

# Parsing

## Part III

# Top Down Parsing

- Find a left-most derivation
- Find (build) a parse tree
- Start building from the root and work down...
- As we search for a derivation
  - Must make choices:
    - Which rule to use
    - Where to use it
- May run into problems!!

# Top-Down Parsing

- Recursive-Descent Parsing
  - Backtracking is needed (If a choice of a production rule does not work, we backtrack to try other alternatives.)
  - It is a general parsing technique, but not widely used.
  - Not efficient
- ~~P~~redictive Parsing
  - no backtracking ✓
  - efficient ✓
  - needs a special form of grammars (LL(1) grammars).
  - Recursive Predictive Parsing is a special form of Recursive Descent parsing without backtracking.
  - Non-Recursive (Table Driven) Predictive Parser is also known as LL(1) parser.

# Recursive Descent Parsing (Backtracking)

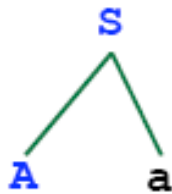
Input: aabbde  
↑

S

1. S  $\rightarrow$  Aa ✗
2.      $\rightarrow$  Ce ✓
3. A  $\rightarrow$  aaB
4.      $\rightarrow$  aaba
5. B  $\rightarrow$  bbb
6. C  $\rightarrow$  aaD
7. D  $\rightarrow$  bbd

# Recursive Descent Parsing (Backtracking)

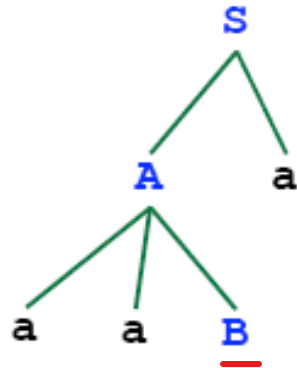
Input: aabbde



1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$  ✓
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

# Recursive Descent Parsing (Backtracking)

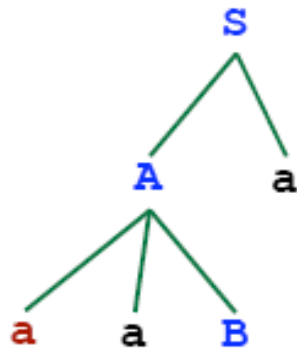
Input: aabbde



1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

# Recursive Descent Parsing (Backtracking)

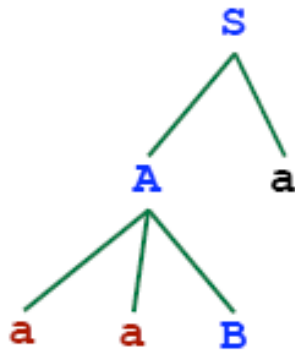
Input: aabbde



1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

# Recursive Descent Parsing (Backtracking)

Input: aabbde

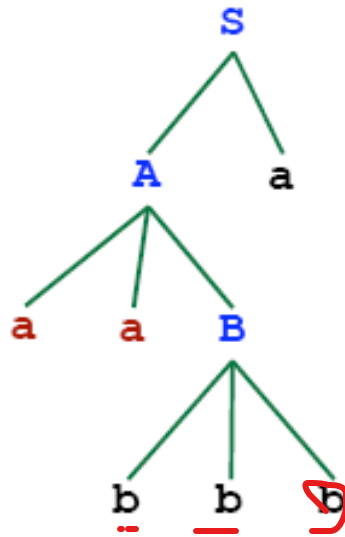


1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$



# Recursive Descent Parsing (Backtracking)

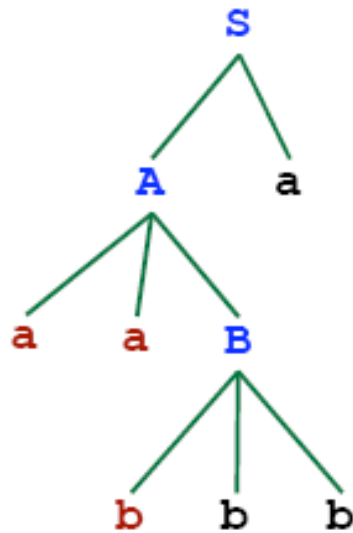
Input: aabbde



1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

# Recursive Descent Parsing (Backtracking)

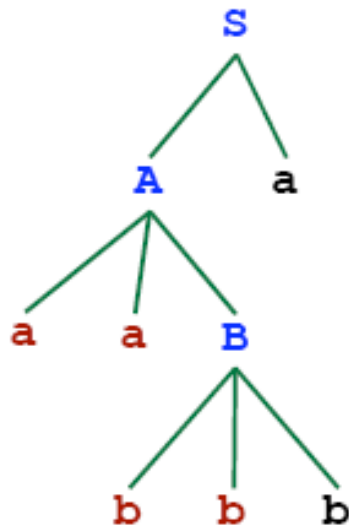
Input: aabbde



1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

# Recursive Descent Parsing (Backtracking)

Input: aabbde

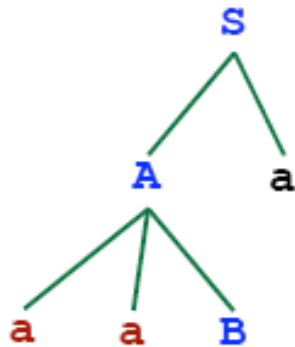


Failure Occurs Here!!!

1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

# Recursive Descent Parsing (Backtracking)

Input: aabbde

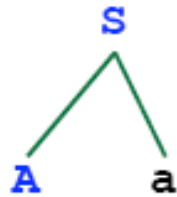


1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

*We need an ability to  
back up in the input!!!*

# Recursive Descent Parsing (Backtracking)

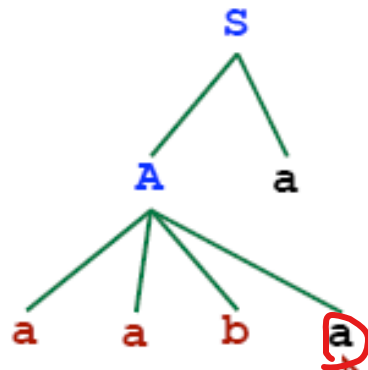
Input: aabbde



1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$  ✓
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

# Recursive Descent Parsing (Backtracking)

Input: aabbde

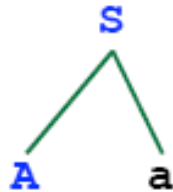


Failure Occurs Here!!!

1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

# Recursive Descent Parsing (Backtracking)

Input: aabbde



1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

# Recursive Descent Parsing (Backtracking)

Input: aabbde



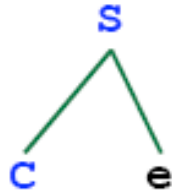
S

1. S  $\rightarrow$  Aa
2.  $\rightarrow$  Ce ✓
3. A  $\rightarrow$  aaB
4.  $\rightarrow$  aaba
5. B  $\rightarrow$  bbb
6. C  $\rightarrow$  aaD
7. D  $\rightarrow$  bbd



# Recursive Descent Parsing (Backtracking)

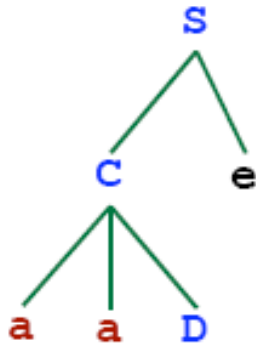
Input: aabbde



1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

# Recursive Descent Parsing (Backtracking)

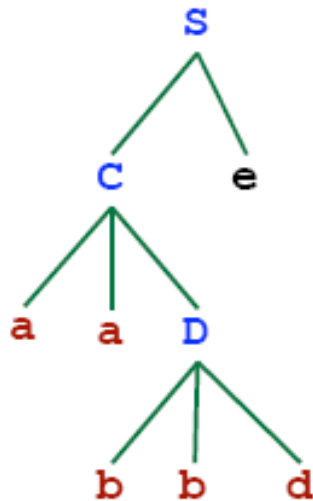
Input: aabbde



1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

# Recursive Descent Parsing (Backtracking)

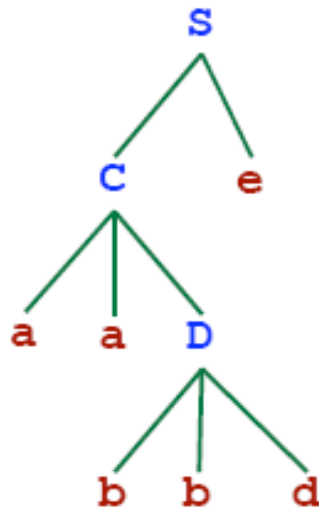
Input: aabbde



1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

# Recursive Descent Parsing (Backtracking)

Input: aabbde



1.  $S \rightarrow Aa$
2.  $\rightarrow Ce$
3.  $A \rightarrow aaB$
4.  $\rightarrow aaba$
5.  $B \rightarrow bbb$
6.  $C \rightarrow aaD$
7.  $D \rightarrow bbd$

Successfully parsed!!

