Syntax Analysis Or Parsing

Lecture 05

FOLLOW SETS

- FOLLOW(A) is the set of terminals (including end marker of input - $) that may follow non-terminal A in some sentential form.
- FOLLOW(A) = $\{c \mid S \Rightarrow^{+} \ldots Ac \ldots\} \cup \{\$\}$ if $S \Rightarrow^{+} \ldots A$
- For example, consider $L \Rightarrow^{+} (())(L)L$

  Both ')' and end of file can follow L
- NOTE: ε is ***never*** in FOLLOW sets

COMPUTING FOLLOW(A)

1. If A is start symbol, put \$ in FOLLOW(A)
2. Productions of the form $B \rightarrow \alpha\ A\ \beta$,

   Add FIRST(β) – {ε} to FOLLOW(A)
3. Productions of the form $B \rightarrow \alpha\ A$ or

   $B \rightarrow \alpha\ A\ \beta$ where $\beta \Rightarrow^* \varepsilon$

   Add FOLLOW(B) to FOLLOW(A)

Example

- $E \rightarrow T E'$
- $E' \rightarrow + T E' \mid \varepsilon$
- $T \rightarrow F T'$
- $T' \rightarrow * F T' \mid \varepsilon$
- $F \rightarrow (E) \mid id$

- FIRST(E) = {(, id}
- FIRST(E′) = {+, ε}
- FIRST(T) = {(, id}
- FIRST(T′) = {*, ε}
- FIRST(F) = {(, id}}

Assume the first non-terminal is the start symbol

- FOLLOW(E) = {$}
- FOLLOW(E′) =
- FOLLOW(T) =
- FOLLOW(T′) =
- FOLLOW(F) =

Using rule #1

1. If A is start symbol, put $ in FOLLOW(A)

EXAMPLE

- $E \rightarrow T\ E'$
- $E' \rightarrow +\ T\ E' \mid \varepsilon$
- $T \rightarrow F\ T'$
- $T' \rightarrow *\ F\ T' \mid \varepsilon$
- $F \rightarrow (\ E\ ) \mid id$

- FIRST(E) = {(, id}
- FIRST(E′) = {+, ε}
- FIRST(T) = {(, id}
- FIRST(T′) = {*, ε}
- FIRST(F) = {(, id}}

- FOLLOW(E) = {$,)}
- FOLLOW(E′) =
- FOLLOW(T) = {+}
- FOLLOW(T′) =
- FOLLOW(F) = {*}

Using rule #2

2. Productions of the form $B \rightarrow \alpha\ A\ \beta$,
Add FIRST(β) – {ε} to FOLLOW(A)

Example

- $E \rightarrow T\ E'$
- $E' \rightarrow +\ T\ E' \mid \varepsilon$
- $T \rightarrow F\ T'$
- $T' \rightarrow *\ F\ T' \mid \varepsilon$
- $F \rightarrow (\ E\ ) \mid id$

- FIRST(E) = {(, id}
- FIRST(E′) = {+, ε}
- FIRST(T) = {(, id}
- FIRST(T′) = {*, ε}
- FIRST(F) = {(, id}

- FOLLOW(E) = {$,)}
- FOLLOW(E′) = FOLLOW(E)
  = {$,)}
- FOLLOW(T) = {+} ∪ FOLLOW(E′)
  = {+, $,)}
- FOLLOW(T′) = FOLLOW(T)
  = {+, $,)}
- FOLLOW(F) = {*} ∪ FOLLOW(T′)
  = {*, +, $,)}

Using rule #3

3. Productions of the form $B \rightarrow \alpha\ A$ or $B \rightarrow \alpha\ A\ \beta$ where $\beta \Rightarrow^{*} \varepsilon$
Add FOLLOW(B) to FOLLOW(A)

Example

- S → (A) | ε
- A → T E
- E → & T E | ε
- T → (A) | a | b | c

  - FIRST(T) =
  - FIRST(E) =
  - FIRST(A) =
  - FIRST(S) =

    - FOLLOW(S) =
    - FOLLOW(A) =
    - FOLLOW(E) =
    - FOLLOW(T) =

Example

- $S \rightarrow (A) \mid \varepsilon$
- $A \rightarrow T\ E$
- $E \rightarrow \&\ T\ E \mid \varepsilon$
- $T \rightarrow (A) \mid a \mid b \mid c$

- FIRST(T) = {(,a,b,c}
- FIRST(E) = {&, ε }
- FIRST(A) = {(,a,b,c}
- FIRST(S) = {(, ε}

- FOLLOW(S) = {$}
- FOLLOW(A) = {) }
- FOLLOW(E) = FOLLOW(A) = {) }
- FOLLOW(T) = FIRST(E) ∪ FOLLOW(E) = {&,)}

EXAMPLE

- $S \rightarrow a\ S\ e \mid B$
- $B \rightarrow b\ B\ C\ f \mid C$
- $C \rightarrow c\ C\ g \mid d \mid \varepsilon$

- FIRST(C) =
- FIRST(B) =
- FIRST(S) =

- FOLLOW(C) =
- FOLLOW(B) =
- FOLLOW(S) = {\$}

Assume the first non-terminal is the start symbol

1. If A is start symbol, put \$ in FOLLOW(A)
2. Productions of the form $B \rightarrow \alpha\ A\ \beta$,
   Add FIRST(β) – {ε} to FOLLOW(A)
3. Productions of the form $B \rightarrow \alpha\ A$ or
   $B \rightarrow \alpha\ A\ \beta$ where $\beta \Rightarrow^{*} \varepsilon$
   Add FOLLOW(B) to FOLLOW(A)

EXAMPLE

- S → a S e | <u>B</u>
- B → b B C f | <u>C</u>
- C → c C g | d | ε

- FIRST(C) = {c,d,ε}
- FIRST(B) = {b,c,d,ε}
- FIRST(S) = {a,b,c,d,ε}

- FOLLOW(C) =

  {f,g} ∪ FOLLOW(B)
  = {c,d,e,f,g,$}

- FOLLOW(B) =
  {c,d} ∪ FOLLOW(S)
  = {c,d,e,$}

- FOLLOW(S) = {$, e }

Follow Example

- $S \rightarrow a\ S\ e \mid B$
- $B \rightarrow b\ B\ C\ f \mid C$
- $C \rightarrow c\ C\ g \mid d \mid \varepsilon$

- FIRST(C) =
- FIRST(B) =
- FIRST(S) =

- FOLLOW(C) =
- FOLLOW(B) =
- FOLLOW(S) = {$}

Assume the first non-terminal is the start symbol

1. If A is start symbol, put $ in FOLLOW(A)
2. Productions of the form $B \rightarrow \alpha A \beta$,
   Add FIRST(β) – {ε} to FOLLOW(A)
3. Productions of the form $B \rightarrow \alpha A$ or
   $B \rightarrow \alpha A \beta$ where $\beta \Rightarrow^* \varepsilon$
   Add FOLLOW(B) to FOLLOW(A)

FOLLOW EXAMPLE

- $S \rightarrow a\ S\ e \mid \underline{B}$
- $B \rightarrow b\ B\ C\ f \mid \underline{C}$
- $C \rightarrow c\ C\ g \mid d \mid \varepsilon$

- FIRST(C) = $\{c,d,\varepsilon\}$
- FIRST(B) = $\{b,c,d,\varepsilon\}$
- FIRST(S) = $\{a,b,c,d,\varepsilon\}$

- FOLLOW(C) =

  $\{f,g\} \cup$ FOLLOW(B)
  $= \{c,d,e,f,g,\$\}$

- FOLLOW(B) =
  $\{c,d\} \cup$ FOLLOW(S)
  $= \{c,d,e,\$\}$

- FOLLOW(S) = $\{\$, e\}$

PREDICTIVE PARSING

- **LL(1) Grammars**
  - Can do predictive parsing
  - Can select the right rule
  - Looking at only the next 1 input symbol
    - First L : Left to Right Scanning
    - Second L: Leftmost derivation
    - 1 : one input symbol look-ahead for predictive decision

- **LL(k) Grammars**
  - Can do predictive parsing
  - Can select the right rule
  - Looking at only the next k input symbols

- **Techniques to modify the grammar:**
  - Left Factoring
  - Removal of Left Recursion

- **LL(k) Language**
  - Can be described with an LL(k) grammar

Table Driven Predictive Parsing

Assume that the grammar is LL(1)

- i.e., Backtracking will never be needed
  - Always know which righthand side to choose (with one look-ahead)
    - No Left Recursion
    - Grammar is Left-Factored.

Example

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +\ T\ E'\ |\ \varepsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T'\ |\ \varepsilon \\
F &\rightarrow (\ E\ )\ |\ \underline{\texttt{id}}
\end{aligned}
$$

Term ...

+Term +Term + ... +Term

Factor ...

*** Factor * Factor * ... * Factor**

Step 1: From grammar, construct table.

Step 2: Use table to parse strings.

Parse Table Construction

M[A,a] refers to table entry at row non-terminal A and column terminal a.

