CSCI 742 - Compiler Construction

Lecture 13

Predictive Parsing Table Construction

Instructor: Hossein Hojjat

February 14, 2018

Recap: Conversion to LL(1)

- Consider the following grammar for additive expressions:

$$S \to E$$
$$E \to E + F \mid \texttt{num}$$
$$F \to (E) \mid \texttt{num}$$

- Grammar has left-recursion so is not LL(1)
- Grammar is LL(1) after removing left-recursion:

$$S \to E$$
$$E \to \texttt{num}\ E'$$
$$E' \to +FE' \mid \epsilon$$
$$F \to (E) \mid \texttt{num}$$

Predictive Parser Implementation

$$S \to E$$
$$E \to \texttt{num}\ E'$$
$$E' \to +FE' \mid \epsilon$$
$$F \to (E) \mid \texttt{num}$$

- Predictive parsing table:

	num	+	(	)	$
S	$\to E$				
E	$\to \texttt{num}\ E'$				
E'		$\to +FE'$		$\to \epsilon$	$\to \epsilon$
F	$\to \texttt{num}$		$\to (E)$		

- Implement a recursive descent parser using the mutually recursive procedures `parse_S`, `parse_E`, `parse_E'` and `parse_F`

Construct Parsing Tables

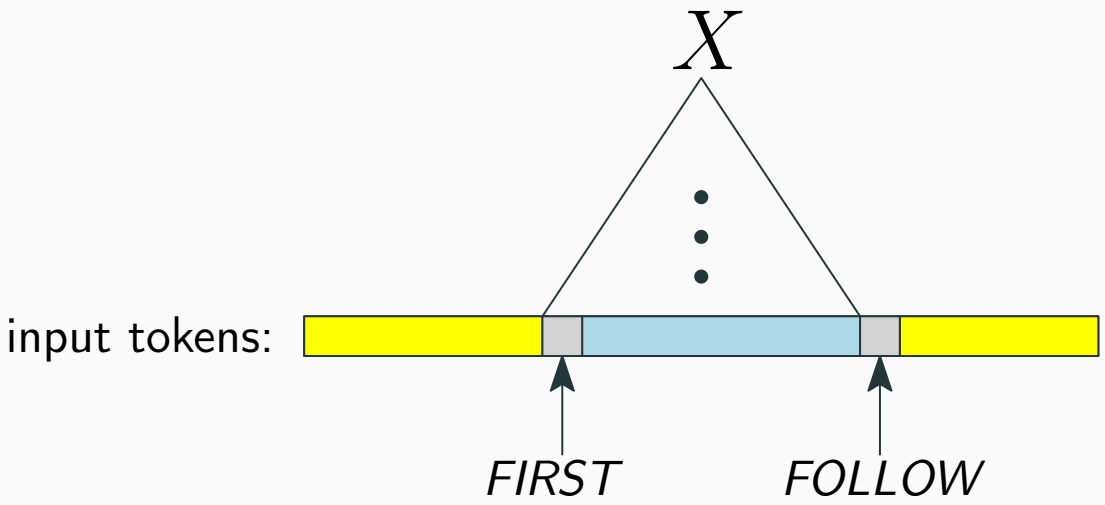
- Need an algorithm to generate predictive parse tables from an LL(1) grammar

$$S \to E$$
$$E \to \texttt{num}\ E'$$
$$E' \to +FE' \mid \epsilon$$
$$F \to (E) \mid \texttt{num}$$

	num	+	(	)	$
S	$\to E$				
E	$\to \texttt{num}\ E'$				
E'		$\to +FE'$		$\to \epsilon$	$\to \epsilon$
F	$\to \texttt{num}$		$\to (E)$		

Prerequisite Definitions

- Parsing table determines for each non-terminal and each look-ahead symbol what one production to use
- *FIRST*(γ) for arbitrary string of terminals and non-terminals γ is: set of symbols that might begin the fully expanded version of γ
- *FOLLOW*(X) for a non-terminal X is:
  set of symbols that might follow the derivation of X in the input token stream



Parse Table Entries

- Consider a production $X \to \gamma$
- Add $\to \gamma$ to the X row for each symbol in *FIRST*(γ)
- If γ can derive ϵ (γ is nullable), add $\to \gamma$ for each symbol in *FOLLOW*(X)
- Grammar is LL(1) if no conflicting entries

$S \to E$
$E \to \texttt{num}\ E'$
$E' \to +FE' \mid \epsilon$
$F \to (E) \mid \texttt{num}$

	num	+	(	)	$
S	$\to E$				
E	$\to \texttt{num}\ E'$				
E'		$\to +FE'$		$\to \epsilon$	$\to \epsilon$
F	$\to \texttt{num}$		$\to (E)$		

Computing nullable

- X is nullable if it can derive the empty string
- If it derives ϵ directly ($X \to \epsilon$)
- If it has a production $X \to Y_1 Y_2 \cdots Y_n$ where all RHS symbols (Y_1, Y_2, $\cdots$, Y_n) are nullable
- **Algorithm:** assume all non-terminals non-nullable, apply rules repeatedly until no change in status

