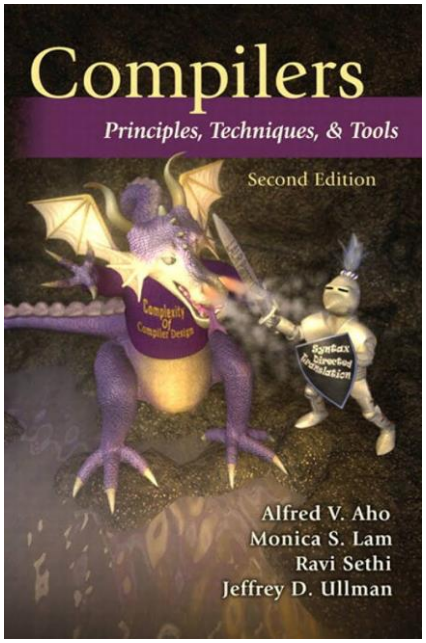
Compiler

Lec 08

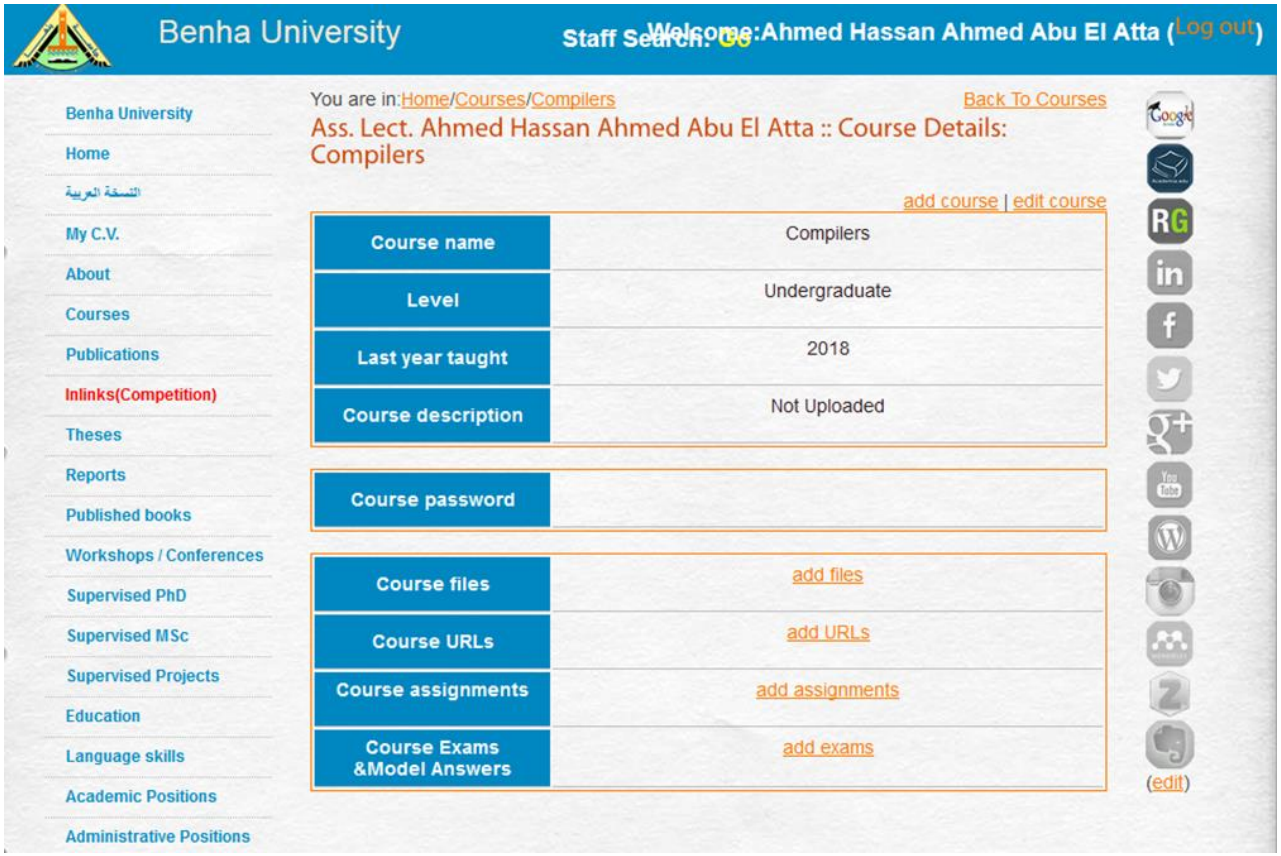
Book

Compilers: Principles, Techniques, and Tools is a computer science textbook by Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman about compiler construction.



