LR Parsing

CMPT 379: Compilers
Instructor: Anoop Sarkar
anoopsarkar.github.io/compilers-class

LR(0) conflicts:

$S' \rightarrow T$
$T \rightarrow F$
$T \rightarrow T * F$
$T \rightarrow id$
$F \rightarrow id \mid (T)$
$F \rightarrow id = T ;$

1: $F \rightarrow id \bullet$
$F \rightarrow id \bullet = T$
Shift/reduce conflict

1: $F \rightarrow id \bullet$
$T \rightarrow id \bullet$
Reduce/Reduce conflict

Need more lookahead: SLR(1)

First(X)

First(X) = set of first terminal symbols in all derivations starting from non-terminal X

$S \rightarrow AB$

$A \rightarrow c \mid \varepsilon$

$B \rightarrow cbB \mid ca$

$B \Longrightarrow cbB$ $\quad$ $B \Longrightarrow ca$

FIRST(B) = {c}

$A \Longrightarrow c$ $\quad$ $A \Longrightarrow \varepsilon$

FIRST(A) = {c, ε}

$S \Longrightarrow AB \Longrightarrow B \Longrightarrow cbB$ $\quad$ $S \Longrightarrow AB \Longrightarrow cB$

$S \Longrightarrow AB \Longrightarrow B \Longrightarrow ca$

FIRST(S) = {c}

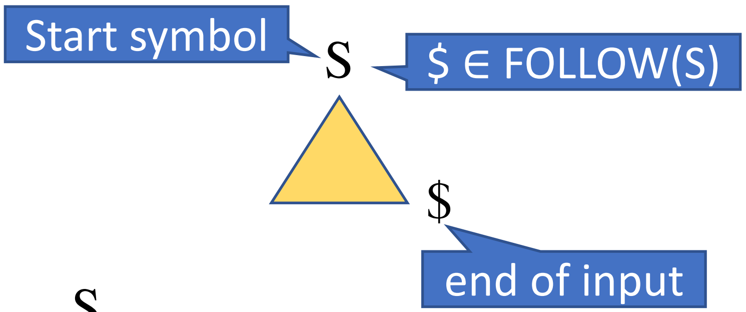
Follow(X)

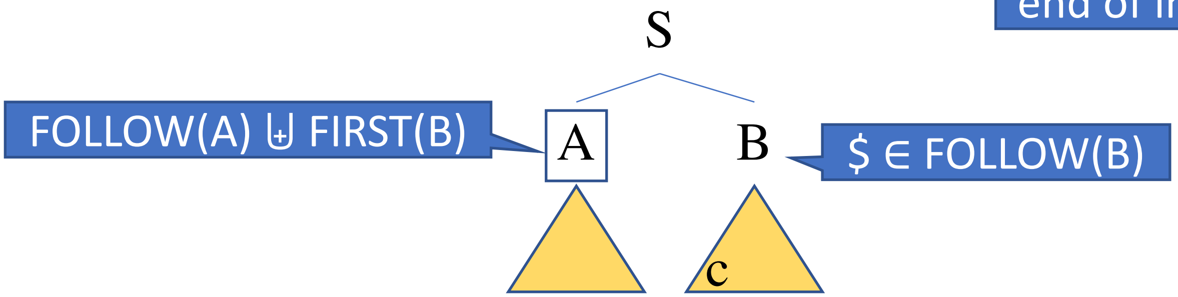
Follow(X) = set of terminal symbols that can follow non-terminal X in all parse trees starting from the start symbol

$S \rightarrow AB$

$A \rightarrow c \mid \varepsilon$

$B \rightarrow cbB \mid ca$





Example First/Follow

$S \to AB$

$A \to c \mid \varepsilon$

$B \to cbB \mid ca$

First(A) = {c, ε}

First(B) = {c}

First(cbB) =

First(ca) = {c}

First(S) = {c}

Follow(A) = {c}

Follow(A) $\cap$

First(c) = {c}

Follow(B) = {$}

Follow(S) = {$}

Example First/Follow

$S \rightarrow cAa$

$A \rightarrow cB \mid B$

$B \rightarrow bcB \mid \varepsilon$

If $X \rightarrow \alpha A a \beta$ and a is terminal, then the set Follow(A) includes a

If $X \rightarrow \alpha A B \beta$ and a is in set First(B), then the set Follow(A) includes a

If $X \rightarrow \alpha A$ then the set Follow(A) includes Follow(X)

First(A) = {b, c, ε}

First(B) = {b, ε}

First(S) = {c}

Follow(A) = {a}

Follow(B) = {a}

Follow(S) = {$}

FIRST and FOLLOW

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

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

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

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

$$\text{and } \gamma \Rightarrow^* \epsilon$$

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 $F \rightarrow id$?

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

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

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

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

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 $C \rightarrow id$?

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

Follow(C) = { (}

SLR(1) : Simple LR(1) Parsing

0: $S' \to \bullet\, T$
$T \to \bullet\, F$
$T \to \bullet\, T * F$
$T \to \bullet\, C\ (T)$
$F \to \bullet\, id$
$F \to \bullet\, id\ {+}{+}$
$F \to \bullet\, (\ T\ )$
$C \to \bullet\, id$

$\xrightarrow{\text{id}}$

1: $F \to id\, \bullet$
$F \to id\, \bullet\ \mathbf{{+}{+}}$
$C \to id\, \bullet$

$S' \to T$
$T \to F \mid T * F \mid C\ (\ T\ )$
$F \to id \mid id\ {+}{+} \mid (\ T\ )$
$C \to id$