For each production $A \rightarrow \alpha$ do

1. For each terminal a in first(α) Add $A \rightarrow \alpha$ to M[A,a]
2. if ε in first(α) add Add $A \rightarrow \alpha$ to M[A,b] for each terminal b in follow(A)
3. if ε in first(α) and $ in follow(A), Add $A \rightarrow \alpha$ to M[A,$]
4. Set each undefined entry of M to error

Table Driven Predictive Parsing

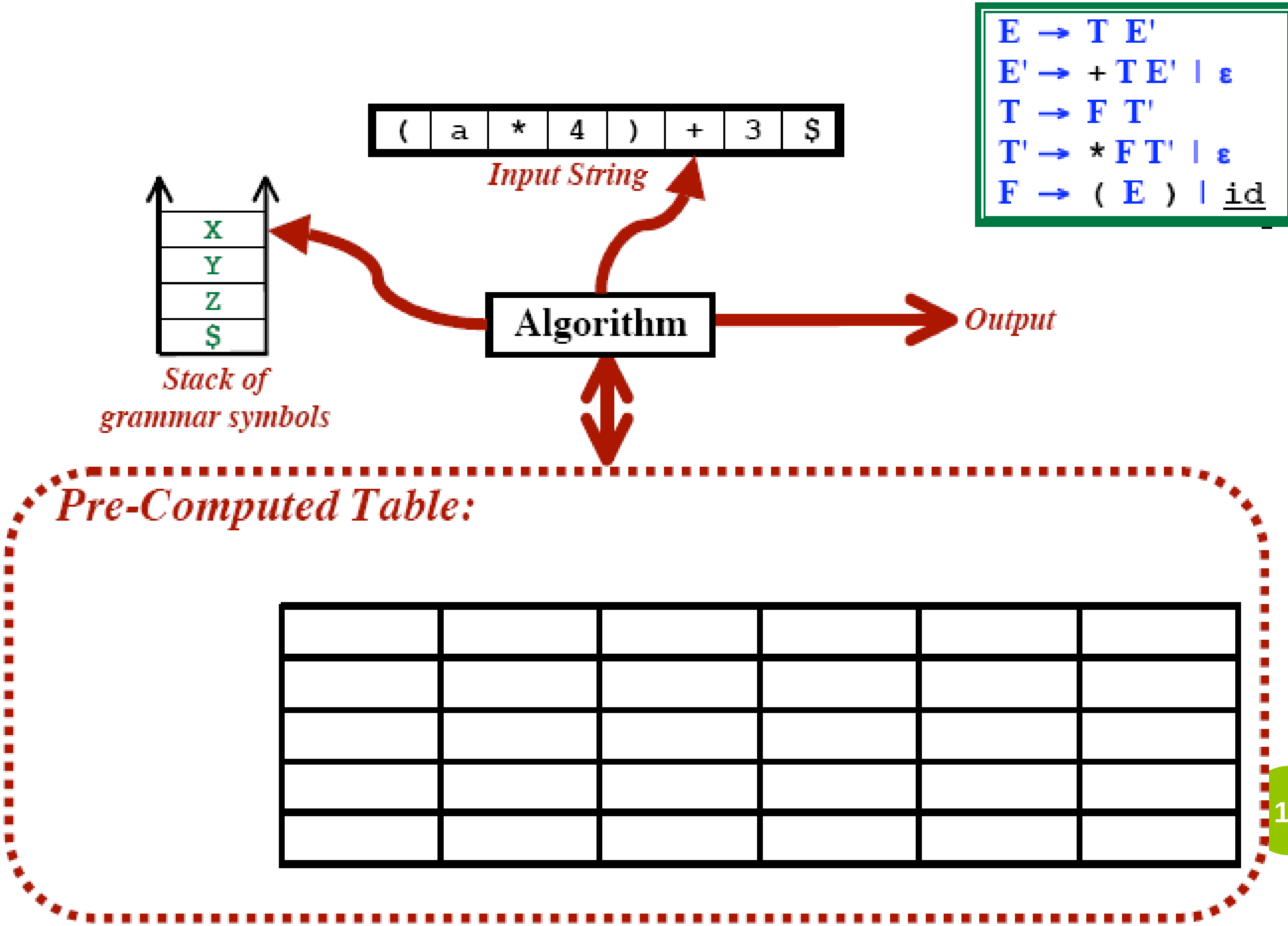
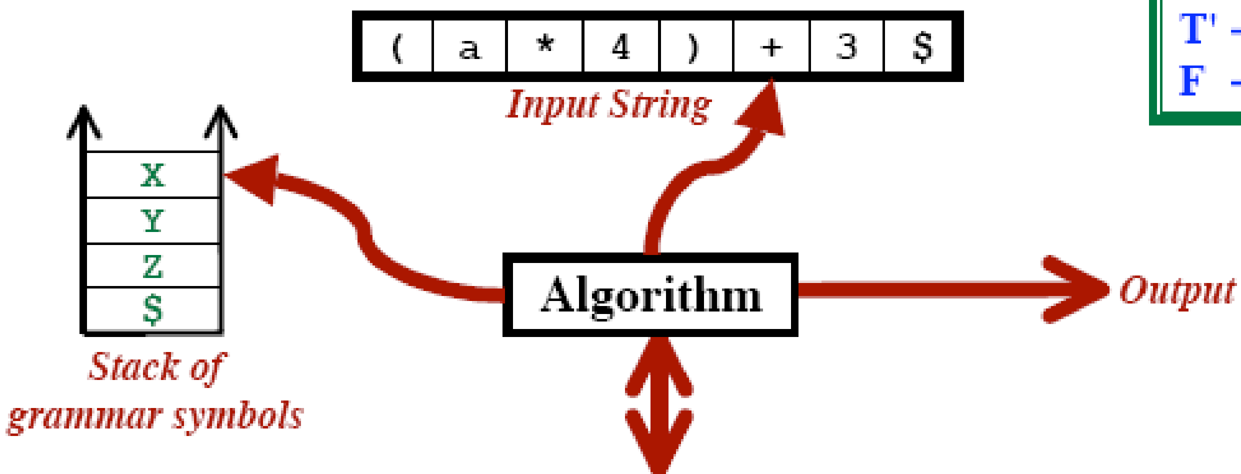


Table Driven Predictive Parsing

E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id



Pre-Computed Table:

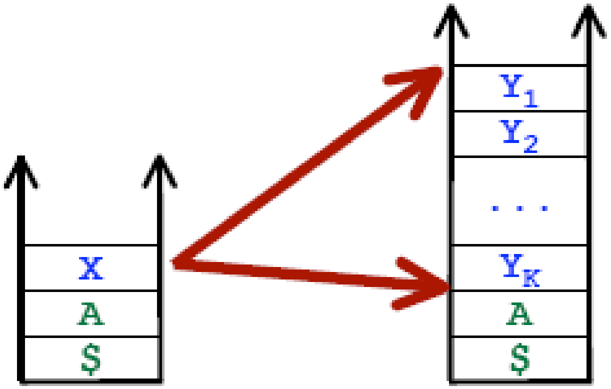
Input Symbols, plus $

Nonterminals

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing Algorithm

```
Set input ptr to first symbol; Place $ after last input symbol
Push $
Push S
repeat
  X = stack top
  a = current input symbol
  if X is a terminal or X = $ then
    if X == a then
      Pop stack
      Advance input ptr
    else
      Error
    endIf
  elseIf Table[X,a] contains a rule then    // call it X → Y_1 Y_2 ... Y_K
    Pop stack
    Push Y_K
    ...
    Push Y_2
    Push Y_1
    Print ("X → Y_1 Y_2 ... Y_K")
  else    // Table[X,a] is blank
    Syntax Error
  endIf
until X == $
```



PREDICTIVE PARSING

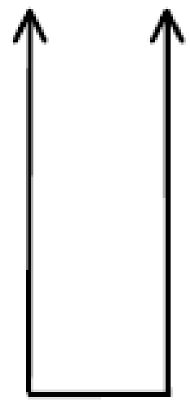
Example

Input:

`(id*id)+id`

Output:

E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id



(	id	*	id	)	+	id	

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

PREDICTIVE PARSING

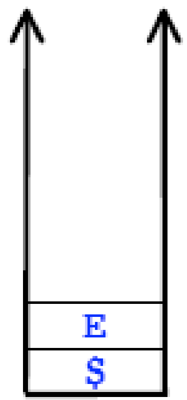
Example

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \varepsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \varepsilon$
$F \rightarrow (\ E\ ) \mid \underline{id}$

Input:

    (id*id)+id

Output:



(	id	*	id	)	+	id	$

Add $ to end of input
Push $
Push E

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

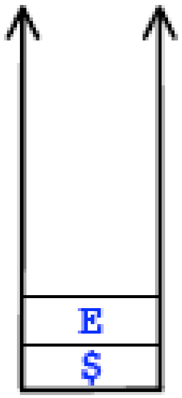
Example

Input:

(id*id)+id

Output:

E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id



(	id	*	id	)	+	id	$

Look at Table [E, '(']

Use rule E→TE'

Pop E

Push E'

Push T

Print E→TE'

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

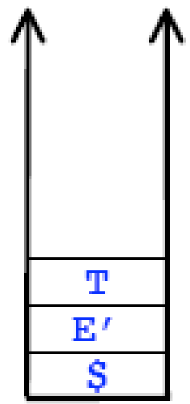
E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E → T E'



