form, then \( \epsilon \) is in \( \text{FOLLOW}(A) \).

To compute \( \text{FIRST}(X) \) for all grammar symbols \( X \), apply the following rules until no more terminals or \( \epsilon \) can be added to any \( \text{FIRST} \) set.
<table>
<thead>
<tr>
<th>Stack</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>$id + id \ast id$</td>
<td>$E \rightarrow TE'$</td>
</tr>
<tr>
<td>$SET$</td>
<td>$id + id \ast id$</td>
<td>$T \rightarrow FT'$</td>
</tr>
<tr>
<td>$SETF$</td>
<td>$id + id \ast id$</td>
<td>$F \rightarrow id$</td>
</tr>
<tr>
<td>$SET'id$</td>
<td>$id + id \ast id$</td>
<td>$T' \rightarrow \epsilon$</td>
</tr>
<tr>
<td>$SET'$</td>
<td>$id \ast id$</td>
<td>$E' \rightarrow +TE'$</td>
</tr>
<tr>
<td>$SETF$</td>
<td>$id \ast id$</td>
<td>$T \rightarrow FT'$</td>
</tr>
<tr>
<td>$SET'id$</td>
<td>$id \ast id$</td>
<td>$F \rightarrow id$</td>
</tr>
<tr>
<td>$SET'$</td>
<td>$\ast id$</td>
<td>$T' \rightarrow \ast FT'$</td>
</tr>
<tr>
<td>$SETF*$</td>
<td>$\ast id$</td>
<td>$T' \rightarrow \ast FT'$</td>
</tr>
<tr>
<td>$SETF$</td>
<td>$id$</td>
<td>$F \rightarrow id$</td>
</tr>
<tr>
<td>$SET'id$</td>
<td>$id$</td>
<td>$E \rightarrow \epsilon$</td>
</tr>
<tr>
<td>$SET'$</td>
<td>$$$</td>
<td>$T' \rightarrow \epsilon$</td>
</tr>
<tr>
<td>$E'$</td>
<td>$$$</td>
<td>$E' \rightarrow \epsilon$</td>
</tr>
</tbody>
</table>

Fig. 5.25. Moves by predictive parser.

1. If $X$ is terminal, then $\text{FIRST}(X)$ is \{X\}.

2. If $X$ is nonterminal and $X \rightarrow a\alpha$ is a production, then add $a$ to $\text{FIRST}(X)$. If $X \rightarrow \epsilon$ is a production, then add $\epsilon$ to $\text{FIRST}(X)$.

3. If $X \rightarrow Y_1Y_2 \cdots Y_k$ is a production, then for all $i$ such that all of $Y_1, \ldots, Y_{i-1}$ are nonterminals and $\text{FIRST}(Y_i)$ contains $\epsilon$ for $j = 1, 2, \ldots, i-1$ (i.e. $Y_1Y_2 \cdots Y_{i-1} \Rightarrow \epsilon$), add every non-$\epsilon$ symbol in $\text{FIRST}(Y_i)$ to $\text{FIRST}(X)$. If $\epsilon$ is in $\text{FIRST}(Y_j)$ for all $j = 1, 2, \ldots, k$, then add $\epsilon$ to $\text{FIRST}(X)$.

Now, we can compute $\text{FIRST}$ for any string $X_1X_2 \cdots X_n$ as follows. Add to $\text{FIRST}(X_1X_2 \cdots X_n)$ all the non-$\epsilon$ symbols of $\text{FIRST}(X_1)$. Also add the non-$\epsilon$ symbols of $\text{FIRST}(X_2)$ if $\epsilon$ is in $\text{FIRST}(X_1)$, the non-$\epsilon$ symbols of $\text{FIRST}(X_2)$ if $\epsilon$ is in both $\text{FIRST}(X_1)$ and $\text{FIRST}(X_2)$, and so on. Finally, add $\epsilon$ to $\text{FIRST}(X_1X_2 \cdots X_n)$ if, for all $i$, $\text{FIRST}(X_i)$ contains $\epsilon$.

To compute $\text{FOLLOW}(A)$ for all nonterminals $A$, apply the following rules until nothing can be added to any $\text{FOLLOW}$ set.
1. \( \epsilon \) is in FOLLOW\((S)\), where \( S \) is the start symbol.

2. If there is a production \( A \rightarrow \alpha B \beta \), then everything in FIRST\((\beta)\) but \( \epsilon \) is in FOLLOW\((B)\). Note that \( \epsilon \) may still wind up in FOLLOW\((B)\) by rule (3).

