

# LR Parsing

CMPT 379: Compilers

Instructor: Anoop Sarkar

[anoopsarkar.github.io/compilers-class](https://anoopsarkar.github.io/compilers-class)

# LR(0) conflicts:

$S' \rightarrow T$

$T \rightarrow F$

$T \rightarrow T * F$

$T \rightarrow id$

$F \rightarrow id \mid ( T )$

$F \rightarrow id = T ;$

11:  $F \rightarrow id \bullet$

$F \rightarrow id \bullet = T$

Shift/reduce conflict

1:  $F \rightarrow id \bullet$

$T \rightarrow id \bullet$

Reduce/Reduce conflict

Need more lookahead: SLR(1)

# FIRST and FOLLOW

$a \in \text{FIRST}(\alpha)$  if  $\alpha \Rightarrow^* a\beta$

if  $\alpha \Rightarrow^* \epsilon$  then  $\epsilon \in \text{FIRST}(\alpha)$

$a \in \text{FOLLOW}(A)$  if  $S \Rightarrow^* \alpha A a \beta$

$a \in \text{FOLLOW}(A)$  if  $S \Rightarrow^* \alpha A \gamma a \beta$

and  $\gamma \Rightarrow^* \epsilon$

# Example First/Follow

$$S \rightarrow AB$$

$$A \rightarrow c \mid \varepsilon$$

$$B \rightarrow cbB \mid ca$$

$$\text{First}(A) = \{c, \varepsilon\}$$

$$\text{Follow}(A) = \{c\}$$

$$\text{First}(B) = \{c\}$$

$$\text{Follow}(A) \cap$$

$$\text{First}(cbB) =$$

$$\text{First}(c) = \{c\}$$

$$\text{First}(ca) = \{c\}$$

$$\text{Follow}(B) = \{\$ \}$$

$$\text{First}(S) = \{c\}$$

$$\text{Follow}(S) = \{\$ \}$$

# Example First/Follow

$$S \rightarrow cAa$$

$$A \rightarrow cB \mid B$$

$$B \rightarrow bcB \mid \varepsilon$$

$$\text{First}(A) = \{b, c, \varepsilon\}$$

$$\text{Follow}(A) = \{a\}$$

$$\text{First}(B) = \{b, \varepsilon\}$$

$$\text{Follow}(B) = \{a\}$$

$$\text{First}(S) = \{c\}$$

$$\text{Follow}(S) = \{\$ \}$$

# SLR(1) : Simple LR(1) Parsing

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

What can the next symbol be when we reduce  $F \rightarrow id$  ?

$$S' \$ \Rightarrow T \$ \Rightarrow F \$ \Rightarrow id \underline{\$} \quad S' \$ \Rightarrow T \$ \Rightarrow T * F \$ \Rightarrow T * id \$ \Rightarrow F * id \$ \Rightarrow id \underline{*} id \$$$
$$S' \$ \Rightarrow T \$ \Rightarrow C(T) \$ \Rightarrow C(F) \$ \Rightarrow C(id) \underline{\$}$$

The top of stack will be  $id$  and the next input symbol will be either  $\$$ , or  $*$  or  $)$

$\text{Follow}(F) = \{ *, ), \$ \}$

# SLR(1) : Simple LR(1) Parsing

$$S' \rightarrow T$$
$$T \rightarrow F \mid T * F \mid C ( T )$$
$$F \rightarrow id \mid id ++ \mid ( T )$$
$$C \rightarrow id$$

What can the next symbol be when we reduce  $C \rightarrow id$  ?

$$S' \$ \Rightarrow T \$ \Rightarrow C(T) \$ \Rightarrow C(F) \$ \Rightarrow C(id) \Rightarrow id(id) \$$$
$$\text{Follow}(C) = \{ ( \}$$

# SLR(1) : Simple LR(1) Parsing

0:  $S' \rightarrow \bullet T$   
 $T \rightarrow \bullet F$   
 $T \rightarrow \bullet T * F$   
 $T \rightarrow \bullet C (T)$   
 $F \rightarrow \bullet id$   
 $F \rightarrow \bullet id ++$   
 $F \rightarrow \bullet ( T )$   
 $C \rightarrow \bullet id$

id

$S' \rightarrow T$   
 $T \rightarrow F \mid T * F \mid C ( T )$   
 $F \rightarrow id \mid id ++ \mid ( T )$   
 $C \rightarrow id$

1:  $F \rightarrow id \bullet$   
 $F \rightarrow id \bullet ++$   
 $C \rightarrow id \bullet$