(	id	*	id	)	+	id	$

Look at Table [E, '(']
Use rule E→TE'
Pop E
Push E'
Push T
Print E→TE'

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

PREDICTIVE PARSING

Example

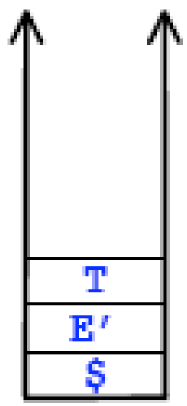
E	→	T E'		
E'	→	+ T E'	|	ε
T	→	F T'		
T'	→	* F T'	|	ε
F	→	(E)	|	id

Input:

(id*id)+id

Output:

E → T E'



(	id	*	id	)	+	id	$

Table [T, '('] = T→FT'

Pop T

Push T'

Push F

Print T→FT'

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

PREDICTIVE PARSING

Example

$E \rightarrow T\,E'$
$E' \rightarrow +\,T\,E' \mid \varepsilon$
$T \rightarrow F\,T'$
$T' \rightarrow *\,F\,T' \mid \varepsilon$
$F \rightarrow (\,E\,) \mid \underline{id}$

Input:

(id*id)+id

Output:

$E \rightarrow T\,E'$
$T \rightarrow F\,T'$

(	id	*	id	)	+	id	$

Stack:

F
T′
E′
$

Table [T, '('] = $T \rightarrow FT'$
Pop T
Push T'
Push F
Print $T \rightarrow FT'$

	id	+	*	(	)	$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

Predictive Parsing

Example

Input:

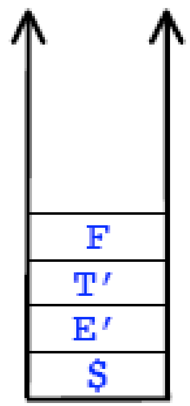
(id*id)+id

Output:

E → T E'

T → F T'

E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id



(	id	*	id	)	+	id	$

Table [F, '('] = F→(E)

Pop F

Push (

Push E

Push)

Print F→(E)

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

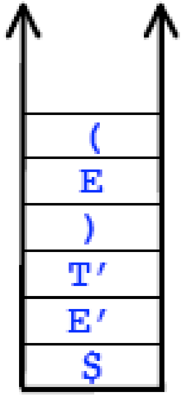
E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)



(	id	*	id	)	+	id	$

Table [F, '('] = F→(E)
Pop F
Push)
Push E
Push (
Print F→(E)

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

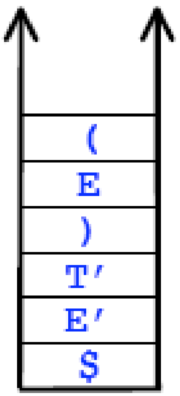
E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)



(	id	*	id	)	+	id	$

Top of Stack matches next input
Pop and Scan

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

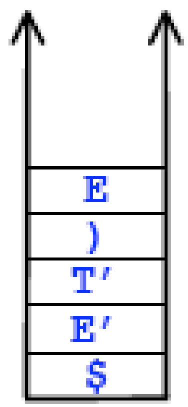
E	→	T E'
E'	→	+ T E' | ε
T	→	F T'
T'	→	* F T' | ε
F	→	(E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)



(	id	*	id	)	+	id	$

Table [E, id] = E→TE'
Pop E
Push E'
Push T
Print E→TE'

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'

(	id	*	id	)	+	id	$

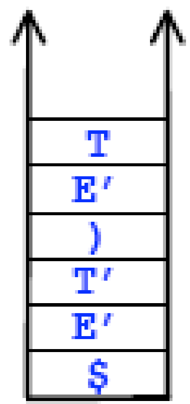


Table [E, id] = E→TE'
Pop E
Push E'
Push T
Print E→TE'

E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

E	→	T E'		
E'	→	+ T E'	|	ε
T	→	F T'		
T'	→	* F T'	|	ε
F	→	(E)	|	id

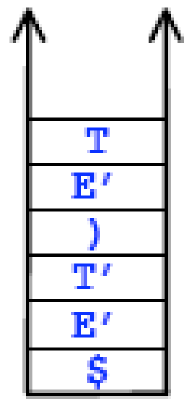
Input:

(id*id)+id

Example

Output:

E → T E'
T → F T'
F → (E)
E → T E'



(	id	*	id	)	+	id	$

Table [T, id] = T→FT'
Pop T
Push T'
Push F
Print T→FT'

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'

Stack
F
T'
E'
)
T'
E'
$

(	id	*	id	)	+	id	$

Table [T, id] = T→FT'
Pop T
Push T'
Push F
Print T→FT'

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

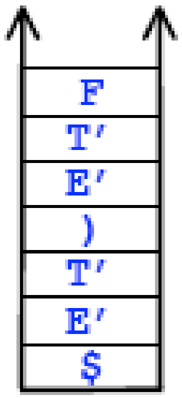
E	→ T E'
E'	→ + T E' | ε
T	→ F T'
T'	→ * F T' | ε
F	→ (E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'



(	id	*	id	)	+	id	$

Table [F, id] = F→id
Pop F
Push id
Print F→id

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

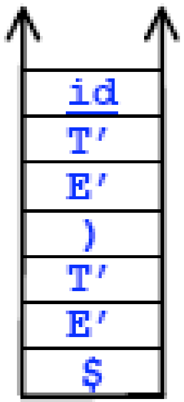
E	→ T E'
E'	→ + T E' | ε
T	→ F T'
T'	→ * F T' | ε
F	→ (E) | id

Input:

(id*id)+id

Output:

E	→ T E'
T	→ F T'
F	→ (E)
E	→ T E'
T	→ F T'
F	→ id



(	id	*	id	)	+	id	$

Table [F, id] = F→id

Pop F

Push id

Print F→id

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

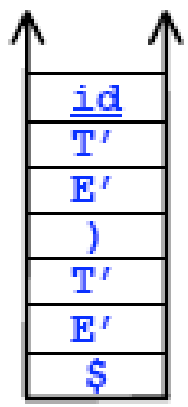
Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id

E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id



(	id	*	id	)	+	id	$

Top of Stack matches next input
Pop and Scan

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E'\ |\ \varepsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T'\ |\ \varepsilon$
$F \rightarrow (\ E\ )\ |\ \underline{id}$

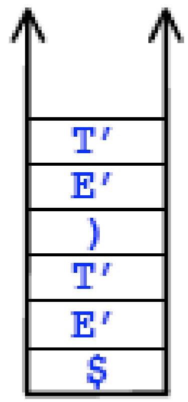
Example

Input:

(id*id)+id

Output:

$E \rightarrow T\ E'$
$T \rightarrow F\ T'$
$F \rightarrow (\ E\ )$
$E \rightarrow T\ E'$
$T \rightarrow F\ T'$
$F \rightarrow \underline{id}$



(	id	*	id	)	+	id	$

Table [T', ''] =* $T' \rightarrow *FT'$
Pop T'
Push T'
Push F
Push ''*
Print $T' \rightarrow *FT'$

	id	+	*	(	)	$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

Predictive Parsing

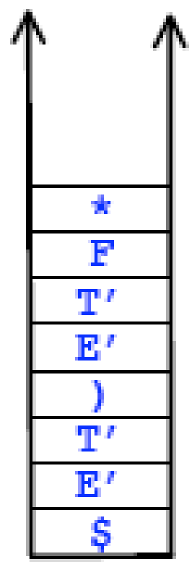
Example

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'



E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

(	id	*	id	)	+	id	$

Table [T', ''] = T'→*FT'*
Pop T'
Push T'
Push F
Push ''*
*Print T'→*FT'*

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

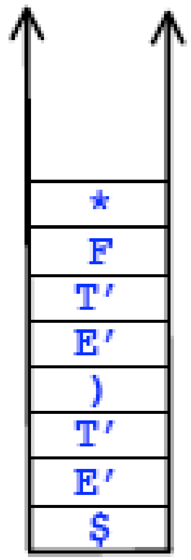
Example

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'



E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

(	id	*	id	)	+	id	$

Top of Stack matches next input
Pop and Scan

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

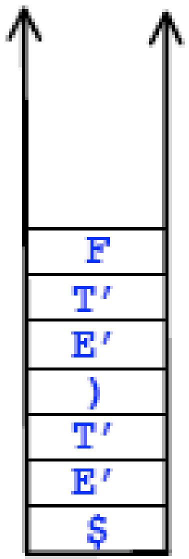
$E \rightarrow T\ E'$	
$E' \rightarrow +\ T\ E'$	ε
$T \rightarrow F\ T'$	
$T' \rightarrow *\ F\ T'$	ε
$F \rightarrow (\ E\ )$	<u>id</u>

Input:

(id*id)+id

Output:

$E \rightarrow T\ E'$
$T \rightarrow F\ T'$
$F \rightarrow (\ E\ )$
$E \rightarrow T\ E'$
$T \rightarrow F\ T'$
$F \rightarrow$ <u>id</u>
$T' \rightarrow *\ F\ T'$



