Compiler Design

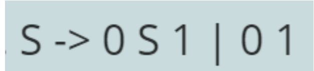
Fatemeh Deldar

Isfahan University of Technology

1403-1404

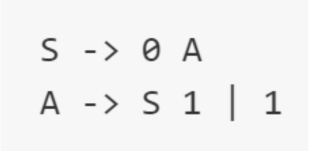
Exercise

- **Example:** Devise predictive parsers and show the parsing tables (You may left-factor and/or eliminate left-recursion from your grammars first.)

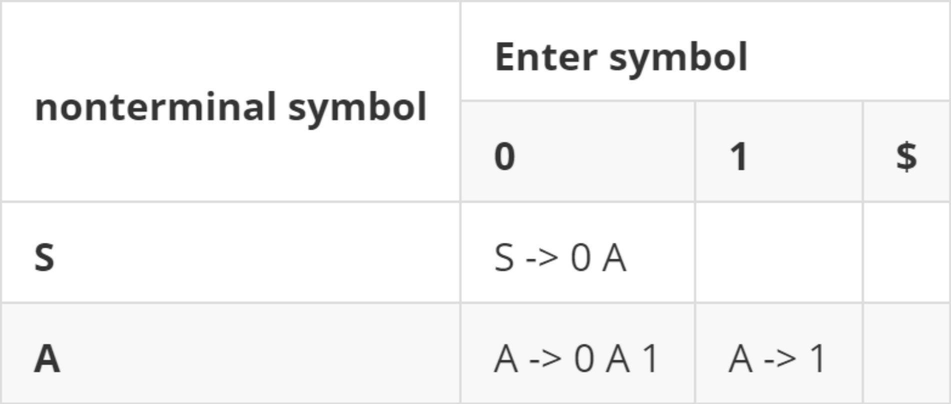


S -> 0 S 1 | 0 1

- Left factoring



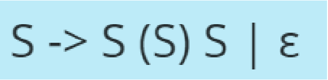
```
S -> 0 A
A -> S 1 | 1
```



nonterminal symbol	Enter symbol		
	0	**1**	**$**
S	S -> 0 A		
A	A -> 0 A 1	A -> 1	

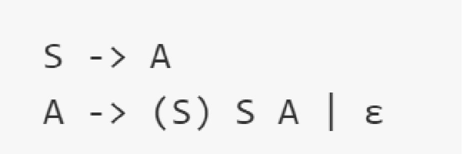
Exercise

- **Example:** Devise predictive parsers and show the parsing tables (You may left-factor and/or eliminate left-recursion from your grammars first.)

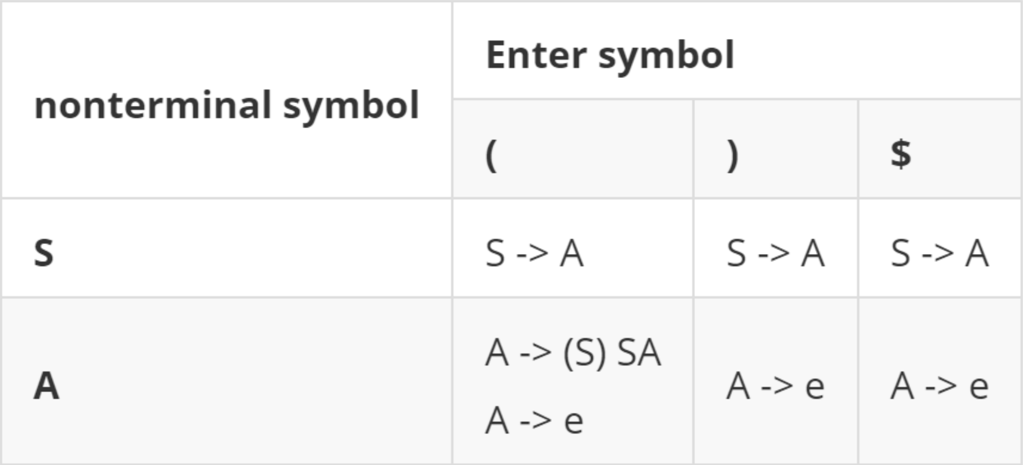


S -> S (S) S | ε

- Eliminate left recursion



```
S -> A
A -> (S) S A | ε
```



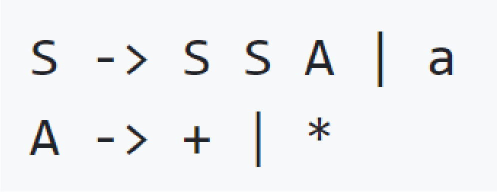
nonterminal symbol	Enter symbol		
	(	)	$
S	S -> A	S -> A	S -> A
A	A -> (S) SA A -> e	A -> e	A -> e

