Example 4.33. Figure 4.31 shows the parsing action and goto functions of an LR parsing table for the following grammar for arithmetic expressions with binary operators + and ∗ :

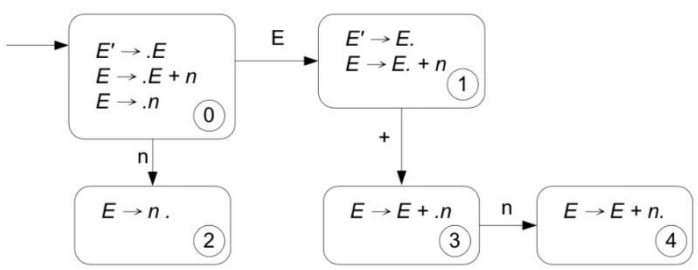
(1) E -> E + T
(2) E -> T
(3) T -> T∗F
(4) T -> F
(5) F -> (E)
(6) F -> **id**

STATE	*action*						*goto*		
	id	+	∗	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Fig. 4.31. Parsing table for expression grammar

STACK	INPUT	ACTION
(1) 0	**id**∗**id** + **id** $	shift
(2) 0 **id** 5	∗**id** + **id** $	reduce by F -> **id**
(3) 0F3	∗**id** + **id** $	reduce by T -> F
(4) 0T2	∗**id** + **id** $	shift
(5) 0T2∗7	**id** + **id** $	shift
(6) 0T2∗7 **id** 5	+ **id** $	reduce by F -> **id**
(7) 0T2∗7 F 10	+ **id** $	reduce by T -> T∗F
(8) 0T2	+ **id** $	reduce by E -> T
(9) 0E1	+ **id** $	shift
(10) 0E1 + 6	**id** $	shift
(11) 0E1 + 6 **id** 5	$	reduce by F -> **id**
(12) 0E1 + 6 F 3	$	reduce by T -> F
(13) 0E1 + 6 T 9	$	E -> E + T
(14) 0E1	$	accept

Figure 4.32 Moves of LR parser on **id**∗**id** + **id**.



Follow (E') = { $ }; Follow (E) = { $, + }

State	Input			Goto
	n	+	$	E
0	s2			1
*1		s3	accept	
2		r ($E \rightarrow n$)	r ($E \rightarrow n$)	
3	s4			
4		r ($E \rightarrow E + n$)	r ($E \rightarrow E + n$)	

GoTo

	+	n	E	$
0		2	1	
1	3			
2				
3		4		
4				

Action

	+	n	$
0		S	
*1	S		A
2	R3		R3
3		S	
4	R2		R2

	Parsing stack	Input	Action
1	$ 0	$n + n + n$ $	shift 2
2	$ 0 n 2	$+ n + n$ $	reduce $E \rightarrow n$
3	$ 0 E 1	$+ n + n$ $	shift 3
4	$ 0 E 1 + 3	$n + n$ $	shift 4
5	$ 0 E 1 + 3 n 4	$+ n$ $	reduce $E \rightarrow E + n$
6	$ 0 E 1	$+ n$ $	shift 3
7	$ 0 E 1 + 3	n $	shift 4
8	$ 0 E 1+ 3 n 4	$	reduce $E \rightarrow E + n$
9	$ 0 E 1	$	accept