-Syntax Analysis-Predictive Parser

FIRST

To compute FIRST(X) for all grammar symbols X, apply the following rules until no more terminals or ϵ can be added to any FIRST set.

1. If X is a terminal, then FIRST(X) = $\{X\}$.
2. If X is a nonterminal and $X \to Y_1 Y_2 \cdots Y_k$ is a production for some $k \geq 1$, then place a in FIRST(X) if for some i, a is in FIRST(Y_i), and ϵ is in all of FIRST(Y_1), ... , FIRST(Y_{i-1}); that is, $Y_1 \cdots Y_{i-1} \stackrel{*}{\Rightarrow} \epsilon$. If ϵ is in FIRST(Y_j) for all $j = 1, 2, \ldots, k$, then add ϵ to FIRST(X). For example, everything in FIRST(Y_1) is surely in FIRST(X). If Y_1 does not derive ϵ, then we add nothing more to FIRST(X), but if $Y_1 \stackrel{*}{\Rightarrow} \epsilon$, then we add FIRST($Y_2$), and so on.
3. If $X \to \epsilon$ is a production, then add ϵ to FIRST(X).

FOLLOW

To compute FOLLOW(A) for all nonterminals A, apply the following rules until nothing can be added to any FOLLOW set.

1. Place \$ in FOLLOW(S), where S is the start symbol, and \$ is the input right endmarker.

2. If there is a production $A \to \alpha B \beta$, then everything in FIRST(β) except ϵ is in FOLLOW(B).

3. If there is a production $A \to \alpha B$, or a production $A \to \alpha B \beta$, where FIRST(β) contains ϵ, then everything in FOLLOW(A) is in FOLLOW(B).

Predictive Parser or LL(1) parser

- Is a top-down parser
- Given **a grammar G** and an **input string** to be tested…….
- Start from the root or start variable S
- Construct the tree in top-down fashion (leaves at the bottom)

