Compiler Design

Fatemeh Deldar

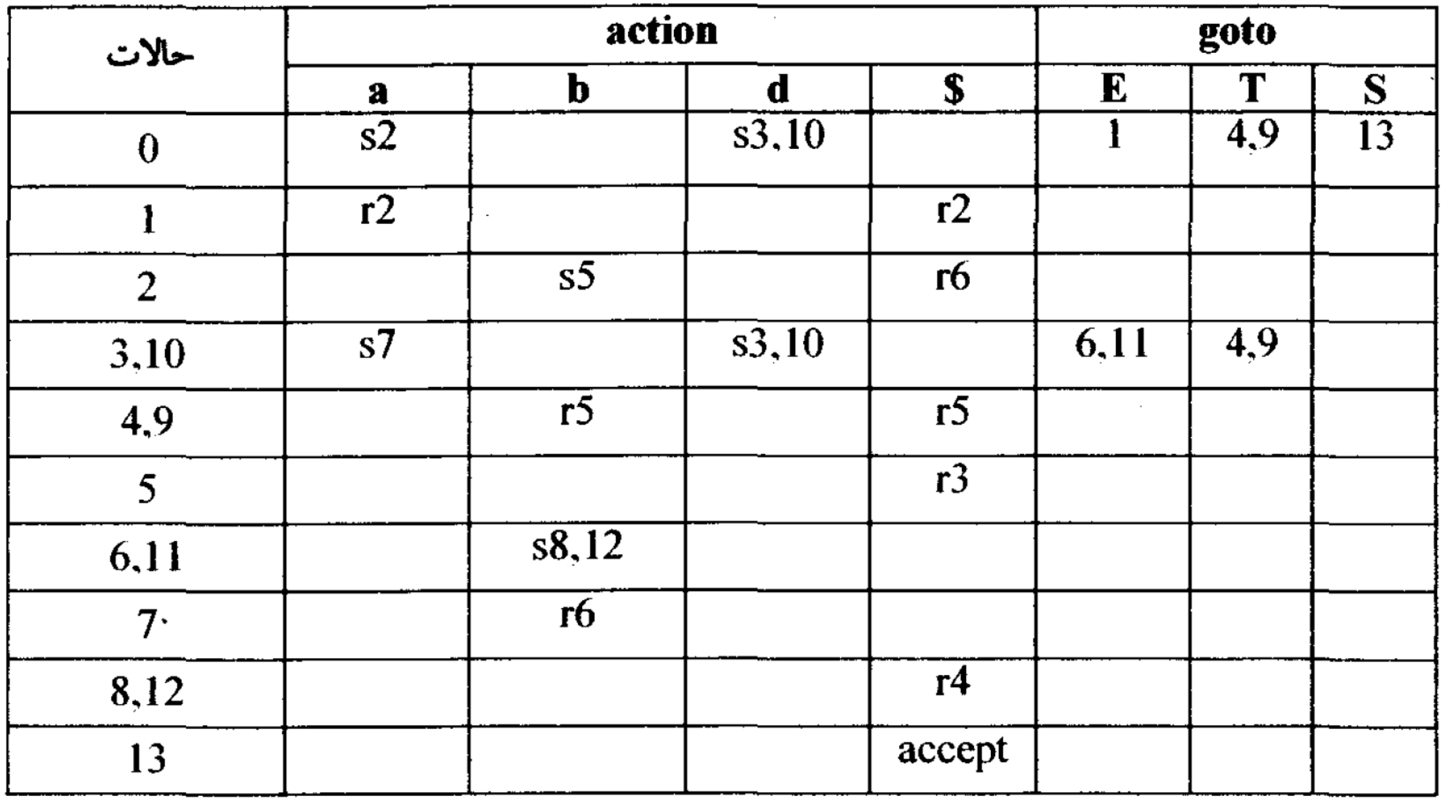
Isfahan University of Technology

1403-1404

LALR(1) Grammar

- مثال: با داشتن گرامر و جدول تجزیه LALR(1) زیر، مراحل تجزیه رشته dab را بنویسید.
(پاسخ اسلاید بعد)

1- A→ S
2- S→ E
3- S→ ab
4- E→ dEb
5- E→ T
6- T→ a



حالات	action				goto		
	a	b	d	$	E	T	S
0	s2		s3,10		1	4,9	13
1	r2			r2			
2		s5		r6			
3,10	s7		s3,10		6,11	4,9	
4,9		r5		r5			
5				r3			
6,11		s8,12					
7·		r6					
8,12				r4			
13				accept			

