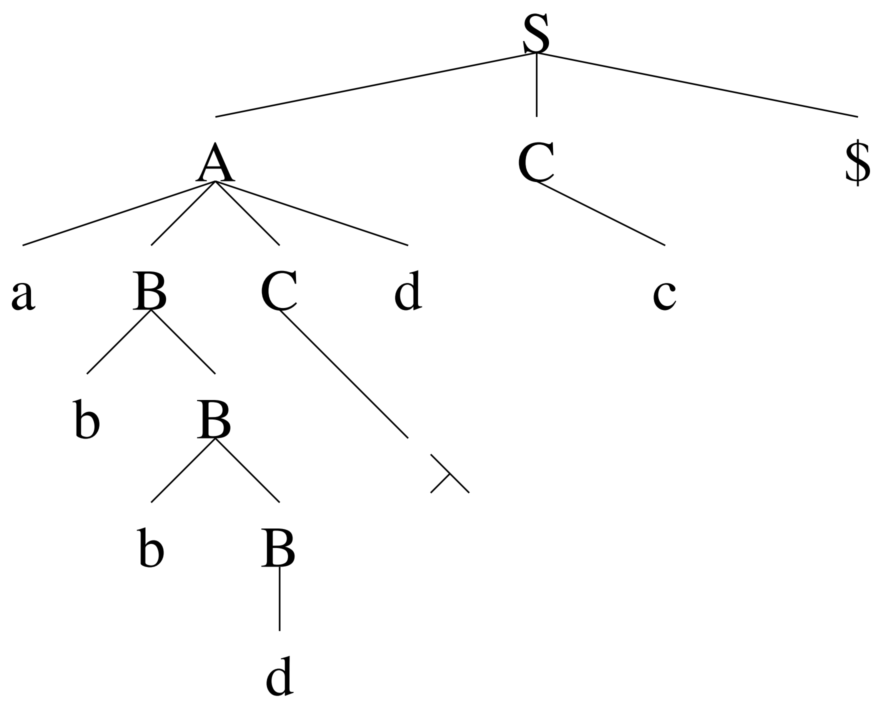
Bottom-up parsing



$$
\begin{array}{llcl}
\boxed{1} & \textbf{S} & \rightarrow & \textbf{A C \$} \\
\boxed{2} & \textbf{C} & \rightarrow & \textbf{c} \\
\boxed{3} & & | & \lambda \\
\boxed{4} & \textbf{A} & \rightarrow & \textbf{a B C d} \\
\boxed{5} & & | & \textbf{B Q} \\
\boxed{6} & & | & \lambda \\
\boxed{7} & \textbf{B} & \rightarrow & \textbf{b B} \\
\boxed{8} & & | & \textbf{d} \\
\boxed{9} & \textbf{Q} & \rightarrow & \textbf{q}
\end{array}
$$

$$
\begin{array}{lll}
\textbf{S} & \Rightarrow & \textbf{A}\ \mathit{C}\ \textbf{\$} \\
 & \Rightarrow & \mathit{A}\ \boxed{\textbf{c}}\ \textbf{\$} \\
 & \Rightarrow & \boxed{\textbf{a B}\ \mathit{C}\ \textbf{d}}\ \textbf{c \$} \\
 & \Rightarrow & \textbf{a}\ \mathit{B}\ \boxed{}\ \textbf{d c \$} \\
 & \Rightarrow & \textbf{a}\ \boxed{\textbf{b}\ \mathit{B}}\ \textbf{d c \$} \\
 & \Rightarrow & \textbf{a b}\ \boxed{\textbf{b}\ \mathit{B}}\ \textbf{d c \$} \\
 & \Rightarrow & \textbf{a b b}\ \boxed{\textbf{d}}\ \textbf{d c \$}
\end{array}
$$