# Recursive-Descent Parsing Algorithm

- A recursive-descent parsing program consists of a set of procedures – one for each non-terminal
- Execution begins with the procedure for the start symbol
  - Announces success if the procedure body scans the entire input

```
void A(){  
    for (j=1 to t){ /* assume there is t number of A-productions */  
        Choose a A-production,  $A_j \rightarrow X_1 X_2 \dots X_k$ ;  
        for (i=1 to k){  
            if ( $X_i$  is a non-terminal)  
                call procedure  $X_i()$ ;  
            else if ( $X_i$  equals the current input symbol  $a$ )  
                advance the input to the next symbol;  
            else backtrack in input and reset the pointer  
        }  
    }  
}
```

## Predictive Parser

When re-writing a non-terminal in a derivation step, a predictive parser can uniquely choose a production rule by just looking the current symbol in the input string.

$A \rightarrow \alpha_1 \mid \dots \mid \alpha_n$

input: ... a .....

↑  
current token

## Predictive Parser (example)

```
stmt → if ..... |  
      while ..... |  
      begin ..... |  
      for .....
```

- When we are trying to write the non-terminal *stmt*, if the current token is `if` we have to choose first production rule.
- When we are trying to write the non-terminal *stmt*, we can uniquely choose the production rule by just looking the current token.
- We eliminate the left recursion in the grammar, and left factor it. But it may not be suitable for predictive parsing (not LL(1) grammar).

## Recursive Predictive Parsing

- Each non-terminal corresponds to a procedure.

Ex:  $A \rightarrow aBb$  (This is only the production rule for A)

proc A {

- match the current token with a, and move to the next token;
- call 'B';
- match the current token with b, and move to the next token;

}



## Recursive Predictive Parsing (cont.)

$A \rightarrow aBb \mid bAB$

```
proc A {  
  case of the current token {  
    'a': - match the current token with a, and move to the next token;  
        - call 'B';  
        - match the current token with b, and move to the next token;  
  
    'b': - match the current token with b, and move to the next token;  
        - call 'A';  
        - call 'B';  
  }  
}
```

## Recursive Predictive Parsing (cont.)

- When to apply  $\varepsilon$ -productions.

$$A \rightarrow aA \mid bB \mid \varepsilon$$

- If all other productions fail, we should apply an  $\varepsilon$ -production. For example, if the current token is not a or b, we may apply the  $\varepsilon$ -production.
- **Most correct choice:** We should apply an  $\varepsilon$ -production for a non-terminal A when the current token is in the follow set of A (which terminals can follow A in the sentential forms).


# Recursive Predictive Parsing (Example)

$A \rightarrow aBe \mid cBd \mid C$

$B \rightarrow bB \mid \epsilon$


$C \rightarrow f$

```
proc A {  
  case of the current token {  
    a: - match the current token with a,  
        and move to the next token;  
        - call B;  
        - match the current token with e,  
        and move to the next token;  
    c: - match the current token with c,  
        and move to the next token;  
        - call B;  
        - match the current token with d,  
        and move to the next token;  
    f: - call C  
  }  
}
```

 first set of C

```
proc C { match the current token with f,  
        and move to the next token; }
```

```
proc B {  
  case of the current token {  
    b: - match the current token with b,  
        and move to the next token;  
        - call B  
    e,d: do nothing  
  }  
}
```

 follow set of B

# First Function

Let  $\alpha$  be a string of symbols (terminals and nonterminals)

**Define:**

$\text{FIRST}(\alpha)$  = The set of terminals that could occur first  
in any string derivable from  $\alpha$   
 $= \{ a \mid \alpha \Rightarrow^* aw, \text{ plus } \epsilon \text{ if } \alpha \Rightarrow^* \epsilon \}$

**Example:**

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

$\text{FIRST}(F) = \{ (, id \}$

$\text{FIRST}(T') = \{ *, \epsilon \}$

$\text{FIRST}(T) = \{ (, id \}$

$\text{FIRST}(E') = \{ +, \epsilon \}$

$\text{FIRST}(E) = \{ (, id \}$

## 1. Computing the First Function

For all symbols  $X$  in the grammar...

if  $X$  is a terminal then  
     $\text{FIRST}(X) = \{ X \}$

if  $X \rightarrow \epsilon$  is a rule then  
    add  $\epsilon$  to  $\text{FIRST}(X)$

if  $X \rightarrow Y_1 Y_2 Y_3 \dots Y_K$  is a rule then  
    if  $a \in \text{FIRST}(Y_1)$  then  
        add  $a$  to  $\text{FIRST}(X)$   
    if  $\epsilon \in \text{FIRST}(Y_1)$  and  $a \in \text{FIRST}(Y_2)$  then  
        add  $a$  to  $\text{FIRST}(X)$   
    if  $\epsilon \in \text{FIRST}(Y_1)$  and  $\epsilon \in \text{FIRST}(Y_2)$  and  $a \in \text{FIRST}(Y_3)$  then  
        add  $a$  to  $\text{FIRST}(X)$   
    ...  
    if  $\epsilon \in \text{FIRST}(Y_i)$  for all  $Y_i$  then  
        add  $\epsilon$  to  $\text{FIRST}(X)$

*Repeat until nothing more can be added to any sets.*

## To Compute the FIRST( $X_1X_2X_3\dots X_N$ )

```
Result = {}
Add everything in FIRST( $X_1$ ), except  $\epsilon$ , to result
if  $\epsilon \in \text{FIRST}(X_1)$  then
    Add everything in FIRST( $X_2$ ), except  $\epsilon$ , to result
    if  $\epsilon \in \text{FIRST}(X_2)$  then
        Add everything in FIRST( $X_3$ ), except  $\epsilon$ , to result
        if  $\epsilon \in \text{FIRST}(X_3)$  then
            Add everything in FIRST( $X_4$ ), except  $\epsilon$ , to result
            ...
            if  $\epsilon \in \text{FIRST}(X_{N-1})$  then
                Add everything in FIRST( $X_N$ ), except  $\epsilon$ , to result
                if  $\epsilon \in \text{FIRST}(X_N)$  then
                    // Then  $X_1 \Rightarrow^* \epsilon, X_2 \Rightarrow^* \epsilon, X_3 \Rightarrow^* \epsilon, \dots X_N \Rightarrow^* \epsilon$ 
                    Add  $\epsilon$  to result
                endif
            endif
        endif
    endif
    ...
    endif
endif
endif
```

## First - Example

- $P \rightarrow i \mid c \mid n T S$
- $Q \rightarrow P \mid a S \mid b S c S T$
- $R \rightarrow b \mid \epsilon$
- $S \rightarrow c \mid R n \mid \epsilon$
- $T \rightarrow R S q$

$$P = i, c, n$$

$$Q = a, b, i, c, n$$

$$R = b, \epsilon$$

$$S = c, \epsilon, b, n$$

$$T = b, c, n, q$$

- $\text{FIRST}(P) = \{i, c, n\}$
- $\text{FIRST}(Q) = \{i, c, n, a, b\}$
- $\text{FIRST}(R) = \{b, \epsilon\}$
- $\text{FIRST}(S) = \{c, b, n, \epsilon\}$
- $\text{FIRST}(T) = \{b, c, n, q\}$

## First - Example

- $S \rightarrow a S e \mid S T S$
- $T \rightarrow R S e \mid Q$
- $R \rightarrow r S r \mid \varepsilon$
- $Q \rightarrow S T \mid \varepsilon$

- $\text{FIRST}(S) = \{a\}$
- $\text{FIRST}(R) = \{r, \varepsilon\}$
- $\text{FIRST}(T) = \{r, a, \varepsilon\}$
- $\text{FIRST}(Q) = \{a, \varepsilon\}$



## 1. FOLLOW Sets

- FOLLOW(A) is the set of terminals (including end marker of input - \$) that may follow non-terminal A in some sentential form.
- $\text{FOLLOW}(A) = \{c \mid S \Rightarrow^+ \dots Ac\dots\} \cup \{\$\}$  if  $S \Rightarrow^+ \dots A$
- For example, consider  $L \Rightarrow^+ (())(L)L$   
Both ')' and end of file can follow L
- NOTE:  $\epsilon$  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  $\text{FIRST}(\beta) - \{\epsilon\}$  to FOLLOW(A)

---

INTUITION: Suppose  $B \rightarrow AX$  and  $\text{FIRST}(X) = \{c\}$

$S \Rightarrow^+ \alpha B \beta \Rightarrow \alpha A X \beta \Rightarrow^+ \alpha A c \delta \beta$

$= \text{FIRST}(X)$



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)

---

INTUITION:

— Suppose  $B \rightarrow Y A$

$S \Rightarrow^+ \alpha B \beta \Rightarrow \alpha Y A \beta$

FOLLOW(B)

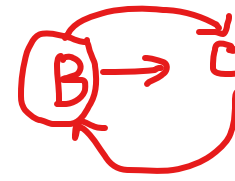


— Suppose  $B \rightarrow A X$  and  $X \Rightarrow^* \lambda$

$S \Rightarrow^+ \alpha B \beta \Rightarrow \alpha A X \beta \Rightarrow^* \alpha A \beta$