LALR(1) Grammar

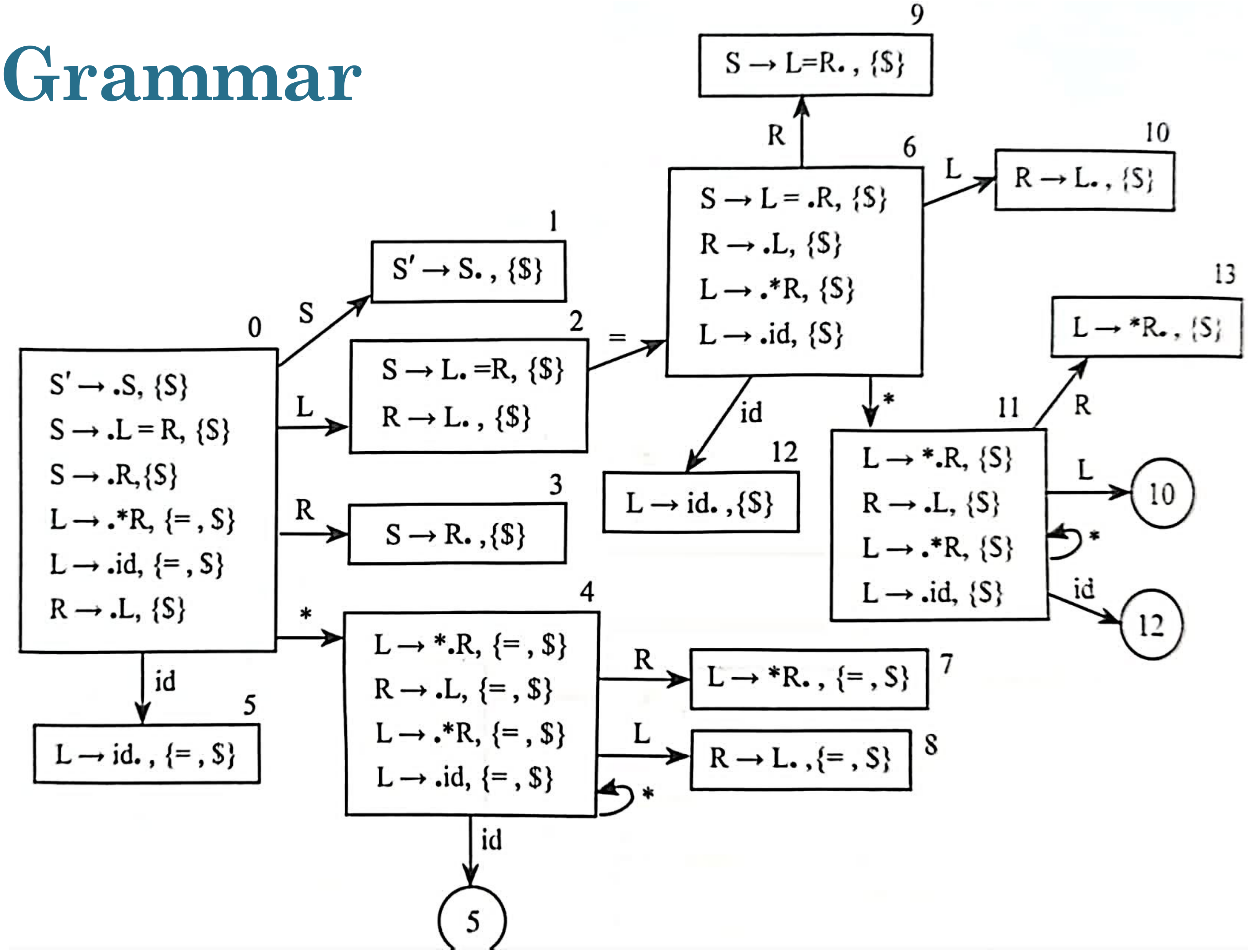
- مثال (ادامه)

پشته	رشته ورودی
0	dab$
0d3,10	ab$
0d3,10a7	b$
0d3,10T	b$
0d3,10T4,9	b$
0d3,10E	b$
0d3,10E6,11	b$
0d3,10E6,11b8,12	$
0E	$
0E1	$
0S	$
0S13 (accept)	$

