CS480
Translators
Top-down Parsing
Chap. 4
Eliminate Left Recursion

• Immediate Left Recursion
  \[ A \rightarrow A\alpha_1 | A\alpha_2 | \ldots | A\alpha_m | \beta_1 | \beta_2 | \ldots | \beta_n \]
  \[ A \rightarrow \beta_1 A' | \beta_2 A' | \ldots | \beta_n A' \]
  \[ A' \rightarrow \alpha_1 A' | \alpha_2 A' | \ldots | \alpha_m A' | \varepsilon \]

• Example:
  \[ E \rightarrow E + E | E \times E | (E) | id \]
  \[ E \rightarrow (E) E' | id E' \]
  \[ E' \rightarrow + E E' | \times E E' | \varepsilon \]
Eliminate Left Recursion cont.

• Example:
  \[ S \rightarrow A \ a \mid b \]
  \[ A \rightarrow A \ c \mid S \ d \mid \varepsilon \]

• Not immediate
  \[ S \Rightarrow A \ a \Rightarrow S \ d \ a \]

• Substitute all S productions in A
  \[ A \rightarrow A \ c \mid A \ a \ d \mid b \ d \mid \varepsilon \]
Eliminate Left Factoring

A→ αβ₁ | αβ₂
• A→ αA’
• A’→ β₁ | β₂

• Example:

stmt → if expr then stmt else stmt | if expr then stmt

stmt → if expr then stmt E
E→ else stmt | ε
Criteria for Parsing

• Efficient – proportional to size
• Determine action by fixed # tokens
• Practical Considerations
  – 1 Lookahead
  – No backtracking
  – LL(1) grammar

• What is LL(k)?
LL Grammars

• Top-down Parsing
  – Recursive descent
    • General
    • Predictive
  – Table-driven
Top Down Parsing

```c
void A() {
    1) Choose an A-production, A → X_1 X_2 ... X_k;
    2) for ( i = 1 to k ) {
        3) if ( X_i is a nonterminal )
            call procedure X_i();
        4) else if ( X_i equals the current input symbol a )
            advance the input to the next symbol;
        5) else /* an error has occurred */;
    }
}
```

- How does this change for the production below?

```
A -> ab | a
```
Defining an LL Grammar

- Need to definitions:
- **First** and **Follow**

Figure 4.15: Terminal $c$ is in FIRST($A$) and $a$ is in FOLLOW($A$)
First(α)

• If α is any string in grammar, First(α) is set of terminals that begin strings derived from α.

  * If α ⇒ ε, then ε is in First(α).

• What does it mean if A→α | β, and First(α) and First(β) are disjoint?
Follow(A)

• If A is a nonterminal in grammar, Follow(A) is set of terminals that can appear immediately to the right of A.
• If A can be the rightmost symbol, then $ is in Follow(A).
• What is $?
Compute First(X) for all symbols

• If X is a terminal, then First(X) = \{X\}
• If X is a nonterminal and X \to Y_1Y_2...Y_k is a production, then place a in First(X) if for some i, a is in First(Y_i) and \epsilon is in all First(Y_1), ..., First(Y_{i-1}). If \epsilon is in First(Y_j) for all j=1, 2, ..., k, then add \epsilon First(X).
• If X \to \epsilon, then \epsilon is in First(X)
Compute Follow(A) for nonterminals

- Place $ in Follow(S), where S is start symbol and $ is the input endmarker.
- If there is a production $A \rightarrow \alpha B \beta$, then everything in First(\beta), except \epsilon, is in Follow(B).
- If there is a production $A \rightarrow \alpha B$ or $A \rightarrow \alpha B \beta$, where \epsilon is in First(\beta), then everything in Follow(A) is Follow(B).
Example First and Follow

\[ E \rightarrow T E' \]
\[ E' \rightarrow + T E' \mid \varepsilon \]
\[ T \rightarrow F T' \]
\[ T' \rightarrow * F T' \mid \varepsilon \]
\[ F \rightarrow ( E ) \mid \text{id} \]

- First(E), First(E’), First(T), First(T’), First(F)?
- Follow(E), Follow(E’), Follow(T), Follow(T’), Follow(F)?