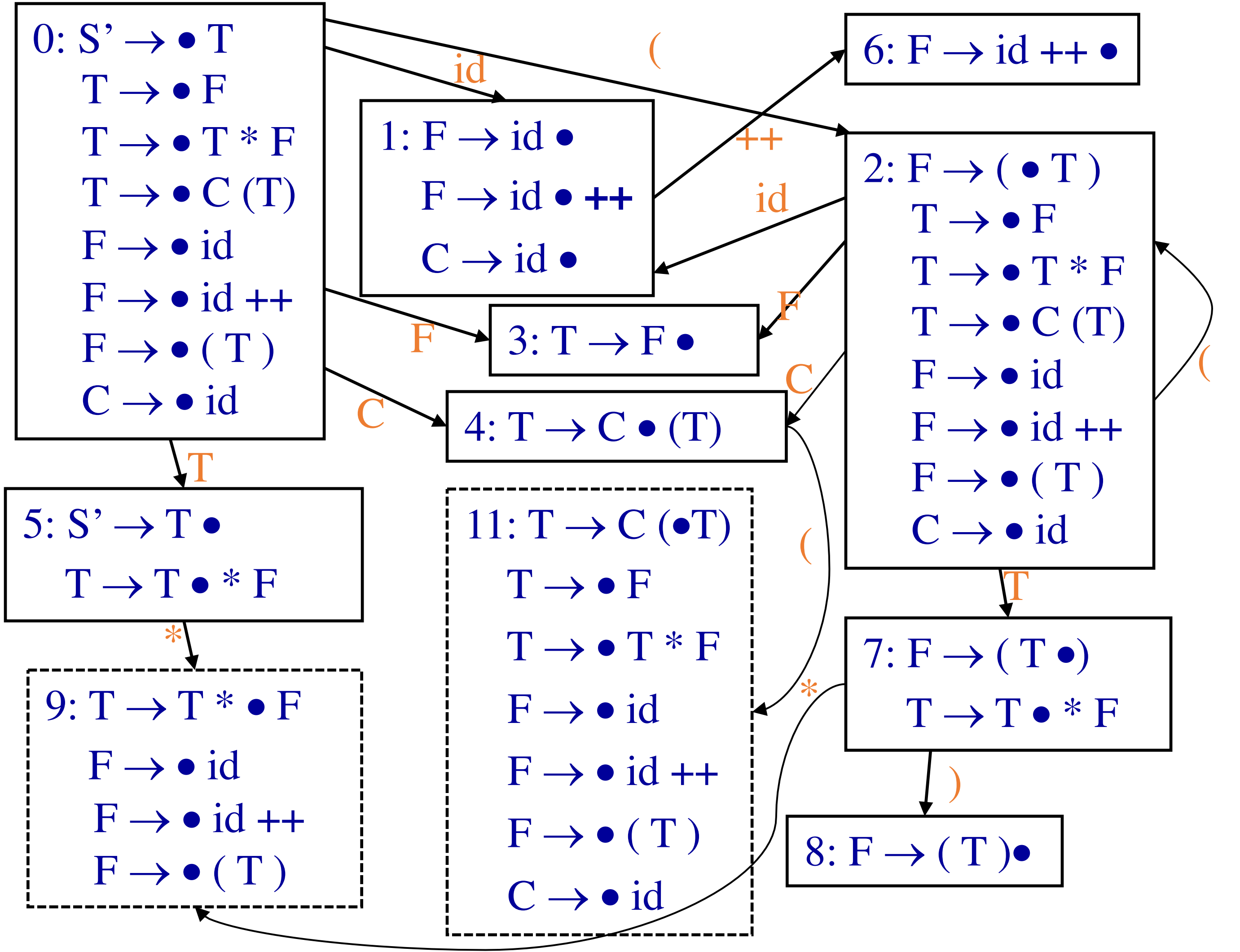
Follow(F) = { *,), $ }

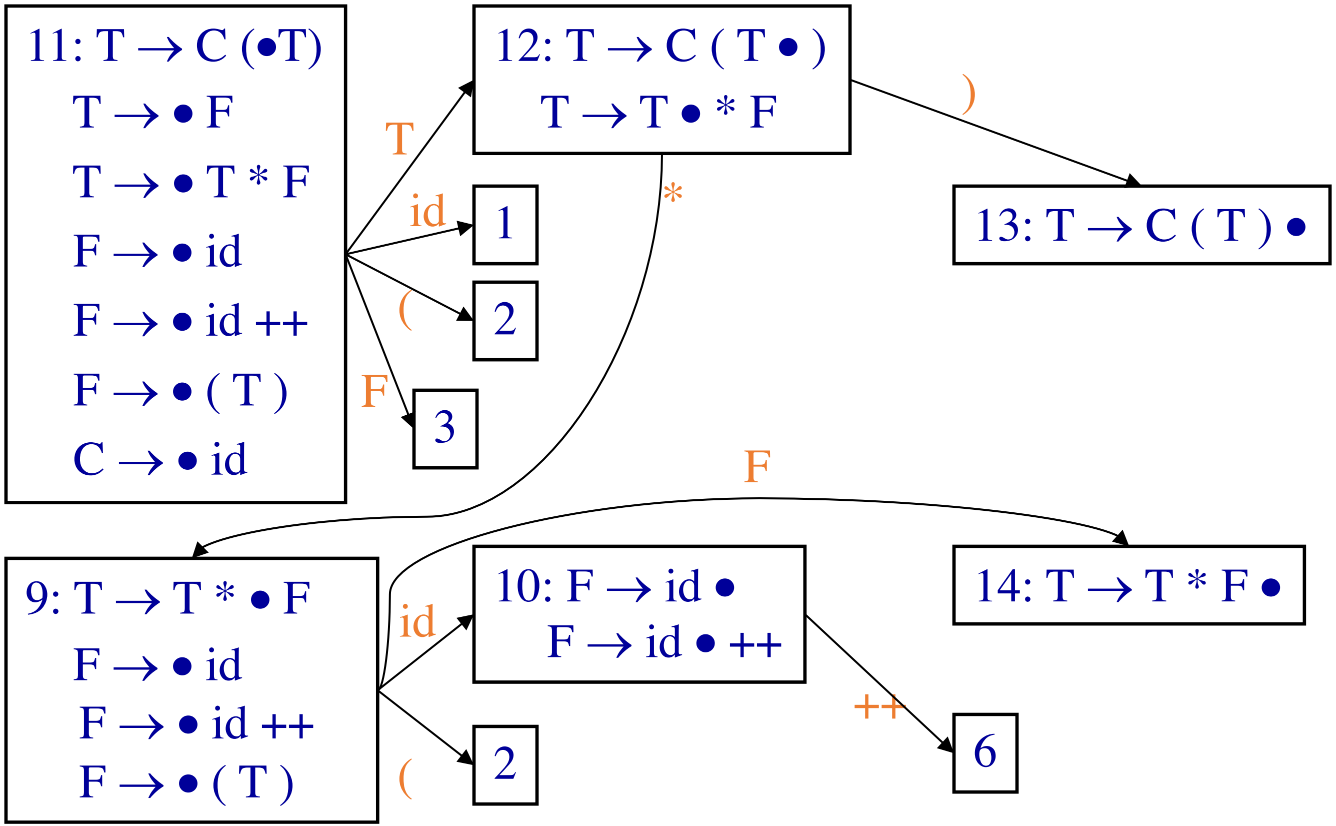
Follow(C) = { (}

action[1,*]= action[1,)] = action[1,$] = Reduce $F \to id$

action[1,(] = Reduce $C \to id$

action[1,++] = Shift


0: S’ → • T
T → • F
T → • T * F
T → • C (T)
F → • id
F → • id ++
F → • (T)
C → • id
1: F → id •
F → id • ++
C → id •
2: F → (• T)
T → • F
T → • T * F
T → • C (T)
F → • id
F → • id ++
F → • (T)
C → • id
3: T → F •
4: T → C • (T)
5: S’ → T •
T → T • * F
6: F → id ++ •
7: F → (T •)
T → T • * F
8: F → (T)•
9: T → T * • F
F → • id
F → • id ++
F → • (T)
11: T → C (•T)
T → • F
T → • T * F
F → • id
F → • id ++
F → • (T)
C → • id
id
(
++
id
F
F
C
C
T
(
(
T
*
*
)



11: T → C (•T)
T → • F
T → • T * F
F → • id
F → • id ++
F → • (T)
C → • id
T
id
(
F
1
2
3
12: T → C (T •)
T → T • * F
)
*
13: T → C (T) •
F
9: T → T * • F
F → • id
F → • id ++
F → • (T)
id
(
10: F → id •
F → id • ++
2
++
6
14: T → T * F •