(	<u>id</u>	*	<u>id</u>	)	+	<u>id</u>	$

Top of Stack matches next input
Pop and Scan

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→<u>id</u>			F→(E)		

Predictive Parsing

Example

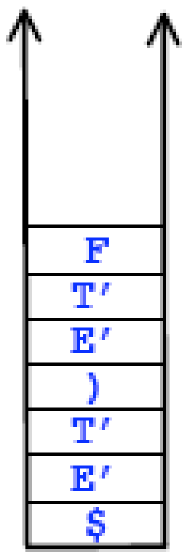
E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'



(	id	*	id	)	+	id	$

Table [F, id] = F→id
Pop F
Push id
Print F→id

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

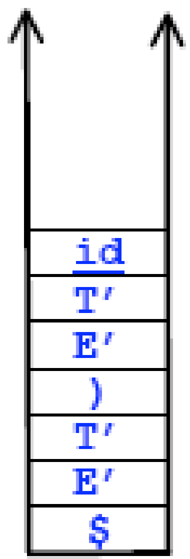
$E \rightarrow T\,E'$
$E' \rightarrow +\,T\,E' \mid \varepsilon$
$T \rightarrow F\,T'$
$T' \rightarrow *\,F\,T' \mid \varepsilon$
$F \rightarrow (\,E\,) \mid \underline{id}$

Input:

(id*id)+id

Output:

$E \rightarrow T\,E'$
$T \rightarrow F\,T'$
$F \rightarrow (\,E\,)$
$E \rightarrow T\,E'$
$T \rightarrow F\,T'$
$F \rightarrow \underline{id}$
$T' \rightarrow *\,F\,T'$
$F \rightarrow \underline{id}$



(	id	*	id	)	+	id	$

Table [F, id] = $F \rightarrow \underline{id}$
Pop F
Push id
Print $F \rightarrow \underline{id}$

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

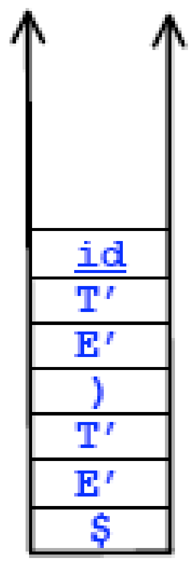
Example

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id



E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

(	id	*	id	)	+	id	$

Top of Stack matches next input
Pop and Scan

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

PREDICTIVE PARSING

Example

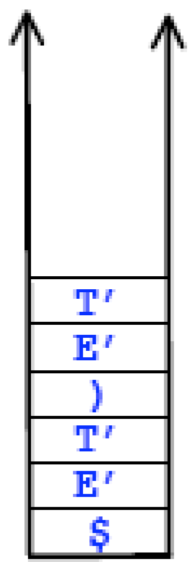
E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id



(	id	*	id	)	+	id	$

Top of Stack matches next input
Pop and Scan

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

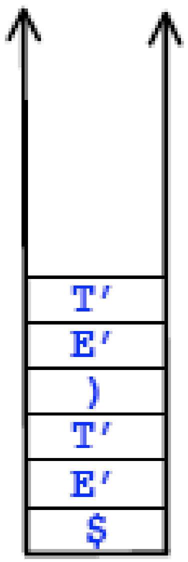
Example

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id



E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

(	id	*	id	)	+	id	$

Table [T', ')'] = T'→ε
Pop T'
Push <nothing>
Print T'→ε

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

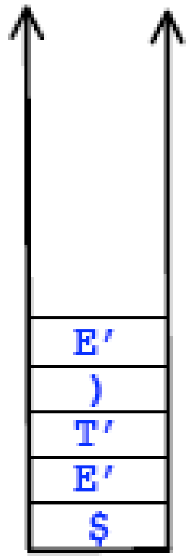
$E \rightarrow T\,E'$
$E' \rightarrow +\,T\,E' \mid \varepsilon$
$T \rightarrow F\,T'$
$T' \rightarrow *\,F\,T' \mid \varepsilon$
$F \rightarrow (\,E\,) \mid \underline{id}$

Input:

(id*id)+id

Output:

$E \rightarrow T\,E'$
$T \rightarrow F\,T'$
$F \rightarrow (\,E\,)$
$E \rightarrow T\,E'$
$T \rightarrow F\,T'$
$F \rightarrow \underline{id}$
$T' \rightarrow *\,F\,T'$
$F \rightarrow \underline{id}$
$T' \rightarrow \varepsilon$



(	id	*	id	)	+	id	$

Table [T', ')'] = $T' \rightarrow \varepsilon$
Pop T'
Push <nothing>
Print $T' \rightarrow \varepsilon$

	id	+	*	(	)	$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

Predictive Parsing

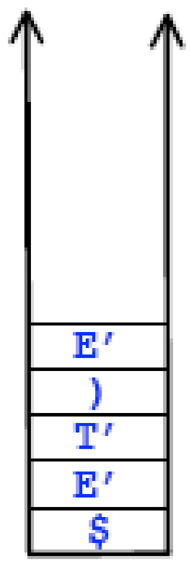
Example

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε



E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

(	id	*	id	)	+	id	$

Table [E', ')'] = E'→ε
Pop E'
Push <nothing>
Print E'→ε

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

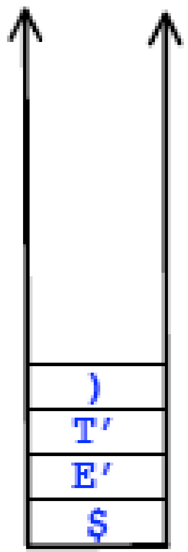
Example

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε



E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

(	id	*	id	)	+	id	$

Table [E', ')'] = E' → ε
Pop E'
Push <nothing>
Print E' → ε

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Predictive Parsing

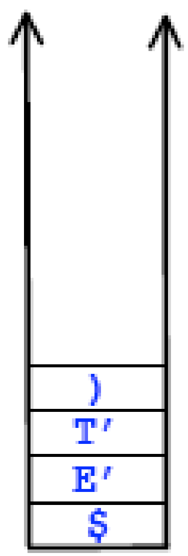
Example

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε



E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

(	id	*	id	)	+	id	$

Top of Stack matches next input
Pop and Scan

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

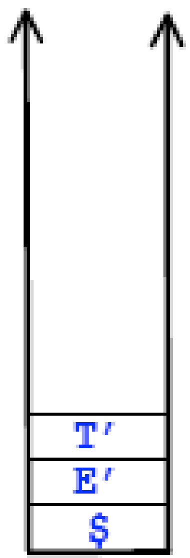
E	→ T E'
E'	→ + T E' | ε
T	→ F T'
T'	→ * F T' | ε
F	→ (E) | id

Input:

(id*id)+id

Output:

E	→ T E'
T	→ F T'
F	→ (E)
E	→ T E'
T	→ F T'
F	→ id
T'	→ * F T'
F	→ id
T'	→ ε
E'	→ ε



(	id	*	id	)	+	id	$

Top of Stack matches next input
Pop and Scan

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

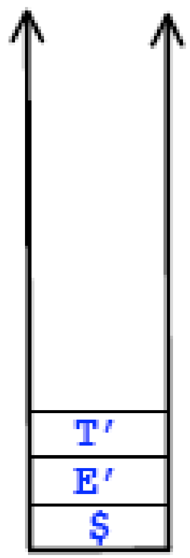
Example

Input:

(id*id)+id

Output:

E	→	T E'
T	→	F T'
F	→	(E)
E	→	T E'
T	→	F T'
F	→	id
T'	→	* F T'
F	→	id
T'	→	ε
E'	→	ε



E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

(	id	*	id	)	+	id	$

Table [T', '+'] = T'→ε
Pop T'
Push <nothing>
Print T'→ε

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

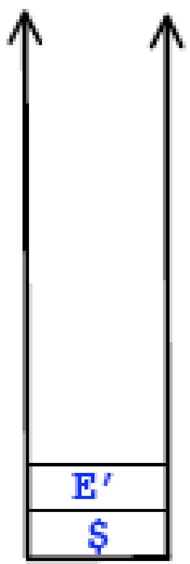
$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \varepsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \varepsilon$
$F \rightarrow (\ E\ ) \mid$ id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε



(	id	*	id	)	+	id	$

Table [T', '+'] = T'→ε
Pop T'
Push <nothing>
Print T'→ε

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

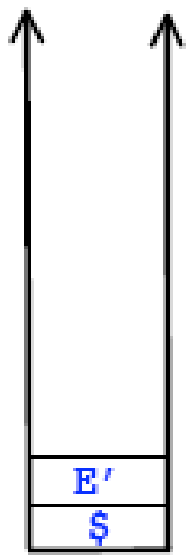
Example

Input:

(id*id)+id

Output:

E	→	T E'
T	→	F T'
F	→	(E)
E	→	T E'
T	→	F T'
F	→	id
T'	→	* F T'
F	→	id
T'	→	ε
E'	→	ε
T'	→	ε



E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

(	id	*	id	)	+	id	$

Table [E', '+'] = E'→+TE'
Pop E'
Push E'
Push T
Push '+'
Print E'→+TE'