Exercise

- **Example:**

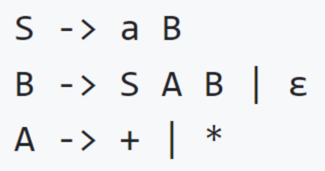
S -> S S + | S S * | a

- Left factoring

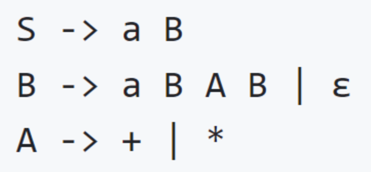


```
S -> S S A | a
A -> + | *
```

- Eliminate left recursion



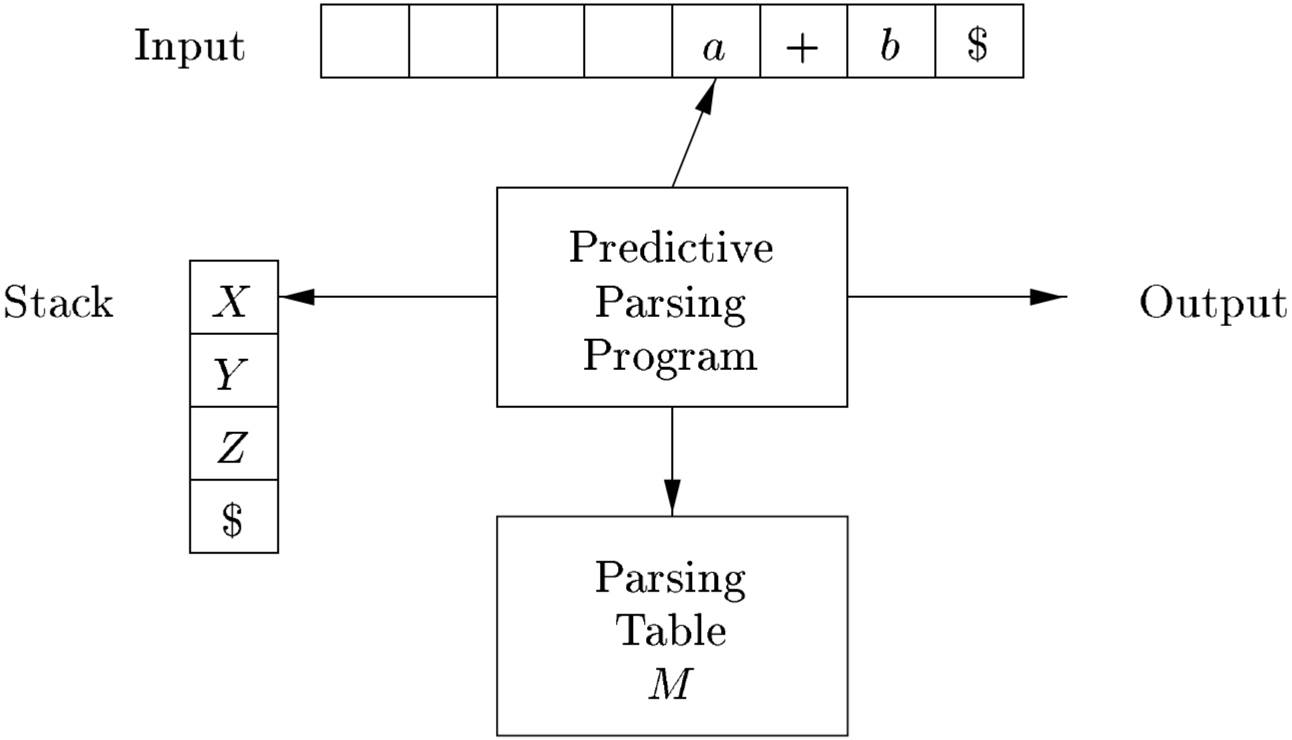
```
S -> a B
B -> S A B | ε
A -> + | *
```



```
S -> a B
B -> a B A B | ε
A -> + | *
```