PowerPoint

http://www.bu.edu.eg/staff/ahmedaboalatah14-courses/14779



Syntax Analysis

PART V

LR(0) Item

An LR(0) item of G is a production of G with the dot at some position of the body:

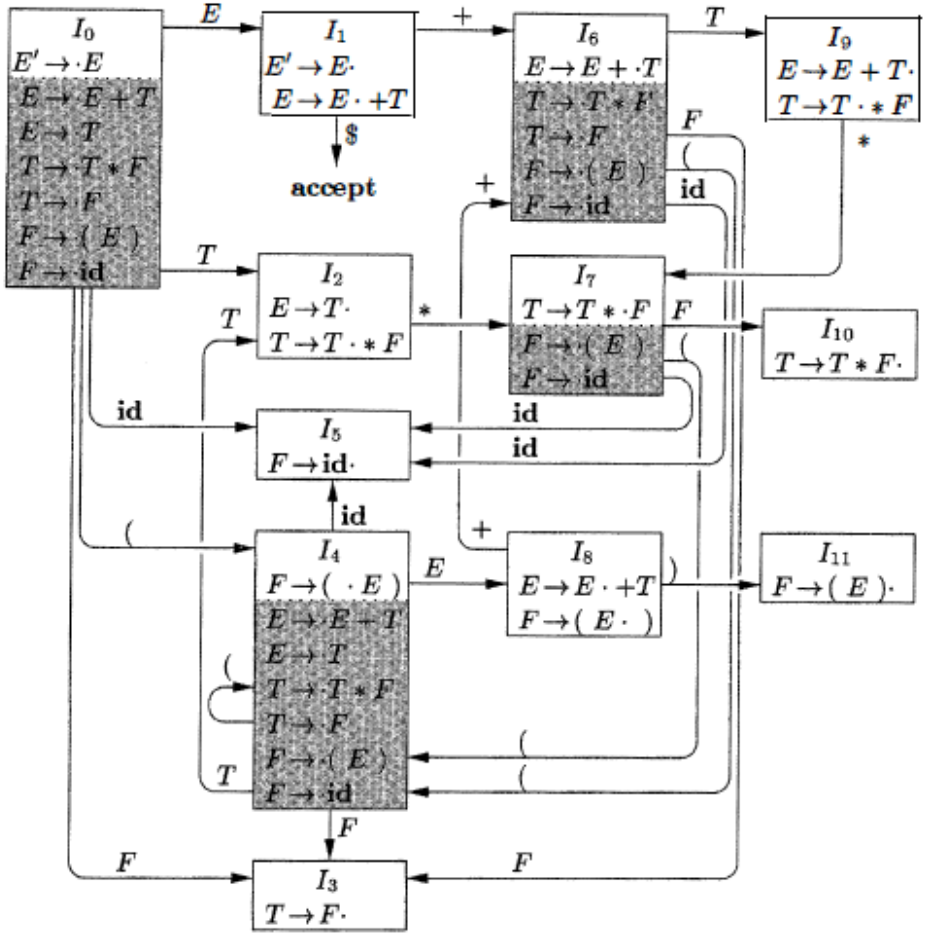
- For A->XYZ we have following items
  - **A->.XYZ**
  - **A->X.YZ**
  - **A->XY.Z**
  - **A->XYZ.**
- In a state having A->.XYZ we hope to see a string derivable from XYZ next on the input.
- The production A->ϵ generates only one item, A -> .