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

SLR Parsing

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

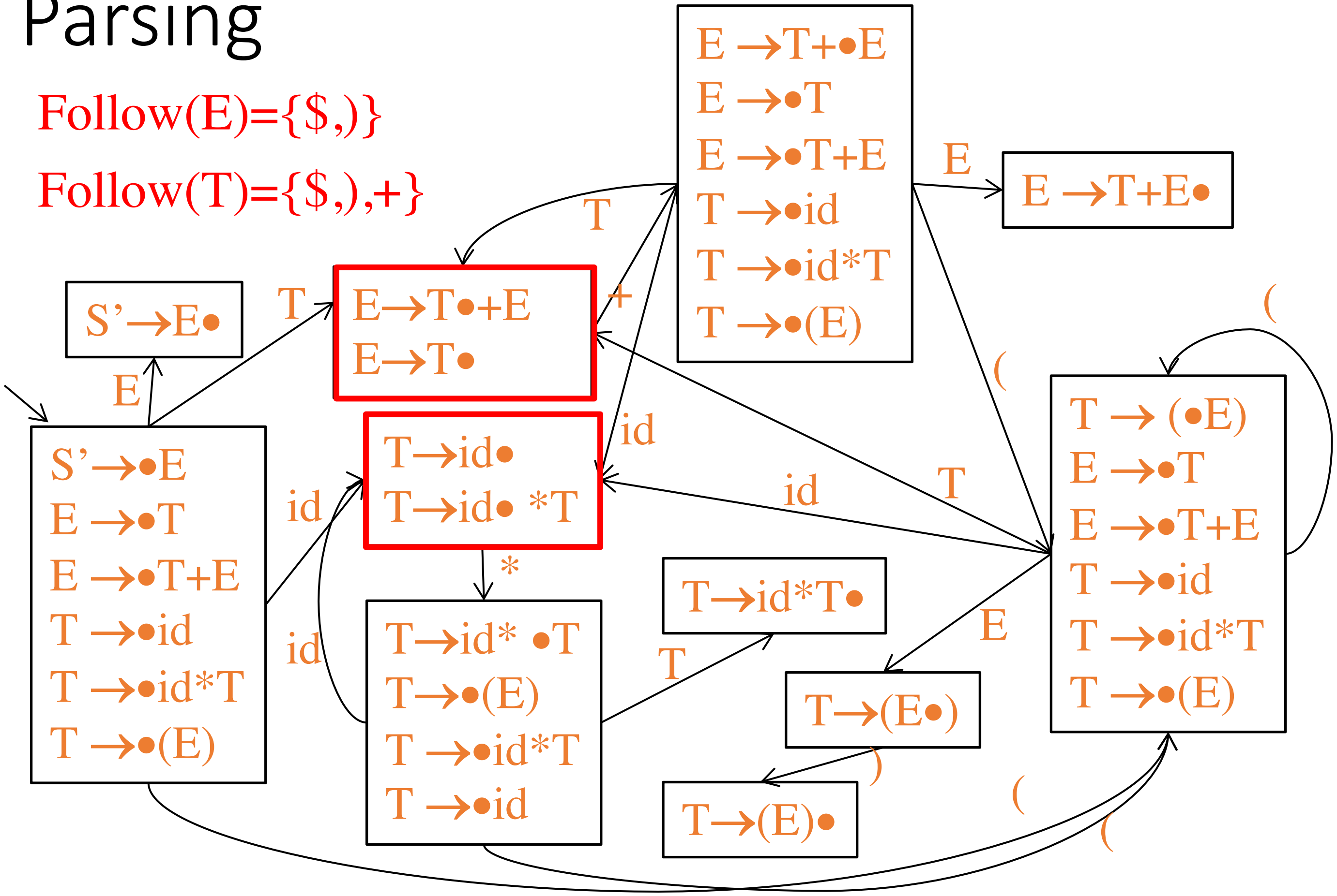
If there are still conflicts under these rules, grammar is not SLR(1)

SLR Parsing

S’ → E
E → T + E
E → T
T → id
T → id * T
T → (E)

Follow(E)={$,)}

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



SLR Parsing

- Let M be the finite-state automaton for viable prefixes of G
- Let $|x_1 \ldots x_n\$$ be initial configuration
- Repeat until configuration is $S|\$$
  - Let $\alpha |\ \omega$ be current configuration
  - Run M on current stack α
  - If M rejects α, report parsing error
    - Stack α is not a viable prefix
  - If M accepts α with items I, let a be the next input
    - Shift $[X \rightarrow \beta \bullet a\ \gamma] \ \in I$
    - Reduce if $[X \rightarrow \beta \bullet] \in I$ 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 actions), grammar is not SLR(1)