LALR(1) Grammar

- **Example**
  - **LR(1) Diagram**

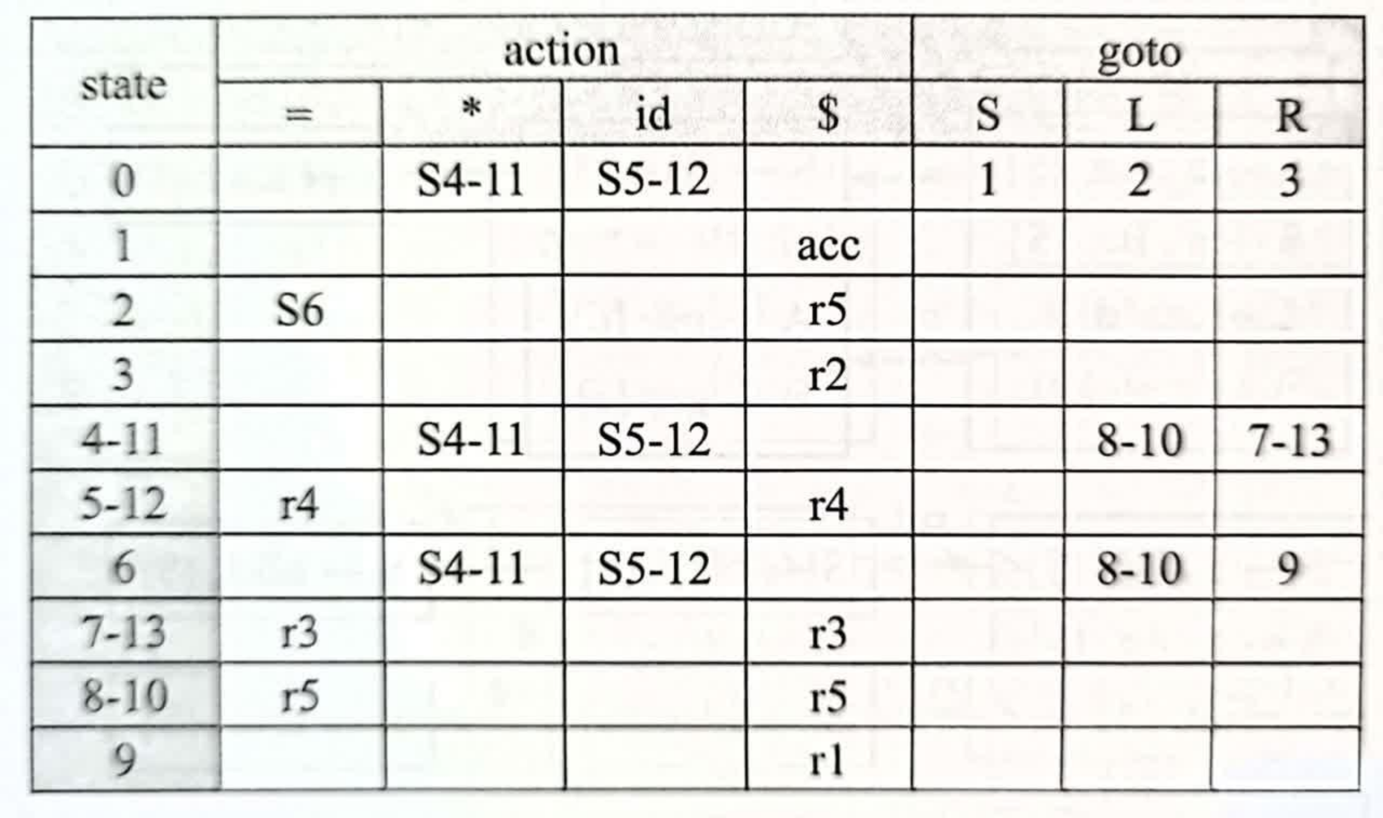
$$\begin{array}{lll} S & \rightarrow & L = R \ \mid \ R \\ L & \rightarrow & *R \ \mid \ \textbf{id} \\ R & \rightarrow & L \end{array}$$



