Syntax Analysis
[Chapter 4 - Part 2]

Lecture 11

Edited by Dr. Ismail Hababeh
German Jordanian University
Adapted from slides by Dr. Robert A. Engelen

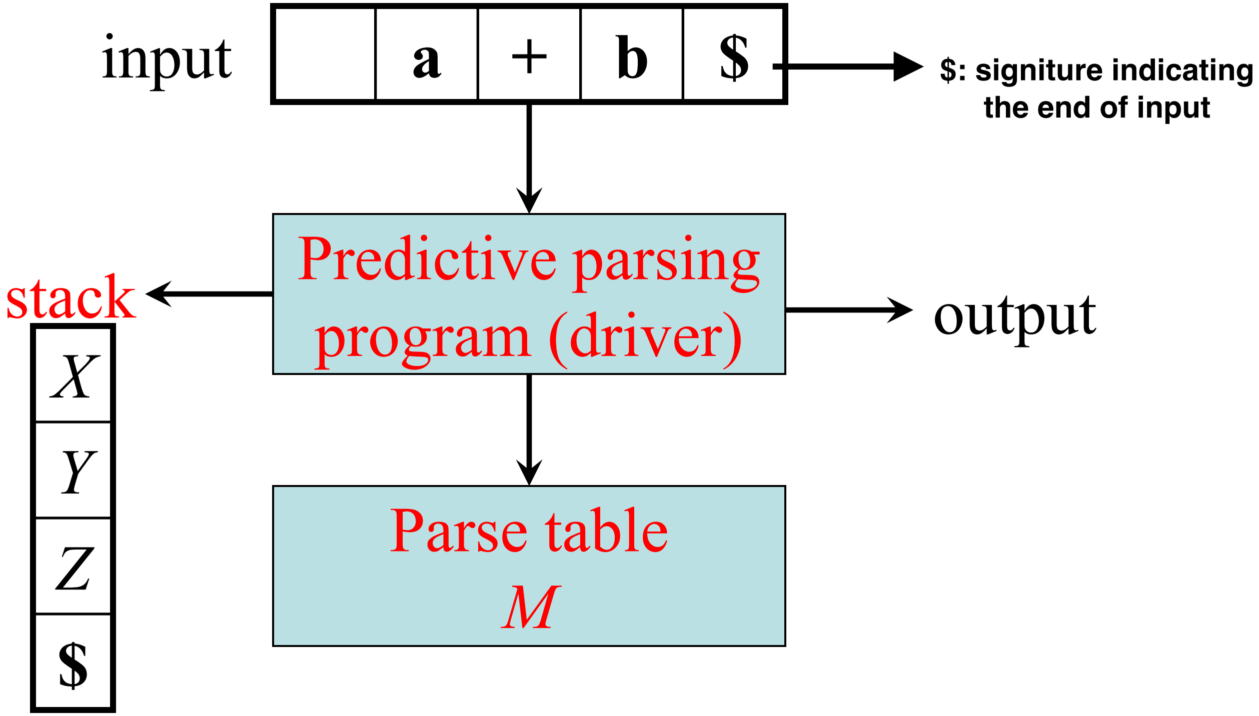
Recursive Predictive Descent Vs. Non-Recursive Predictive Descent Parsing

- Recursive Predictive Descent Parser:
  - A top-down method of syntax analysis in which a set of recursive procedures is used to process input.
  - It uses procedures for every non-terminal entity to parse strings.
  - It may or may not require backtracking.
- Non-Recursive Predictive Descent Parser:
  - It finds out productions by replacing input string.
  - Uses lookahead which points to next input symbols, that's make it backtracking free.

Non-Recursive Predictive Parsing

Non-terminals, terminals, productions, start state

- Given an LL(1) grammar $G=(N,T,P,S)$ construct a table $M[A,a]$ for $A \in N$, $a \in T$ and use a driver (predictive parsing program) with a stack (We use a stack instead of recursive calls).


input
a
+
b
$
$: signiture indicating the end of input
Predictive parsing program (driver)
stack
X
Y
Z
$
output
Parse table
M

Constructing a Predictive Parsing Table

```
for each production $A \rightarrow \alpha$ do
    for each $a \in \mathrm{FIRST}(\alpha)$ do
        add $A \rightarrow \alpha$ to $M[A,a]$
    enddo
    if $\varepsilon \in \mathrm{FIRST}(\alpha)$ then
        for each $b \in \mathrm{FOLLOW}(A)$ do
            add $A \rightarrow \alpha$ to $M[A,b]$
        enddo
    endif
enddo
Mark each undefined entry in M table to be error
```