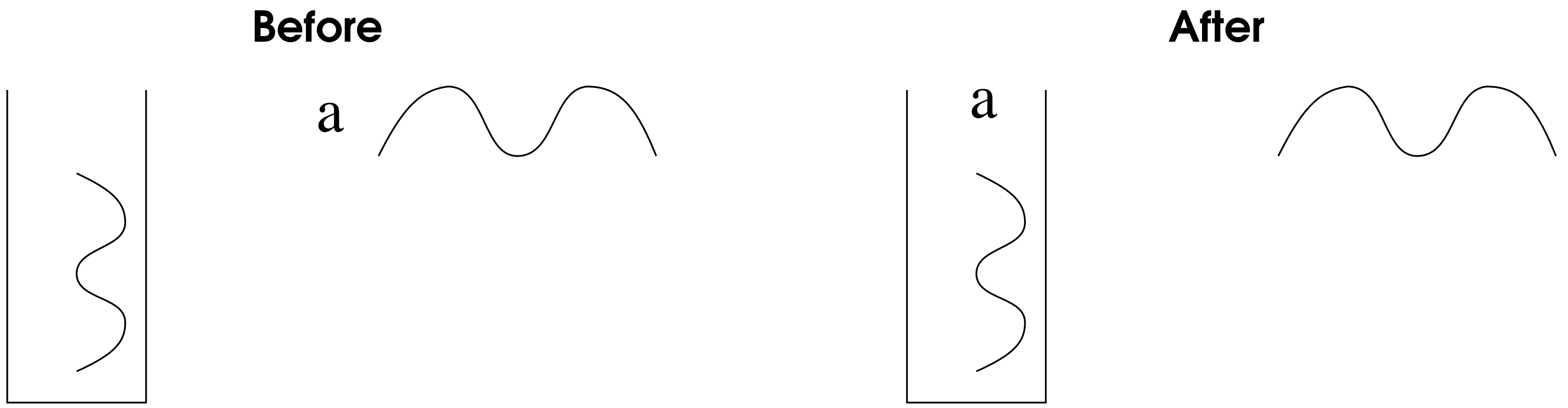
A bottom-up parse is essentially a rightmost derivation, run in *reverse*. Instead of replacing a nonterminal by a string, we recognize the string as *reducing* to the nonterminal.

The parsing engine issues the following instructions:

shift: a symbol is moved from input to top-of-stack.

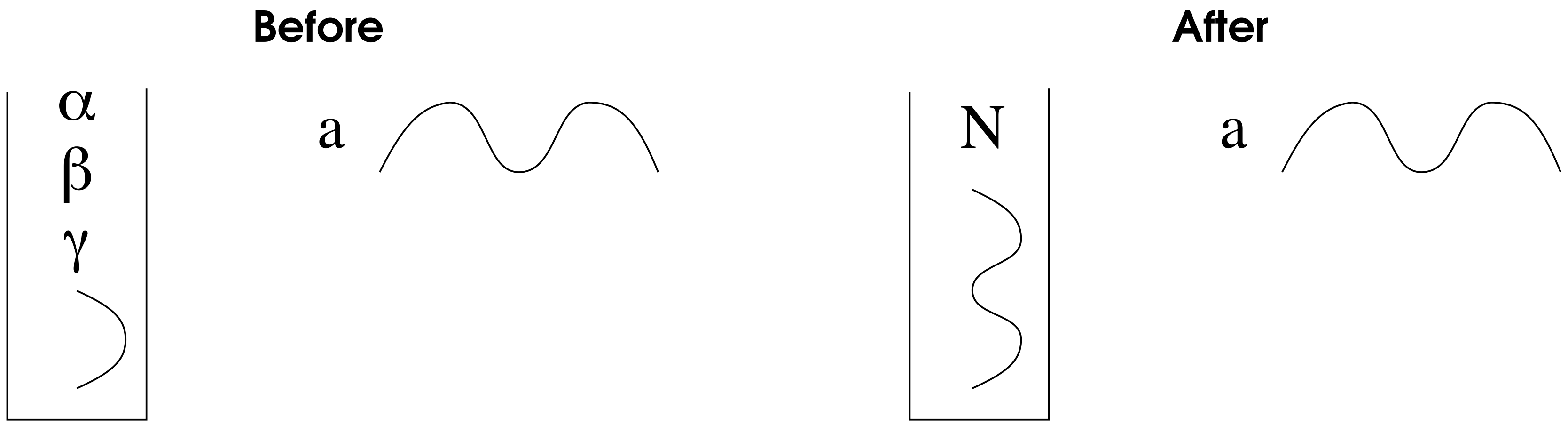
reduce r**:** the stack is modified by applying rule r.

Shift



- Like the top-down parser, the bottom-up parser checks for errors on a shift. The parse table we shall construct indicates when a shift is error-free.
- Actually, instead of pushing a symbol onto the stack, we push a *state*, which indexes the parse table and represents the current possibilities of the parse.

Reduce



- If the rule applied is $N \rightarrow \omega$, where ω has m symbols, then m symbols are popped off the stack, and a symbol representing N is pushed.
- It's important to remember that a canonical parse can perform reductions *only* at the top-of-stack.

Rightmost derivation in reverse – slow motion

Stack	Input	Activity
	a b b d d c $	Shift
a	b b d d c $	Shift
a b	b d d c $	Shift
a b b	d d c $	Shift
a b b d	d c $	Reduce $B \to d$
a b b B	d c $	Reduce $B \to bB$
a b B	d c $	Reduce $B \to bB$
a B	d c $	Reduce $C \to \lambda$
a B C	d c $	Shift
a B C d	c $	Reduce $A \to aBCd$
A	c $	Shift
A c	$	Reduce $C \to c$
A C	$	Shift
A C $		Reduce $S \to AC\$$
S		Accept

This is LR-style parsing: a scan from the left that produces a rightmost derivation.

We could have tried to apply $C \to \lambda$ at any point during the parse, but most would not have made progress toward an accept. Where parse table construction is successful, the table directs the parse towards an accept if one is possible.