FOLLOW(B)





## Example

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

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

- $\text{FOLLOW}(C) = g, \$, c, d, e, \phi$

- $\text{FOLLOW}(B) = \{c, d, f, \$, e\}$

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

Using rule #1

Assume the first non-terminal is  
the start symbol

## Example

- $S \rightarrow a \underline{S} e \mid B$
- $B \rightarrow b \underline{B C} f \mid C$
- $C \rightarrow c \underline{C} g \mid d \mid \epsilon$
- $\text{FIRST}(C) = \{c, d, \epsilon\}$
- $\text{FIRST}(B) = \{b, c, d, \epsilon\}$
- $\text{FIRST}(S) = \{a, b, c, d, \epsilon\}$
- $\text{FOLLOW}(C) = \{f, g\}$
- $\text{FOLLOW}(B) = \{c, d, f\}$
- $\text{FOLLOW}(S) = \{\$, e\}$

Using rule #2

## Example

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

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

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

Using rule #3

## Example

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

- $\text{FIRST}(T) = \{ (, a, b, c \}$
- $\text{FIRST}(E) = \{ \&, \varepsilon \}$
- $\text{FIRST}(A) = \{ (, a, b, c \}$
- $\text{FIRST}(S) = \{ (, \varepsilon \}$

- $\text{FOLLOW}(S) = \{ \$ \}$
- $\text{FOLLOW}(A) = \{ , \}$
- $\text{FOLLOW}(E) = \{ , \}$
- $\text{FOLLOW}(T) = \{ \&, ) \}$

## Example

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

- $\text{FIRST}(T) = \{ (, a, b, c \}$
- $\text{FIRST}(E) = \{ \&, \varepsilon \}$
- $\text{FIRST}(A) = \{ (, a, b, c \}$
- $\text{FIRST}(S) = \{ (, \varepsilon \}$

- $\text{FOLLOW}(S) = \{ \$ \}$
- $\text{FOLLOW}(A) = \{ ) \}$
- $\text{FOLLOW}(E) =$   
 $\text{FOLLOW}(A) = \{ ) \}$
- $\text{FOLLOW}(T) =$   
 $\text{FIRST}(E) \cup \text{FOLLOW}(A) \cup$   
 $\text{FOLLOW}(E) = \{ \&, ) \}$



# Predictive Parsing

Will never backtrack!

## Requirement:

For every rule:

$$A \rightarrow \alpha_1 \mid \alpha_2 \mid \alpha_3 \mid \dots \mid \alpha_N$$

We must be able to choose the correct alternative  
by looking only at the **next** symbol

May peek ahead to the **next** symbol (token).

## Example

$A \rightarrow aB$   
 $\rightarrow cD$   
 $\rightarrow E$

Assuming  $a, c \notin \text{FIRST}(E)$

## Example

$\text{Stmt} \rightarrow \underline{\text{if}} \text{ Expr} \dots$   
 $\rightarrow \underline{\text{for}} \text{ LValue} \dots$   
 $\rightarrow \underline{\text{while}} \text{ Expr} \dots$   
 $\rightarrow \underline{\text{return}} \text{ Expr} \dots$   
 $\rightarrow \underline{\text{ID}} \dots$

# 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

$E \rightarrow T E'$

$E' \rightarrow + T E' \mid \epsilon$

$T \rightarrow F T'$

$T' \rightarrow * F T' \mid \epsilon$

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

Term ...

+Term +Term + ... +Term

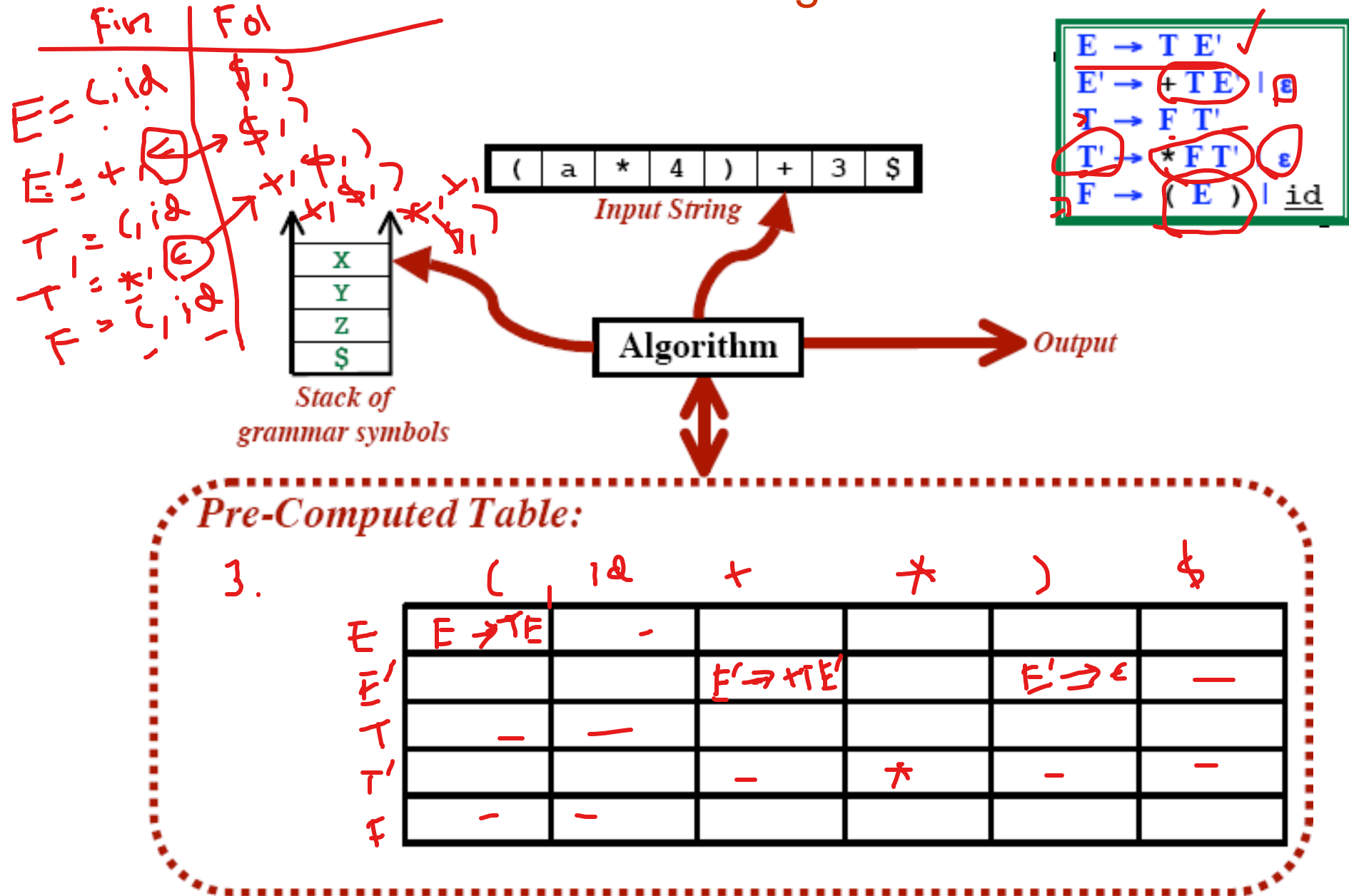
Factor ...

