Precedence and Associativity to Resolve conflicts

Consider the Grammar

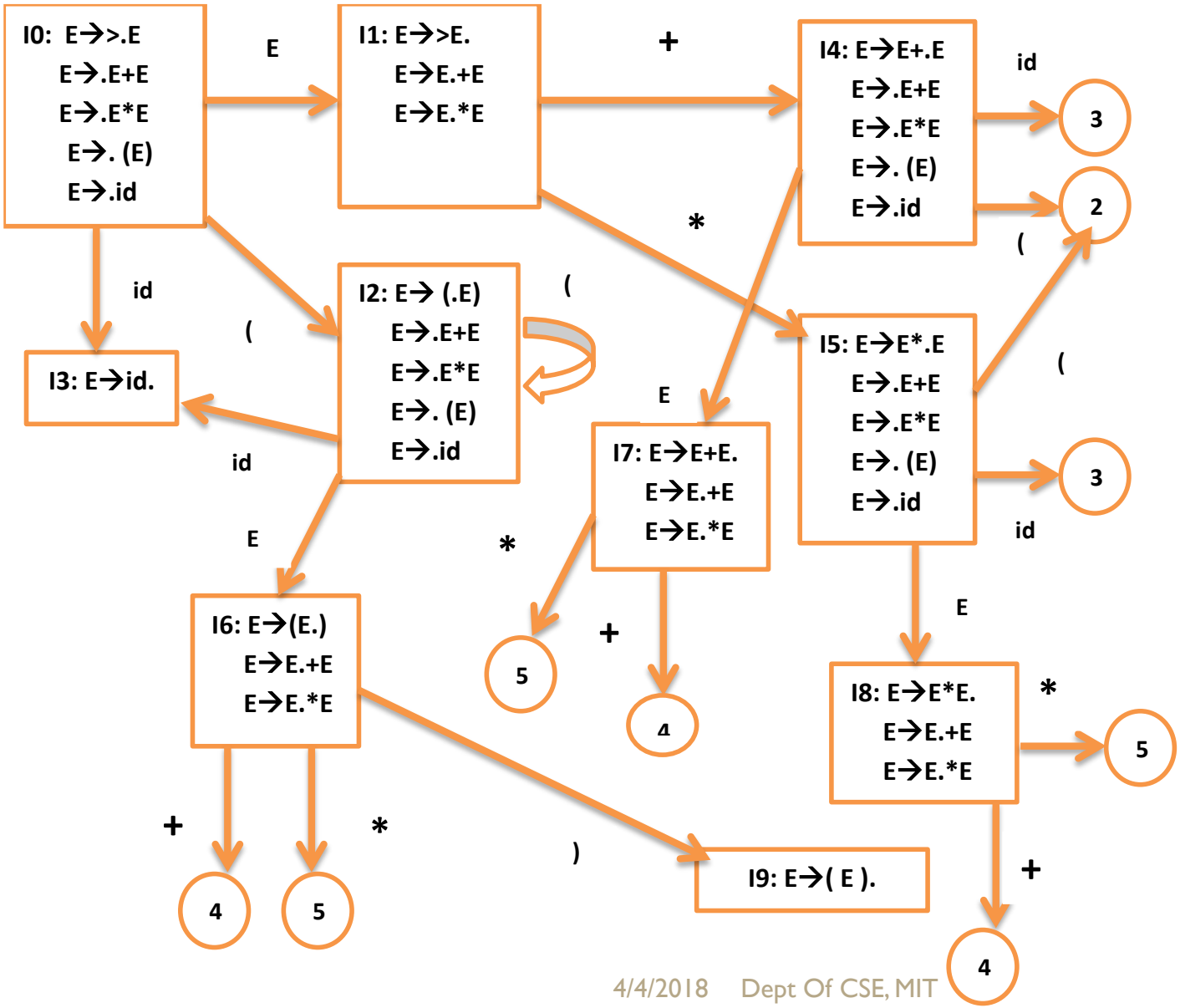
E→E+E

E→E*E

E→(E)

E→id

DFA:


I0: E→>.E
E→.E+E
E→.E*E
E→. (E)
E→.id
E
I1: E→>E.
E→E.+E
E→E.*E
+
I4: E→E+.E
E→.E+E
E→.E*E
E→. (E)
E→.id
id
3
2
(
*
id
(
I2: E→ (.E)
E→.E+E
E→.E*E
E→. (E)
E→.id
(
I3: E→id.
id
E
I5: E→E*.E
E→.E+E
E→.E*E
E→. (E)
E→.id
(
3
id
E
I7: E→E+E.
E→E.+E
E→E.*E
*
+
5
4
I6: E→(E.)
E→E.+E
E→E.*E
E
I8: E→E*E.
E→E.+E
E→E.*E
*
5
+
4
+
*
)
I9: E→(E).