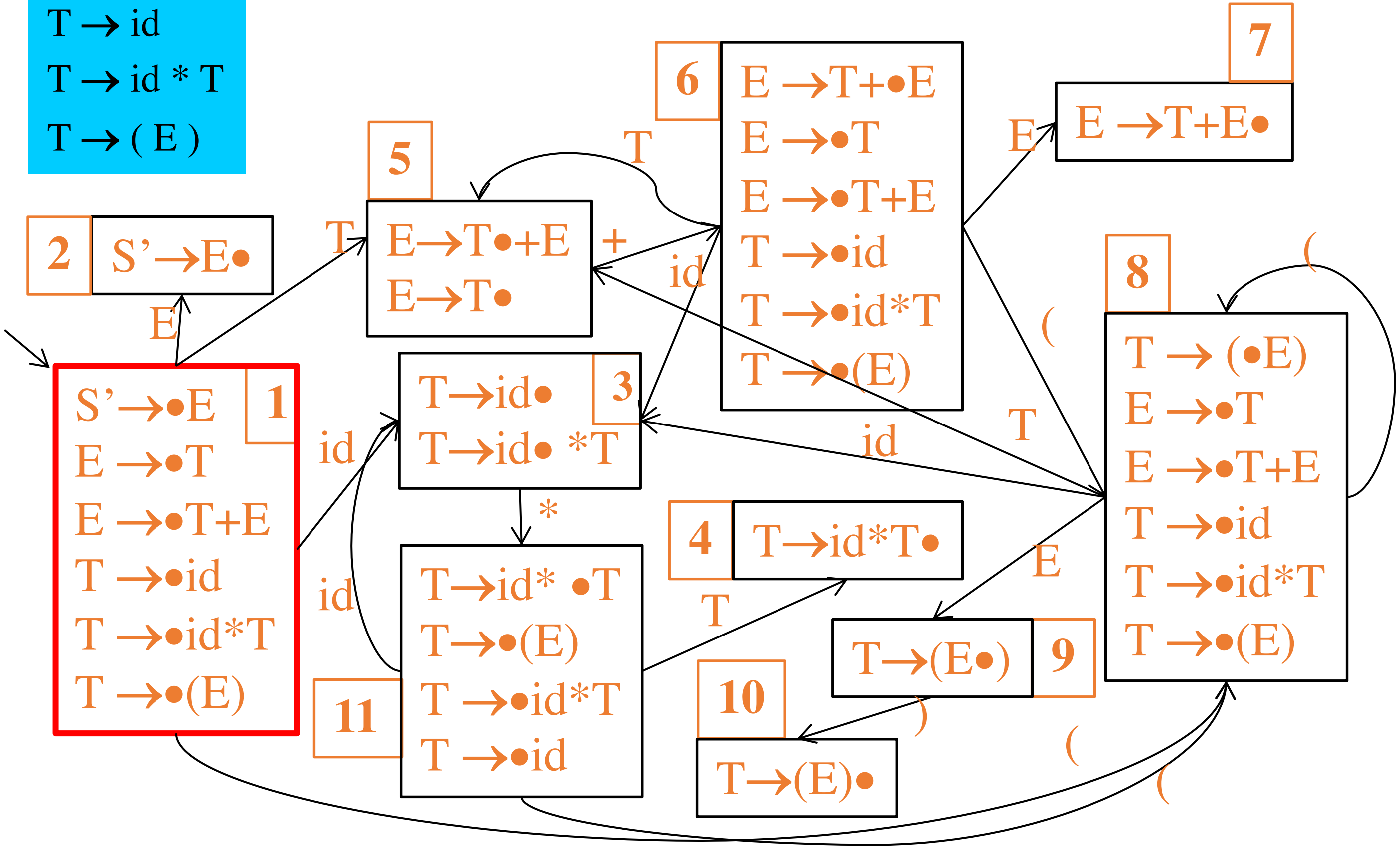
Trace `id*id`

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

Input	Stack	Action
| id * id $		

S' → E
E → T + E
E → T
T → id
T → id * T
T → (E)

| id * id $



Trace **`id*id`**

S’ → E
E → T + E
E → T
T → id
T → id * T
T → (E)

Input	Stack	Action
| id * id \$ id | * id \$	1	Shift 3

S' → E

E → T + E

E → T

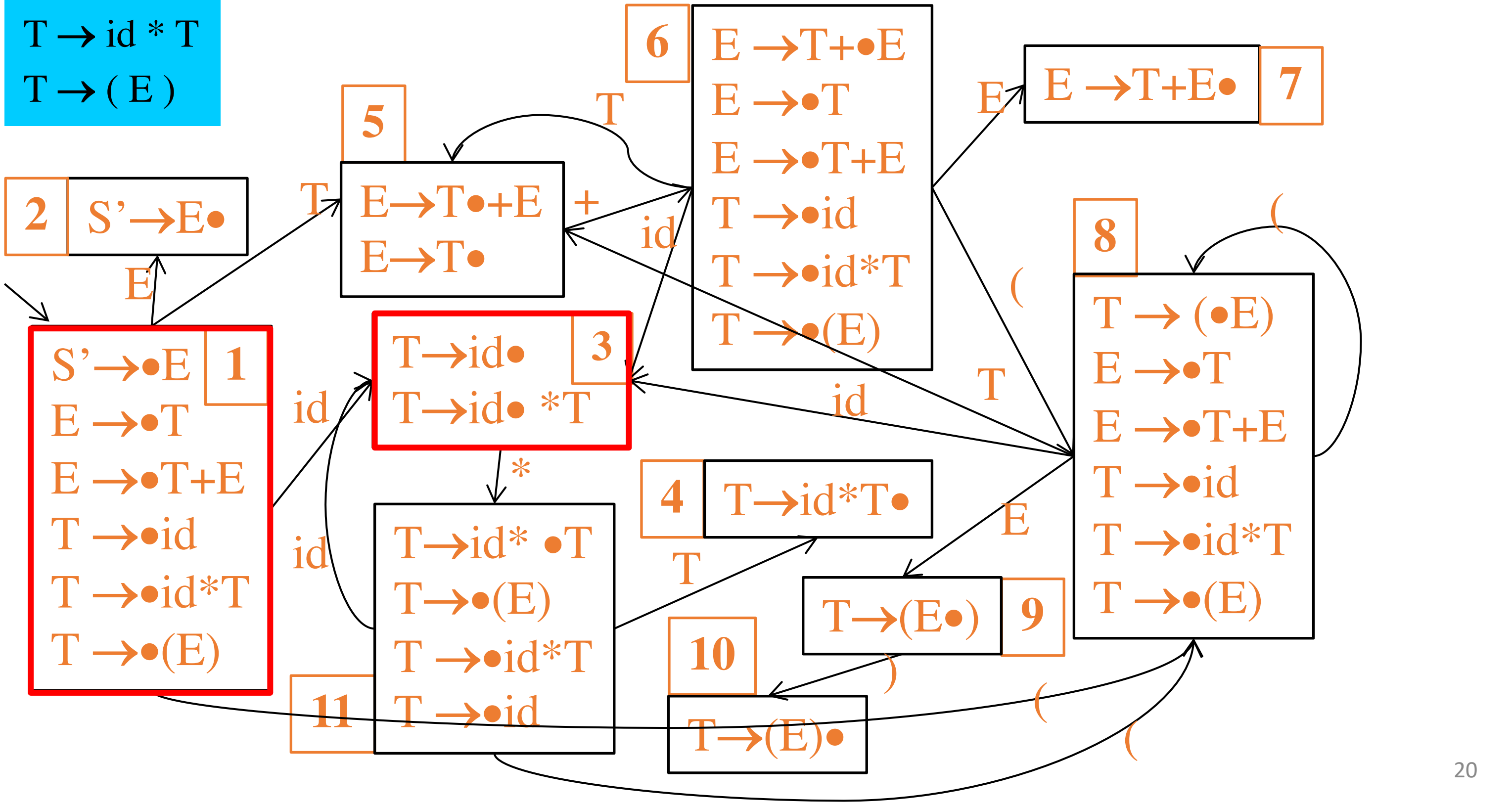
T → id

T → id * T

T → (E)

id | * id $

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



Trace `id*id`

S' → E
E → T + E
E → T
T → id
T → id * T
T → (E)

Input	Stack	Action
| id * id $	1	Shift 3
id | * id $	1 3 * ∉ Follow(T)	Shift 11
id * | id $		