$\text{Follow}(F) = \{ *, ), \$ \}$

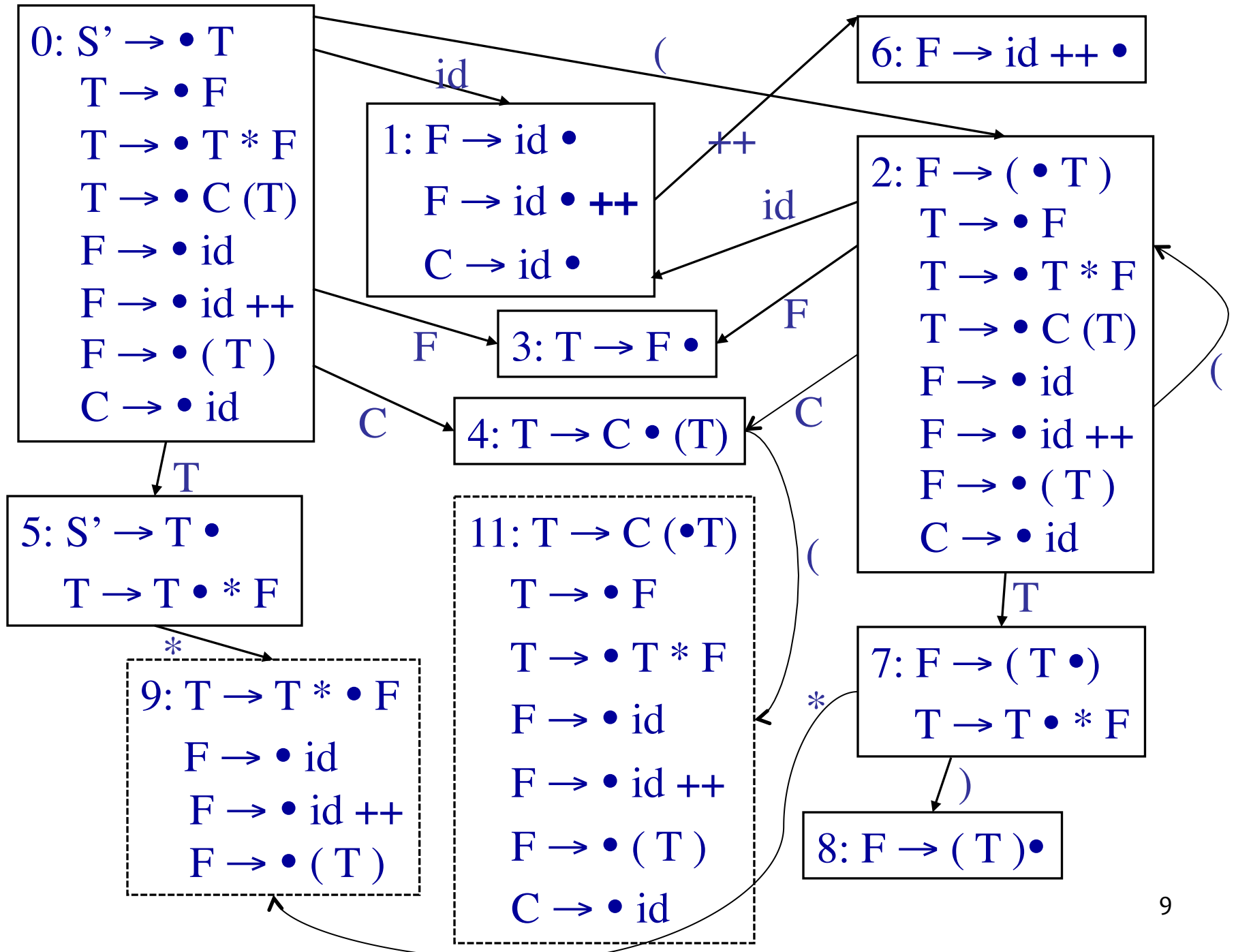
$\text{Follow}(C) = \{ ( \}$

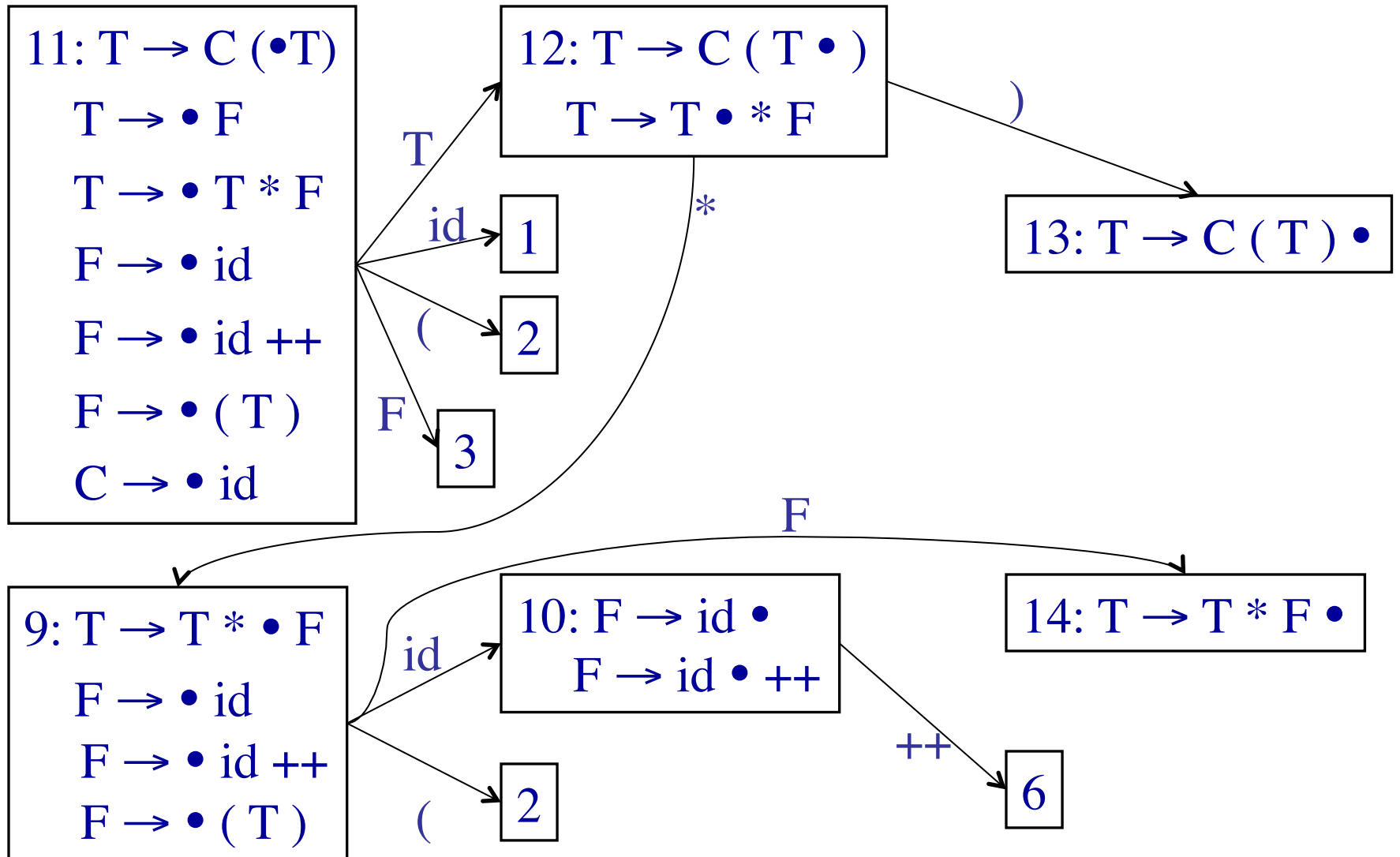
$\text{action}[1,*] = \text{action}[1,)] = \text{action}[1,\$] = \text{Reduce } F \rightarrow id$

$\text{action}[1,(] = \text{Reduce } C \rightarrow id$

$\text{action}[1,++] = \text{Shift}$







Productions	
1	$T \rightarrow F$
2	$T \rightarrow T * F$
3	$T \rightarrow C(T)$
4	$F \rightarrow id$
5	$F \rightarrow id ++$
6	$F \rightarrow (T)$
7	$C \rightarrow id$

	*	(	)	id	++	\$	T	F	C
0		S2		S1			5	3	4
1	R4	R7	R4		S2	R4			
2		S2		S1			7	3	4
3	R1		R1			R1			
4		S11							
5	S9					A			
6	R5		R5			R5			
7	S9		S8						
8	R6		R6			R6			
9		S2		S10				14	
10	R4		R4		S6	R4			
11		S2		S1			12	3	
12	S9		S13						
13	R3		R3			R3			
14	R2		R2			R2			

If there is still conflicts under  
These rules, grammar is not SLR

# SLR Parsing

- Assume:
  - Stack contains  $\alpha$  and next input is  $t$
  - DFA on input  $\alpha$  terminates in state  $s$
- Reduce by  $X \rightarrow \beta$  if
  - $s$  contains item  $X \rightarrow \beta \cdot$
  - $t \in \text{Follow}(X)$
- Shift if
  - $s$  contains item  $X \rightarrow \beta \cdot t$
  - If  $X \rightarrow \beta \cdot$  is in  $s$  then  $t$  cannot be in  $\text{Follow}(X)$