LALR(1) Grammar

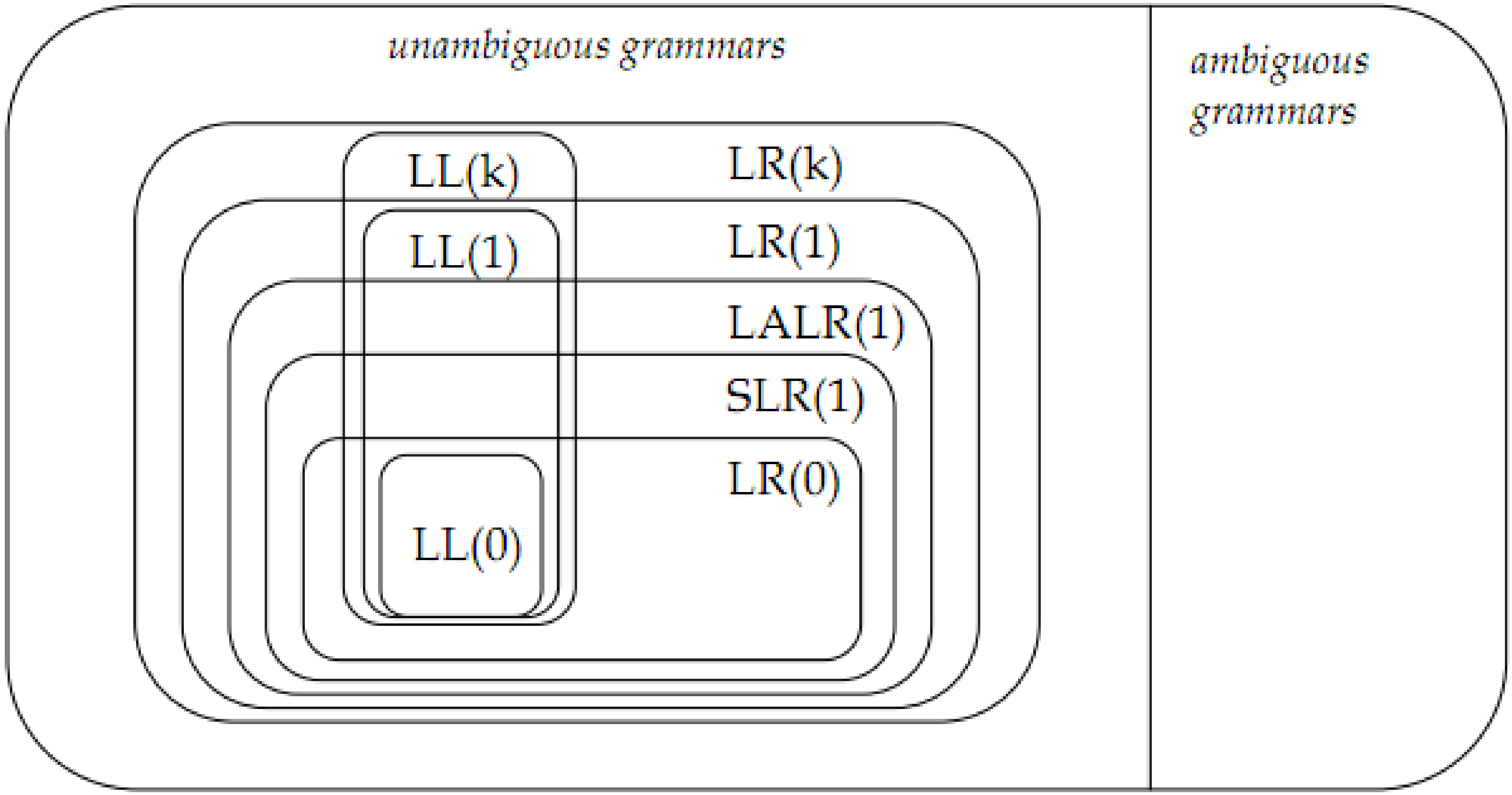
- **Example**
  - **LALR(1) parse table**

$$\begin{array}{lcl} S & \rightarrow & L = R \mid R \\ L & \rightarrow & *R \mid \textbf{id} \\ R & \rightarrow & L \end{array}$$



state	action				goto		
	=	*	id	$	S	L	R
0		S4-11	S5-12		1	2	3
1				acc			
2	S6			r5			
3				r2			
4-11		S4-11	S5-12			8-10	7-13
5-12	r4			r4			
6		S4-11	S5-12			8-10	9
7-13	r3			r3			
8-10	r5			r5			
9				r1			