Closure of Item Sets

If I is a set of items for a grammar G, then Closure(I) is the set of items constructed from I by the two rules:

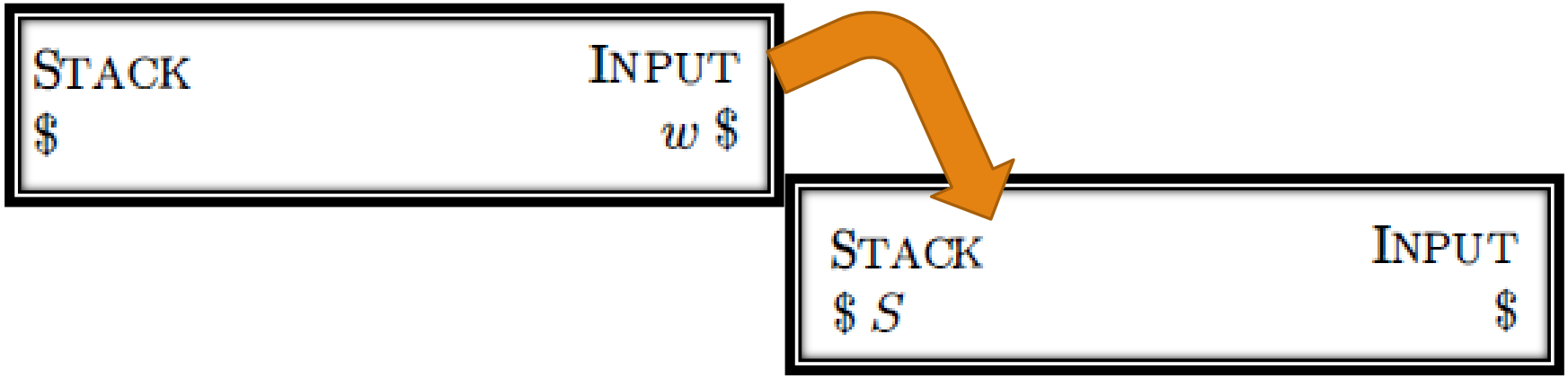
1. Initially, add every item in I to CLOSURE(I).
2. If $A \to \alpha \cdot B\beta$ is in CLOSURE(I) and $B \to \gamma$ is a production, then add the item $B \to \cdot\gamma$ to CLOSURE(I), if it is not already there. Apply this rule until no more new items can be added to CLOSURE(I).

LR(0) Automaton



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

$S' \rightarrow S.$



To construct the canonical LR(O) collection for a grammar, we define an augmented grammar

Augmented Grammar

If G is a grammar with start symbol S, then G' , the augmented grammar for G, is G with a new start symbol S' and production S'→S

The purpose of this new starting production is to indicate to the parser when it should stop parsing and announce acceptance of the input.

That is, acceptance occurs when and only when the parser is about to reduce by S'→S.

Example LR(1)