$S' \rightarrow E$

$E \rightarrow T + E$

$E \rightarrow T$

$T \rightarrow id$

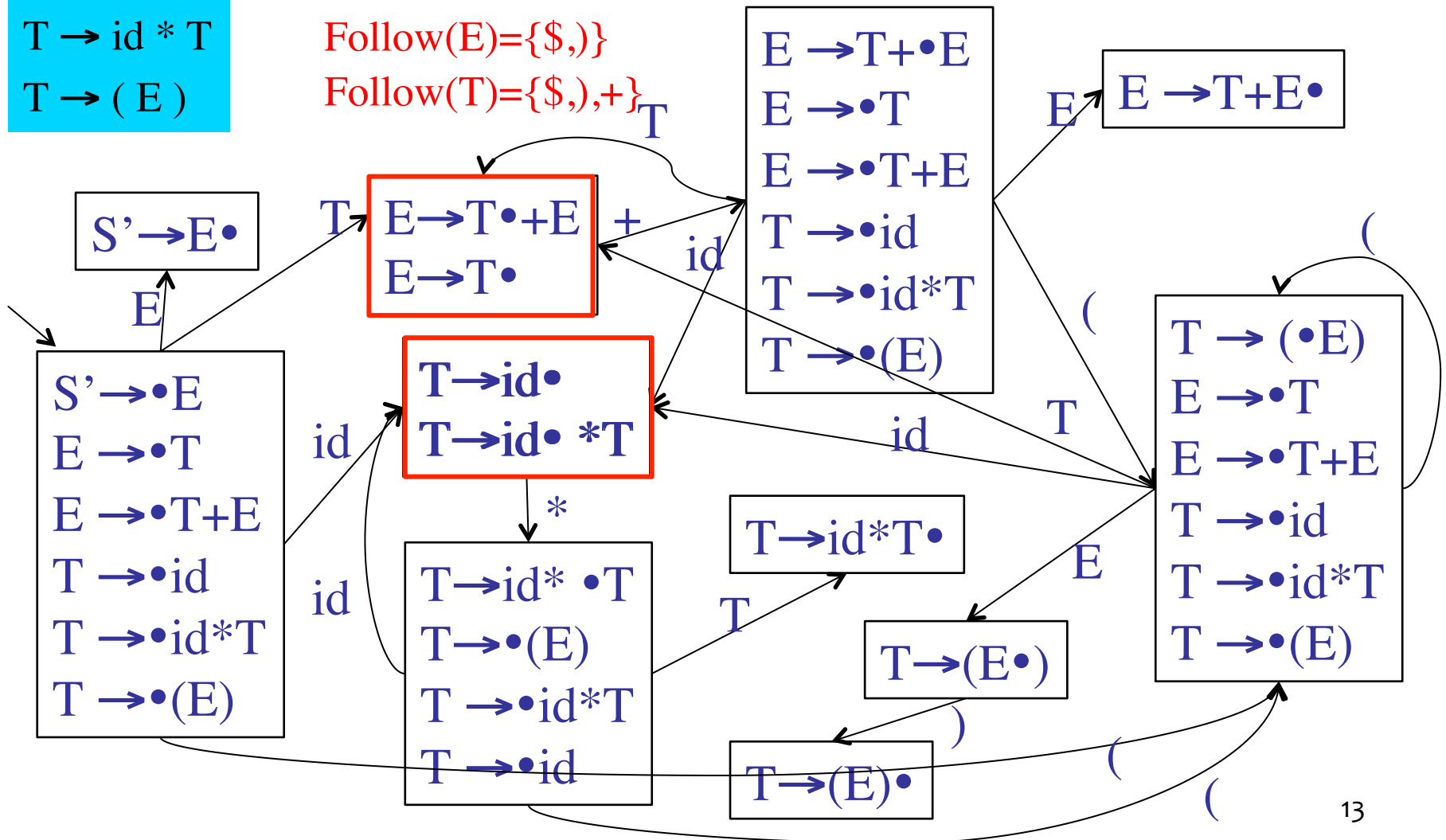
$T \rightarrow id * T$

$T \rightarrow (E)$

# SLR Parsing

$Follow(E) = \{\$, \,)\}$

$Follow(T) = \{\$, \,)\, , \, +\}$



# SLR Parsing

- Let  $M$  be DFA for viable prefixes of  $G$
- Let  $|x_1 \dots x_n \$$  be initial configuration
- Repeat until configuration is  $S| \$$ 
  - Let  $\alpha| \omega$  be current configuration
  - Run  $M$  on current stack  $\alpha$
  - If  $M$  rejects  $\alpha$ , report parsing error
    - Stack  $\alpha$  is not a viable prefix
  - If  $M$  accepts  $\alpha$  with items  $I$ , let  $a$  be the next input
    - Shift  $[X \rightarrow \beta \cdot a \gamma] \in I$
    - Reduce if  $[X \rightarrow \beta \cdot] \in I$  and  $a \in \text{Follow}(X)$
    - Report parsing error if neither applies