Computing *FIRST*

Definition. $\mathit{FIRST}(\gamma) = \{a \mid \gamma \Rightarrow^* a\beta\} \cup \{\epsilon \mid \gamma \Rightarrow^* \epsilon\}$

1. $\mathit{FIRST}(a) = \{a\}$

2. For all productions $X \to Y_1 \cdots Y_n$

- Add $\mathit{FIRST}(Y_1) - \{\epsilon\}$ to $\mathit{FIRST}(X)$. Stop if $\epsilon \notin \mathit{FIRST}(Y_1)$
- Add $\mathit{FIRST}(Y_2) - \{\epsilon\}$ to $\mathit{FIRST}(X)$. Stop if $\epsilon \notin \mathit{FIRST}(Y_2)$
- $\cdots$
- Add $\mathit{FIRST}(Y_n) - \{\epsilon\}$ to $\mathit{FIRST}(X)$. Stop if $\epsilon \notin \mathit{FIRST}(Y_n)$
- Add ϵ to $\mathit{FIRST}(X)$

Exercise

Question.

- Compute the *FIRST* sets for non-terminals

$$
\begin{aligned}
E &\to TE' \\
E' &\to +TE' \mid \epsilon \\
T &\to FT' \\
T' &\to *FT' \mid \epsilon \\
F &\to (E) \mid \texttt{num}
\end{aligned}
$$

Exercise

Question.

- Compute the *FIRST* sets for non-terminals

$$
\begin{aligned}
E &\to TE' \\
E' &\to +TE' \mid \epsilon \\
T &\to FT' \\
T' &\to *FT' \mid \epsilon \\
F &\to (E) \mid \texttt{num}
\end{aligned}
$$

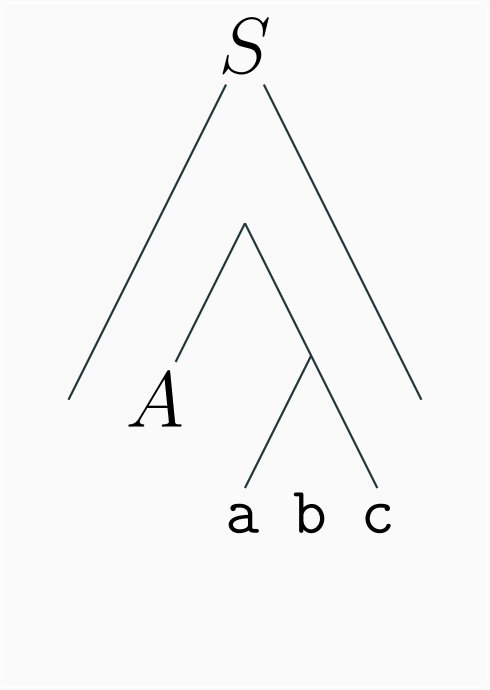
Answer.

$\mathit{FIRST}(E) = \{\ (\ ,\ \texttt{num}\ \}$ $\qquad$ $\mathit{FIRST}(E') = \{+\ ,\ \epsilon\}$

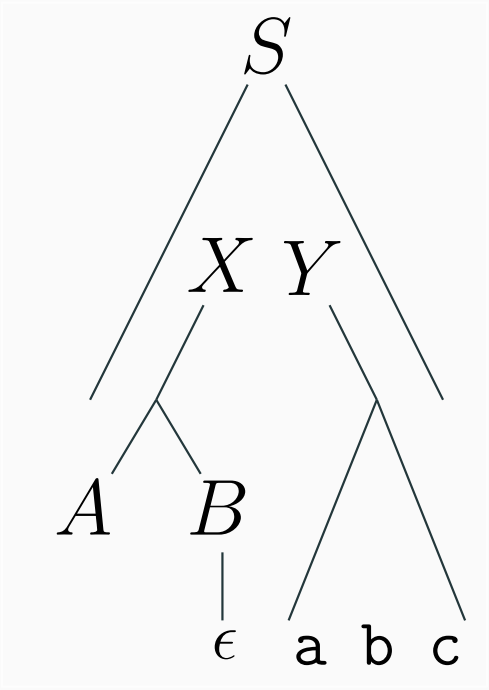
$\mathit{FIRST}(T) = \{\ (\ ,\ \texttt{num}\ \}$ $\qquad$ $\mathit{FIRST}(T') = \{*\ ,\ \epsilon\}$

$$\mathit{FIRST}(F) = \{\ (\ ,\ \texttt{num}\}$$

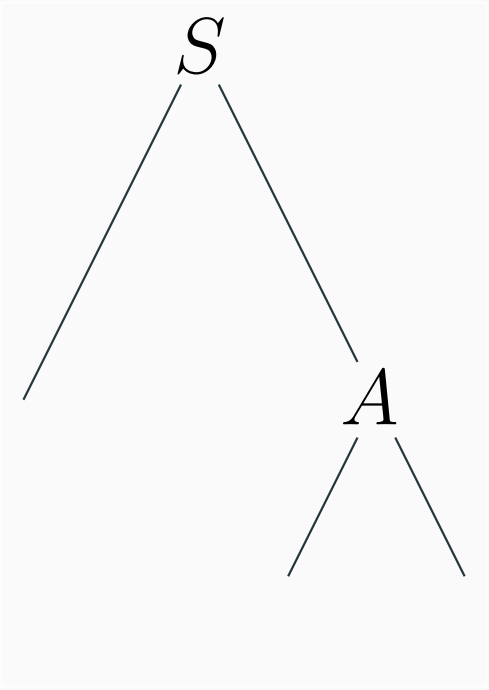
Computing *FOLLOW*



a is in *FOLLOW*(A)



a is in *FOLLOW*(A)