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

PREDICTIVE PARSING

Example

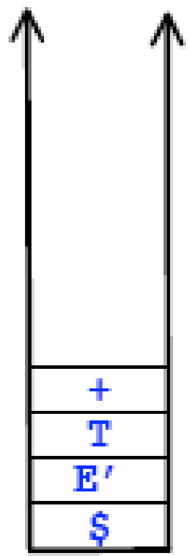
E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E	→	T E'
T	→	F T'
F	→	(E)
E	→	T E'
T	→	F T'
F	→	id
T'	→	* F T'
F	→	id
T'	→	ε
E'	→	ε
T'	→	ε
E'	→	+ T E'



(	id	*	id	)	+	id	$

Table [E', '+'] = E'→+TE'
Pop E'
Push E'
Push T
Push '+'
Print E'→+TE'

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

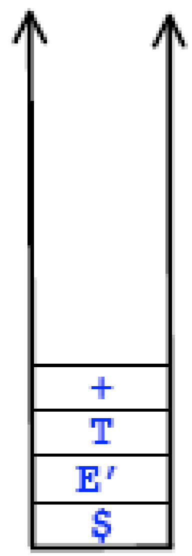
E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε
E' → + T E'



(	id	*	id	)	+	id	$

Top of Stack matches next input
Pop and Scan

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

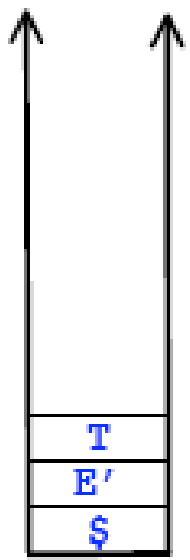
Example

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε
E' → + T E'



E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

(	id	*	id	)	+	id	$

Top of Stack matches next input
Pop and Scan

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

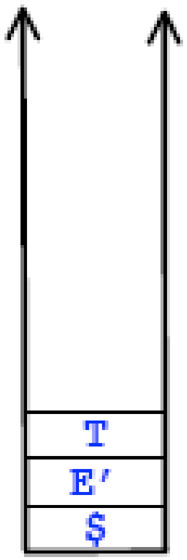
$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E'\ \vert\ \varepsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T'\ \vert\ \varepsilon$
$F \rightarrow (\ E\ )\ \vert\ \underline{id}$

Input:

(id*id)+id

Output:

$E \rightarrow T\ E'$
$T \rightarrow F\ T'$
$F \rightarrow (\ E\ )$
$E \rightarrow T\ E'$
$T \rightarrow F\ T'$
$F \rightarrow \underline{id}$
$T' \rightarrow *\ F\ T'$
$F \rightarrow \underline{id}$
$T' \rightarrow \varepsilon$
$E' \rightarrow \varepsilon$
$T' \rightarrow \varepsilon$
$E' \rightarrow +\ T\ E'$



(	id	*	id	)	+	id	$

Table [T, id] = $T \rightarrow FT'$
Pop T
Push T'
Push F
Print $T \rightarrow FT'$

	id	+	*	(	)	$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

Predictive Parsing

Example

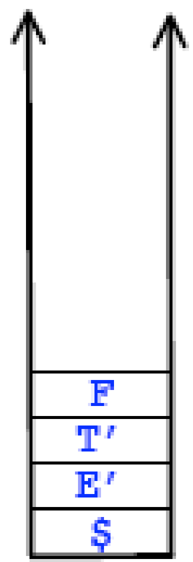
E	→	T E'
E'	→	+ T E' | ε
T	→	F T'
T'	→	* F T' | ε
F	→	(E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε
E' → + T E'
T → F T'



(	id	*	id	)	+	id	$

Table [T, id] = T→FT'
Pop T
Push T'
Push F
Print T→FT'

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

PREDICTIVE PARSING

Example

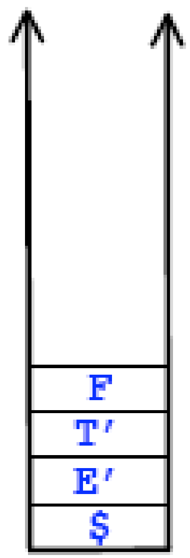
E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε
E' → + T E'
T → F T'



(	id	*	id	)	+	id	$

Table [F, id] = F→id
Pop F
Push id
Print F→id

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

PREDICTIVE PARSING

Example

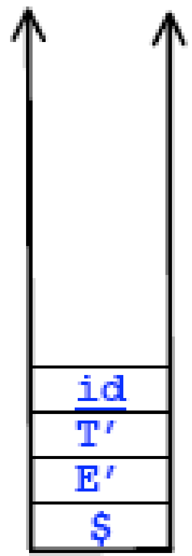
E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε
E' → + T E'
T → F T'
F → id



(	id	*	id	)	+	id	$

Table [F, id] = F→id
Pop F
Push id
Print F→id

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

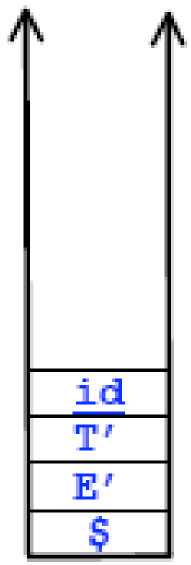
E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε
E' → + T E'
T → F T'
F → id



(	id	*	id	)	+	id	$

Top of Stack matches next input
Pop and Scan

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

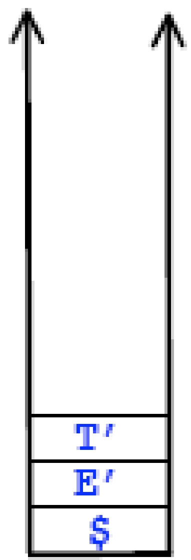
E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε
E' → + T E'
T → F T'
F → id



(	id	*	id	)	+	id	$

Top of Stack matches next input
Pop and Scan

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

PREDICTIVE PARSING

Example

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E'\ |\ \varepsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T'\ |\ \varepsilon$
$F \rightarrow (\ E\ )\ |\ \underline{id}$

Input:

(id*id)+id

Output:

$E \rightarrow T\ E'$
$T \rightarrow F\ T'$
$F \rightarrow (\ E\ )$
$E \rightarrow T\ E'$
$T \rightarrow F\ T'$
$F \rightarrow \underline{id}$
$T' \rightarrow *\ F\ T'$
$F \rightarrow \underline{id}$
$T' \rightarrow \varepsilon$
$E' \rightarrow \varepsilon$
$T' \rightarrow \varepsilon$
$E' \rightarrow +\ T\ E'$
$T \rightarrow F\ T'$
$F \rightarrow \underline{id}$

T'
E'
$

(	id	*	id	)	+	id	$

Table [T', $] = $T' \rightarrow \varepsilon$
Pop T'
Push <nothing>
Print $T' \rightarrow \varepsilon$

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

Example

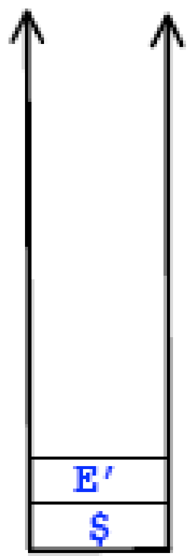
E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε
E' → + T E'
T → F T'
F → id
T' → ε



(	id	*	id	)	+	id	$

Table [T', $] = T'→ε
Pop T'
Push <nothing>
Print T'→ε

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

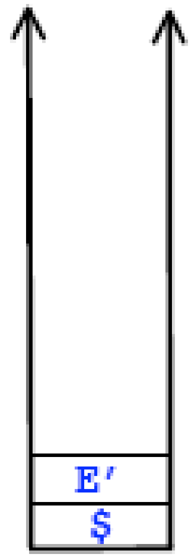
Example

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε
E' → + T E'
T → F T'
F → id
T' → ε



E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id

(	id	*	id	)	+	id	$

Table [E', $] = E'→ε
Pop E'
Push <nothing>
Print E'→ε

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

Predictive Parsing

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E'\ |\ \varepsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T'\ |\ \varepsilon$
$F \rightarrow (\ E\ )\ |\ \underline{id}$

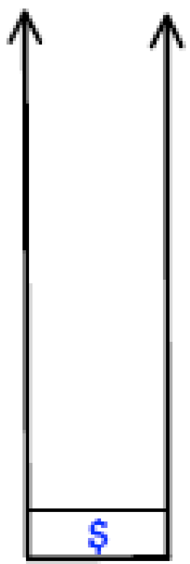
Example

Input:

(id*id)+id

Output:

$E \rightarrow T\ E'$
$T \rightarrow F\ T'$
$F \rightarrow (\ E\ )$
$E \rightarrow T\ E'$
$T \rightarrow F\ T'$
$F \rightarrow \underline{id}$
$T' \rightarrow *\ F\ T'$
$F \rightarrow \underline{id}$
$T' \rightarrow \varepsilon$
$E' \rightarrow \varepsilon$
$T' \rightarrow \varepsilon$
$E' \rightarrow +\ T\ E'$
$T \rightarrow F\ T'$
$F \rightarrow \underline{id}$
$T' \rightarrow \varepsilon$
$E' \rightarrow \varepsilon$