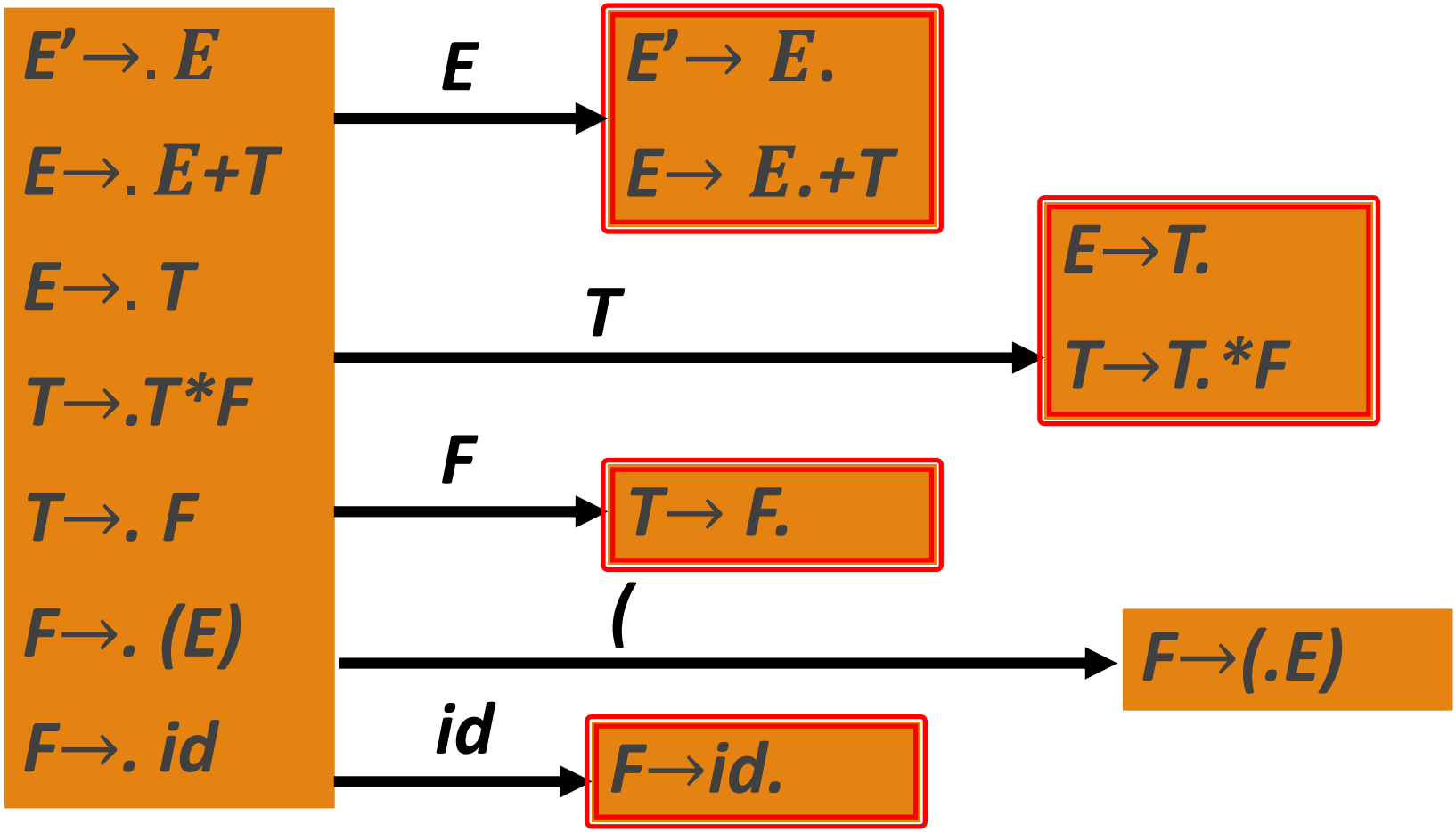
$$
\begin{aligned}
E &\rightarrow E + T \mid T \\
T &\rightarrow T * F \mid F \\
F &\rightarrow (E) \mid \mathbf{id}
\end{aligned}
$$