Initial Parse Table

State	id	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			accept	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			7
5	s3			s2			8
6		s4	s5		s9		
7		r1,s4	r1,s5		r1	r1	
8		r2,s4	r2,s5		r2	r2	
9		r3	r3		r3	r3	

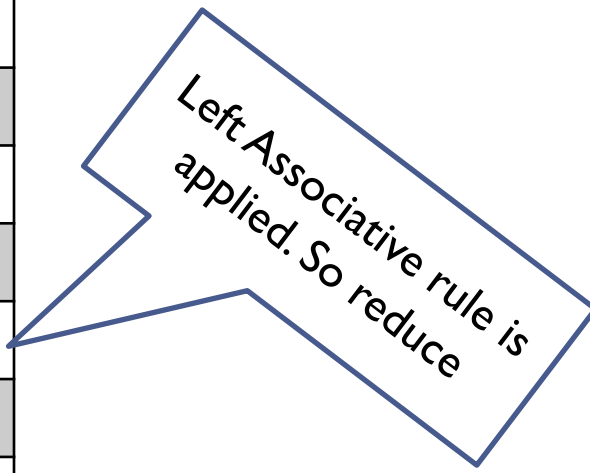
How to resolve conflict??

- Conflict occurs in state 7 and 8.
- Consider four different cases
  - id+id+id
  - id*id*id
  - id+id*id
  - id*id+id
- Solution: In case of '+' and '*' give more precedence to *

  In case of '+' and '+' or '*' and '*' go for left associativity.

Case1: id+id+id

Stack	Symbol	Input	Action
0		id+id+id$	shift
03	id	+id+id$	reduce; e→id
01	E	+id+id$	shift
014	E+	+id+id$	shift
0143	E+id	+id$	reduce; e→id
0147	E+E	+id$	conflict; reduce
01	E	+id$	shift
014	E+	id$	shift
0143	E+id	$	reduce; e→id
0147	E+E	$	reduce; e→e+e
01	E	$	accept

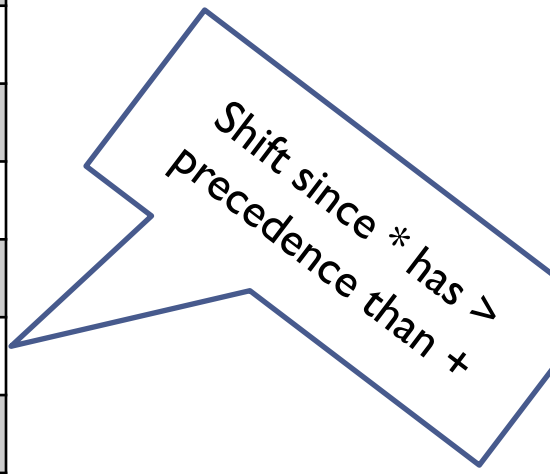


Parse table will be

State	id	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			accept	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			7
5	s3			s2			8
6		s4	s5		s9		
7		r1	r1,s5		r1	r1	
8		r2,s4	r2,s5		r2	r2	
9		r3	r3		r3	r3	

Case 2: id+id*id

Stack	Symbol	Input	Action
0		id+id*id$	shift
03	id	+id*id$	reduce; e→id
01	E	+id*id$	shift
014	E+	Id*id$	shift
0143	E+id	*id$	reduce; e→id
0147	E+E	*id$	conflict; Shift
01475	E+E*	id$	shift
014753	E+E*id	$	Reduce E→id
014758	E+E*E	$	reduce; E→E*E
0147	E+E	$	reduce; E→E+E
01	E	$	accept