S’ → E

E → T + E

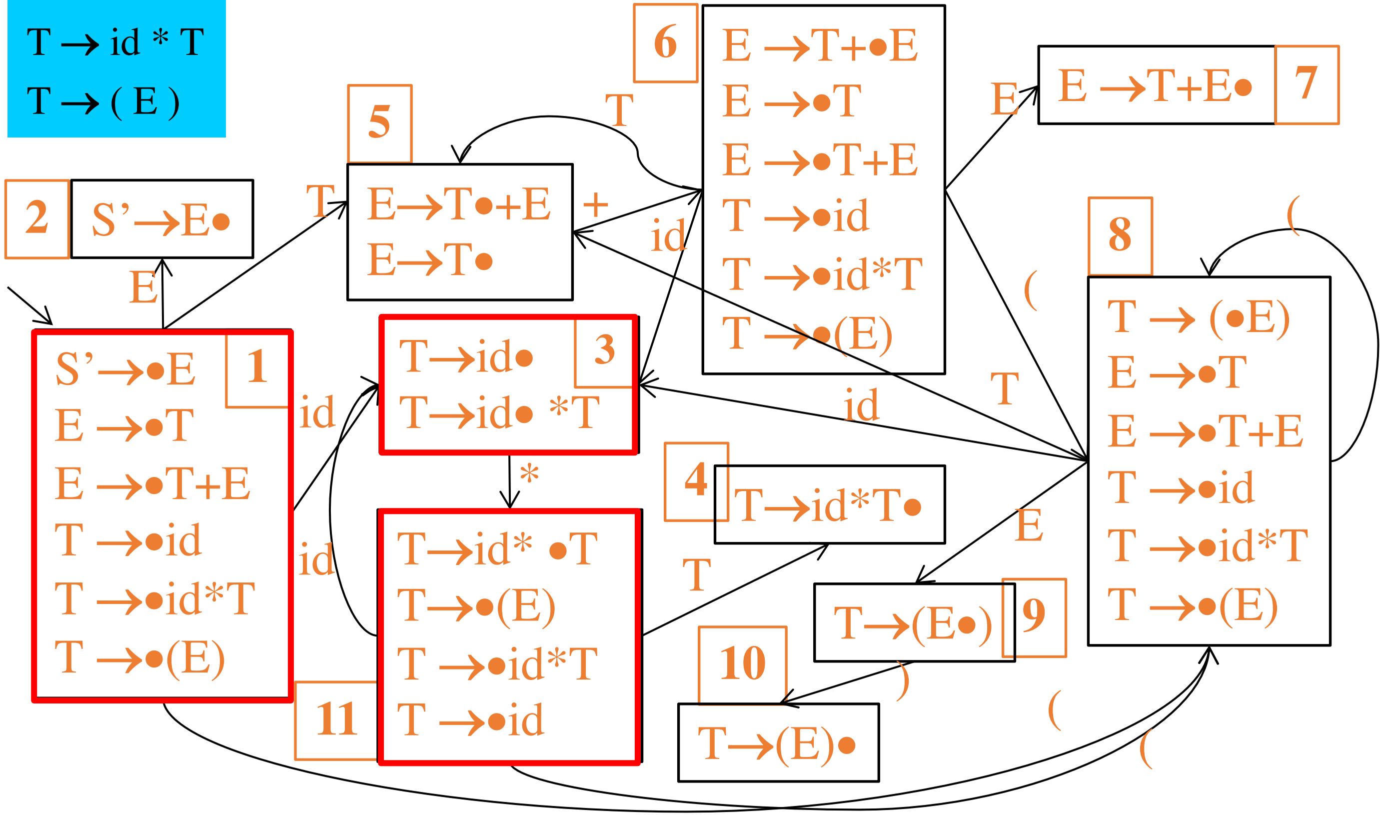
E → T

T → id

T → id * T

T → (E)

id * | id $



Trace `id*id`

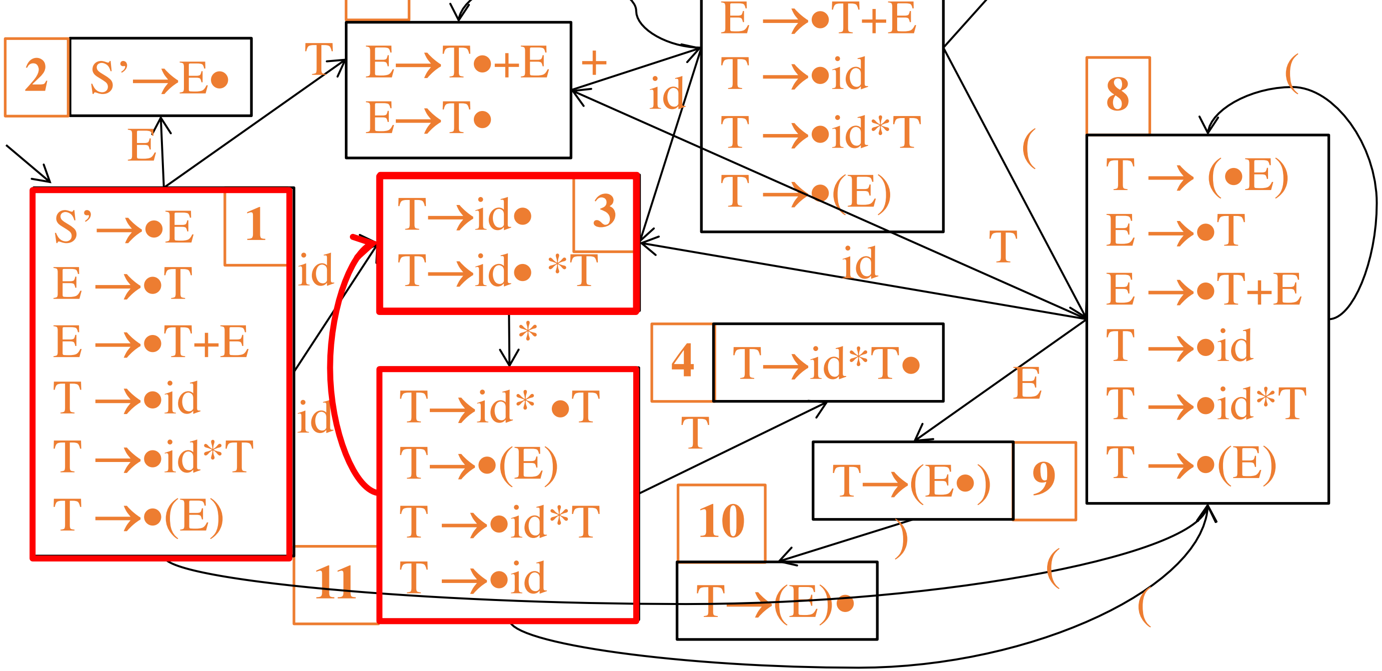
$S' \to E$
$E \to T + E$
$E \to T$
$T \to id$
$T \to id * T$
$T \to (E)$

Input	Stack	Action
| id * id $	1	Shift 3
id | * id $	1 3 * ∉ Follow(T)	Shift 11
id * | id $	1 3 11	Shift 3
id * id | $		

S’ → E
E → T + E
E → T
T → id
T → id * T
T → (E)

