WARSAW UNIVERSITY OF TECHNOLOGY
DEVELOPMENT PROGRAMME

Compiling Techniques ECOTE

part 6 - Predictive parser

DSc eng. Ilona Bluemke

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!

EUROPEAN UNION
EUROPEAN
SOCIAL FUND

Project is co-financed by European Union within European Social Fund

Predictive parser

Algorithm to construct **a predictive parsing table**:

- **$A \rightarrow \alpha$** and ***a* in First(α),** whenever A is an active symbol and *a* is the current input symbol, the parser will expand **A** by α
- If $\alpha = \varepsilon$ or $\alpha \Rightarrow^{*} \varepsilon$ the parser will expand **A** by α if the current input symbol is in $Follow_1(A)$, or if **$** on the input has been reached and ε is in $Follow_1(A)$

Method:

1. For each production **$A \to \alpha$** of the grammar, do steps **2** and **3**
2. For each terminal ***a*** in **$First_1(\alpha)$,** add **$A \to \alpha$** to **M[A, *a*]**
3. If **ε** is in $First_1(\alpha)$, add **$A \to \alpha$** to **M[A, *b*]** for each terminal ***b*** in $Follow_1(A)$. If **ε** is in $First_1(\alpha)$ and in $Follow_1(A)$, add $A \to \alpha$ to **M[A, $]**
4. Make each undefined entry of M **error**.

Consider the grammar with productions:

$S \rightarrow i\ C\ t\ S\ S' \mid a$

$S' \rightarrow e\ S \mid \varepsilon$ Follow(S')= {ε, e}

$C \rightarrow b$

The parsing table:

	a	b	e	i	t	$
S	$S \rightarrow a$			$S \rightarrow iCtSS'$		
S'			$S' \rightarrow \varepsilon$ $S' \rightarrow eS$			$S' \rightarrow \varepsilon$
C		$C \rightarrow b$				

M [S', e] contains two productions, since **$Follow_1$(S')={e, ε}**

The grammar **is ambiguous**, the ambiguity is manifested by a choice what production to use when an e (**else**) is seen. To resolve the ambiguity **S'→eS** should be chosen (associating **else** with the closest **then**)

A grammar whose parsing table has no multiply-defined entries is said to be **LL(1).**

Example

$S \rightarrow AS'$

$S' \rightarrow + AS' \mid \varepsilon$

$A \rightarrow BA'$

$A' \rightarrow * BA' \mid \varepsilon$

$B \rightarrow (S) \mid a$

$First_1(B) = \{(, a\}$ $First_1(A') = \{*, \varepsilon\}$

$First_1(S') = \{+, \varepsilon\}$ $First_1(S) = First_1(A) = \{(, a\}$

$Follow_1(S) = Follow_1(S') = \{), \varepsilon\}$

$Follow_1(A) = Follow_1(A') = \{+,), \varepsilon\}$

$Follow_1(B) = \{+, *,), \varepsilon\}$

	a	+	*	(	)	$
S	$S \rightarrow AS'$			$S \rightarrow AS'$		
S'		$S' \rightarrow +AS'$			$S' \rightarrow \varepsilon$	$S' \rightarrow \varepsilon$
A	$A \rightarrow BA'$			$A \rightarrow BA'$		
A'		$A' \rightarrow \varepsilon$	$A' \rightarrow *BA'$		$A' \rightarrow \varepsilon$	$A' \rightarrow \varepsilon$
B	$B \rightarrow a$			$B \rightarrow (S)$		

DEFINITION

A **grammar G is LL(1)** if and only if whenever

$A \rightarrow \alpha \mid \beta$ are two distinct productions of G the following conditions hold:

1. There is no terminal ***a*** such that α and β derive strings beginning with ***a***
2. At most one of α and β can derive the empty string.

   If $\beta \Rightarrow^* \varepsilon$ then α does not derive any strings beginning with a terminal in $Follow_1(A)$.