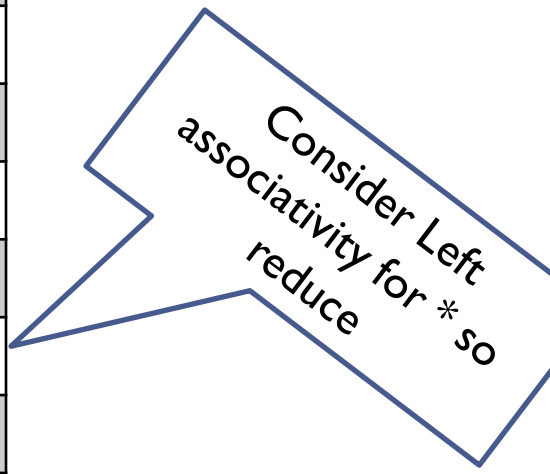


Parse table will be

State	id	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			accept	
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,s4	r2,s5		r2	r2	
9		r3	r3		r3	r3	

Case 3: id*id*id

Stack	Symbol	Input	Action
0		id*id*id$	Shift
03	id	*id*id$	reduce; E→id
01	E	*id*id$	shift
015	E*	id*id$	shift
0153	E*id	*id$	reduce; E→id
0158	E*E	*id$	conflict; Reduce
01	E	*id$	shift
015	E*	id$	shift
0153	E*id	$	reduce; E→id
0158	E*E	$	reduce; E→E*E
01	E	$	accept

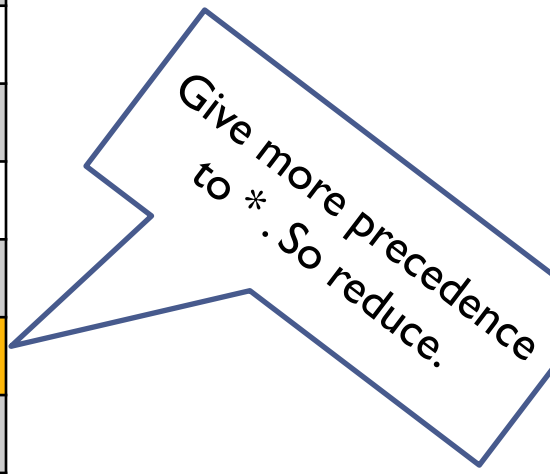


Parse table will be

State	id	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			accept	
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,s4	r2		r2	r2	
9		r3	r3		r3	r3	

Case 3: id * id + id

Stack	Symbol	Input	Action
0		id*id+id$	Shift
03	id	*id+id$	reduce; E→id
01	E	*id+id$	shift
015	E*	id+id$	shift
0153	E*id	+id$	reduce; E→id
0158	E*E	+id$	conflict; Reduce
01	E	+id$	shift
014	E+	id$	shift
0143	E+id	$	reduce; E→id
0147	E+E	$	reduce; E→E+E
01	E	$	accept



Parse table will be

State	id	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			accept	
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	

The "Dangling-Else" ambiguity

$$
\begin{array}{lll}
\textit{stmt} & \rightarrow & \textbf{if } \textit{expr} \textbf{ then } \textit{stmt} \textbf{ else } \textit{stmt} \\
& | & \textbf{if } \textit{expr} \textbf{ then } \textit{stmt} \\
& | & \textbf{other}
\end{array}
$$

Simplifying the above grammar

$$
\begin{array}{lll}
S' & \rightarrow & S \\
S & \rightarrow & i\ S\ e\ S \mid i\ S \mid a
\end{array}
$$