**If there is any conflict in the last step (more than two valid action), grammar is not SLR(k) in practice  $k=1$**

# Trace 'id\*id'

configuration (Stackinput)	DFA halt state	Action
id * id \$		

$S' \rightarrow E$

$E \rightarrow T + E$

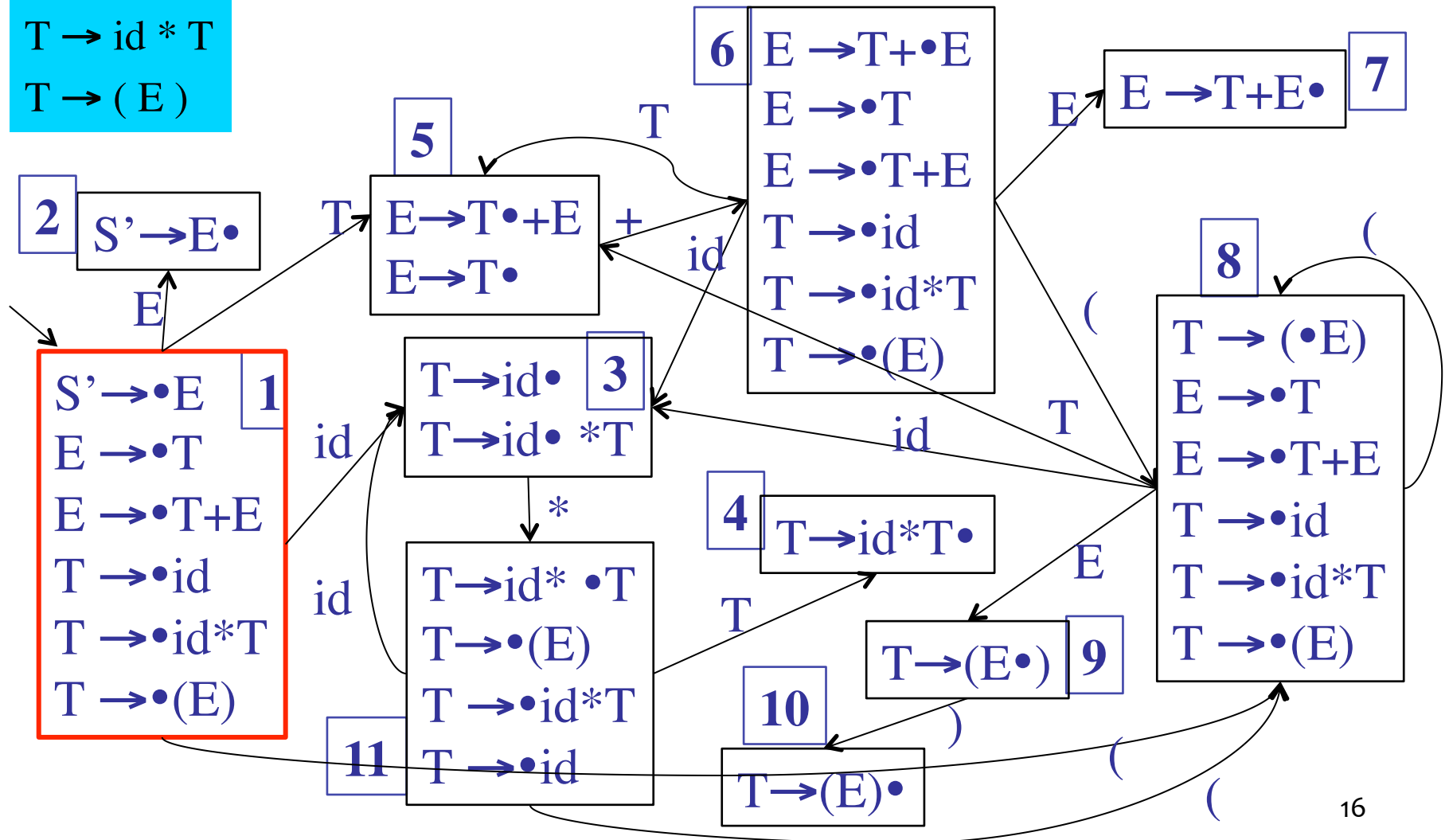
$E \rightarrow T$

$T \rightarrow id$

$T \rightarrow id * T$

$T \rightarrow (E)$

| id \* id \$





# Trace 'id\*id'

configuration (Stackinput)	DFA halt state	Action
id * id \$ id   * id \$	1	Shift

$S' \rightarrow E$

$E \rightarrow T + E$

$E \rightarrow T$

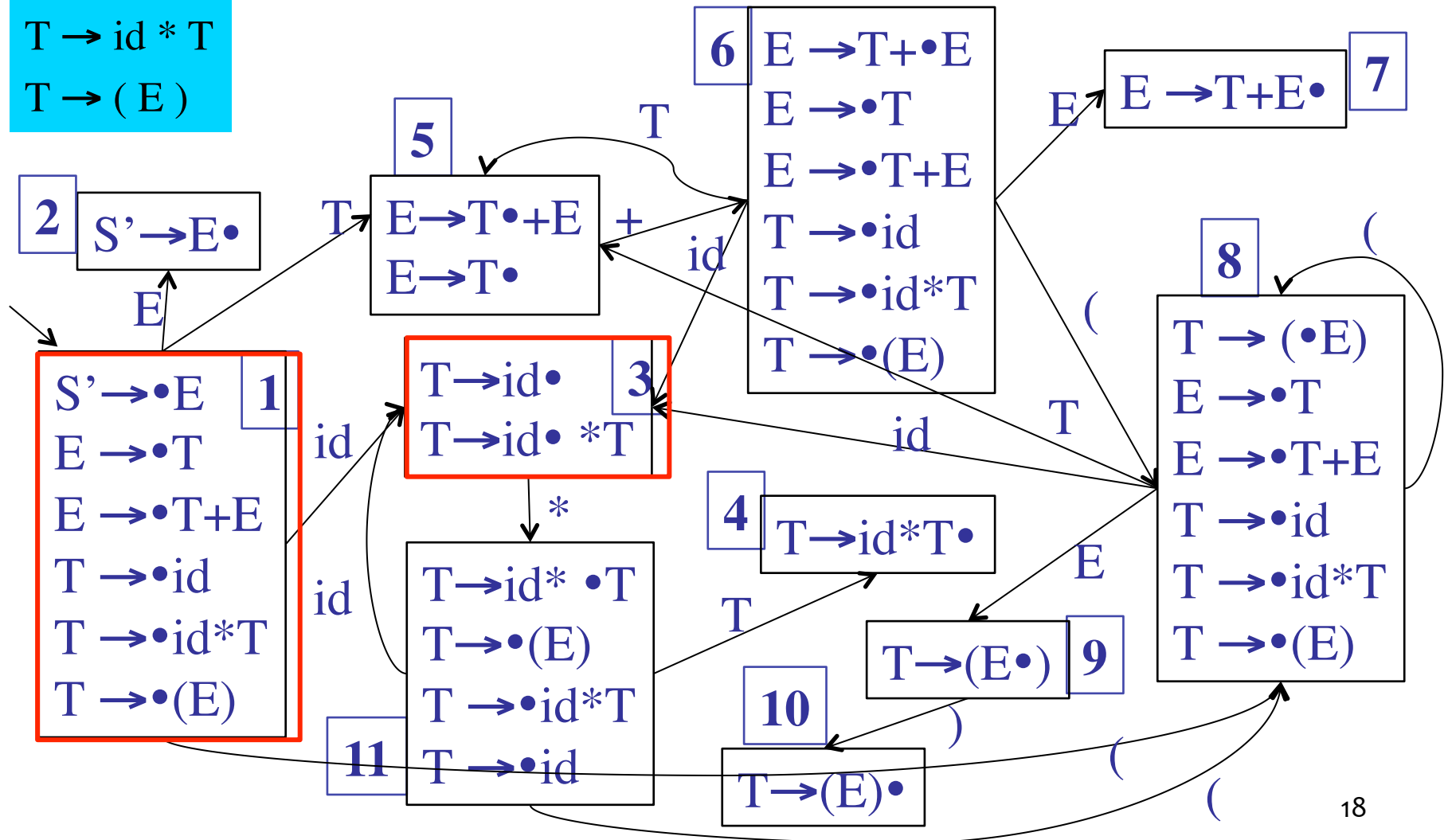
$T \rightarrow id$

$T \rightarrow id * T$

$T \rightarrow (E)$

$Follow(T) = \{\$, ), +\}$

$id \mid * id \$$



# Trace 'id\*id'

configuration (Stackinput)	DFA halt state	Action
id * id \$	1	Shift
id   * id \$	3    * $\notin \text{Follow}(T)$	Shift
id *   id \$		

$S' \rightarrow E$

$E \rightarrow T + E$

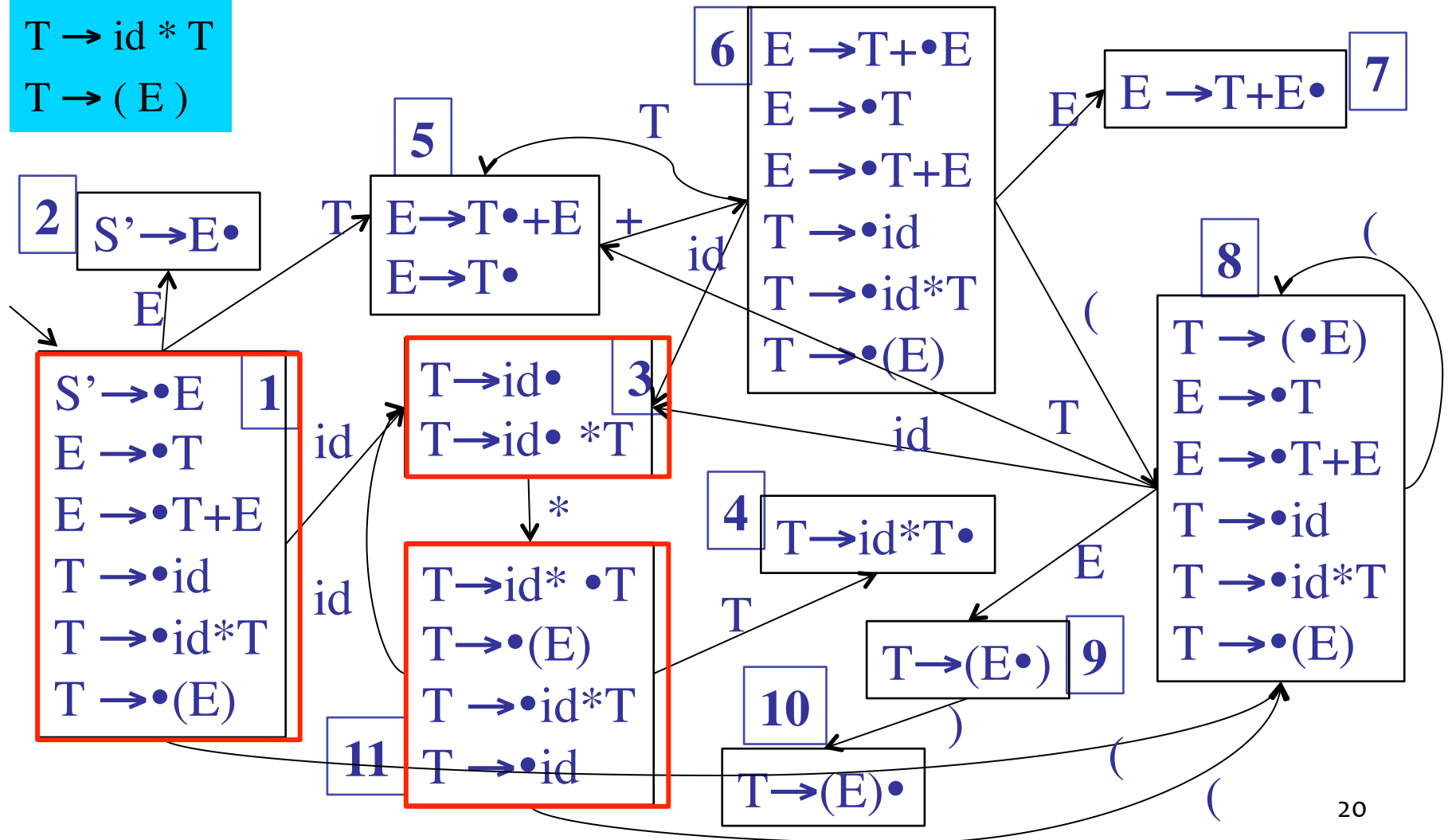
$E \rightarrow T$

$T \rightarrow id$

$T \rightarrow id * T$

$T \rightarrow (E)$

$id * id \$$



# Trace 'id\*id'

configuration (Stack input)	DFA halt state	Action
id * id \$	1	Shift
id   * id \$	3    * $\notin \text{Follow}(T)$	Shift
id *   id \$	11	Shift
id * id   \$		