Example

$S \rightarrow AS'$ $\quad S' \rightarrow +AS' \mid \varepsilon$ $\quad A \rightarrow BA'$

$A' \rightarrow *BA' \mid \varepsilon$ $\quad B \rightarrow (S) \mid a$

S – only one production - OK

S' – two production derive different strings (+, ε)

only one production can derive ε, + is not in Follow(S') – OK

A – only one production - OK

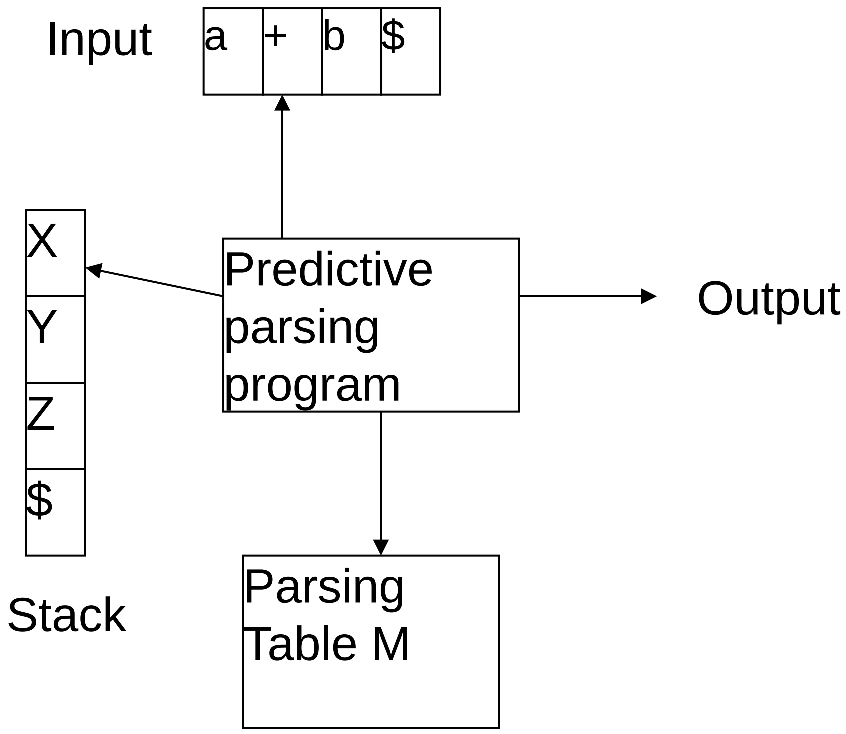
A' – two production derive different strings (*, ε)

only one production can derive ε , * is not in FOLLOW (A') – OK

B – two production derive different strings („(„, a) – OK

So **it is LL (1) grammar**

Nonrecursive predictive parser



$ - the end marker, the bottom of a stack

Initially stack contains the start symbol **S** of the grammar on top of $

M[A,a] – A nonterminal, **a** terminal

Parser program:

Considers **X** – the symbol on the top of the stack and **a** – the current input symbol. These symbols determine the action of the parser:

1. **If X=a=$**, the parser halts and announces the successful completion
2. **If X=a ≠$,** the parser pops **X** off the stack and advances the input pointer to the next input symbol

3.

If **X** is a nonterminal, the program consults entry **M [X,a]** of parsing table M. The entry will be either an **X** production or an error entry.

a) **If M [X,a] = {X→UVW}** the parser replaces **X** on top of the stack by **WVU** (U on top), as output the parser prints the number of production used

b) **If M [X,a] = error** the parser calls error recovery routine

Example

1. $S \to AS'$
2. $S' \to + AS' \mid \varepsilon$
4. $A \to BA'$
5. $A' \to * BA' \mid \varepsilon$
7. $B \to (S) \mid a$

$First_1(B) = \{(, a\}$ $First_1(A') = \{*, \varepsilon\}$
$First_1(S') = \{+, \varepsilon\}$ $First_1(S) = First_1(A) = \{(, a\}$
$Follow_1(S) = Follow_1(S') = \{), \varepsilon\}$
$Follow_1(A) = Follow_1(A') = \{+,), \varepsilon\}$
$Follow_1(B) = \{+, *,), \varepsilon\}$

	a	+	*	(	)	$
S	$S \to AS'$			$S \to AS'$		
S'		$S' \to +AS'$			$S' \to \varepsilon$	$S' \to \varepsilon$
A	$A \to BA'$			$A \to BA'$		
A'		$A' \to \varepsilon$	$A' \to *BA'$		$A' \to \varepsilon$	$A' \to \varepsilon$
B	$B \to a$			$B \to (S)$		