\* Factor \* Factor \* ... \* Factor

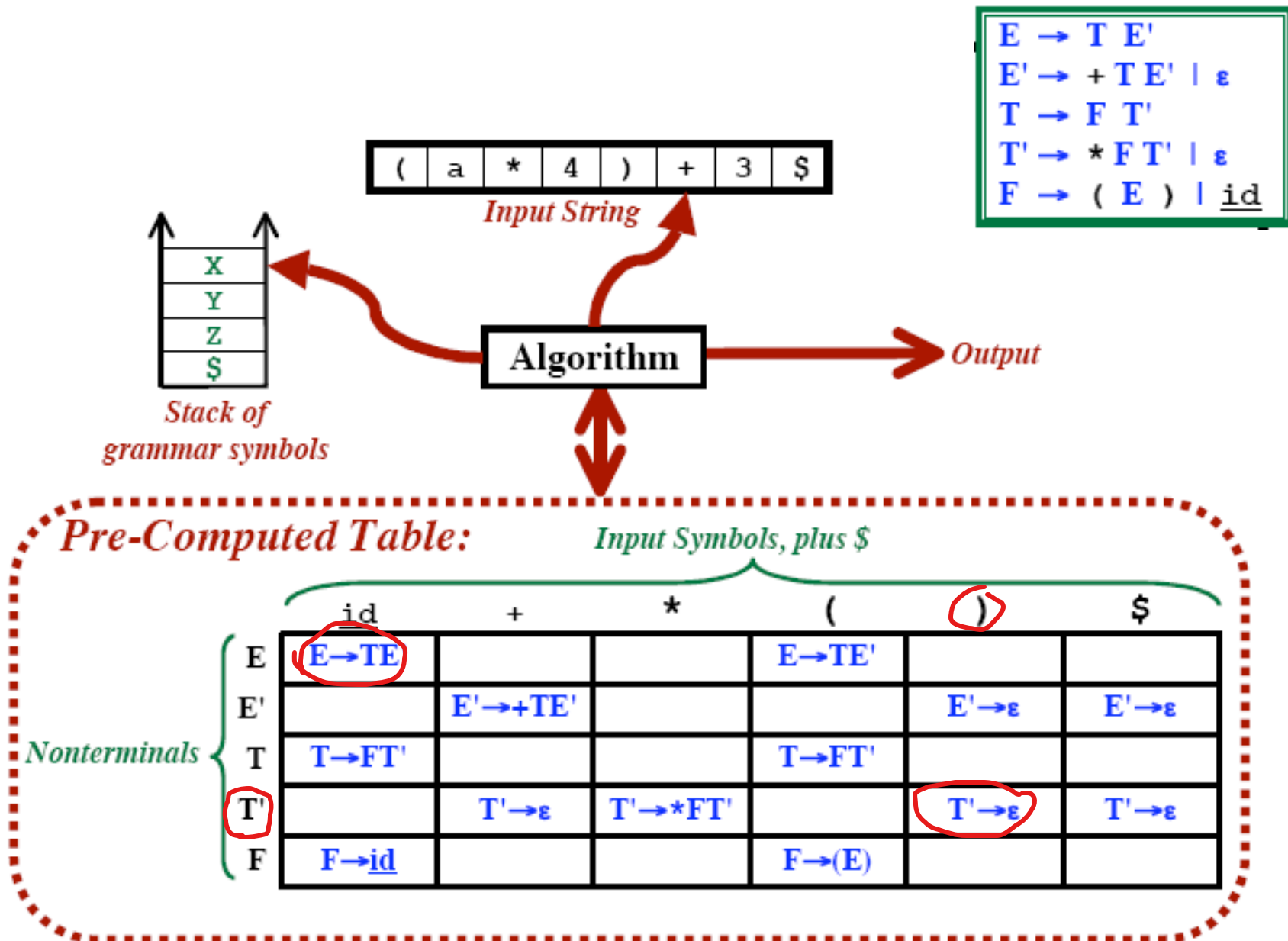
Step 1: From grammar, construct table.

Step 2: Use table to parse strings.

