■ **Example 4.17.** Consider again grammar (4.11), repeated below:

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \varepsilon$

$F \rightarrow (E) \mid id$

- Then:

  FIRST(E) = FIRST(T) = FIRST(F) = {(,id } .

  FIRST(E') = {+, ε }

  FIRST(T') = {*, ε }

  FOLLOW(E) = FOLLOW(E') = {) , $ }

  FOLLOW(T) = FOLLOW(T') = { +,) , $ }

  FOLLOW(F) = { +, *,) , $ }

Grammar Analysis Algorithms (Cont'd)

- Follow(A)
  - A is any nonterminal
  - Follow(A) is the set of terminals that my follow A in some sentential form

  $\text{Follow}(A)=\{a\in V_t \mid S\Rightarrow^* \ldots Aa \ldots\}\cup$
  $\{\text{if } S\Rightarrow^+ \alpha A \text{ then } \{\lambda\} \text{ else } \varnothing\}$
- First(α)
  - The set of all the terminal symbols that can begin a sentential form derivable from α
  - If α is the right-hand side of a production, then First(α) contains terminal symbols that begin strings derivable from α

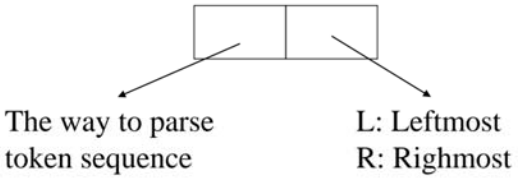
  $\text{First}(\alpha)=\{a\in V_t \mid \alpha\Rightarrow^* a\beta\}\cup$
  $\{\text{if } \alpha\Rightarrow^* \lambda \text{ then } \{\lambda\} \text{ else } \varnothing\}$

Grammar Analysis Algorithms (Cont'd)

- Definition of C data structures and subroutines
  - first_set[X]
    - contains terminal symbols and λ
    - X is any single vocabulary symbol
  - follow_set[A]
    - contains terminal symbols and λ
    - A is a nonterminal symbol

Parsers and Recognizers (Cont'd)

- Naming of parsing techniques



- Top-down
  - LL
- Bottom-up
  - LR

Parsers and Recognizers (Cont'd)

- Two general approaches to parsing
  - Top-down parser
    - Expanding the parse tree (via predictions) in a depth-first manner
    - Preorder traversal of the parse tree
    - ***Predictive*** in nature
    - lm
    - LL

Parsers and Recognizers (Cont'd)

  - Buttom-down parser
    - Beginning at its bottom (the leaves of the tree, which are terminal symbols) and determining the productions used to generate the leaves
    - Postorder traversal of the parse tree
    - rm
    - LR

The LL(1) Predict Function

- Given the productions

  $A \rightarrow \alpha_1$

  $A \rightarrow \alpha_2$

  …

  $A \rightarrow \alpha_n$

- During a (leftmost) derivation

  $\ldots A \ldots \Rightarrow \ldots \alpha_1 \ldots$ ***or***

  $\ldots \alpha_2 \ldots$ ***or***

  $\ldots \alpha_n \ldots$

- Deciding which production to match
  - Using lookahead symbols

The LL(1) Predict Function

Single Symbol Lookahead

$$\text{Predict}(A \rightarrow X_1 \cdots X_m) = \begin{cases} \text{if } \lambda \in \text{First}(X_1 \cdots X_m) \\ \quad (\text{First}(X_1 \cdots X_m) - \lambda) \cup \text{Follow}(A) \\ \text{else} \\ \quad \text{First}(X_1 \cdots X_m) \end{cases}$$

- The limitation of LL(1)
  - LL(1) contains exactly those grammars that have disjoint predict sets for productions that share a common left-hand side

1 <program> → **begin** <statement list> **end**