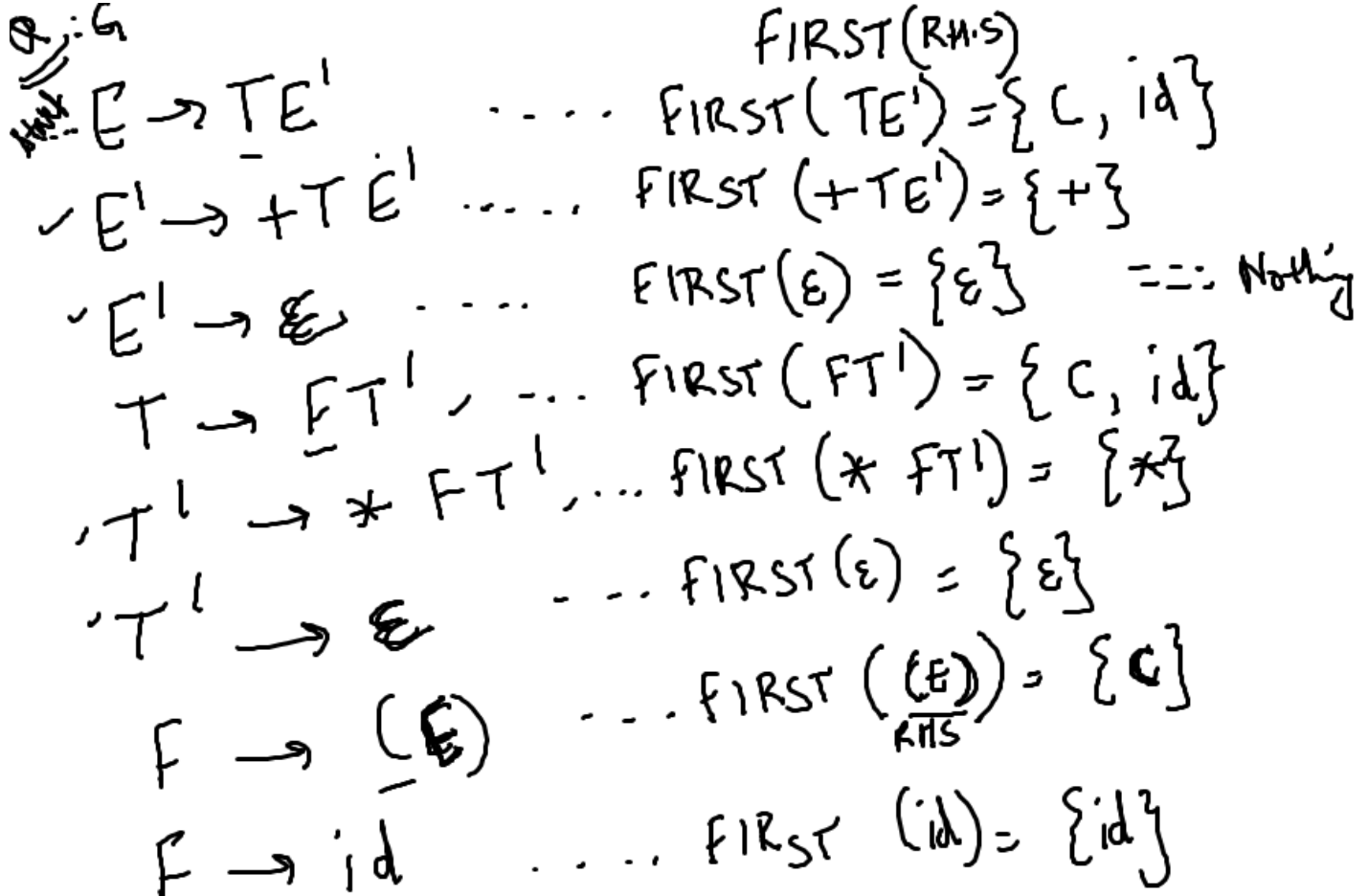
PREDICTIVE PARSER

- Remove left recursion (if any) from the grammar A→Da (in G) D→A
- Construct parsing table from given grammar
- Table showing sequence of moves for the given input string & parsing table

Step 1: Remove left recursion (if any) from the grammar: Check if for any variable A, $A \stackrel{+}{\Rightarrow} A\alpha$

$$
\begin{array}{lcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathbf{id}
\end{array}
\tag{4.28}
$$

FIRST(R.H.S) and FOLLOW(Non-Term)



FOLLOW(E) = {$,)} ✓

FOLLOW(E') = FOLLOW(E) = {$,)} ✓

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

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

FOLLOW(P) = FIRST(T') = {*, +, $,)}

FIRST

term = Σ = {+, *, (,), id, $}

N.T = {E, E', T, T', F}

Step 2: Construct parsing table from given grammar

Algorithm 4.31 : Construction of a predictive parsing table.

INPUT: Grammar G.

OUTPUT: Parsing table M.

METHOD: For each production $A \to \alpha$ of the grammar, do the following:

1. For each terminal a in FIRST(α), add $A \to \alpha$ to $M[A, a]$.
2. If ϵ is in FIRST(α), then for each terminal b in FOLLOW(A), add $A \to \alpha$ to $M[A, b]$. If ϵ is in FIRST(α) and \$ is in FOLLOW(A), add $A \to \alpha$ to $M[A, \$]$ as well.

If, after performing the above, there is no production at all in $M[A, a]$, then set $M[A, a]$ to **error** (which we normally represent by an empty entry in the table). □

Example 4.32 : For the expression grammar (4.28), Algorithm 4.31 produces the parsing table in Fig. 4.17. Blanks are error entries; nonblanks indicate a production with which to expand a nonterminal.

NON-TERMINAL	INPUT SYMBOL					
	id	+	*	(	)	\$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$	$T' \to \epsilon$
F	$F \to$ **id**			$F \to (E)$		

Figure 4.17: Parsing table M for Example 4.32

Step 3: Sequence of moves for input id+id*id

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	output $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	output $T \to FT'$
	$\mathbf{id}\ T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	output $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\ \mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\ \mathbf{id} * \mathbf{id}\$$	output $T' \to \epsilon$
$\mathbf{id}$	$+\ TE'\$$	$+\ \mathbf{id} * \mathbf{id}\$$	output $E' \to +\ TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$
$\mathbf{id} +$	$FT'E'\$$	$\mathbf{id} * \mathbf{id}\$$	output $T \to FT'$
$\mathbf{id} +$	$\mathbf{id}\ T'E'\$$	$\mathbf{id} * \mathbf{id}\$$	output $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$T'E'\$$	$* \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$*\ FT'E'\$$	$* \mathbf{id}\$$	output $T' \to * FT'$
$\mathbf{id} + \mathbf{id} *$	$FT'E'\$$	$\mathbf{id}\$$	match $*$
$\mathbf{id} + \mathbf{id} *$	$\mathbf{id}\ T'E'\$$	$\mathbf{id}\$$	output $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$T'E'\$$	$\$$	match $\mathbf{id}$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$E'\$$	$\$$	output $T' \to \epsilon$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$\$$	$\$$	output $E' \to \epsilon$

Figure 4.21: Moves made by a predictive parser on input $\mathbf{id} + \mathbf{id} * \mathbf{id}$

- Outcome of Syntax Analysis stage: ($,$) : YES/accept (NO SYNTAX ERROR in id+id*id)

Some tips for LL(1) parser (Predictive parser)…

- While making the PARSING TABLE (Step 2) follow the rules…..
- For all productions LHS→RHS, {go for the FIRST(RHS); if FIRST contains epsilon go for FOLLOW (LHS)} ; Write this production in appropriate cell number. [Repeat for all productions in the input grammar]
- While making SEQUENCE OF MOVES TABLE (Step 3) follow the rules…..
- 3 columns: STACK, INPUT, ACTION; refer to PARSING TABLE for each ACTION
- Explain ACTION in detail, eg. “Refer to cell no. (F, id) and substitute F by id”
- Replacement done on the top of the stack (leftmost position on the stack)
- Declare MATCH!! if top of stack matches and cancels out with leftmost symbol of input string
- Repeat till you get ($, $) in (STACK, INPUT) for NO SYNTAX ERROR IN INPUT!