$ is in *FOLLOW*(A)

Computing *FOLLOW*

Definition. $\mathit{FOLLOW}(X) = \{a \mid S \Rightarrow^* \beta X a \gamma\}$

1. Compute the *FIRST* sets for all non-terminals first

2. Add \$ to $\mathit{FOLLOW}(S)$ (S is the start non-terminal)

3. For all productions $Z \to \cdots X Y_1 \cdots Y_n$

- Add $\mathit{FIRST}(Y_1) - \{\epsilon\}$ to $\mathit{FOLLOW}(X)$. Stop if $\epsilon \notin \mathit{FIRST}(Y_1)$
- Add $\mathit{FIRST}(Y_2) - \{\epsilon\}$ to $\mathit{FOLLOW}(X)$. Stop if $\epsilon \notin \mathit{FIRST}(Y_2)$
- $\cdots$
- Add $\mathit{FIRST}(Y_n) - \{\epsilon\}$ to $\mathit{FOLLOW}(X)$. Stop if $\epsilon \notin \mathit{FIRST}(Y_n)$
- Add $\mathit{FOLLOW}(Z)$ to $\mathit{FOLLOW}(X)$

Exercise

Question.

- Compute the *FOLLOW* sets for non-terminals (E is the start)

$$
\begin{aligned}
E &\to TE' \\
E' &\to +TE' \mid \epsilon \\
T &\to FT' \\
T' &\to *FT' \mid \epsilon \\
F &\to (E) \mid \texttt{num}
\end{aligned}
$$

Exercise

Question.

- Compute the *FOLLOW* sets for non-terminals (E is the start)

$$
\begin{aligned}
E &\to TE' \\
E' &\to +TE' \mid \epsilon \\
T &\to FT' \\
T' &\to *FT' \mid \epsilon \\
F &\to (E) \mid \texttt{num}
\end{aligned}
$$

Answer.

$$\mathit{FOLLOW}(E) = \{\ ) \ , \ \$\} \qquad \mathit{FOLLOW}(E') = \{\ ) \ , \ \$\}$$
$$\mathit{FOLLOW}(T) = \{\ ) \ , \ + \ , \$\} \qquad \mathit{FOLLOW}(T') = \{\ ) \ , \ + \ , \$\}$$
$$\mathit{FOLLOW}(F) = \{\ ) \ , \ * \ , \ + \ , \$\}$$

Ambiguous Grammars

- Construction of predictive parse table for ambiguous grammar results in conflicts
  - but converse does not hold

$$S \to S + S \mid S * S \mid \mathtt{num}$$

- $\mathit{FIRST}(S + S) = \mathit{FIRST}(S * S) = \mathit{FIRST}(\mathtt{num}) = \{\mathtt{num}\}$

	num	+	$
S	$\to \mathtt{num}, \to S + S, \to S * S$		

Exercise

Construct the predictive parsing table for the following LL(1) grammar:

$$
\begin{aligned}
\mathit{Stmt} &\to \texttt{if}\ \mathit{Expr}\ \texttt{then}\ \mathit{Stmt}\ \texttt{else}\ \mathit{Stmt} && (1)\\
&\mid \texttt{while}\ \mathit{Expr}\ \texttt{do}\ \mathit{Stmts} && (2)\\
&\mid \texttt{block}\ \mathit{Stmts} && (3)\\
\mathit{Stmts} &\to \mathit{Stmt}\ \texttt{;}\ \mathit{Stmts} && (4)\\
&\mid \epsilon && (5)\\
\mathit{Expr} &\to \texttt{id} && (6)
\end{aligned}
$$

	id	;	block	do	while	else	then	if	$
Stmt									
Stmts									
Expr									

Exercise

Construct the predictive parsing table for the following LL(1) grammar:

$$
\begin{aligned}
\mathit{Stmt} &\to \texttt{if}\ \mathit{Expr}\ \texttt{then}\ \mathit{Stmt}\ \texttt{else}\ \mathit{Stmt} && (1)\\
&\mid \texttt{while}\ \mathit{Expr}\ \texttt{do}\ \mathit{Stmts} && (2)\\
&\mid \texttt{block}\ \mathit{Stmts} && (3)\\
\mathit{Stmts} &\to \mathit{Stmt}\ \texttt{;}\ \mathit{Stmts} && (4)\\
&\mid \epsilon && (5)\\
\mathit{Expr} &\to \texttt{id} && (6)
\end{aligned}
$$

	id	;	block	do	while	else	then	if	$
Stmt			3		2			1	
Stmts		5	4		4	5		4	5
Expr	6								