$S' \rightarrow E$

$E \rightarrow T + E$

$E \rightarrow T$

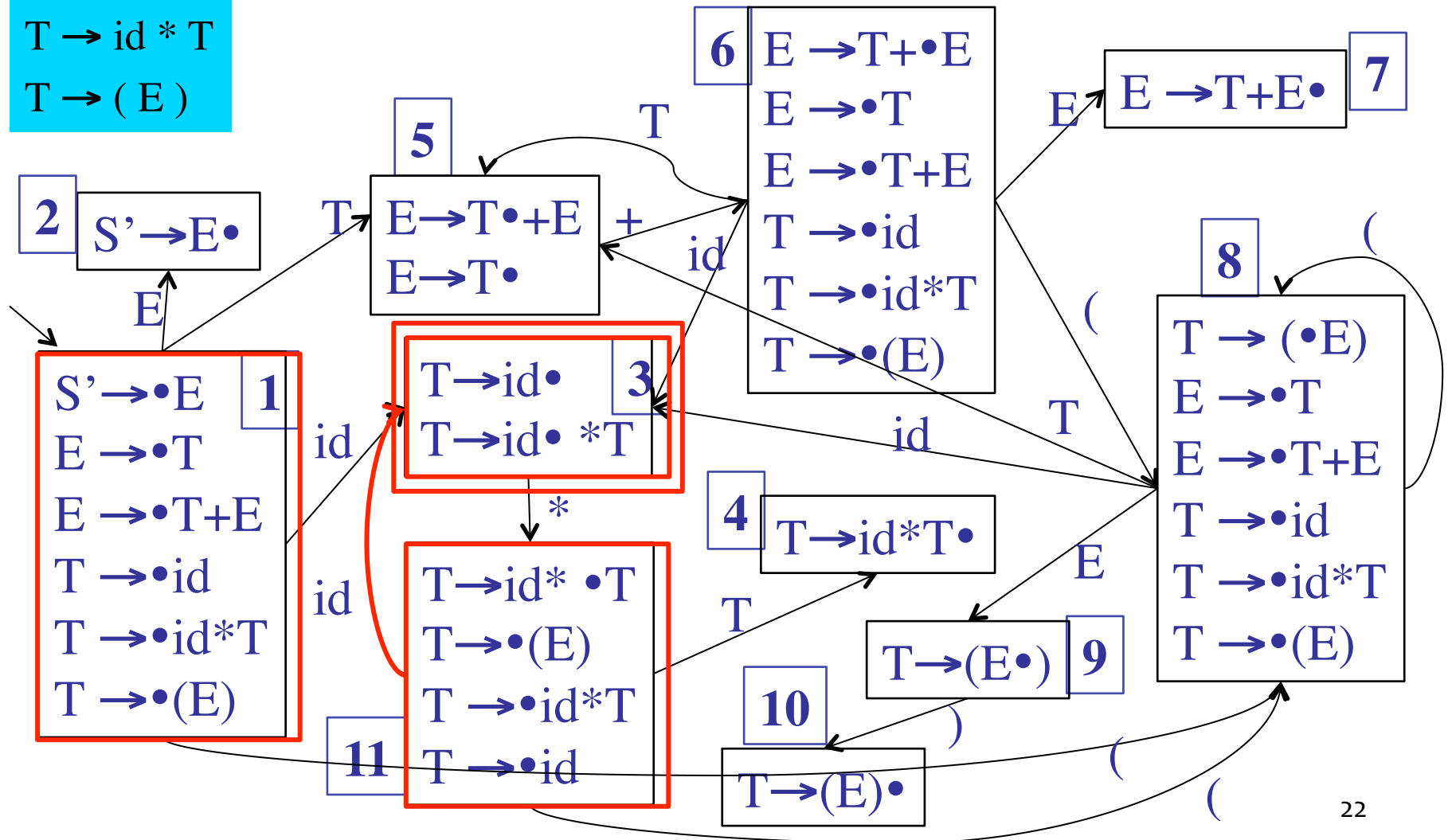
$T \rightarrow id$

$T \rightarrow id * T$

$T \rightarrow (E)$

$Follow(T) = \{\$, ), +\}$

$id * id \mid \$$



# Trace 'id\*id'

configuration (Stack input)	DFA halt state	Action
id * id \$	1	Shift
id   * id \$	3    * $\notin \text{Follow}(T)$	Shift
id *   id \$	11	Shift
id * id   \$	3    \$ $\in \text{Follow}(T)$	Reduce $T \rightarrow id$
id * T   \$		

$S' \rightarrow E$

$E \rightarrow T + E$

$E \rightarrow T$

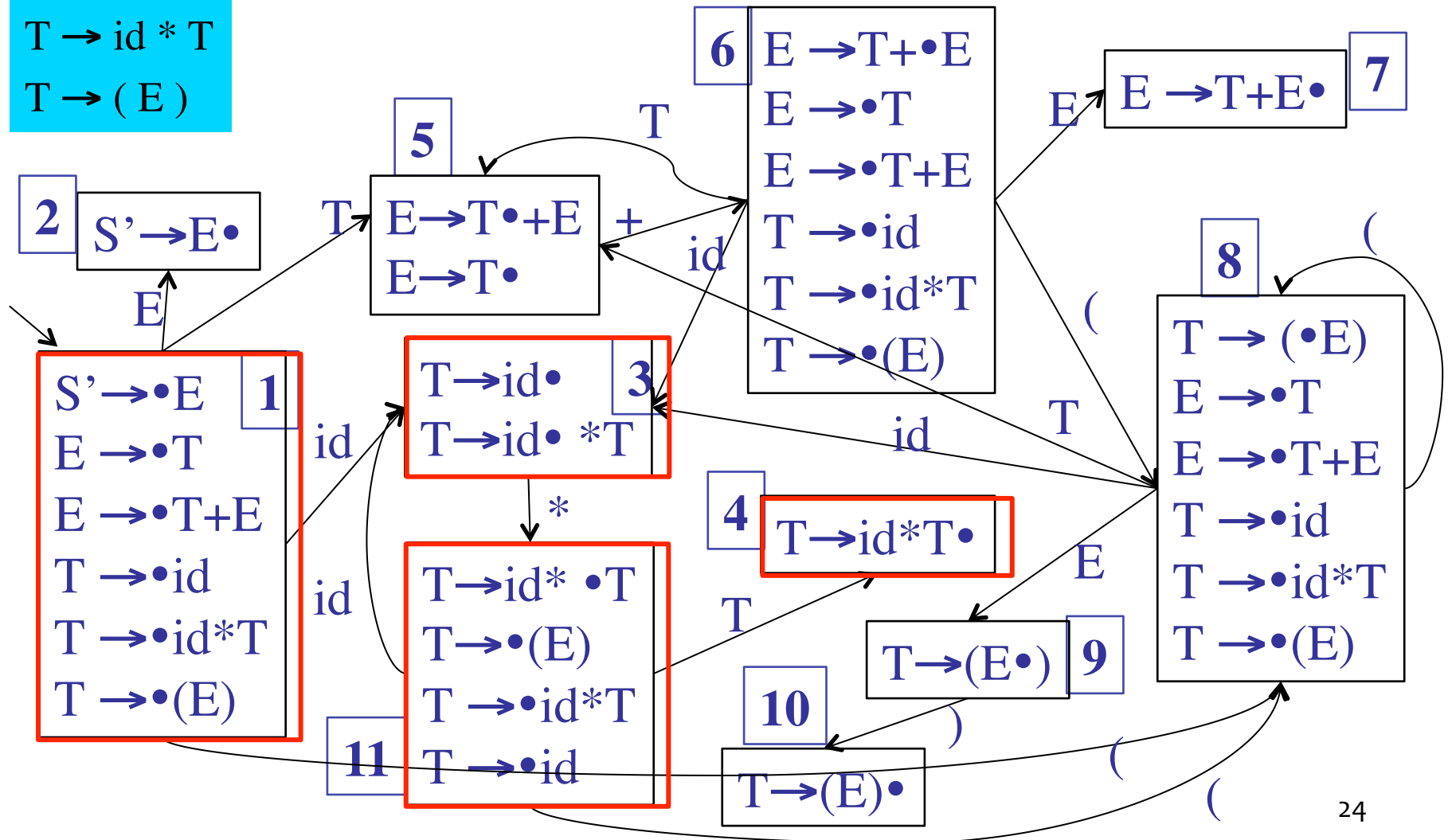
$T \rightarrow id$

$T \rightarrow id * T$

$T \rightarrow (E)$

$Follow(T) = \{\$, ), +\}$

$id * T \mid \$$





# Trace 'id\*id'

configuration (Stack input)	DFA halt state	Action
id * id \$	1	Shift
id   * id \$	3    * $\notin \text{Follow}(T)$	Shift
id *   id \$	11	Shift
id * id   \$	3    \$ $\in \text{Follow}(T)$	Reduce $T \rightarrow \text{id}$
id * T   \$	4    \$ $\in \text{Follow}(T)$	Reduce $T \rightarrow \text{id} * T$
T   \$		