id * id | $

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



Trace `id*id`

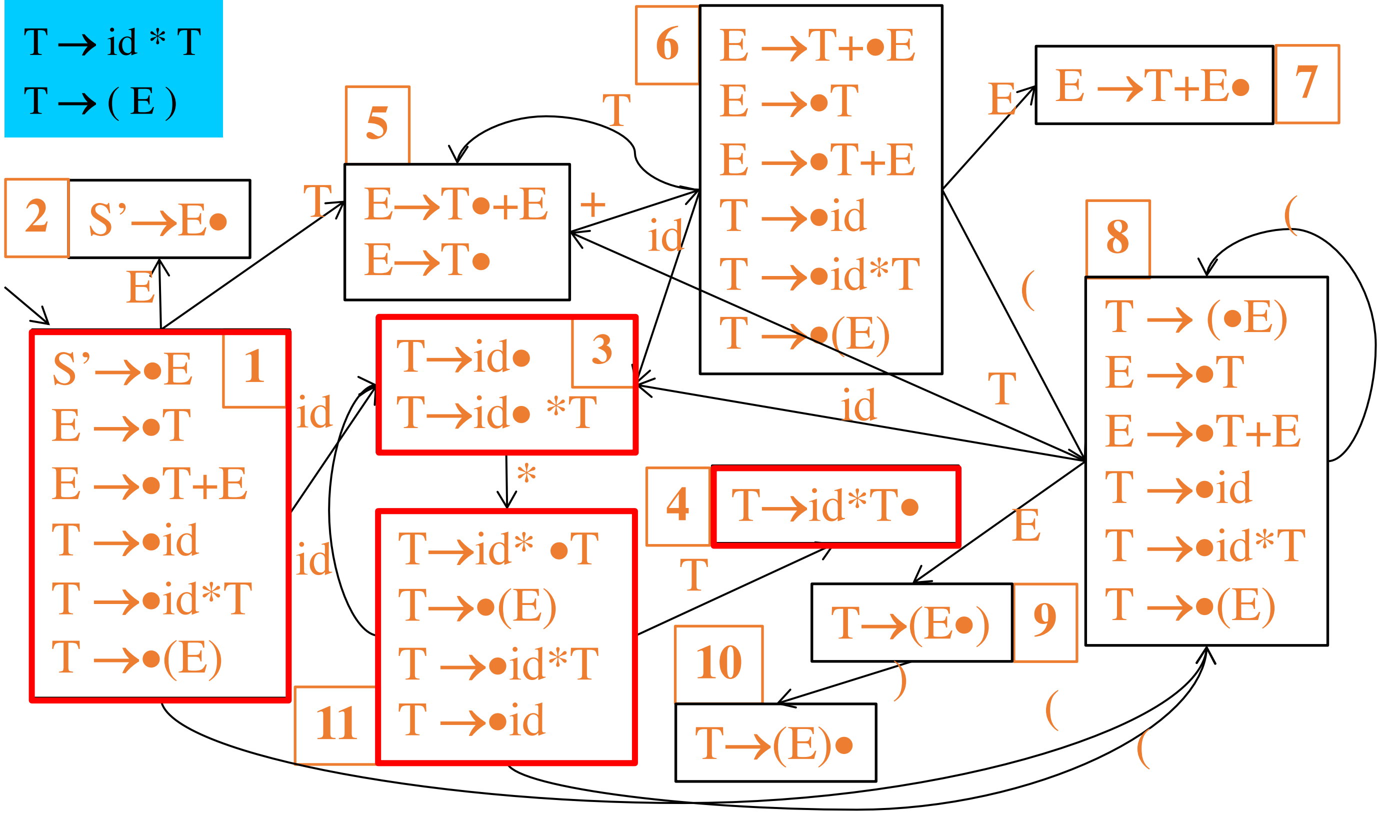
$S' \rightarrow E$
$E \rightarrow T + E$
$E \rightarrow T$
$T \rightarrow id$
$T \rightarrow id * T$
$T \rightarrow (E)$

Input	Stack	Action
| id * id $	1	Shift
id | * id $	1 3 * ∉ Follow(T)	Shift
id * | id $	1 3 11	Shift
id * id | $	1 3 11 3 $ ∈ Follow(T)	Reduce T → id
id * T | $		

S' → E
E → T + E
E → T
T → id
T → id * T
T → (E)

id * T | $

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



Trace `id*id`

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

Input	Stack	Action
| id * id $	1	Shift
id | * id $	1 3 $* \notin$ Follow(T)	Shift
id * | id $	1 3 11	Shift
id * id | $	1 3 11 3 $\$ \in$ Follow(T)	Reduce $T \to id$
id * T | $	1 3 11 4 $\$ \in$ Follow(T)	Reduce $T \to id * T$
T | $		

S’ → E

E → T + E

E → T

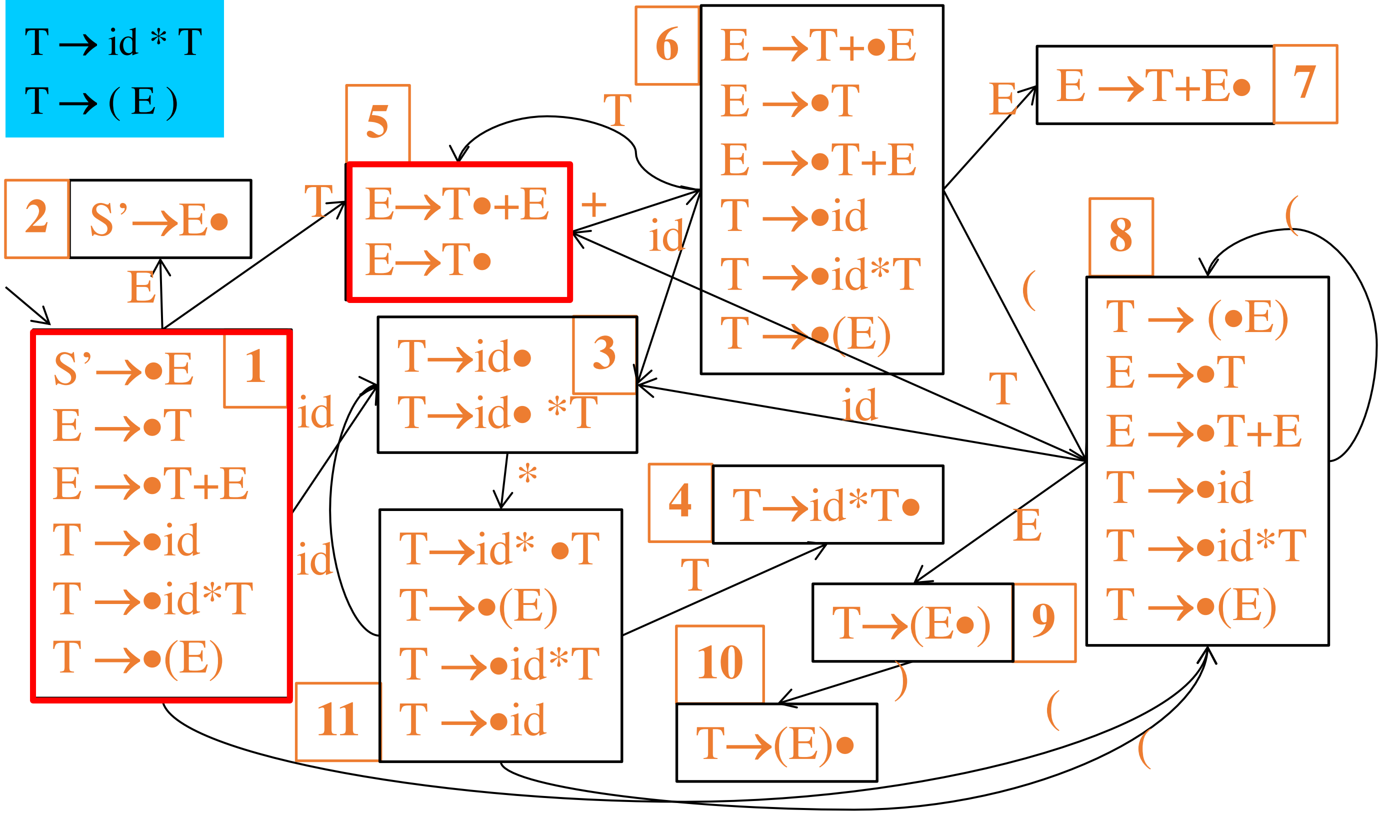
T → id

T → id * T

T → (E)