(	id	*	id	)	+	id	$

Table [E', $] = $E' \rightarrow \varepsilon$
Pop E'
Push <nothing>
Print $E' \rightarrow \varepsilon$

	id	+	*	(	)	$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

Predictive Parsing

Example

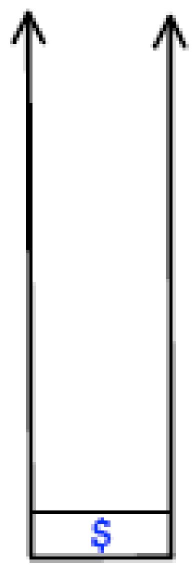
Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε
E' → + T E'
T → F T'
F → id
T' → ε
E' → ε

E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | id



(	id	*	id	)	+	id	$

Input symbol == $
Top of stack == $
Loop terminates with success

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ε	E'→ε
T	T→FT'			T→FT'		
T'		T'→ε	T'→*FT'		T'→ε	T'→ε
F	F→id			F→(E)		

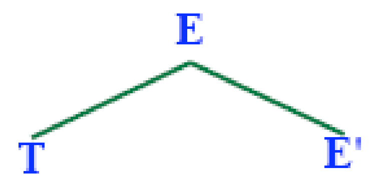
Reconstructing the Parse Tree

Input:

`(id*id)+id`

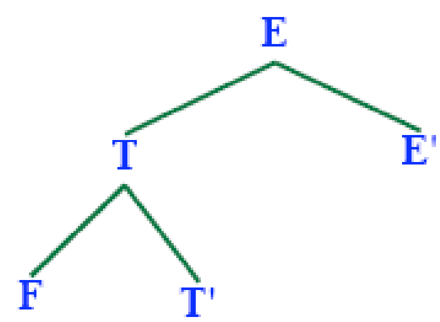
Output:

E → T E'



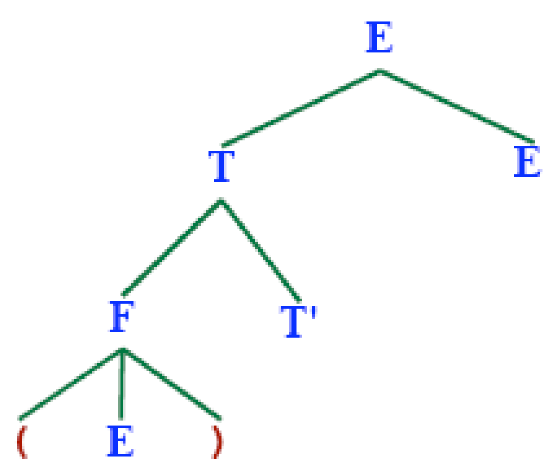
Output:

E → T E'
T → F T'



Output:

E → T E'
T → F T'
F → (E)



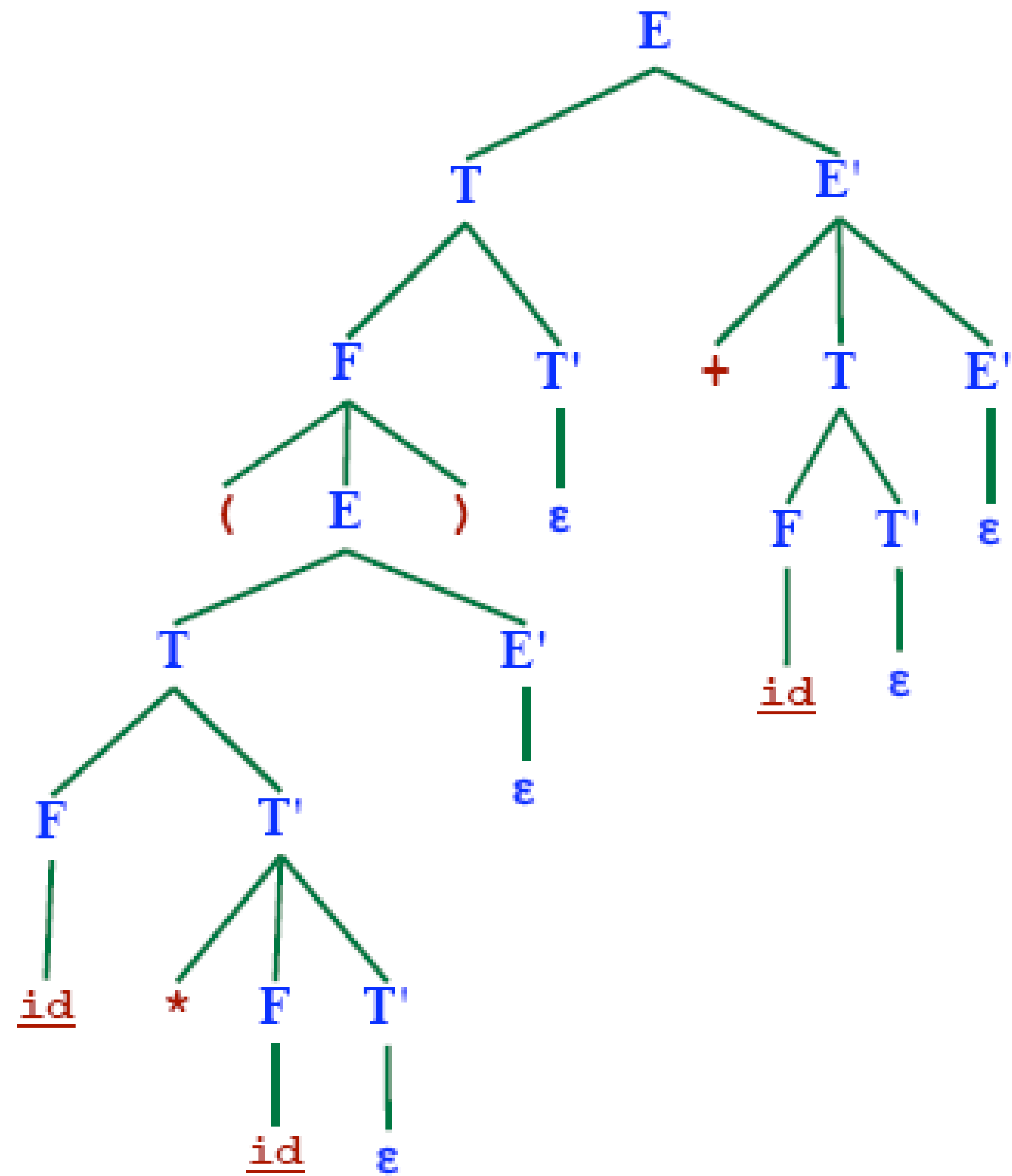
Reconstructing the Parse Tree

Input:

(id*id)+id

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε
E' → + T E'
T → F T'
F → id
T' → ε
E' → ε



Reconstructing the Parse Tree

Input:

`(id*id)+id`

Output:

E → T E'
T → F T'
F → (E)
E → T E'
T → F T'
F → id
T' → * F T'
F → id
T' → ε
E' → ε
T' → ε
E' → + T E'
T → F T'
F → id
T' → ε
E' → ε

Leftmost Derivation:

E
T E'
F T' E'
(E) T' E'
(T E') T' E'
(F T' E') T' E'
(id T' E') T' E'
(id * F T' E') T' E'
(id * id T' E') T' E'
(id * id E') T' E'
(id * id) T' E'
(id * id) E'
(id * id) + T E'
(id * id) + F T' E'
(id * id) + id T' E'
(id * id) + id E'
(id * id) + id

Example: The "Dangling Else" Grammar

1. $S \rightarrow$ i E t S S'
2. $S \rightarrow$ o
3. $S' \rightarrow$ e S
4. $S' \rightarrow \varepsilon$
5. $E \rightarrow$ b

←

1. $S \rightarrow$ if E then S S'
2. $S \rightarrow$ otherStmt
3. $S' \rightarrow$ else S
4. $S' \rightarrow \varepsilon$
5. $E \rightarrow$ boolExpr

Example: The "Dangling Else" Grammar

1. S → <u>i</u> E <u>t</u> S S'
2. S → <u>o</u>
3. S' → <u>e</u> S
4. S' → ε
5. E → <u>b</u>

<u>i</u> <u>b</u> <u>t</u> <u>i</u> <u>b</u> <u>t</u> <u>o</u> <u>e</u> <u>o</u>

FIRST(S) = { <u>i</u>, <u>o</u> }	FOLLOW(S) = { <u>e</u>, $ }
FIRST(S') = { <u>e</u>, ε }	FOLLOW(S') = { <u>e</u>, $ }
FIRST(E) = { <u>b</u> }	FOLLOW(E) = { <u>t</u> }