$S' \rightarrow E$

$E \rightarrow T + E$

$E \rightarrow T$

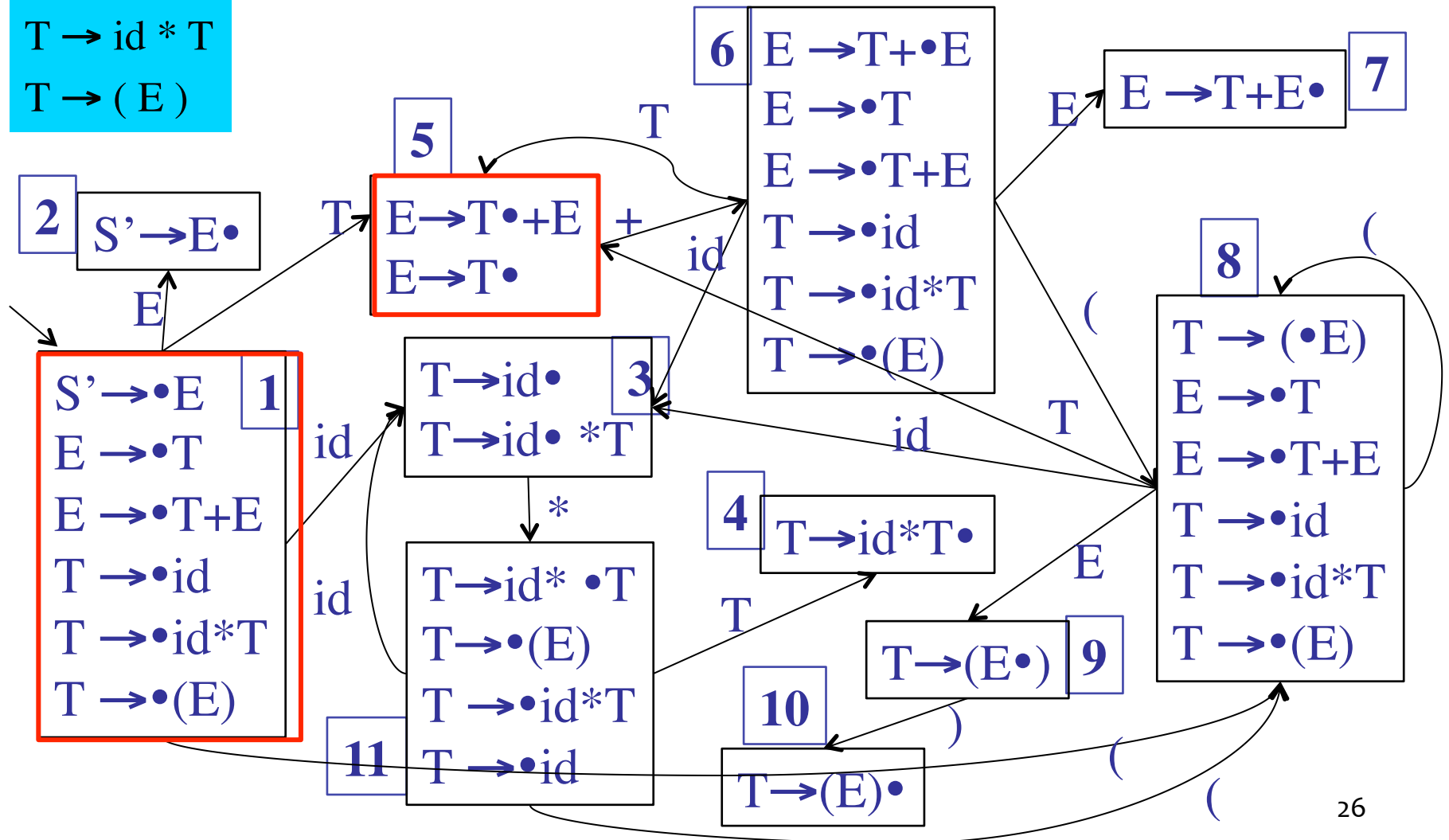
$T \rightarrow id$

$T \rightarrow id * T$

$T \rightarrow (E)$

$Follow(E) = \{\$, )\}$

$T \mid \$$



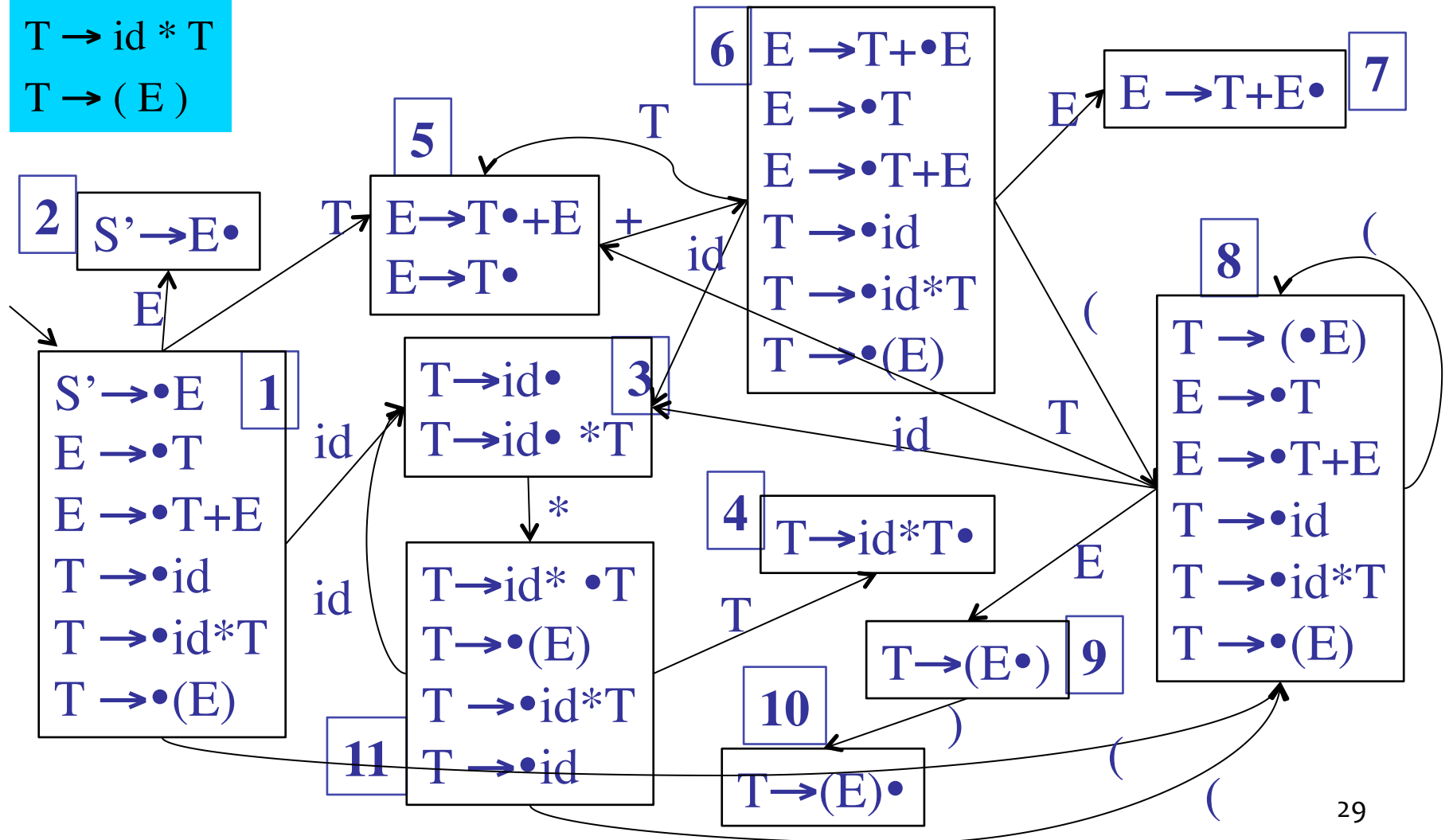
# Trace 'id\*id'

configuration (Stack input)	DFA halt state	Action
id * id \$	1	Shift
id   * id \$	3    * $\notin \text{Follow}(T)$	Shift
id *   id \$	11	Shift
id * id   \$	3    \$ $\in \text{Follow}(T)$	Reduce $T \rightarrow \text{id}$
id * T   \$	4    \$ $\in \text{Follow}(T)$	Reduce $T \rightarrow \text{id} * T$
T   \$	5    \$ $\in \text{Follow}(E)$	Reduce $E \rightarrow T$
E   \$		

# Trace 'id\*id'

configuration (Stack input)	DFA halt state	Action
id * id \$	1	Shift
id   * id \$	3    * $\notin \text{Follow}(T)$	Shift
id *   id \$	11	Shift
id * id   \$	3    \$ $\in \text{Follow}(T)$	Reduce $T \rightarrow \text{id}$
id * T   \$	4    \$ $\in \text{Follow}(T)$	Reduce $T \rightarrow \text{id} * T$
T   \$	5    \$ $\in \text{Follow}(T)$	Reduce $E \rightarrow T$
E   \$		Accept