where,

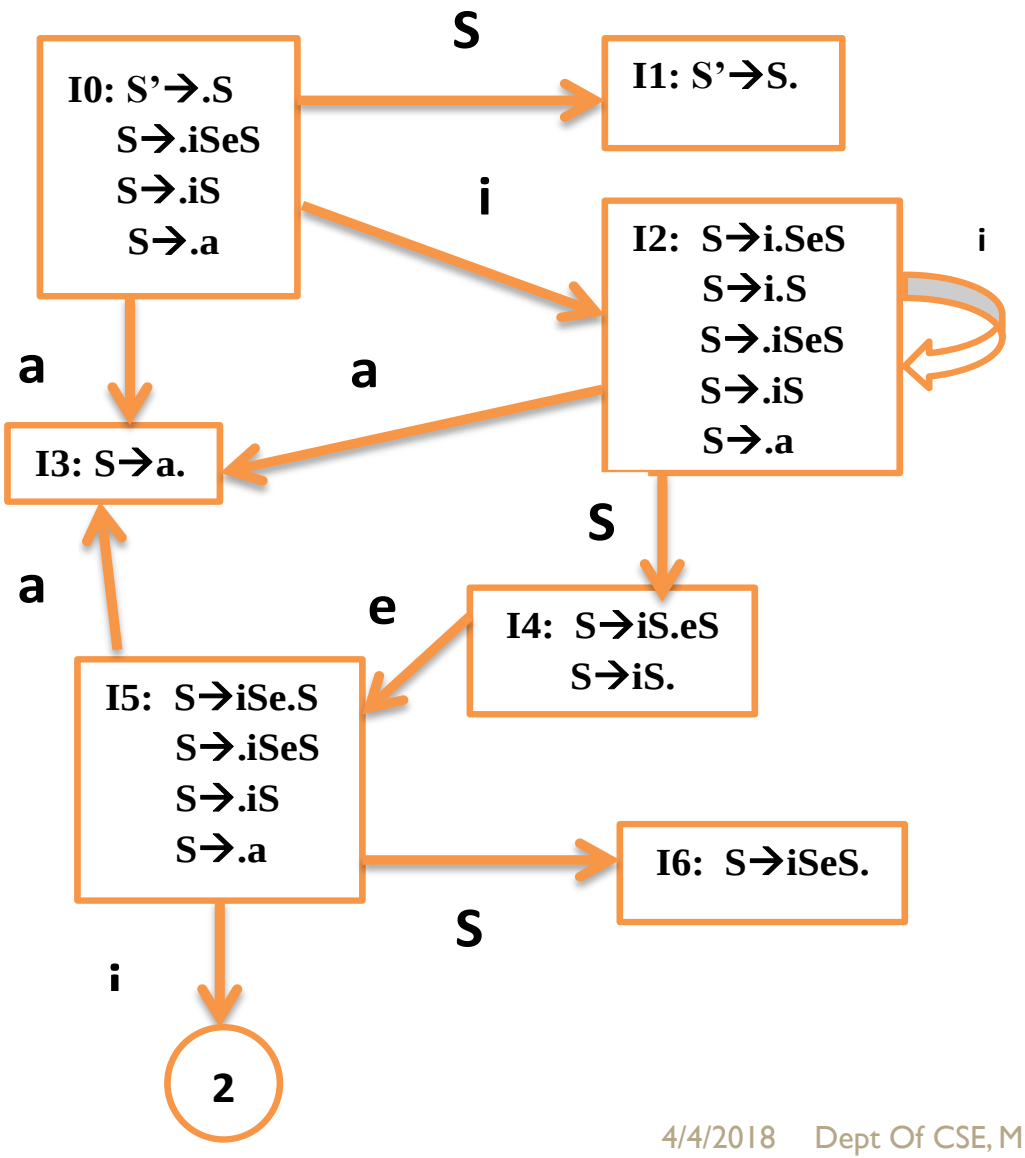
i - if expr then

e- else

a- all other statements

S- stmt

LR(0) automaton



Parse table

States	i	e	a	$	S
0	s2				
1				accept	
2	s2				4
3		r3		r3	
4		s5,r2		r2	
5	s2		s3		
6		r1		r1	

Conflict; So use disambiguating rule to match else with nearest unmatched then. Hence Shift

Resulting Parse table

States	i	e	a	$	S
0	s2		s3		1
1				accept	
2	s2		s3		4
3		r3		r3	
4		S5		r2	
5	s2		s3		
6		r1		r1	

Parsing Actions for "iiaea"

Stack	Symbols	Input	Action
0		iiaea$	Shift
02	I	iaea$	Shift
022	ii	aea$	Shift
0223	iia	ea$	Reduce S→a
0224	iiS	ea$	Shift
02245	iiSe	a$	Shift
022453	iiSea	$	S→a
022456	iiSes	$	S→iSeS
024	iS	$	S→iS
01	S	$	accept