Hierarchy of Grammar Classes



Ambiguous Grammars

- *Every ambiguous grammar fails to be LR*
- *Every LR grammar is not ambiguous*
- **However, certain types of ambiguous grammars are quite useful in the specification and implementation of languages**
  - **We can specify dis-ambiguating rules that allow only one parse tree for each sentence**
- ***Compilers usually use precedence and associativity to resolve conflicts***

Ambiguous Grammars

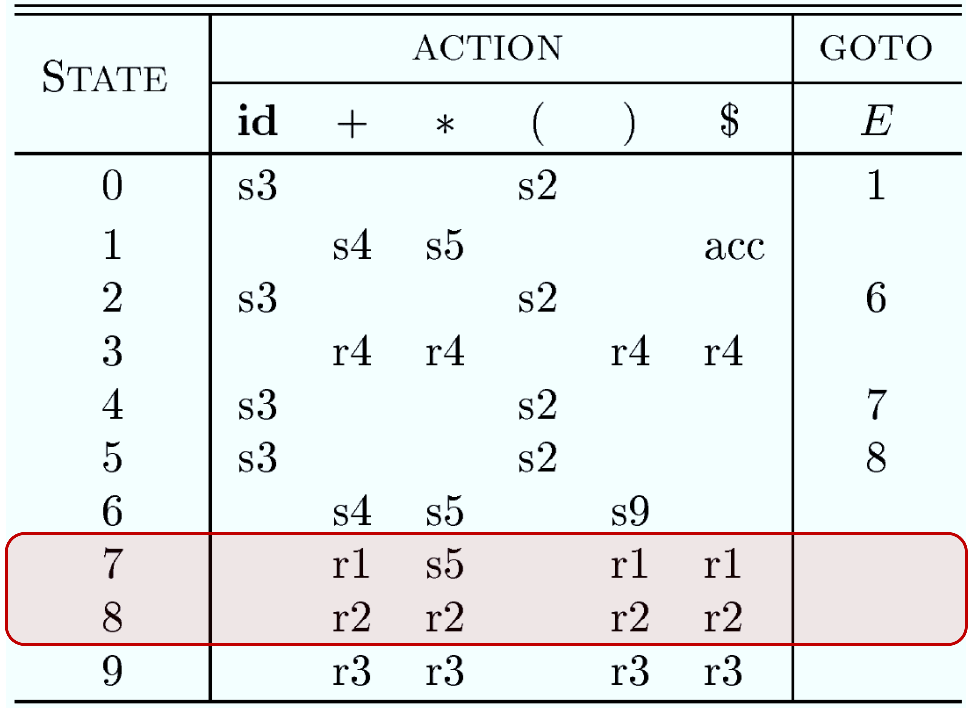
- **Precedence and Associativity to Resolve Conflicts**
- **Example** $E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}$
- ***This grammar is ambiguous because it does not specify the associativity or precedence of the operators + and ****
- This grammar is equivalent to the above grammar, which is not ambiguous

$$\begin{array}{lcl} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T * F \mid F \\ F & \rightarrow & (E) \mid \mathbf{id} \end{array}$$