FOLLOW Computation

- FOLLOW(A) = the set of **terminals** that can immediately follow nonterminal A

```
FOLLOW($A$) =
    for all ($B \to \alpha A \beta$) $\in P$ do
        add FIRST($\beta$)\{$\varepsilon$} to FOLLOW($A$)
        // FIRST($\beta$)\{$\varepsilon$}  means everything in First($\beta$) except {$\varepsilon$}
    for all ($B \to \alpha A \beta$) $\in P$ and $\varepsilon \in$ FIRST($\beta$) do
        add FOLLOW($B$) to FOLLOW($A$)
    for all ($B \to \alpha A$) $\in P$ do
        add FOLLOW($B$) to FOLLOW($A$)
    if $A$ is the start symbol $S$ then
        add $ to FOLLOW($A$)
        // $ is the input right end-marker
```

FOLLOW Computation - Example

$$
\begin{aligned}
E &\rightarrow T\ E_R \\
E_R &\rightarrow +T\ E_R \mid \varepsilon \\
T &\rightarrow F\ T_R \\
T_R &\rightarrow *\ F\ T_R \mid \varepsilon \\
F &\rightarrow (E) \mid \mathbf{id}
\end{aligned}
$$

FOLLOW(E) = FOLLOW(E_R) = {), \$}

Since E is the start symbol, FOLLOW(E) must contain \$. The production body (E) *explains why the right parenthesis ")" is in FOLLOW(E). For E_R, note that this nonterminal appears only at the end of bodies of E productions. Thus, FOLLOW(E_R) must be the same as FOLLOW(E).*

FOLLOW(T) = FOLLOW(T_R) = {+,), \$}

FOLLOW(F) = {+, *,), \$}

Predictive Parsing Table - Example

$E \rightarrow T\, E_R$

$E_R \rightarrow +\, T\, E_R \mid \varepsilon$

$T \rightarrow F\, T_R$

$T_R \rightarrow *\, F\, T_R \mid \varepsilon$

$F \rightarrow (\, E\,) \mid$ **id**

Production	FIRST(α)	FOLLOW(*A*)
$E \rightarrow T\, E_R$	**(id**	**$)**
$E_R \rightarrow +\, T\, E_R$	+	**$)**
$E_R \rightarrow \varepsilon$	ε	**$)**
$T \rightarrow F\, T_R$	**(id**	**+ $)**
$T_R \rightarrow *\, F\, T_R$	*	**+ $)**
$T_R \rightarrow \varepsilon$	ε	**+ $)**
$F \rightarrow (\, E\,)$	**(**	*** + $)**
$F \rightarrow$ **id**	**id**	*** + $)**

	id	+	*	**(**	ε	**$**
E	$E \rightarrow T\, E_R$			$E \rightarrow T\, E_R$		
E_R		$E_R \rightarrow +\, T\, E_R$			$E_R \rightarrow \varepsilon$	$E_R \rightarrow \varepsilon$
T	$T \rightarrow F\, T_R$			$T \rightarrow F\, T_R$		
T_R			$T_R \rightarrow *\, F\, T_R$		$T_R \rightarrow \varepsilon$	$T_R \rightarrow \varepsilon$
F	$F \rightarrow$ **id**			$F \rightarrow (\, E\,)$		

Predictive Parsing Program (Driver)

```
While (X <> $)
{
push($)
push($S$)
$a$ := $lookahead$
repeat
        $X$ := pop()
        if $X$ is a terminal or $X$ = $ then
                match($X$) // move to next token, $a$ := $lookahead$
        else if $M[X,a] = X \to Y_1 Y_2 \ldots Y_k$ then
                push($Y_k$, $Y_{k-1}$, …, $Y_2$, $Y_1$) // such that $Y_1$ is on top
                produce output and/or invoke actions
        else    error()
        endif
until $X$ = $
}
```

Top-Down Table-Driven Predictive LL(1) Parsing – (leftmost derivation) Example

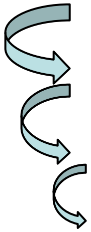
$E \rightarrow T\ E_R$

$E_R \rightarrow +\ T\ E_R \mid \varepsilon$

$T \rightarrow F\ T_R$

$T_R \rightarrow *\ F\ T_R \mid \varepsilon$

$F \rightarrow (\ E\ ) \mid \textbf{id}$



Stack	Input	Production applied
$\$E$	**id+id*id$**	
$\$E_R T$	**id+id*id$**	$E \rightarrow T\ E_R$
$\$E_R T_R F$	**id+id*id$**	$T \rightarrow F\ T_R$
$\$E_R T_R$**id**	**id+id*id$**	$F \rightarrow$ **id**
$\$E_R T_R$	**+id*id$**	
$\$E_R$	**+id*id$**	$T_R \rightarrow \varepsilon$
$\$E_R T+$	**+id*id$**	$E_R \rightarrow +\ T\ E_R$
$\$E_R T$	**id*id$**	
$\$E_R T_R F$	**id*id$**	$T \rightarrow F\ T_R$
$\$E_R T_R$**id**	**id*id$**	$F \rightarrow$ **id**
$\$E_R T_R$	***id$**	$T_R \rightarrow \varepsilon$
$\$E_R T_R F*$	***id$**	$T_R \rightarrow *\ F\ T_R$
$\$E_R T_R F$	**id$**	
$\$E_R T_R$**id**	**id$**	$F \rightarrow$ **id**
$\$E_R T_R$	**$**	$T_R \rightarrow \varepsilon$
$\$E_R$	**$**	$E_R \rightarrow \varepsilon$
$	**$**	