3. If there is a production \( A \rightarrow \alpha B \), or a production \( A \rightarrow \alpha B \beta \) where FIRST\((\beta)\) contains \( \epsilon \) (i.e., \( \beta \xrightarrow{*} \epsilon \)), then everything in FOLLOW\((A)\) is in FOLLOW\((B)\).

**Example 5.19.** Consider again grammar (5.9)

\[
\begin{align*}
E & \rightarrow TE' \\
E' & \rightarrow +TE' \mid \epsilon \\
T & \rightarrow FT' \\
T' & \rightarrow \ast FT' \mid \epsilon \\
F & \rightarrow (E) \mid \text{id}
\end{align*}
\]

Then:

\[
\begin{align*}
\text{FIRST}(E) &= \text{FIRST}(T) = \text{FIRST}(F) = \{(,), \text{id}\}. \\
\text{FIRST}(E') &= \{+, \epsilon\} \\
\text{FIRST}(T') &= \{\ast, \epsilon\} \\
\text{FOLLOW}(E) &= \text{FOLLOW}(E') = \{, \epsilon\} \\
\text{FOLLOW}(T) &= \text{FOLLOW}(T') = \{+, ,\}, \epsilon\} \\
\text{FOLLOW}(F) &= \{+, \ast, ,\}, \epsilon\}
\end{align*}
\]

For example, \( \text{id} \) and left parenthesis are added to FIRST\((F)\) by rule (2) in the definition of FIRST. Then by rule (3) with \( i = 1 \), the production \( T \rightarrow FT' \) implies that \( \text{id} \) and left parenthesis are in FIRST\((T)\) as well.

To compute FOLLOW sets, we put \( \epsilon \) in FOLLOW\((E)\) by rule (1). By rule (2) applied to production \( F \rightarrow (E) \), the right parenthesis is also in FOLLOW\((E)\). By rule (3) applied to production \( E \rightarrow TE' \), \( \epsilon \) and right parenthesis are in FOLLOW\((E')\). Since \( E' \xrightarrow{*} \epsilon \), they are also in FOLLOW\((T)\). For a last example of how the FOLLOW rules are applied, the production \( E \rightarrow TE' \) implies, by rule (2), that everything other than \( \epsilon \) in FIRST\((E')\) must be placed in FOLLOW\((T)\). We have already seen that \( \epsilon \) is in FOLLOW\((T)\) anyway. □
Construction of Parsing Tables

The following algorithm can be used to construct a predictive parsing table for a grammar $G$. The idea behind the algorithm is simple. Suppose $A → α$ is a production with $a$ in FIRST($α$). Then, whenever the parser has $A$ on top of the stack with $a$ the current input symbol, the parser will expand $A$ by $α$. The only complication occurs when $α = ε$ or $α → ε$. In this case, we should also expand $A$ by $α$ if the current input symbol is in FOLLOW($A$), or if the $\$$ on the input has been reached and $ε$ is in FOLLOW($A$).

**Algorithm 5.4.** Constructing a predictive parsing table.

*Input.* Grammar $G$.

*Output.* Parsing table $M$.

*Method.*

1. For each production $A → α$ of the grammar, do steps 2 and 3.
2. For each terminal $a$ in FIRST($α$), add $A → α$ to $M[A, a]$.
3. If $ε$ is in FIRST($α$), add $A → α$ to $M[A, b]$ for each terminal $b$ in FOLLOW($A$). If $ε$ is in FIRST($α$) and in FOLLOW($A$), add $A → a$ to $M[A, \$$].
4. Make each undefined entry of $M$ error. □

**Example 5.20.** Let us apply Algorithm 5.4 to grammar (5.9). Since FIRST($TE'$) = FIRST($T$) = $\{[, \text{id}]\}$, production $E → TE'$ causes $M[E, [\} \text{id}]$ to acquire the entry $E → TE'$.

Production $E' → +TE'$ causes $M[E', +]$ to acquire $E' → +TE'$. Production $E' → ε$ causes $M[E', ]$ and $M[E', \$$] to acquire $E' → ε$ since FOLLOW($E'$) = $\{[, \} \text{id}]\}$.

The parsing table produced by Algorithm 5.4 for $G$ was shown in Fig. 5.24. □

**LL(1) Grammars**

Algorithm 5.4 can be applied to any grammar $G$ to produce a parsing table $M$. For some grammars, however, $M$ may have some entries that are multiply-defined. For example, if $G$ is left-recursive or ambiguous, then $M$ will have at least one multiply-defined entry.

**Example 5.21.** Consider the grammar from Example 5.16.

$S → iCtSS' \mid a$

$S' → eS \mid ε$ \hspace{1cm} (5.11)

$C → b$

The parsing table for grammar (5.11) is shown in Fig. 5.26.