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

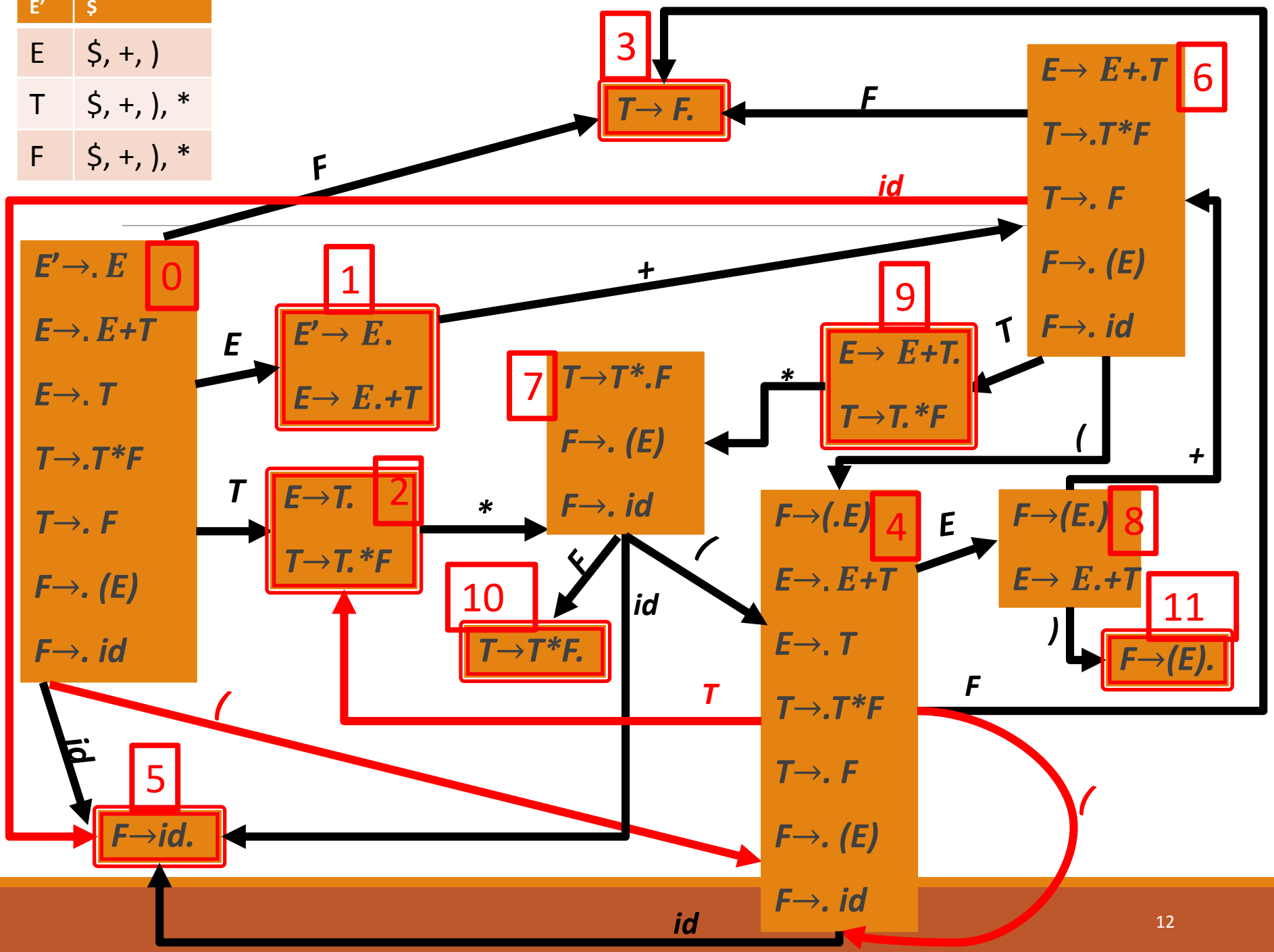
E'→. E

E'→. E
E→. E+T
E→. T
T→.T*F
T→. F
F→. (E)
F→. id

Example



E'	$
E	$, +,)
T	$, +,), *
F	$, +,), *


0
E'→. E
E→. E+T
E→. T
T→. T*F
T→. F
F→. (E)
F→. id
1
E'→ E.
E→ E.+T
2
E→T.
T→T.*F
3
T→ F.
4
F→(.E)
E→. E+T
E→. T
T→. T*F
T→. F
F→. (E)
F→. id
5
F→id.
6
E→ E+.T
T→. T*F
T→. F
F→. (E)
F→. id
7
T→T*.F
F→. (E)
F→. id
8
F→(E.)
E→ E.+T
9
E→ E+T.
T→T.*F
10
T→T*F.
11
F→(E).
E
T
F
id
+
*
(
)
id
id

Example LR(1)

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			

Moves of an LR parser on id * id + id

	STACK	SYMBOLS	INPUT	ACTION
(1)	0		**id** $*$ **id** $+$ **id** \$	shift
(2)	0 5	**id**	$*$ **id** $+$ **id** \$	reduce by $F \to$ **id**
(3)	0 3	F	$*$ **id** $+$ **id** \$	reduce by $T \to F$
(4)	0 2	T	$*$ **id** $+$ **id** \$	shift
(5)	0 2 7	$T*$	**id** $+$ **id** \$	shift
(6)	0 2 7 5	$T*$ **id**	$+$ **id** \$	reduce by $F \to$ **id**
(7)	0 2 7 10	$T*F$	$+$ **id** \$	reduce by $T \to T*F$
(8)	0 2	T	$+$ **id** \$	reduce by $E \to T$
(9)	0 1	E	$+$ **id** \$	shift
(10)	0 1 6	$E+$	**id** \$	shift
(11)	0 1 6 5	$E+$ **id**	\$	reduce by $F \to$ **id**
(12)	0 1 6 3	$E+F$	\$	reduce by $T \to F$
(13)	0 1 6 9	$E+T$	\$	reduce by $E \to E+T$
(14)	0 1	E	\$	accept