Handling Errors - Panic Mode Recovery

- The first level improvement.
- The parser discards input until it encounters a *synchronizing token*.
- These tokens are chosen so that the parser can make a fresh beginning.
- Examples:

  In C/Java, ; and } are synchronizing tokens.

Handling Errors - Panic Mode Recovery

Choosing the synchronization tokens:

- For nonterminal A, place all FOLLOW(A) in the synchronization set of (pop *A)* and skip input till FOLLOW(A) is seen *like; and }*
- If we add symbols of FIRST(A) to the synchronization set for nonterminal A, it may be possible to resume parsing according to A if a symbol in FIRST(A) appears in the input.

Handling Errors - Panic Mode Recovery

How it works:

- If the parser looks up entry M[A,a] and finds that it is empty, then the input a is skipped.
- If the entry is synchronized, then the nonterminal on the top of the stack is popped in attempt to resume parsing.

Example:) id * id $ → skip input → id * id $

Rase an Error when an empty parse table entry is reached

Handling Errors - Panic Mode Recovery
Example

Perform the panic mode recovery for the following language productions:

$$E \rightarrow T\,E_R$$
$$E_R \rightarrow +\,T\,E_R \mid \varepsilon$$
$$T \rightarrow F\,T_R$$
$$T_R \rightarrow *\,F\,T_R \mid \varepsilon$$
$$F \rightarrow (\,E\,) \mid \mathbf{id}$$

Follow

FOLLOW(E) = { \$) }
FOLLOW(E_R) = { \$) }
FOLLOW(T) = { + \$) }
FOLLOW(T_R) = { + \$) }
FOLLOW(F) = { * + \$) }

Handling Errors - Panic Mode Recovery

Example Solution

Add synchronizing actions to undefined entries based on nonterminal FOLLOWs

FOLLOW(E) = { $) }
FOLLOW(E_R) = { $) }
FOLLOW(T) = { + $) }
FOLLOW(T_R) = { + $) }
FOLLOW(F) = { * + $) }

	id	+	*	**(**	**)**	**$**
E	$E \rightarrow T\,E_R$			$E \rightarrow T\,E_R$	***synch***	***synch***
E_R		$E_R \rightarrow +\,T\,E_R$			$E_R \rightarrow \varepsilon$	$E_R \rightarrow \varepsilon$
T	$T \rightarrow F\,T_R$	***synch***		$T \rightarrow F\,T_R$	***synch***	***synch***
T_R		$T_R \rightarrow \varepsilon$	$T_R \rightarrow *\,F\,T_R$		$T_R \rightarrow \varepsilon$	$T_R \rightarrow \varepsilon$
F	$F \rightarrow$ **id**	***synch***	***synch***	$F \rightarrow$ **(** E **)**	***synch***	***synch***

synch: pop A and skip input until synch token is found or skip until FIRST(A) found

Handling Errors – Phrase Level Recovery

- Locally replace some prefix of the remaining input by some string.
- Simple cases are exchanging ; with , and = with ==.
- Difficulties occur when the real error occurred long before an error was detected.

Handling Errors – Phrase Level Recovery

- Fill the blank entries in the predictive parsing table with pointers to error routines that may change, insert, or delete symbols on the input and issue appropriate error messages.
- Change input stream by inserting missing *
- Example: **id id** is changed into **id * id**

Handling Errors – Phrase Level Recovery

	id	+	*	(	)	**$**
E	$E \rightarrow T\, E_R$			$E \rightarrow T\, E_R$	***synch***	***synch***
E_R		$E_R \rightarrow +\, T\, E_R$			$E_R \rightarrow \varepsilon$	$E_R \rightarrow \varepsilon$
T	$T \rightarrow F\, T_R$	***synch***		$T \rightarrow F\, T_R$	***synch***	***synch***
T_R	***insert ****	$T_R \rightarrow \varepsilon$	$T_R \rightarrow *\, F\, T_R$		$T_R \rightarrow \varepsilon$	$T_R \rightarrow \varepsilon$
F	$F \rightarrow$ **id**	***synch***	***synch***	$F \rightarrow (\, E\,)$	***synch***	***synch***

insert *: insert missing * and redo the production