# Table Driven Predictive Parsing

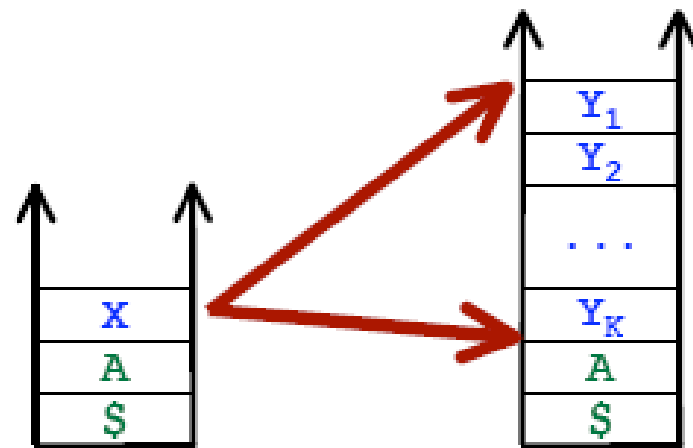


# Table Driven Predictive Parsing



# 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  $X \rightarrow Y_1 Y_2 \dots Y_K$ 
    Pop stack
    Push  $Y_K$ 
    ...
    Push  $Y_2$ 
    Push  $Y_1$ 
    Print (" $X \rightarrow Y_1 Y_2 \dots Y_K$ ")
  else // Table[X,a] is blank
    Syntax Error
  endif
until X == $
```



# Predictive Parsing

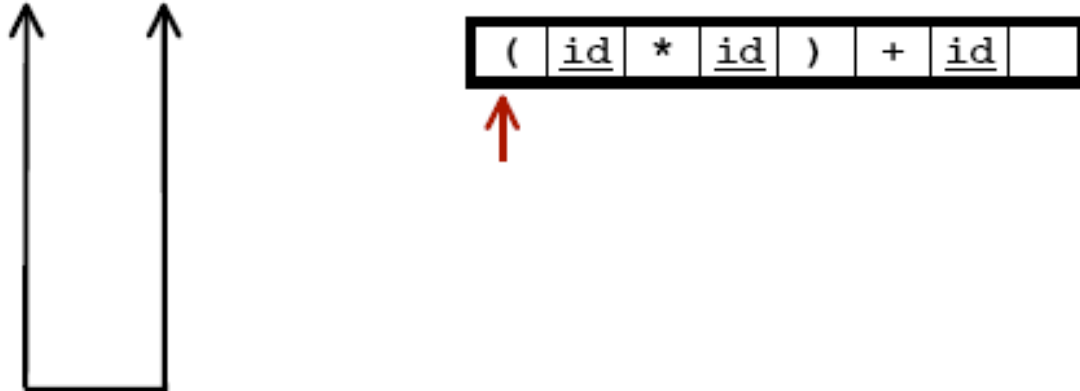
Input:

(id\*id)+id

Output:

**Example**

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



	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

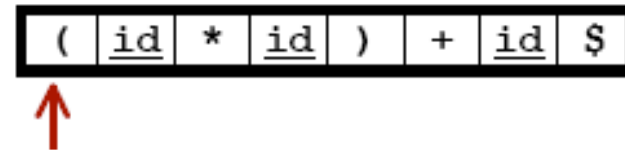
Input:

(id\*id)+id

Output:

Example

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



*Add \$ to end of input*

*Push \$*

*Push E*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		



# Predictive Parsing

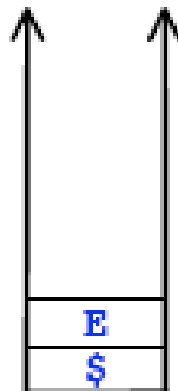
Input:

(id\*id)+id

Output:

Example

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



( id \* id ) + id \$



Look at Table [ E, '(' ]

Use rule  $E \rightarrow TE'$

Pop E

Push E'

Push T

Print  $E \rightarrow TE'$

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

# Predictive Parsing

Input:

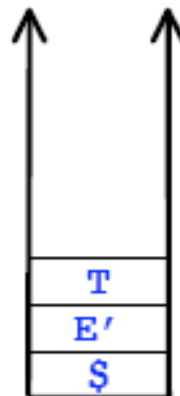
(id\*id)+id

Output:

E → T E'

Example

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



( id \* id ) + id \$



Look at Table [ E, '(' ]

Use rule  $E \rightarrow TE'$

Pop E

Push E'

Push T

Print  $E \rightarrow TE'$

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

# Predictive Parsing

Input:

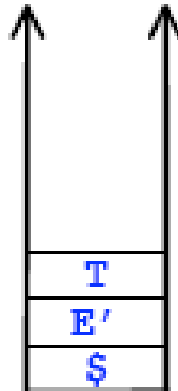
(id\*id)+id

Output:

$E \rightarrow T E'$

Example

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



( id \* id ) + id \$



*Table [ T, '(' ] =  $T \rightarrow FT'$*

*Pop T*

*Push T'*

*Push F*

*Print  $T \rightarrow FT'$*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

# Predictive Parsing

Input:

(id\*id)+id

Output:

E → T E'  
T → F T'

Example

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

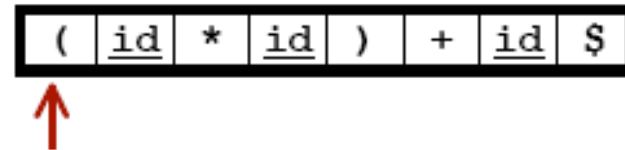
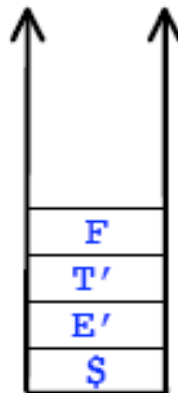


Table [ T, '(' ] = T → FT'

Pop T

Push T'

Push F

Print T → FT'

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

Input:

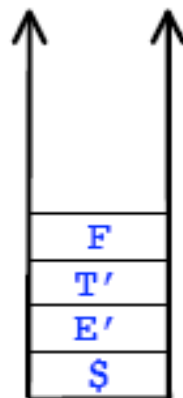
(id\*id)+id

Output:

E → T E'  
T → F T'

Example

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)

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

Input:

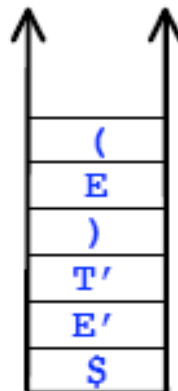
(id\*id)+id

Output:

$E \rightarrow T E'$   
 $T \rightarrow F T'$   
 $F \rightarrow ( E )$

Example

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



( id \* id ) + id \$

$\uparrow$   
 $Table [ F, '(' ] = F \rightarrow (E)$   
*Pop F*  
*Push )*  
*Push E*  
*Push (*  
*Print  $F \rightarrow (E)$*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

# Predictive Parsing

Input:

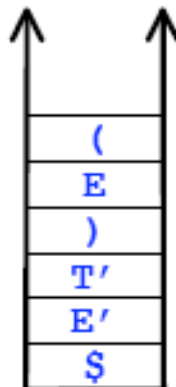
(id\*id)+id

Output:

E → T E'  
T → F T'  
F → ( E )

Example

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*

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

Input:

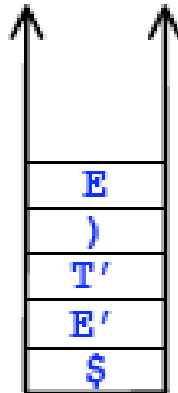
(id\*id)+id

Output:

$E \rightarrow T E'$   
 $T \rightarrow F T'$   
 $F \rightarrow ( E )$

Example

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



( id \* id ) + id \$

$\uparrow$   
 $Table [ E, id ] = E \rightarrow TE'$   
 Pop E  
 Push E'  
 Push T  
 Print  $E \rightarrow TE'$

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		



# Predictive Parsing

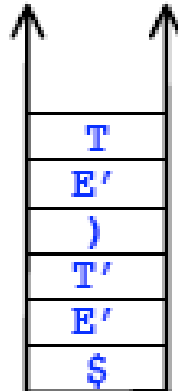
Input:

(id\*id)+id

Output:

$E \rightarrow T E'$   
 $T \rightarrow F T'$   
 $F \rightarrow ( E )$   
 $E \rightarrow T E'$

Example



( id \* id ) + id \$

$\uparrow$   
 $Table [ E, id ] = E \rightarrow TE'$   
 Pop E  
 Push E'  
 Push T  
 Print  $E \rightarrow TE'$

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

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

# Predictive Parsing

Input:

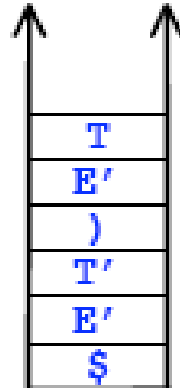
(id\*id)+id

Output:

$E \rightarrow T E'$   
 $T \rightarrow F T'$   
 $F \rightarrow ( E )$   
 $E \rightarrow T E'$

Example

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



( id \* id ) + id \$



*Table [ T, id ] =  $T \rightarrow FT'$*

*Pop T*

*Push T'*

*Push F*

*Print  $T \rightarrow FT'$*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

# Predictive Parsing

Input:

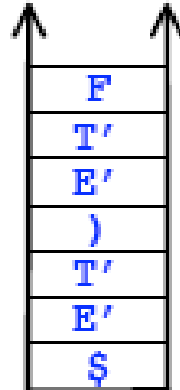
(id\*id)+id

Output:

$E \rightarrow T E'$   