Moves made by parser on input a+a*a$

Stack	Input	Output	
$S	a+a*a$		
$S'A	a+a*a$	$S \rightarrow AS'$	(1)
$S'A'B	a+a*a$	$A \rightarrow BA'$	(4)
$S'A'a	a+a*a$	$B \rightarrow a$	(8)
$S'A'	+a*a$	$A' \rightarrow \varepsilon$	(6)
$S'	+a*a$		
$S'A+	+a*a$	$S' \rightarrow +AS'$	(2)
$S'A	a*a$		

Moves made by parser on input a+a*a$ -2

Stack	Input	Output	
$S'A'B	a*a$	$A \rightarrow BA'$	(4)
$S'A'a	a*a$	$B \rightarrow a$	(8)
$S'A'	*a$		
$S'A'B*	*a$	$A' \rightarrow *BA'$	(5)
$S'A'B	a$		
$S'A'a	a$	$B \rightarrow a$	(8)
$S'A'	$		
$S'	$	$A' \rightarrow \varepsilon$	(6)
$	$	$S' \rightarrow \varepsilon$	(3)

Moves made by parser on input a*a+a$

Stack	Input	Output	
$S	a*a+a$		
$S'A	a*a+a$	$S \rightarrow AS'$	(1)
$S'A'B	a*a+a$	$A \rightarrow BA'$	(4)
$S'A'a	a*a+a$	$B \rightarrow a$	(8)
$S'A'	*a+a$	$A' \rightarrow *BA'$	(5)
$S'A'B*	*a+a$		
$S'A'B	a+a$	$B \rightarrow a$	(8)
$S'A'a	a+a$		
$S'A'	+a$	$A' \rightarrow \varepsilon$	(6)
$S'	+a$		

Moves made by parser on input a*a+a$

Stack	Input	Output	
$S'	+a$	$S' \rightarrow +AS'$	(2)
$S'A+	+a$		
$S'A	a$	$A \rightarrow BA'$	(4)
$S'A'B	a$	$B \rightarrow a$	(8)
$S'A'a	a$		
$S'A'	$	$A' \rightarrow \varepsilon$	(6)
$S'	$	$S' \rightarrow \varepsilon$	(3)
$	$		

- **Draw derivation tree**

Exercises

Consider grammar:

1. $S \to a \mid \wedge \mid (T)$
4. $T \to S\ T'$
5. $T' \to ,S\ T \mid \varepsilon$

$First_1(S) = \{a, (, \wedge\}$

$First_1(T) = First_1(S) = \{a, (, \wedge\}$

$First_1(T') = \{$„,”$, \varepsilon\}$

$Follow_1(S) = \{\varepsilon,$ „ , ”$, a, (, \wedge,)\}$

$Follow_1(T) = \{) \}$

$Follow_1(T') = \{) \}$

Construct the parsing table

	a		(	)	,	$
S	$S \to a$	$S \to \wedge$	$S \to (T)$			
T	$T \to S\ T'$	$T \to S\ T'$	$T \to S\ T'$			
T'				$T' \to \varepsilon$	$T' \to ,S\ T'$	

Moves of parser for (a, a)

Stack	Input	Output	
\$S	(a, a) \$	$S \rightarrow (T)$	(3)
\$)T(	(a, a) \$		
\$)T	a, a) \$	$T \rightarrow S\ T'$	(4)
\$)T'S	a, a) \$	$S \rightarrow a$	(1)
\$)T'a	a, a) \$		
\$)T'	, a) \$	$T' \rightarrow ,S\ T$	(5)
\$)T' S,	, a) \$		
\$)T S	a) \$	$S \rightarrow a$	(1)
\$)T a	a) \$		

Moves of parser for (a, a) -2

Stack	Input	Output	
$)T'	) $	$T' \rightarrow \varepsilon$	(6)
$)	) $		
$	$		

- **Draw derivation tree for (a, a)**

Show that the grammar is LL(1) using definition

$S \to a \mid \wedge \mid (T)$

Rule 1 from definition fulfilled (each symbol is different)

$T \to S\ T'$

only one production, nothing to check

$T' \to ,\ S\ T \mid \varepsilon$

Only one of α and β can derive the empty string.

If $\beta \Rightarrow^* \varepsilon$ then α does not derive any strings beginning with a terminal in $Follow_1(T') = \{)\}$. fulfilled

Compiling Techniques ECOTE

end of part 6 - Predictive parser

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
EUROPEAN UNION
EUROPEAN SOCIAL FUND
Project is co-financed by European Union within European Social Fund