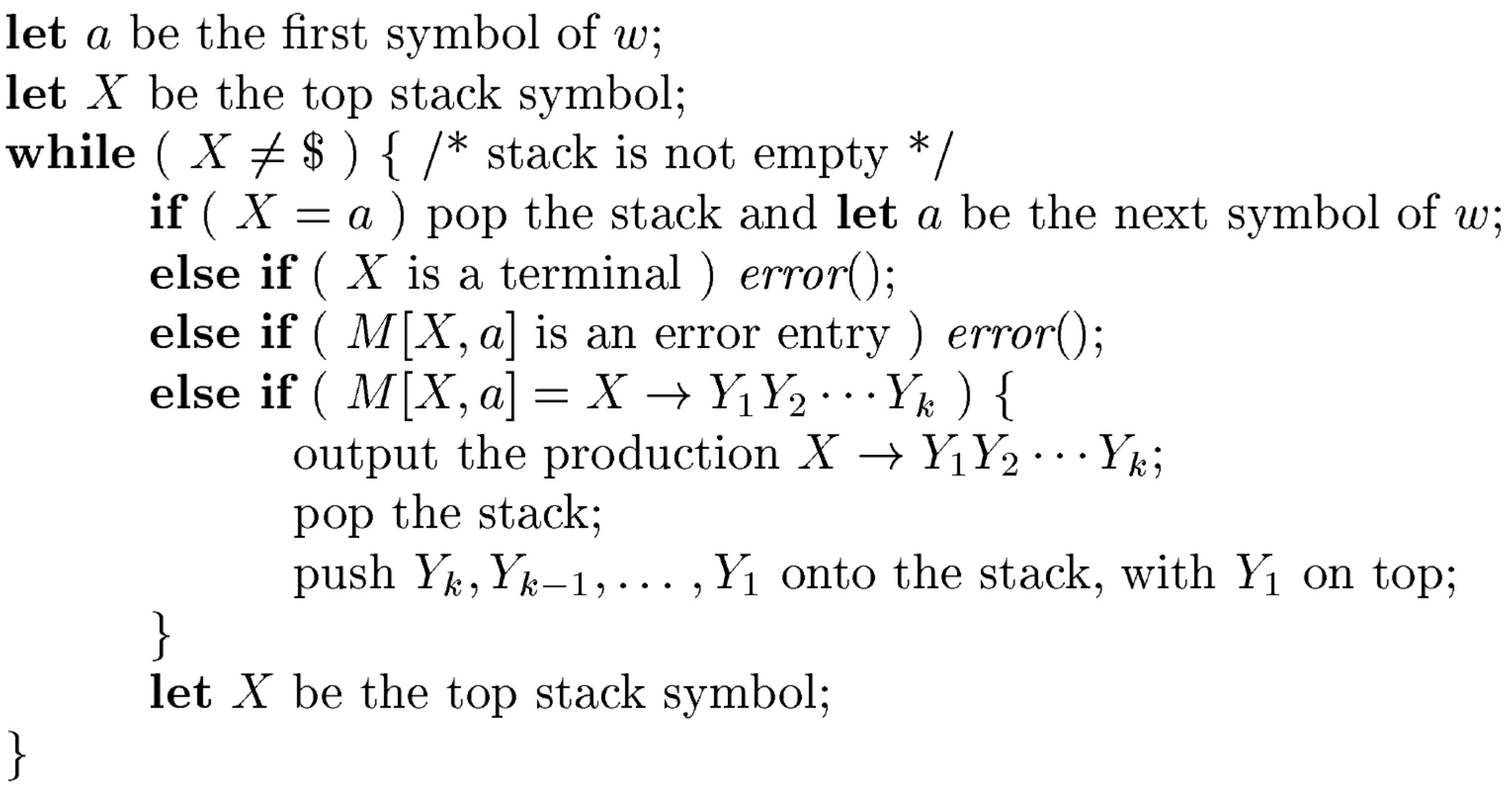
Nonrecursive Predictive Parsing

- A nonrecursive predictive parser can be built by maintaining a stack explicitly, rather than implicitly via recursive calls



Nonrecursive Predictive Parsing

- **Table-driven predictive parsing**
  - Initially, $w\$$ in the input buffer and the start symbol S of G on top of the stack, above \$



```
let $a$ be the first symbol of $w$;
let $X$ be the top stack symbol;
while ( $X \neq \$$ ) { /* stack is not empty */
        if ( $X = a$ ) pop the stack and let $a$ be the next symbol of $w$;
        else if ( $X$ is a terminal ) $error()$;
        else if ( $M[X, a]$ is an error entry ) $error()$;
        else if ( $M[X, a] = X \rightarrow Y_1 Y_2 \cdots Y_k$ ) {
                output the production $X \rightarrow Y_1 Y_2 \cdots Y_k$;
                pop the stack;
                push $Y_k, Y_{k-1}, \ldots, Y_1$ onto the stack, with $Y_1$ on top;
        }
        let $X$ be the top stack symbol;
}
```

- **Example**

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

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

Error Recovery in Predictive Parsing

- An error is detected during predictive parsing when
  1. The terminal on top of the stack does not match the next input symbol
  2. Nonterminal A is on top of the stack, a is the next input symbol, and $M[A, a]$ is error (i.e., the parsing-table entry is empty)
- **Panic Mode**
  - ***Panic-mode error recovery*** is based on the idea of skipping over symbols on the input until a token in a selected set of **synchronizing tokens** appears
  - Its effectiveness depends on the choice of synchronizing set