Example: The "Dangling Else" Grammar

1. S → i E t S S'
2. S → o
3. S' → e S
4. S' → ε
5. E → b

Look at Rule 1: S → i E t S S'
If we are looking for an S
and the next symbol is in FIRST(i E t S S')...
Add that rule to the table

i b t i b t o e o

FIRST(S) = { i, o }
FIRST(S') = { e, ε }
FIRST(E) = { b }

FOLLOW(S) = { e, $ }
FOLLOW(S') = { e, $ }
FOLLOW(E) = { t }

	o	b	e	i	t	$
S						
S'						
E						

EXAMPLE: THE "DANGLING ELSE" GRAMMAR

1. S → i E t S S'
2. S → o
3. S' → e S
4. S' → ε
5. E → b

Look at Rule 1: S → i E t S S'
If we are looking for an S
and the next symbol is in FIRST(i E t S S')...
Add that rule to the table

i b t i b t o e o

FIRST(S) = { i, o }   FOLLOW(S) = { e, $ }
FIRST(S') = { e, ε }   FOLLOW(S') = { e, $ }
FIRST(E) = { b }   FOLLOW(E) = { t }

	o	b	e	i	t	$
S				S → iEtSS'		
S'						
E						

Example: The "Dangling Else" Grammar

1. S → i E t S S'
2. S → o
3. S' → e S
4. S' → ε
5. E → b

Look at Rule 2: S → o
If we are looking for an S
and the next symbol is in FIRST(o)...
Add that rule to the table

i b t i b t o e o

FIRST(S) = { i, o }
FIRST(S') = { e, ε }
FIRST(E) = { b }

FOLLOW(S) = { e, $ }
FOLLOW(S') = { e, $ }
FOLLOW(E) = { t }

	o	b	e	i	t	$
S	S → o			S → iEtSS'		
S'						
E						

Example: The "Dangling Else" Grammar

1. S → i E t S S'
2. S → o
3. S' → e S
4. S' → ε
5. E → b

Look at Rule 5: E → b
If we are looking for an E
and the next symbol is in FIRST(b)...
Add that rule to the table

i b t i b t o e o

FIRST(S) = { i, o } FOLLOW(S) = { e, $ }
FIRST(S') = { e, ε } FOLLOW(S') = { e, $ }
FIRST(E) = { b } FOLLOW(E) = { t }

	o	b	e	i	t	$
S	S → o			S → iEtSS'		
S'						
E		E → b				

EXAMPLE: THE "DANGLING ELSE" GRAMMAR

1. $S \rightarrow$ i E t S S'
2. $S \rightarrow$ o
3. $S' \rightarrow$ e S
4. $S' \rightarrow \varepsilon$
5. $E \rightarrow$ b

Look at Rule 3: $S' \rightarrow$ e S
If we are looking for an S'
and the next symbol is in FIRST(e S)...
Add that rule to the table

i b t i b t o e o

FIRST(S) = { i, o }   FOLLOW(S) = { e, $ }
FIRST(S') = { e, ε }   FOLLOW(S') = { e, $ }
FIRST(E) = { b }   FOLLOW(E) = { t }

	o	b	e	i	t	$
S	S → o			S → iEtSS'		
S'			S' → e S			
E		E → b				

Example: The "Dangling Else" Grammar

1. $S \rightarrow$ i E t S S'
2. $S \rightarrow$ o
3. $S' \rightarrow$ e S
4. $S' \rightarrow \varepsilon$
5. $E \rightarrow$ b

Look at Rule 4: $S' \rightarrow \varepsilon$

If we are looking for an S'

and $\varepsilon \in$ FIRST(rhs)...

Then if $ $\in$ FOLLOW(S')...

Add that rule under $

i b t i b t o e o

FIRST(S) = { i, o }   FOLLOW(S) = { e, $ }

FIRST(S') = { e, ε }   FOLLOW(S') = { e, $ }

FIRST(E) = { b }   FOLLOW(E) = { t }

	o	b	e	i	t	$
S	S → o			S → iEtSS'		
S'			S' → e S			
E		E → b				

Example: The "Dangling Else" Grammar

1. $S \rightarrow \underline{\mathtt{i}}\ E\ \underline{\mathtt{t}}\ S\ S'$
2. $S \rightarrow \underline{\mathtt{o}}$
3. $S' \rightarrow \underline{\mathtt{e}}\ S$
4. $S' \rightarrow \varepsilon$
5. $E \rightarrow \underline{\mathtt{b}}$

Look at Rule 4: $S' \rightarrow \varepsilon$

If we are looking for an S'

and $\varepsilon \in$ FIRST(rhs)...

Then if \$ $\in$ FOLLOW(S')...

Add that rule under \$

i b t i b t o e o

FIRST(S) = { i, o }
FIRST(S') = { e, ε }
FIRST(E) = { b }

FOLLOW(S) = { e, \$ }
FOLLOW(S') = { e, \$ }
FOLLOW(E) = { t }

	o	b	e	i	t	\$
S	$S \rightarrow \mathtt{o}$			$S \rightarrow \mathtt{i}E\mathtt{t}SS'$		
S'			$S' \rightarrow \mathtt{e}\ S$			$S' \rightarrow \varepsilon$
E		$E \rightarrow \mathtt{b}$				

Example: The "Dangling Else" Grammar

1. $S \rightarrow \underline{i}\ E\ \underline{t}\ S\ S'$
2. $S \rightarrow \underline{o}$
3. $S' \rightarrow \underline{e}\ S$
4. $S' \rightarrow \varepsilon$
5. $E \rightarrow \underline{b}$

Look at Rule 4: $S' \rightarrow \varepsilon$

If we are looking for an S'

and $\varepsilon \in$ FIRST(rhs)...

Then if $\underline{e} \in$ FOLLOW(S')...

Add that rule under $\underline{e}$

$\underline{i}\ \underline{b}\ \underline{t}\ \underline{i}\ \underline{b}\ \underline{t}\ \underline{o}\ \underline{e}\ \underline{o}$

FIRST(S) = { $\underline{i}$, $\underline{o}$ } FOLLOW(S) = { $\underline{e}$, \$ }

FIRST(S') = { $\underline{e}$, ε } FOLLOW(S') = { $\underline{e}$, \$ }

FIRST(E) = { $\underline{b}$ } FOLLOW(E) = { $\underline{t}$ }

	o	b	e	i	t	$
S	$S \rightarrow \underline{o}$			$S \rightarrow \underline{i}E\underline{t}SS'$		
S'			$S' \rightarrow \underline{e}\ S$ $S' \rightarrow \varepsilon$			$S' \rightarrow \varepsilon$
E		$E \rightarrow \underline{b}$				

Example: The "Dangling Else" Grammar

1. S → i E t S S'
2. S → o
3. S' → e S
4. S' → ε
5. E → b

CONFLICT!

Two rules in one table entry.

The grammar is not LL(1)!

i b t i b t o e o

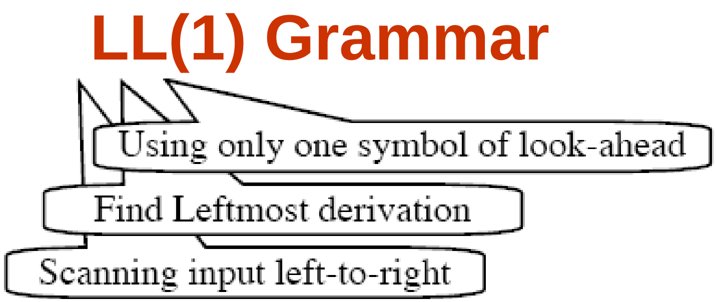
FIRST(S) = { i, o } FOLLOW(S) = { e, $ }

FIRST(S') = { e, ε } FOLLOW(S') = { e, $ }

FIRST(E) = { b } FOLLOW(E) = { t }

	o	b	e	i	t	$
S	S → o			S → iEtSS'		
S'			S' → e S S' → ε			S' → ε
E		E → b				

LL(1) GRAMMAR



- LL(1) grammars
  - Are never ambiguous.
  - Will never have left recursion.
- ***Furthermore…***
  - If we are looking for an “A” and the next symbol is “b”,
  - Then only one production must be possible
- Although elimination of left recursion and left factoring is easy.
  - Some grammar will never be a LL(1) grammar.

LL(1) Grammar

A Grammar which is not LL(1)

- What do we have to do it if the resulting parsing table contains multiply defined entries?
  - If we didn't eliminate left recursion, eliminate the left recursion in the grammar.
  - If the grammar is not left factored, we have to left factor the grammar.
  - If its (new grammar's) parsing table still contains multiply defined entries, that grammar is ambiguous or it is inherently not a LL(1) grammar.
- A left recursive grammar cannot be a LL(1) grammar.
  - $A \rightarrow A\alpha \mid \beta$

    ➔ any terminal that appears in FIRST(β) also appears FIRST($A\alpha$) because $A\alpha \Rightarrow \beta\alpha$.