 $T \rightarrow F T'$   
 $F \rightarrow ( E )$   
 $E \rightarrow T E'$   
 $T \rightarrow F T'$

Example

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



( id \* id ) + id \$



*Table [ T, id ] =  $T \rightarrow FT'$*

*Pop T*

*Push T'*

*Push F*

*Print  $T \rightarrow FT'$*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

# Predictive Parsing

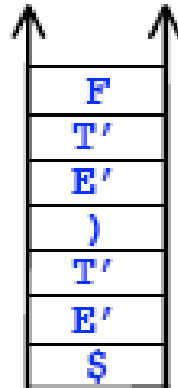
Input:

(id\*id)+id

Output:

$E \rightarrow T E'$   
 $T \rightarrow F T'$   
 $F \rightarrow ( E )$   
 $E \rightarrow T E'$   
 $T \rightarrow F T'$

Example



( id \* id ) + id \$



*Table [ F, id ] = F → id*

*Pop F*

*Push id*

*Print F → id*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

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

# Predictive Parsing

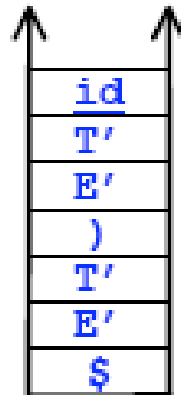
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



Table [ F, id ] = F → id

Pop F

Push id

Print F → id

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

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

# Predictive Parsing

Input:

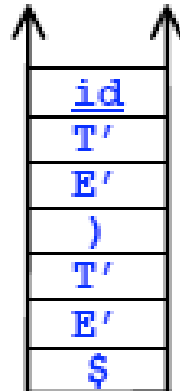
(id\*id)+id

Output:

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

Example

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



( id \* id ) + id \$

↑  
*Top of Stack matches next input*  
*Pop and Scan*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

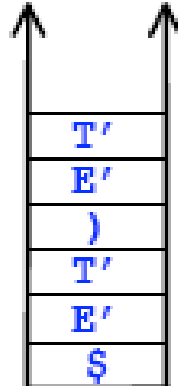
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



*Table* [  $T'$ , '\*' ] =  $T' \rightarrow *FT'$

*Pop*  $T'$

*Push*  $T'$

*Push*  $F$

*Push* '('

*Print*  $T' \rightarrow *FT'$

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

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

# Predictive Parsing

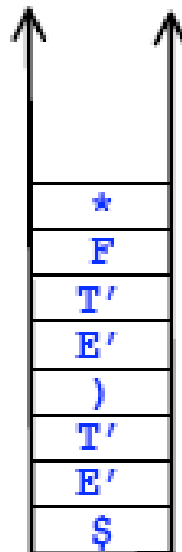
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



Table [  $T'$ , '\*' ] =  $T' \rightarrow *FT'$

Pop  $T'$

Push  $T'$

Push  $F$

Push '\*'

Print  $T' \rightarrow *FT'$

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

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



# Predictive Parsing

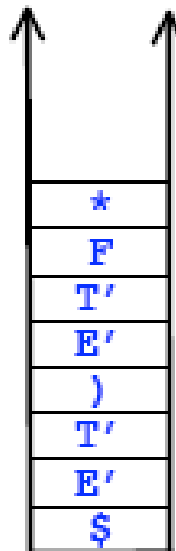
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



*Top of Stack matches next input  
Pop and Scan*

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

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

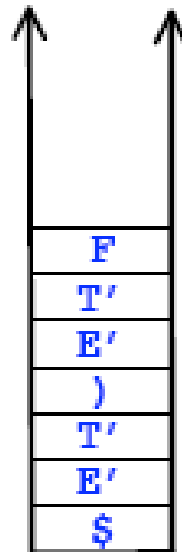
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



*Top of Stack matches next input  
Pop and Scan*

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

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

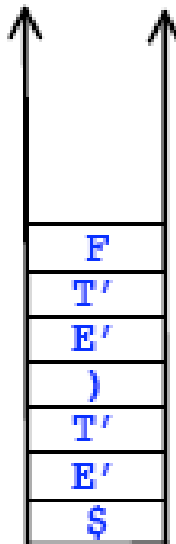
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



*Table* [  $F$ ,  $\underline{id}$  ] =  $F \rightarrow \underline{id}$

*Pop*  $F$

*Push*  $\underline{id}$

*Print*  $F \rightarrow \underline{id}$

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

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

# Predictive Parsing

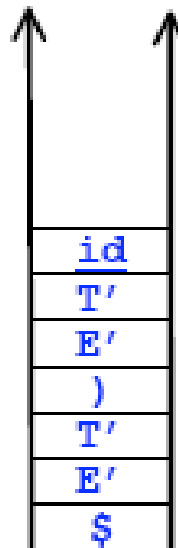
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



*Table [ F, id ] = F → id*

*Pop F*

*Push id*

*Print F → id*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

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

# Predictive Parsing

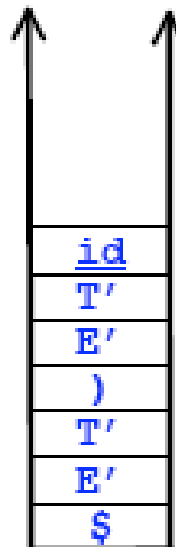
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



*Top of Stack matches next input  
Pop and Scan*

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

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

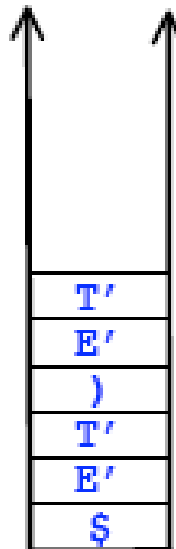
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



*Top of Stack matches next input  
Pop and Scan*

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

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

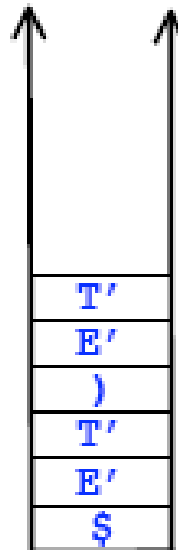
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



*Table [ T', ')' ] =  $T' \rightarrow \epsilon$*

*Pop  $T'$*

*Push <nothing>*

*Print  $T' \rightarrow \epsilon$*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

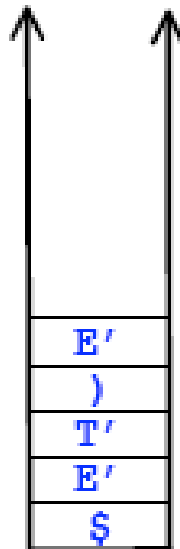
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



*Table [ T', ')' ] =  $T' \rightarrow \epsilon$*

*Pop T'*

*Push <nothing>*

*Print  $T' \rightarrow \epsilon$*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		



# Predictive Parsing

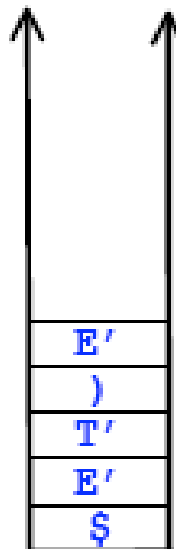
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



*Table [ E', ')' ] =  $E' \rightarrow \epsilon$*

*Pop  $E'$*