$S' \rightarrow E$   
 $E \rightarrow T + E$   
 $E \rightarrow T$   
 $T \rightarrow id$   
 $T \rightarrow id * T$   
 $T \rightarrow (E)$



# Constructing SLR states

- Begin with item  $S' \rightarrow \bullet S$ , calculate related items (**closure**)
- Determine following states: what states can be reached on a single input token or non-terminal (**GOTO**)
- Construct closure of each resulting states

# SLR(1) Construction

1. Construct  $F = \{I_0, I_1, \dots, I_n\}$
2. a) if  $\{A \rightarrow \alpha \bullet\} \in I_i$  and  $A \neq S'$   
then  $\text{action}[i, b] := \text{reduce } A \rightarrow \alpha$   
for all  $b \in \text{Follow}(A)$   
b) if  $\{S' \rightarrow S \bullet\} \in I_i$   
then  $\text{action}[i, \$] := \text{accept}$   
c) if  $\{A \rightarrow \alpha \bullet a \beta\} \in I_i$  and  $\text{Successor}(I_i, a) = I_j$   
then  $\text{action}[i, a] := \text{shift } j$
3. if  $\text{Successor}(I_i, A) = I_j$  then  $\text{goto}[i, A] := j$

# SLR(1) Construction (cont'd)

4. All entries not defined are errors
  5. Make sure  $I_0$  is the initial state
- Note: SLR(1) only reduces  $\{A \rightarrow \alpha \bullet\}$  if lookahead in  $\text{Follow}(A)$
  - Shift and reduce items or more than one reduce item can be in the same configuration set as long as lookaheads are disjoint

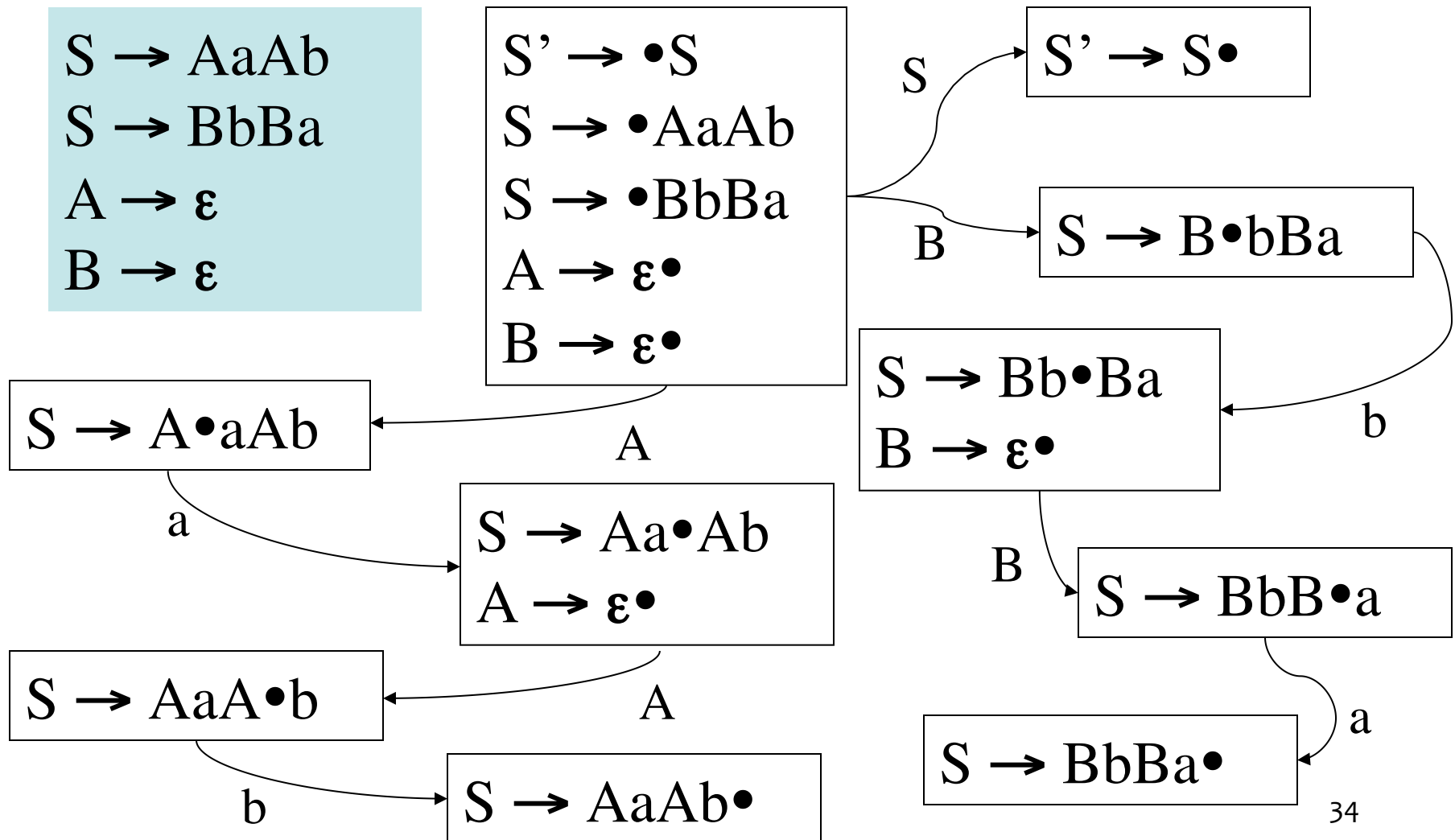


# SLR(1) Conditions

- A grammar is SLR(1) if for each configuration set:
  - For any item  $\{A \rightarrow \alpha \bullet x \beta : x \in T\}$  there is no  $\{B \rightarrow \gamma \bullet : x \in \text{Follow}(B)\}$
  - For any two items  $\{A \rightarrow \alpha \bullet\}$  and  $\{B \rightarrow \beta \bullet\}$   $\text{Follow}(A) \cap \text{Follow}(B) = \emptyset$

LR(0) Grammars  $\subset$  SLR(1) Grammars

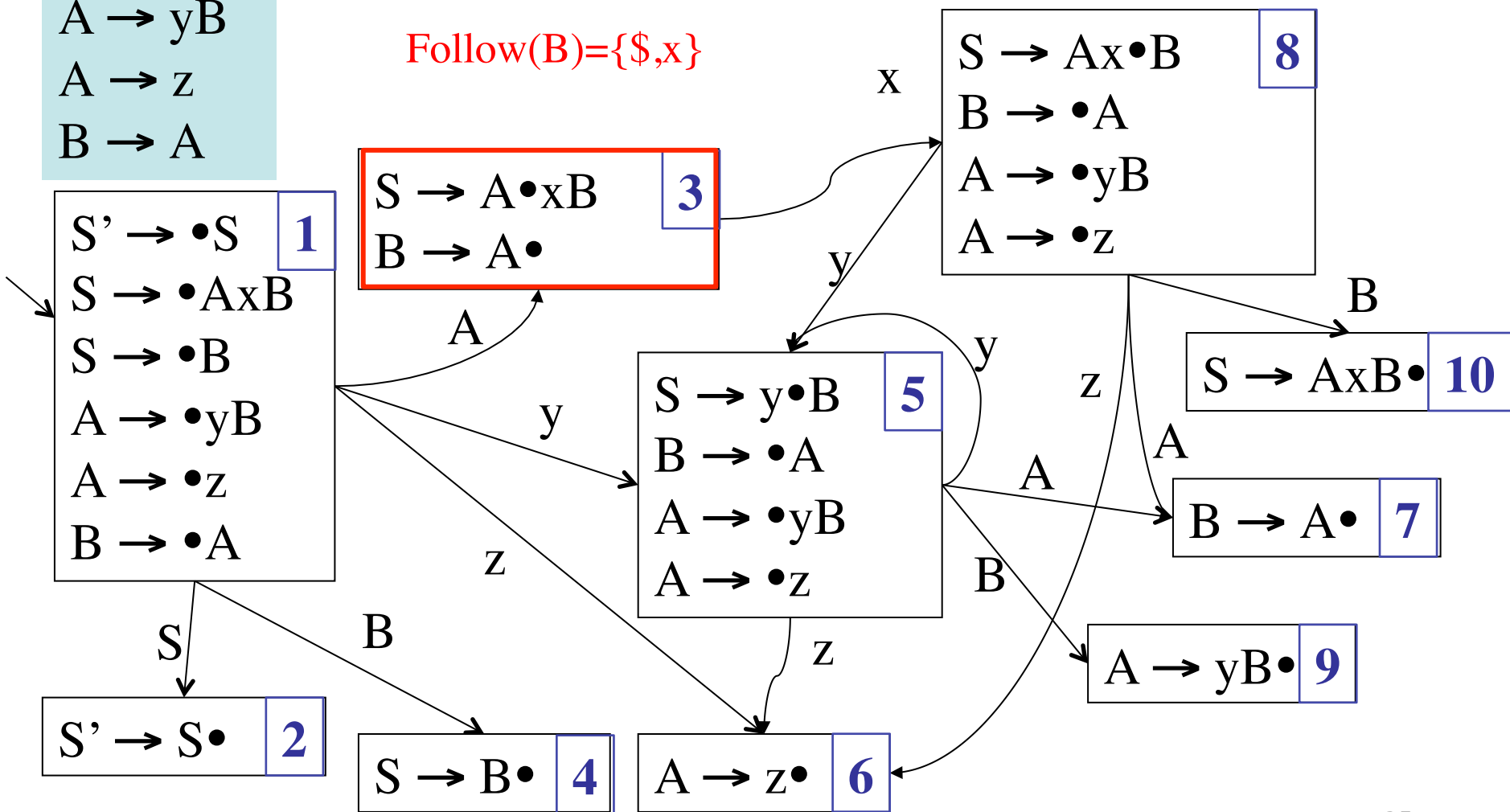
# Is this grammar SLR(1)?