    ➔ If β is ε, any terminal that appears in FIRST(α) also appears in FIRST($A\alpha$) and FOLLOW(A).
- A grammar is not left factored, it cannot be a LL(1) grammar
  - $A \rightarrow \alpha\beta_1 \mid \alpha\beta_2$

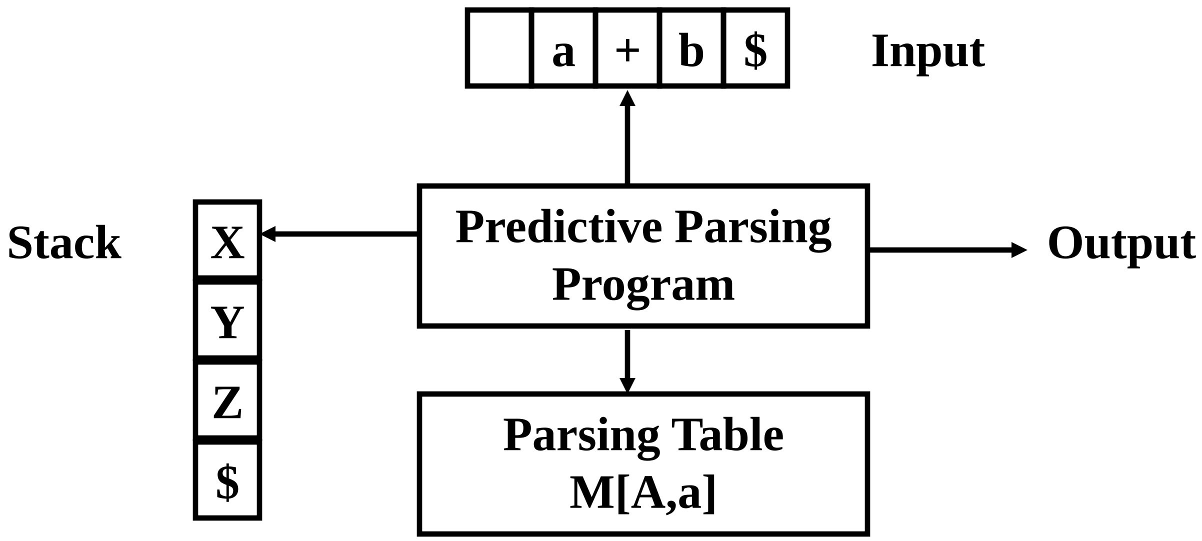
    ➔any terminal that appears in FIRST($\alpha\beta_1$) also appears in FIRST($\alpha\beta_2$).
- An ambiguous grammar cannot be a LL(1) grammar.

Properties of LL(1) Grammar

- A grammar G is LL(1) if an only if whenever $A \rightarrow \alpha \mid \beta$ are two distinct productions of G the following conditions hold:
  1. For no terminal a do both α and β derive strings beginning with a.
  2. At most one of α and β can derive the empty string.
  3. If then $\beta \Rightarrow^* \varepsilon$, then α does not derive any string beginning with a terminal in FOLLOW(A).

ERROR RECOVERY

When Do Errors Occur? Recall Predictive Parser Function:



1. If X is a terminal and it doesn't match input.
2. If M[X, Input] is empty – No allowable actions

Error Recovery

Options

1. Skip over input symbols, until we can resume parsing

   Corresponds to ignoring tokens
2. Pop stack, until we can resume parsing

   Corresponds to inserting missing material
3. Some combination of 1 and 2
4. "Panic Mode" - Use Synchronizing tokens
   - Identify a set of synchronizing tokens.
   - Skip over tokens until we are positioned on a synchronizing token.
   - Pop stack until we can resume parsing.

Error Recovery: Skip Input Symbols

Example:

Decided to use rule

$S \rightarrow$ **IF** E **THEN** S **ELSE** S **END**

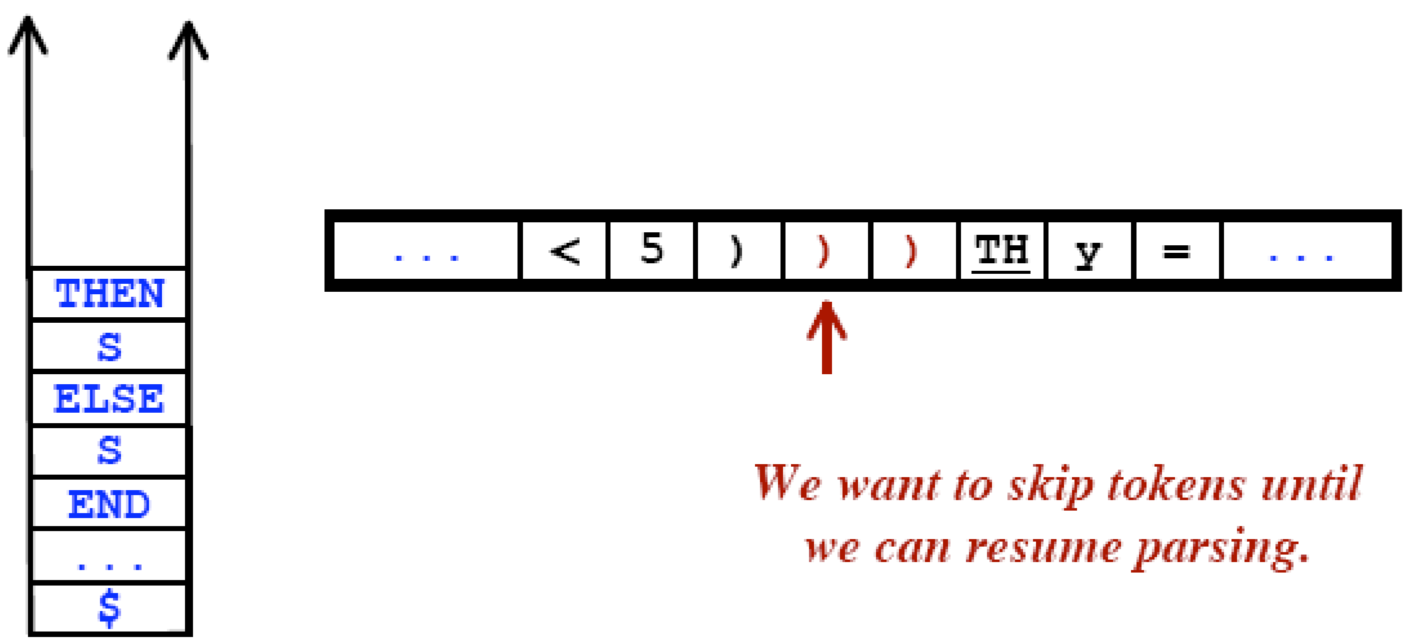
Stack tells us what we are expecting next in the input.

We've already gotten **IF** and E

Assume there are extra tokens in the input.

```
if (x<5))) then y = 7; ...
```

A syntax error occurs here.



We want to skip tokens until we can resume parsing.

Error Recovery: Pop The Stack

Example:

Decided to use rules

$S \rightarrow$ **IF** E **THEN** S **ELSE** S **END**

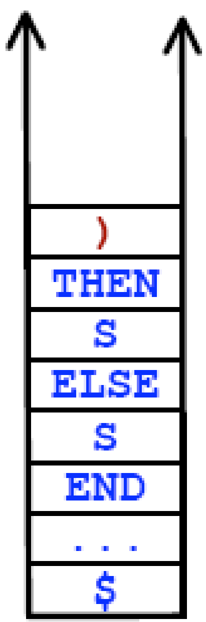
$E \rightarrow$ **(** E **)**

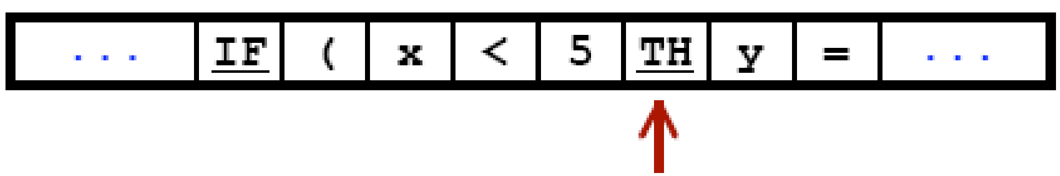
We've already gotten **`if (`** E

Assume there are missing tokens.

`if (x < 5 then y = 7;...`

A syntax error occurs here.





We want to pop the stack until we can resume parsing.

Error Recovery: Panic Mode

The "*Synchronizing Set*" of tokens

... is determined by the compiler writer beforehand

Example: { SEMI-COLON, RIGHT-BRACE }

Skip input symbols until we find something in the synchronizing set.

Idea:

Look at the non-terminal on the stack top.

Choose the synchronizing set based on this non-terminal.

Assume A is on the stack top

Let SynchSet = FOLLOW(A)

Skip tokens until we see something in FOLLOW(A)

Pop A from the stack.

Should be able to keep going.

Idea:

Look at the non-terminals in the stack (e.g., A, B, C, ...)

Include FIRST(A), FIRST(B), FIRST(C), ... in the SynchSet.

Skip tokens until we see something in FIRST(A), FIRST(B), FIRST(C), ...

Pop stack until **NextToken** ∈ FIRST(**NonTerminalOnStackTop**)

Error Recovery - Table Entries

Each blank entry in the table indicates an error.
Tailor the error recovery for each possible error.
Fill the blank entry with an error routine.
    The error routine will tell what to do.

Example:

	id	SEMI	RPAREN	LPAREN	...	$
E			E4			
E'			E5			
...						

Choose the SynchSet based on the particular error

Error-Handling Code

```
...
E4:
          SynchSet = { SEMI, IF, THEN }
          SkipTokensTo (SynchSet)
          Print ("Unexpected right paren")
          Pop stack
          break
E5:       ...
...
```

QUESTIONS ?