*Push <nothing>*

*Print  $E' \rightarrow \epsilon$*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

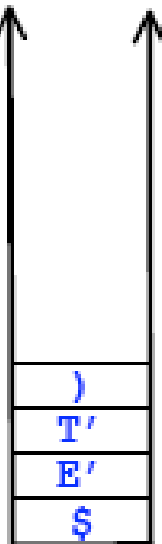
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



*Table [  $E'$ , '(' ] =  $E' \rightarrow \epsilon$*

*Pop  $E'$*

*Push <nothing>*

*Print  $E' \rightarrow \epsilon$*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

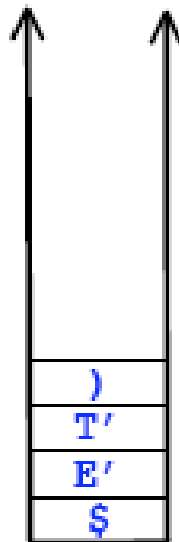
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



*Top of Stack matches next input  
Pop and Scan*

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

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

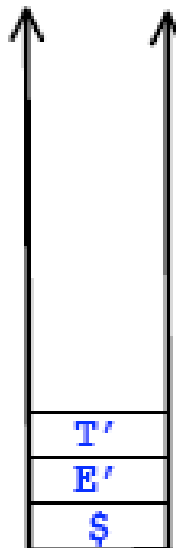
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



Top of Stack matches next input  
Pop and Scan

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

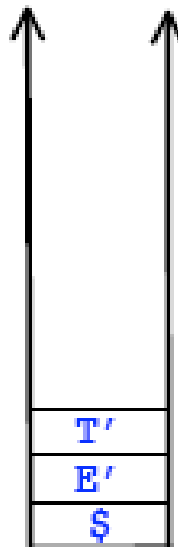
Input:

(id\*id)+id

Output:

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

Example



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

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



*Table [ T', '+' ] =  $T' \rightarrow \epsilon$*

*Pop T'*

*Push <nothing>*

*Print  $T' \rightarrow \epsilon$*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

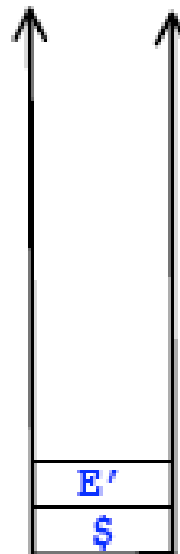
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



*Table [ T', '+' ] =  $T' \rightarrow \epsilon$*

*Pop T'*

*Push <nothing>*

*Print  $T' \rightarrow \epsilon$*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

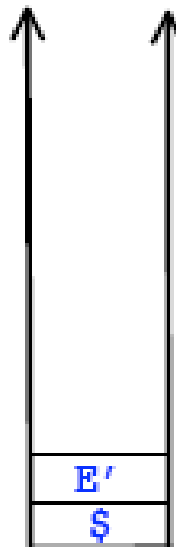
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



*Table* [  $E'$ , '+' ] =  $E' \rightarrow +TE'$

*Pop*  $E'$

*Push*  $E'$

*Push*  $T$

*Push* '+'

*Print*  $E' \rightarrow +TE'$

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

# Predictive Parsing

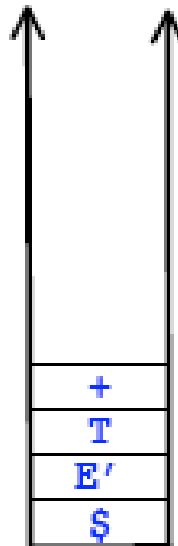
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



*Table* [  $E'$ , '+' ] =  $E' \rightarrow +TE'$

*Pop*  $E'$

*Push*  $E'$

*Push*  $T$

*Push* '+'

*Print*  $E' \rightarrow +TE'$

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow (E)$		

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



# Predictive Parsing

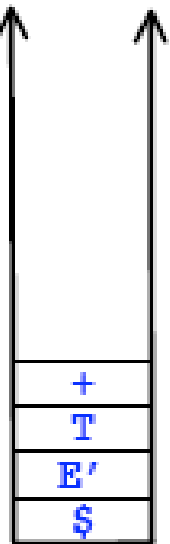
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



*Top of Stack matches next input  
Pop and Scan*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

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

# Predictive Parsing

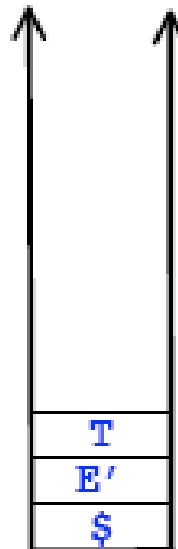
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



*Top of Stack matches next input  
Pop and Scan*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

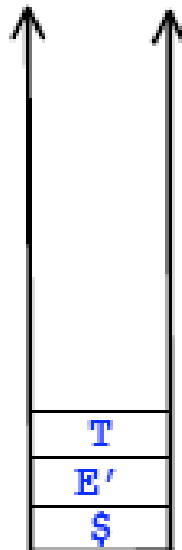
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



*Table [ T, id ] =  $T \rightarrow FT'$*

*Pop T*

*Push T'*

*Push F*

*Print  $T \rightarrow FT'$*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

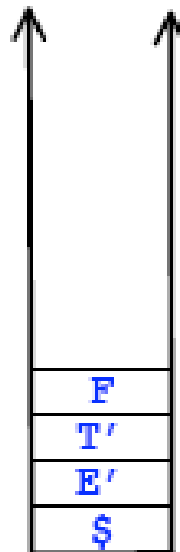
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



*Table [ T, id ] = T → FT'*

*Pop T*

*Push T'*

*Push F*

*Print T → FT'*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

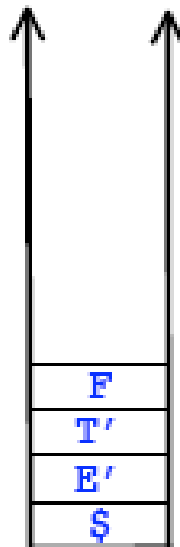
Input:

(id\*id)+id

Output:

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

Example



( id \* id ) + id \$



*Table* [  $F, \underline{id}$  ] =  $F \rightarrow \underline{id}$

*Pop*  $F$

*Push*  $\underline{id}$

*Print*  $F \rightarrow \underline{id}$

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

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

# Predictive Parsing

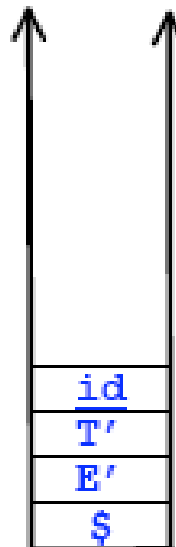
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



*Table [ F, id ] = F → id*

*Pop F*

*Push id*

*Print F → id*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

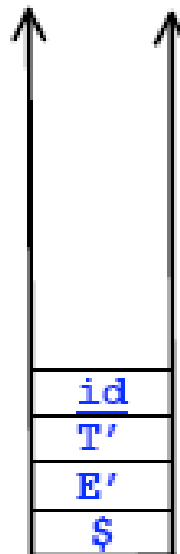
Input:

(id\*id)+id

Output:

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

Example



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

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



*Top of Stack matches next input  
Pop and Scan*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

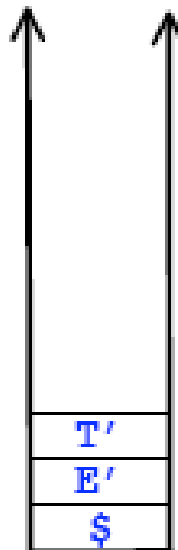
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$

Top of Stack matches next input  
Pop and Scan

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		



# Predictive Parsing

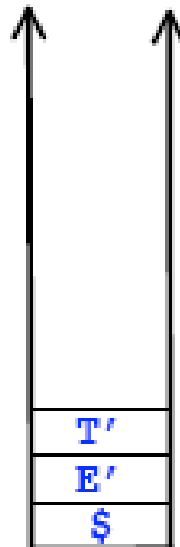
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



*Table [ T', \$ ] = T' → ε*

*Pop T'*

*Push <nothing>*

