Compiler Design

Lecture-14

Constructing the Parsing Table I: First and Follow

Topics Covered

- Left factoring of a grammar
- Predictive Parser
- Constructing the Parsing Table I: First and Follow

Left-Factoring a Grammar II

- Here is the procedure used to left-factor a grammar:
  - For each non-terminal A, find the longest prefix α common to two or more of its alternatives.
  - Replace all the A productions:

    $A \rightarrow \alpha\beta 1 \mid \alpha\beta 2 \ldots \mid \alpha\beta n \mid \gamma$

    (where γ represents all alternatives that do not begin with α)
  - By:

    $A \rightarrow \alpha A' \mid \gamma$

    $A' \rightarrow \beta 1 \mid \beta 2 \mid \ldots \mid \beta n$

Left-Factoring a Grammar III

- Here is an example of a common grammar that needs left factoring:

$$S \rightarrow iEtS \mid iEtSeS \mid a$$
$$E \rightarrow b$$

(i stands for “if”; t stands for “then”; and e stands for “else”)

- Left factored, this grammar becomes:

$$S \rightarrow iEtSS' \mid a$$
$$S' \rightarrow eS \mid \epsilon$$
$$E \rightarrow b$$

Predictive Parser: Details

- The key problem during predictive parsing is that of determining the production to be applied for a non-terminal.

- This is done by using a parsing table.

- A parsing table is a two-dimensional array M[A,a] where A is a non-terminal, and a is a terminal or the symbol $, menaing "end of input string".

- The other inputs of a predictive parser are:
  - The input buffer, which contains the string to be parsed followed by $.
  - The stack which contains a sequence of grammar symbols with, initially, $S (end of input string and start symbol) in it.

Predictive Parser: Informal Procedure

- The predictive parser considers X, the symbol on top of the stack, and a, the current input symbol. It uses, M, the parsing table.
  - If X=a=$ ➔ halt and return success
  - If X=a≠$ ➔ pop X off the stack and advance input pointer to the next symbol
  - If X is a non-terminal ➔ Check M[X,a]
    - If the entry is a production rule, then replace X on the stack by the Right Hand Side of the production
    - If the entry is blank, then halt and return failure

Predictive Parser: An Example

	id	+	*	(	)	$
E	E→ TE'			E→ TE'		
E'		E' →+ TE'			E' →ϵ	E' →ϵ
T	T→ FT'			T→ FT'		
T'		T' →ϵ	T' →* FT'		T' →ϵ	T' →ϵ
F	F→ id			F→ (E)		

Parsing Table

Algorithm Trace →

Stack	Input	Output
$E	id+id*id$	
$E'T	id+id*id$	E → TE'
$E'T'F	id+id*id$	T → FT'
$E'T'id	id+id*id$	F → id
$E'T'	+id*id$	
$E'	+id*id$	T' → ϵ
$E'T+	+id*id$	E' → +TE'
$E'T	id*id$	
$E'T'F	id*id$	T → FT'
$E'T'id	id*id$	F → id
$E'T'	*id$	
$E'T'F*	*id$	T' → *FT'
$E'T'F	id$	
$E'T'id	id$	F → id
$E'T'	$	
$E'	$	T' → ϵ
$	$	E' → ϵ

Constructing the Parsing Table I: First and Follow

- <u>First(α)</u> is the set of terminals that begin the strings derived from α. <u>Follow(A)</u> is the set of terminals a that can appear to the right of A. First and Follow are used in the construction of the parsing table.
- <u>Computing First:</u>
  - If X is a terminal, then First(X) is {X}
  - If X → ϵ is a production, then add ϵ to First(X)
  - If X is a non-terminal and X → Y1 Y2 … Yk is a production, then place a in First(X) if for some i, a is in First(Yi) and ϵ is in all of First(Y1)…First(Yi-1)

Constructing the Parsing Table II: First and Follow

- <u>Computing Follow:</u>
  - Place \$ in Follow(S), where S is the start symbol and \$ is the input right endmarker.
  - If there is a production $A \rightarrow \alpha B \beta$, then everything in First(β) except for ϵ is placed in Follow(B).
  - If there is a production $A \rightarrow \alpha B$, or a production $A \rightarrow \alpha B \beta$ where First(β) contains ϵ, then everything in Follow(A) is in Follow(B)

Example: $E \rightarrow TE'$ $\quad E' \rightarrow +TE' \mid \epsilon$
$T \rightarrow FT'$ $\quad T' \rightarrow *FT' \mid \epsilon$
$F \rightarrow (E) \mid id$

First(E) = First(T) = First(F) = {(, id} First(E') = {+, ϵ}
First(T') = {*, ϵ}

Follow(E) = Follow(E') = {),\$}
Follow(F)={+,*,),\$}
Follow(T) = Follow(T') = {+,),\$}

Constructing the Parsing Table III

- Algorithm for constructing a predictive parsing table:
  1. For each production $A \rightarrow \alpha$ of the grammar, do steps 2 and 3
  2. For each terminal a in First(α), add $A \rightarrow \alpha$ to M[A, a]
  3. If ϵ is in First(α), add $A \rightarrow \alpha$ to M[A, b] for each terminal b in Follow(A). If ϵ is in First(α), add $A \rightarrow \alpha$ to M[A,b] for each terminal b in Follow(A). If ϵ is in First(α) and \$ is in Follow(A), add $A \rightarrow \alpha$ to M[A, \$].
  4. Make each undefined entry of M be an error.

LL(1) Grammars

- A grammar whose parsing table has no multiply-defined entries is said to be LL(1)
- No ambiguous or left-recursive grammar can be LL(1).
- A grammar G is LL(1) iff whenever $A \rightarrow \alpha \mid \beta$ are two distinct productions of G, then the following conditions hold:
  - For no terminal a do both α and β derive strings beginning with a
  - At most one of α and β can derive the empty string
  - If β can (directly or indirectly) derive ϵ, then α does not derive any string beginning with a terminal in Follow(A).