T | $

Follow(E)={$,)}



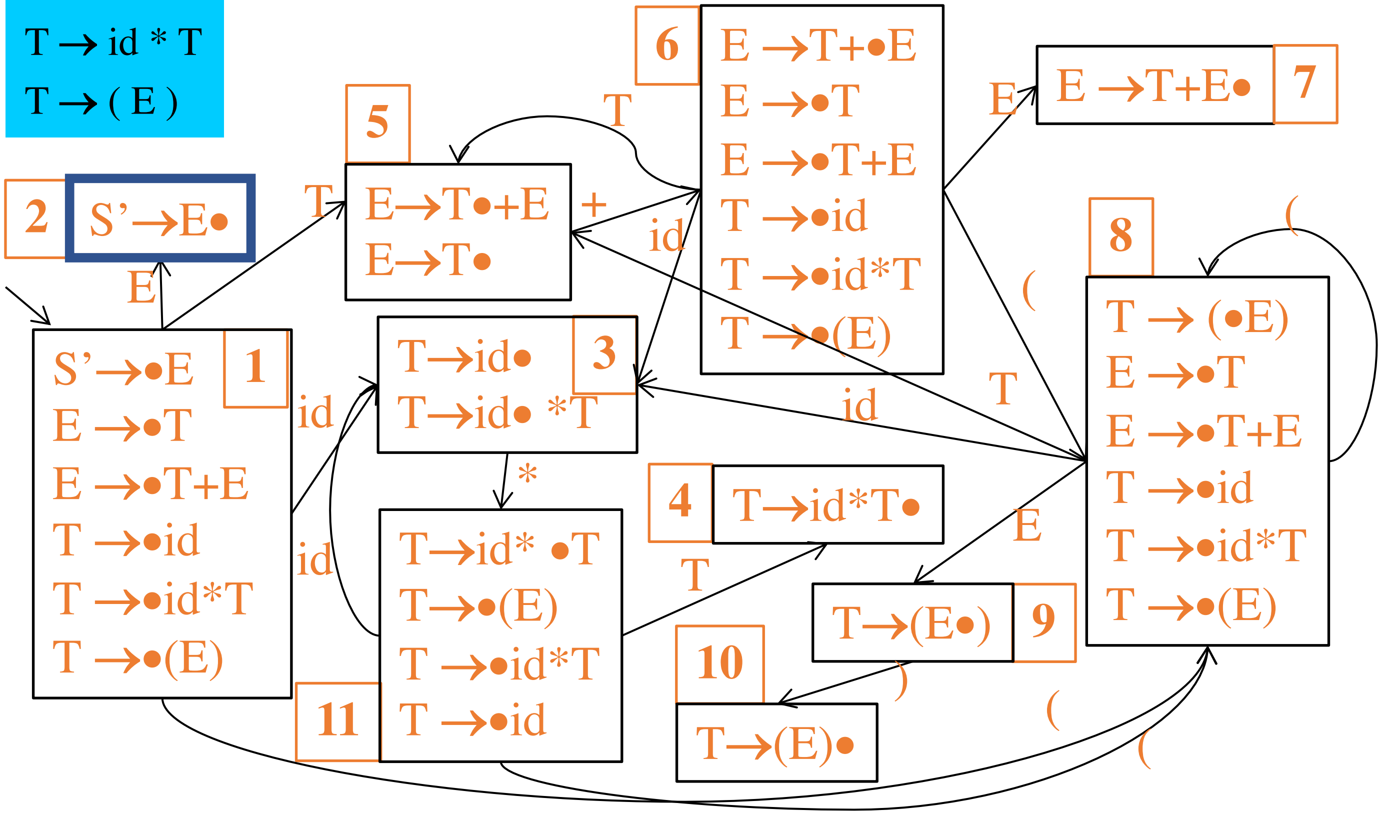
Trace `id*id`

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

Input	Stack	Action
| id * id $	1	Shift
id | * id $	1 3 $* \notin$ Follow(T)	Shift
id * | id $	1 3 11	Shift
id * id | $	1 3 11 3 $\$ \in$ Follow(T)	Reduce $T \to id$
id * T | $	1 3 11 4 $\$ \in$ Follow(T)	Reduce $T \to id * T$
T | $	1 5 $\$ \in$ Follow(T)	Reduce $E \to T$
E | $		

S' → E
E → T + E
E → T
T → id
T → id * T
T → (E)

E | $



Trace `id*id`

S' → E
E → T + E
E → T
T → id
T → id * T
T → (E)

Input	Stack	Action
| id * id $	1	Shift
id | * id $	1 3 * ∉ Follow(T)	Shift
id * | id $	1 3 11	Shift
id * id | $	1 3 11 3 $ ∈ Follow(T)	Reduce T → id
id * T | $	1 3 11 4 $ ∈ Follow(T)	Reduce T → id * T
T | $	1 5 $ ∈ Follow(T)	Reduce E→T
E | $	1 2 $ ∈ Follow(E)	Accept

SLR(1) Construction

1. Construct $F = \{I_0, I_1, \ldots I_n\}$
2. a) if $\{A \rightarrow \alpha \bullet\} \in I_i$ and A != S'

   then action[i, b] := reduce $A \rightarrow \alpha$ for all $b \in$ Follow(A)

   b) if $\{S' \rightarrow S\bullet\} \in I_i$

   then action[i, $] := accept

   c) if $\{A \rightarrow \alpha \bullet a\beta\} \in I_i$ and Successor(I_i, a) = I_j

   then action[i, a] := shift j
3. if Successor(I_i, A) = I_j and A is a non-terminal then 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 is in Follow(A)
- Shift and reduce items or more than one reduce item can be in the same itemset as long as lookaheads are disjoint