*Print T' → ε*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

# Predictive Parsing

Input:

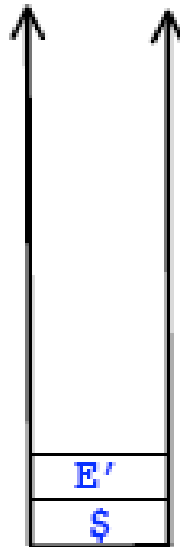
(id\*id)+id

Output:

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

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

Example



( id \* id ) + id \$



$Table [ T', \$ ] = T' \rightarrow \epsilon$

Pop  $T'$

Push <nothing>

Print  $T' \rightarrow \epsilon$

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

# Predictive Parsing

Input:

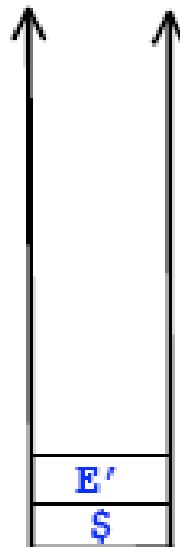
(id\*id)+id

Output:

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

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

Example



( id \* id ) + id \$



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

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

# Predictive Parsing

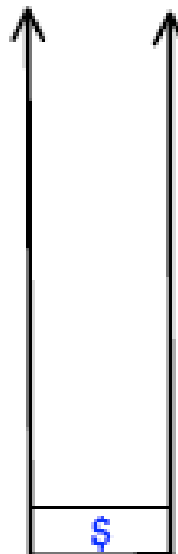
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$

*Table [ E', \$ ] =  $E' \rightarrow \epsilon$*

*Pop E'*

*Push <nothing>*

*Print  $E' \rightarrow \epsilon$*

		<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$				$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$				$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$				$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$			$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$				$F \rightarrow ( E )$		

# Predictive Parsing

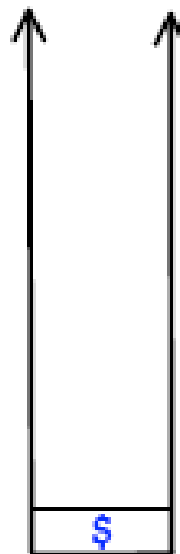
Input:

(id\*id)+id

Output:

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

Example



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

( id \* id ) + id \$



*Input symbol == \$*

*Top of stack == \$*

*Loop terminates with success*

	<u>id</u>	+	*	(	)	\$
E	$E \rightarrow T E'$			$E \rightarrow T E'$		
E'		$E' \rightarrow + T E'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow F T'$			$T \rightarrow F T'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow * F T'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \underline{id}$			$F \rightarrow ( E )$		

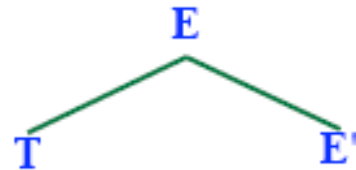
# Reconstructing the Parse Tree

Input:

(id\*id)+id

Output:

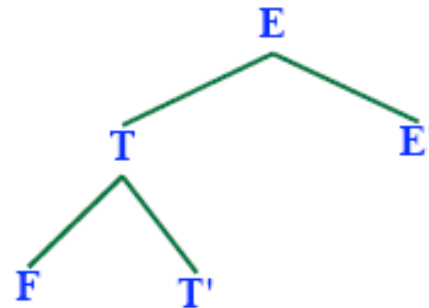
$E \rightarrow T E'$



Output:

$E \rightarrow T E'$

$T \rightarrow F T'$

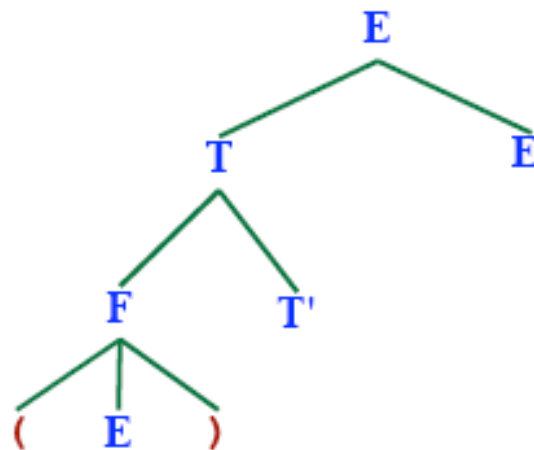


Output:

$E \rightarrow T E'$

$T \rightarrow F T'$

$F \rightarrow ( E )$



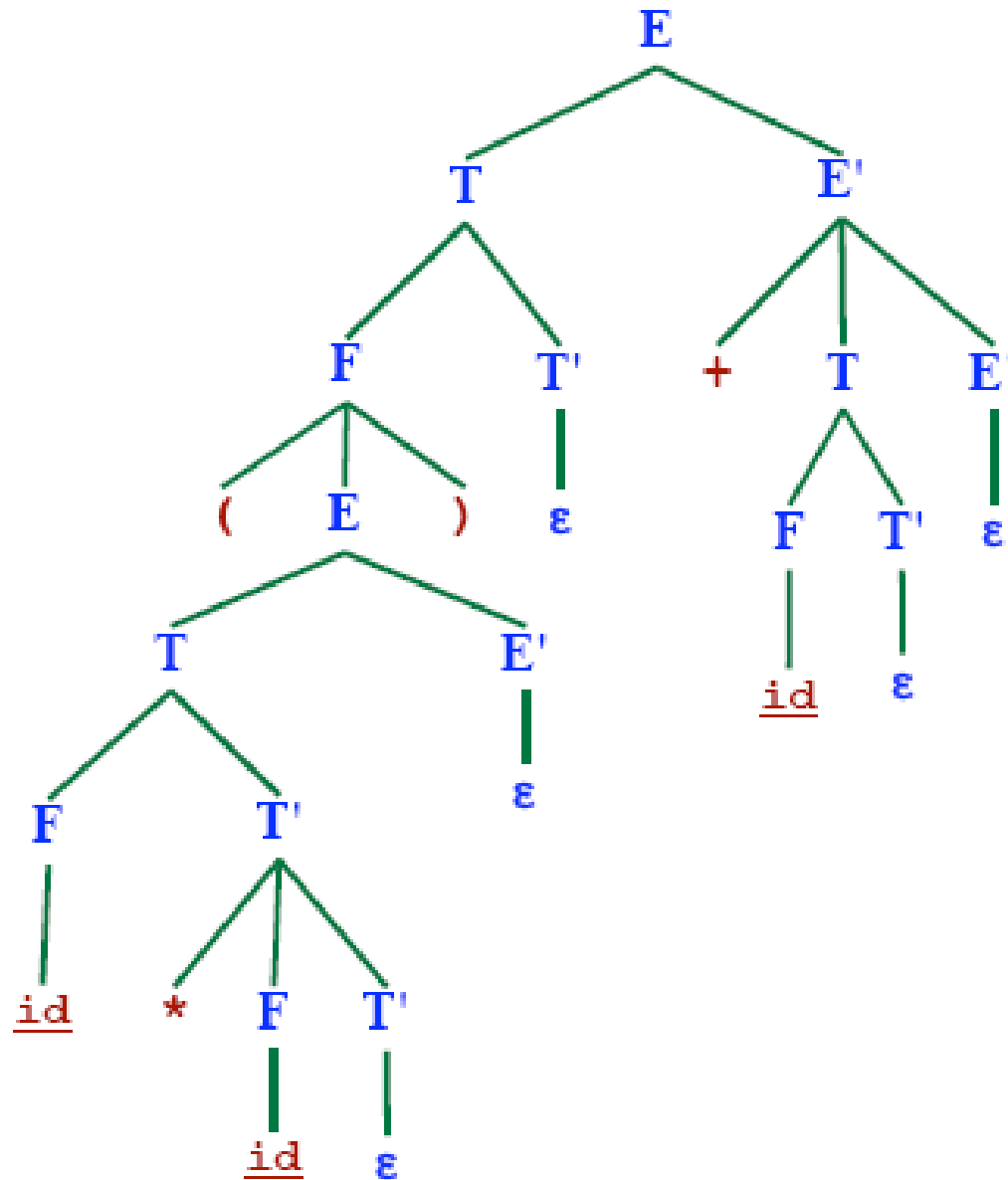
# Reconstructing the Parse Tree

Input:

(id\*id)+id

Output:

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



# Reconstructing the Parse Tree

Input:

(id\*id)+id

Output:

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

Leftmost Derivation:

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



## Transition Diagram for Predictive Parsers

- Useful for visualizing predictive parsers.
- To construct Transition Diagram from a grammar
  - Eliminate left recursion
  - Left factor the grammar
  - Then for each nonterminal  $A$ 
    - Create an initial and final state
    - For each production  $A \rightarrow X_1 X_2 \dots X_k$ , create a path from the initial to the final state, with edges labeled  $X_1, X_2, \dots, X_k$ . If  $A \rightarrow \epsilon$ , the path is an edge labeled  $\epsilon$ .

# Transition Diagram for Predictive Parsers

- Predictive parser begins in the start state for the start symbol
- Suppose at any time it is in state **s** with an edge
  - labeled by a terminal **a** to state **t**



- If the next input is **a** the parser advances in input and moves to state **t**
- If the edge from **s** to **t** is labeled by  $\epsilon$ , then the parser moves immediately to state **t** without advancing the input
- labeled by a nonterminal **A**



- Parser goes to the start state for **A**
- If it ever reaches the final state of **A** it will immediately go back to state **t**

## Transition Diagram for Predictive Parsers

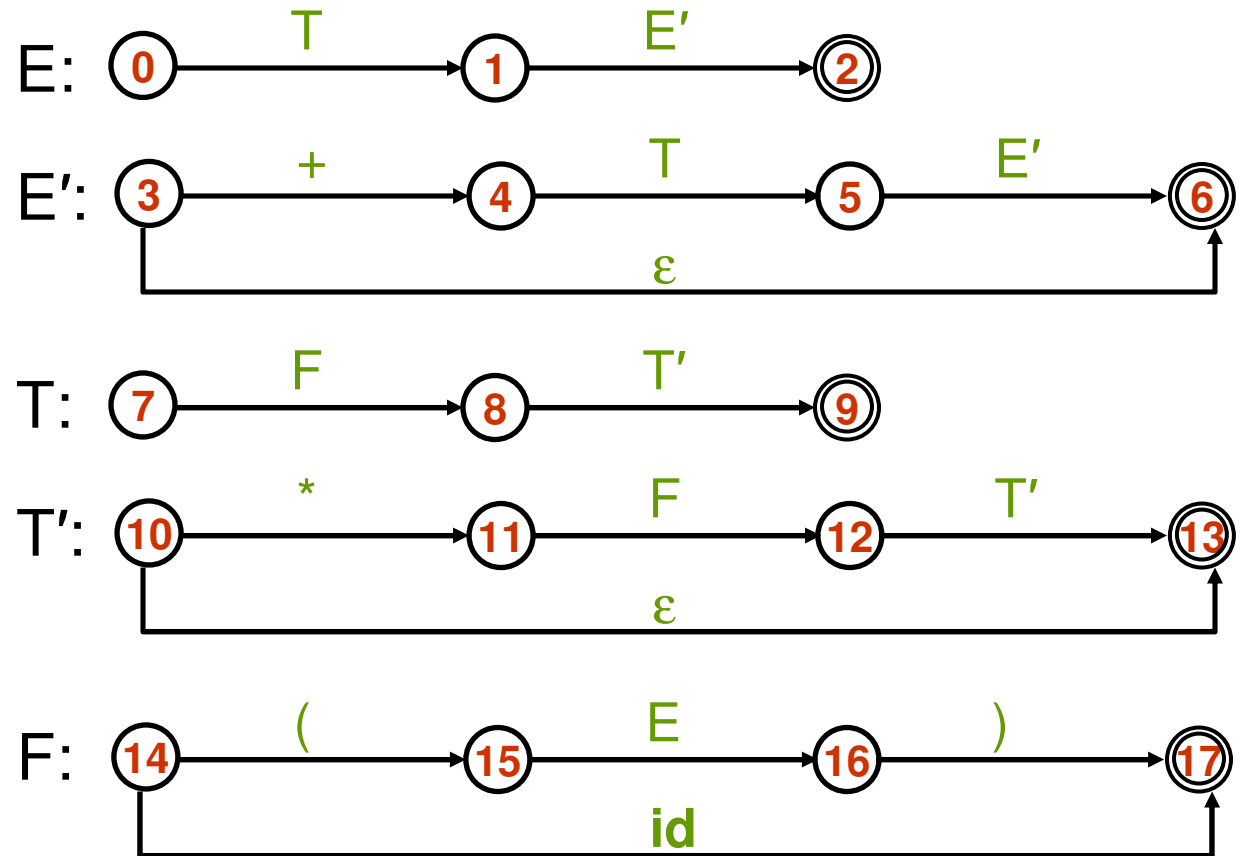
$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \varepsilon$

$T \rightarrow FT'$

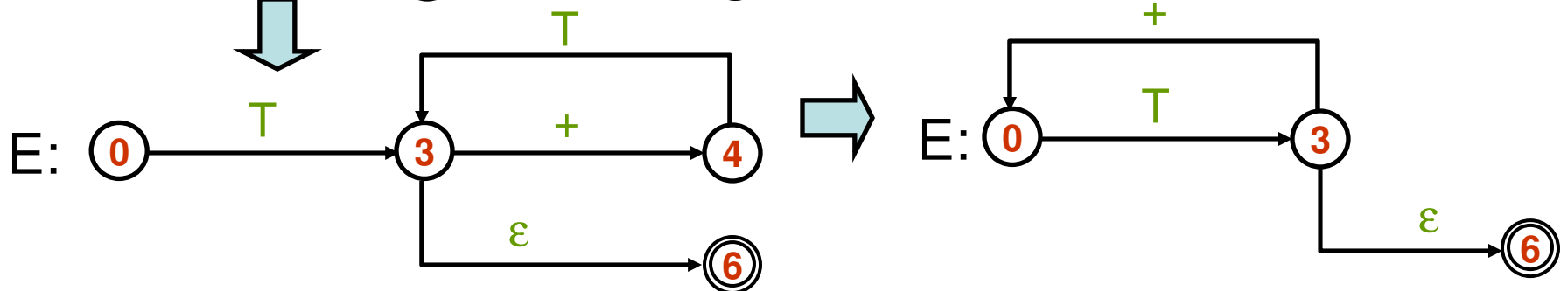
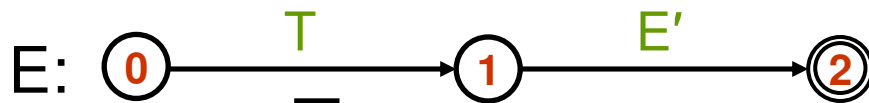
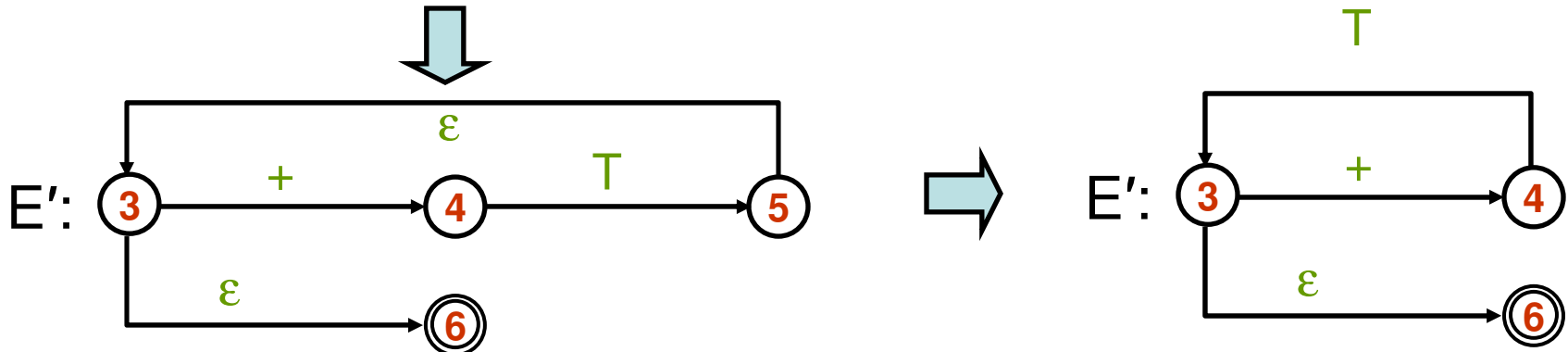
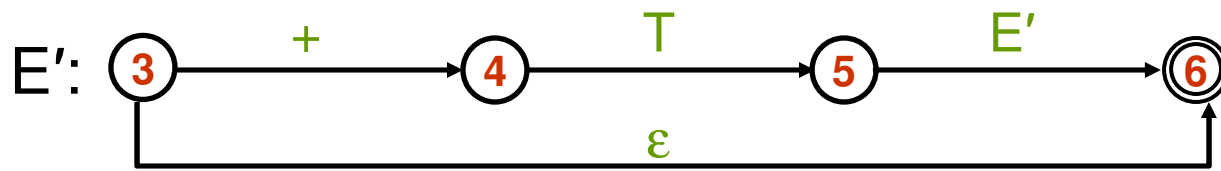
$T' \rightarrow *FT' \mid \varepsilon$

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



# Simplification of Transition Diagrams

- TDs can be simplified by substituting one in another



## Simplification of Transition Diagrams

- Complete set of TDs
- A C implementation of this simplified version of the parser runs 20-25% faster than the original version