$S' \rightarrow S$   
 $S \rightarrow AxB$   
 $S \rightarrow B$   
 $A \rightarrow yB$   
 $A \rightarrow z$   
 $B \rightarrow A$

# Is this grammar SLR(1)?

$\text{Follow}(B) = \{\$, x\}$



1)  $S \rightarrow AxB$

2)  $S \rightarrow B$

3)  $A \rightarrow yB$

4)  $A \rightarrow z$

5)  $B \rightarrow A$

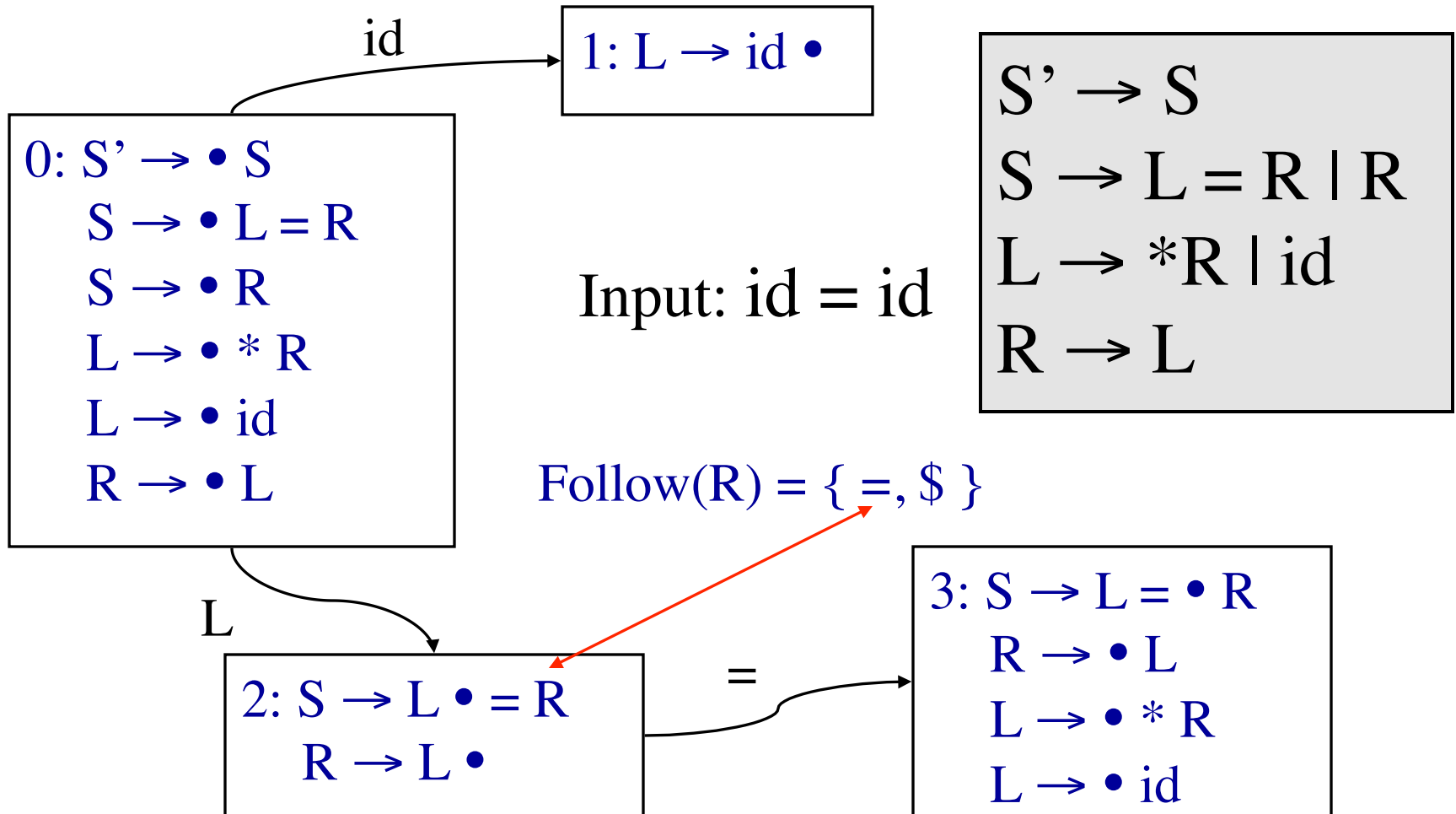
# SLR Parsing Table

	x	y	z	\$	S	A	B
1		S5	S6		2	3	4
2				ACC!			
3	S8,R5			R5			
4				R2			
5		S5	S6			7	9
6	R4			R4			
7	R5			R5			
8		S5	S6			7	10
9	R3			R3			
10				R1			

Grammar is not SLR

Reduce is a bad  
choice

# SLR limitation: lack of context



$$S' \rightarrow S$$
$$S \rightarrow L = R \mid R$$
$$L \rightarrow *R \mid \text{id}$$
$$R \rightarrow L$$
$$\text{Follow}(R) = \{ =, \$ \}$$
$$2: S \rightarrow L \bullet = R$$
$$R \rightarrow L \bullet$$

Find all lookaheads  
for reduce  $R \rightarrow L \bullet$

$S'$   
|  
 $S$   
|  
 $R$   $\$$   
|  
 $L$   
|  
 $\text{id}$

$S'$   
|  
 $S$   
|  
 $L$   $=$   $R$   $\$$   
|        |  
 $\text{id}$       $L$   
         |  
          $\text{id}$

$S'$   
|  
 $S$   
|  
 $R$   
|  
 $L$   
|  
 $*$   $R$   $\$$   
     |  
      $L$   
     |  
      $\text{id}$

$S'$   
|  
 $S$   
|  
 $L$   $=$   $R$   $\$$   
|        |  
 $*$       $R$   $=$   $L$   
     |        |  
      $L$       $\text{id}$   
     |  
      $\text{id}$   $\swarrow$   
          $\text{Problem?}$

No!  $R \rightarrow L \bullet$  reduce  
and  $S \rightarrow L \bullet = R$  do  
not co-occur due to  
the  $L \rightarrow *R$  rule

# Solution: Canonical LR(1)

- Extend definition of configuration
  - Remember lookahead
- New closure method
- Extend definition of Successor

# LR(1) Parsing

- Limit introduced by SLR parsing in using Follow set to decide reductions
- Idea: augment LR items with 1 character lookahead  $[B \rightarrow A\bullet, \$]$  making an LR(1) item
  - Reduce to  $B$  only if lookahead token is  $\$$
- More accurate than just Follow set
- Similar to SLR parsing just use LR(1) items rather than LR(0) items