SLR(1) Conditions

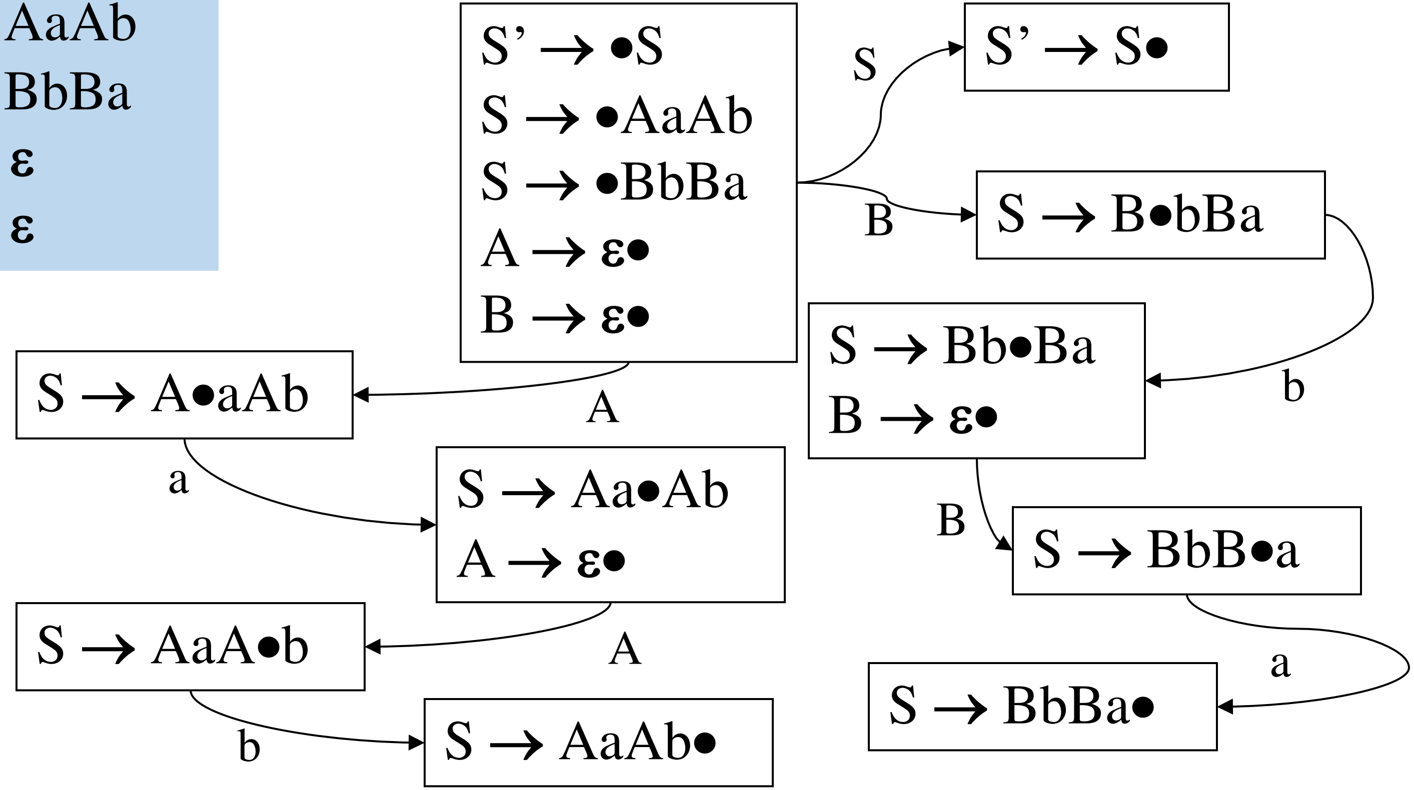
- A grammar is SLR(1) if for each configuration set:
  - For any item $\{A \to \alpha \bullet x\beta : x \in T\}$ there is no $\{B \to \gamma\bullet : x \in \text{Follow}(B)\}$
  - For any two items $\{A \to \alpha\bullet\}$ and $\{B \to \beta\bullet\}$ $\text{Follow}(A) \cap \text{Follow}(B) = \varnothing$

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

Extra Slides

Is this grammar SLR(1)?

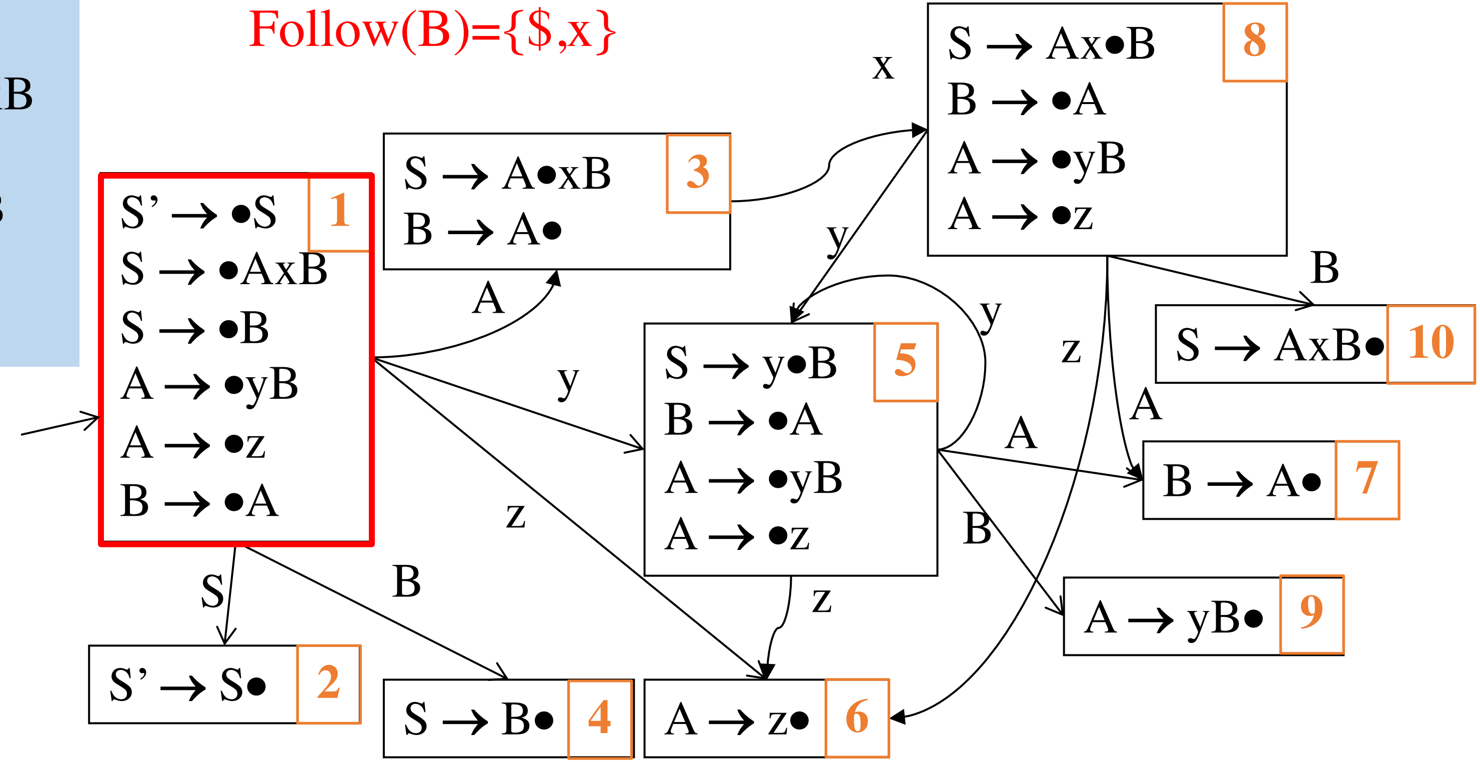
S → AaAb
S → BbBa
A → ε
B → ε



Is this grammar SLR(1)?

S’→ S
S → AxB
S → B
A → yB
A → z
B → A

Follow(B)={$,x}



SLR Parsing Table

Grammar is not SLR

	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			

0) $S' \rightarrow S$
1) $S \rightarrow AxB$
2) $S \rightarrow B$
3) $A \rightarrow yB$
4) $A \rightarrow z$
5) $B \rightarrow A$