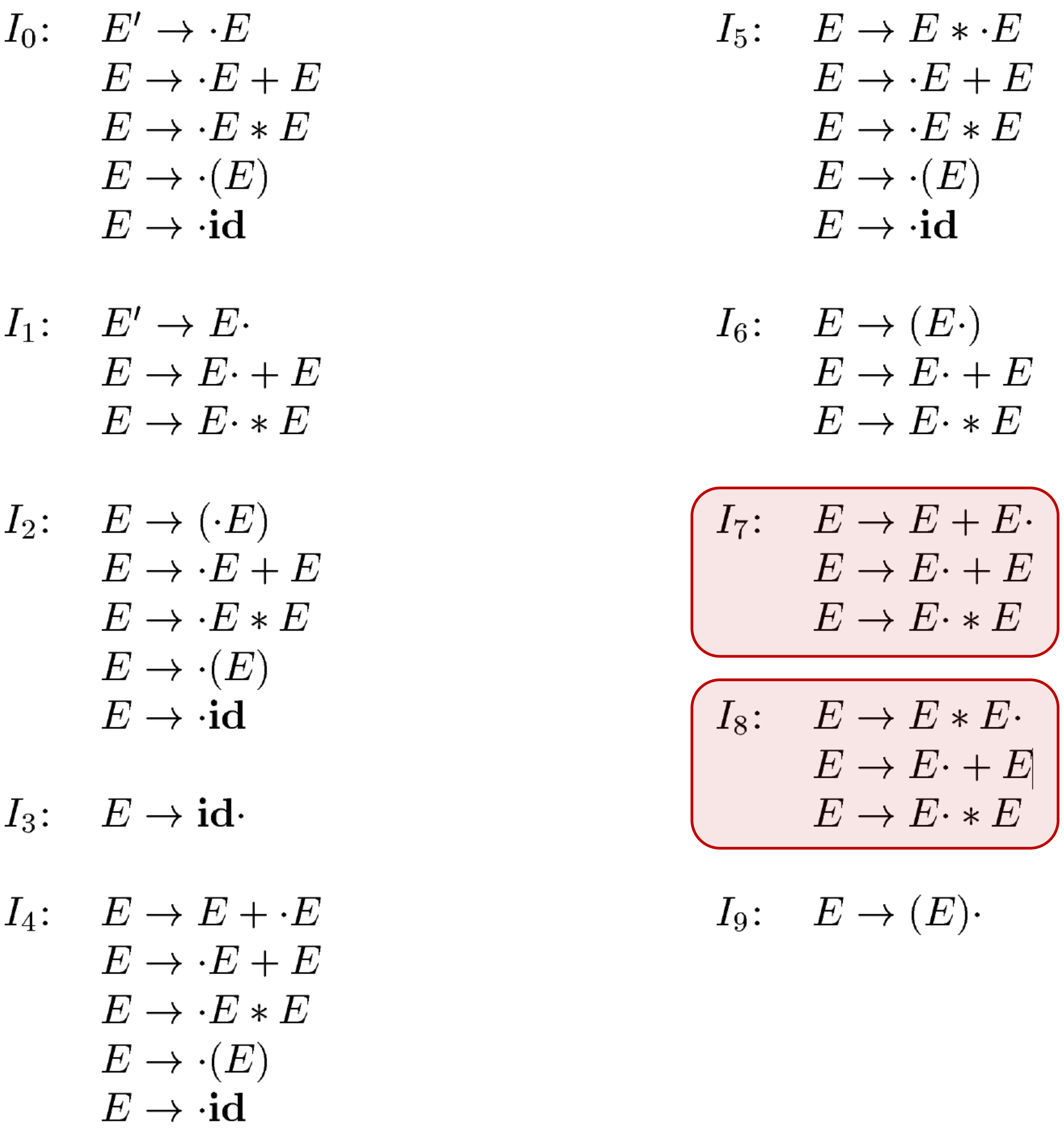
- **There are two reasons why we might prefer to use the ambiguous grammar**
  - We can easily change the associativity and precedence of the operators without disturbing the productions or the number of states in the resulting parser
  - The parser for the ambiguous grammar will not waste time reducing by single productions

Ambiguous Grammar

- **Assuming + and * are left associative and * takes precedence over +**



STATE	ACTION						GOTO
	id	$+$	$*$	(	)	\$	E
0	s3			s2			1
1		s4	s5			acc	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			7
5	s3			s2			8
6		s4	s5		s9		
7		r1	s5		r1	r1	
8		r2	r2		r2	r2	
9		r3	r3		r3	r3	



I_0: $E' \to \cdot E$
$E \to \cdot E + E$
$E \to \cdot E * E$
$E \to \cdot(E)$
$E \to \cdot\mathbf{id}$

I_1: $E' \to E\cdot$
$E \to E\cdot + E$
$E \to E\cdot * E$

I_2: $E \to (\cdot E)$
$E \to \cdot E + E$
$E \to \cdot E * E$
$E \to \cdot(E)$
$E \to \cdot\mathbf{id}$

I_3: $E \to \mathbf{id}\cdot$

I_4: $E \to E + \cdot E$
$E \to \cdot E + E$
$E \to \cdot E * E$
$E \to \cdot(E)$
$E \to \cdot\mathbf{id}$

I_5: $E \to E * \cdot E$
$E \to \cdot E + E$
$E \to \cdot E * E$
$E \to \cdot(E)$
$E \to \cdot\mathbf{id}$

I_6: $E \to (E\cdot)$
$E \to E\cdot + E$
$E \to E\cdot * E$

I_7: $E \to E + E\cdot$
$E \to E\cdot + E$
$E \to E\cdot * E$

I_8: $E \to E * E\cdot$
$E \to E\cdot + E$
$E \to E\cdot * E$

I_9: $E \to (E)\cdot$