LR table construction

Each state of the parser represents parsing possibilities after processing a given prefix of the input string.

To construct the canonical $LR(0)$ set of states:

1. Each state begins with a *kernel* that represents progress through certain rules of the grammar:

(3)	X	→	y	•	z
	W	→	x z y	•	A
	F	→	a B C y	•	

   The dot (•) shows the progress through the rule achieved by moving into this state.
2. When • is next to a nonterminal, we must add into this state the *closure* by expanding all rules of the nonterminal:

(3)	A	→	•	b c d
	A	→	•	z A

We then label each component of the state with an action, indicating transfer to some other state, reduction by a rule, or accept:

(3)	X	→	y	•	z	Goto State	17
	W	→	x z y	•	A	Goto State	5
	F	→	a B C y	•		Reduce by rule	$\boxed{5}$
	A	→		•	b c d	Goto State	2
	A	→		•	z A	Goto State	17

which may create a new state:

(17)	X	→	y z	•		Reduce by rule	$\boxed{10}$
	A	→	z	•	A	Goto State	1
	A	→		•	b c d	Goto State	2
	A	→		•	z A	Goto State	18

Table construction

(1)	S	→	• A C $	Goto State 2
	A	→	• a B C d	Goto State 3
		|	• B Q	Goto State 4
		|	•	Reduce by rule 6
	B	→	• b B	Goto State 5
		|	• d	Goto State 6

(2)	S	→	A • C $	Goto State 7
	C	→	• c	Goto State 8
		|	•	Reduce by rule 3

(3)	A	→	a • B C d	Goto State 9
	B	→	• b B	Goto State 5
		|	• d	Goto State 6

(4)	A	→	B • Q	Goto State 10
	Q	→	• q	Goto State 11

(5)	B	→	b • B	Goto State 12
	B	→	• b B	Goto State 5
		|	• d	Goto State 6

(6)	B	→	d •	Reduce by rule 8

(7)	S	→	A C • $	Goto State 13

(8)	C	→	c •	Reduce by rule 2

(9)	A	→	a B • C d	Goto State 14
	C	→	• c	Goto State 8
		|	•	Reduce by rule 3

(10)	A	→	B Q •	Reduce by rule 5

(11)	Q	→	q •	Reduce by rule 9

(12)	B	→	b B •	Reduce by rule 7

(13)	S	→	A C $ •	☺

(14)	A	→	a B C • d	Goto State 15

(15)	A	→	a B C d •	Reduce by rule 4

Conflict resolution

$\boxed{1}$	S	→	A C $
$\boxed{2}$	C	→	c
$\boxed{3}$		|	λ
$\boxed{4}$	A	→	a B C d
$\boxed{5}$		|	B Q
$\boxed{6}$		|	λ
$\boxed{7}$	B	→	b B
$\boxed{8}$		|	d
$\boxed{9}$	Q	→	q

	First	Follow
S	$\{a, b, d, c, \$\}$	$\{\ \}$
A	$\{a, b, d, \lambda\}$	$\{c, \$\}$
B	$\{b, d\}$	$\{c, d, q\}$
C	$\{c, \lambda\}$	$\{d, \$\}$
Q	$\{q\}$	$\{c, \$\}$

Within a state, how do we resolve whether to shift or reduce when either action seems appropriate?

(1)	S	→	• A C $	Goto State 2
	A	→	• a B C d	Goto State 3
		|	• B Q	Goto State 4
		|	•	Reduce by rule $\boxed{6}$
	B	→	• b B	Goto State 5
		|	• d	Goto State 6

Examining the $Follow$ information shows that only those input symbols in $\{c, \$\}$ can follow an A. In state (1) we therefore Reduce by rule $\boxed{6}$ **only when "c" or "$" appears next in the input. Since these symbols are disjoint from the input symbols that cause shifts into other states ($\{a, b, d\}$), we can resolve the apparent conflict.**

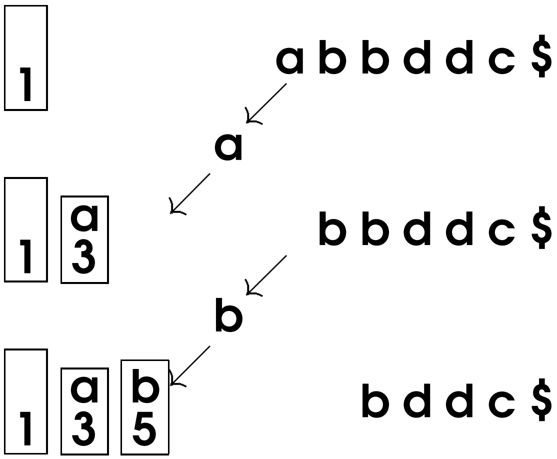
In general, a state might have an apparent shift/reduce or reduce/reduce conflict. The more expensive table construction methods generally provide better conflict resolution.

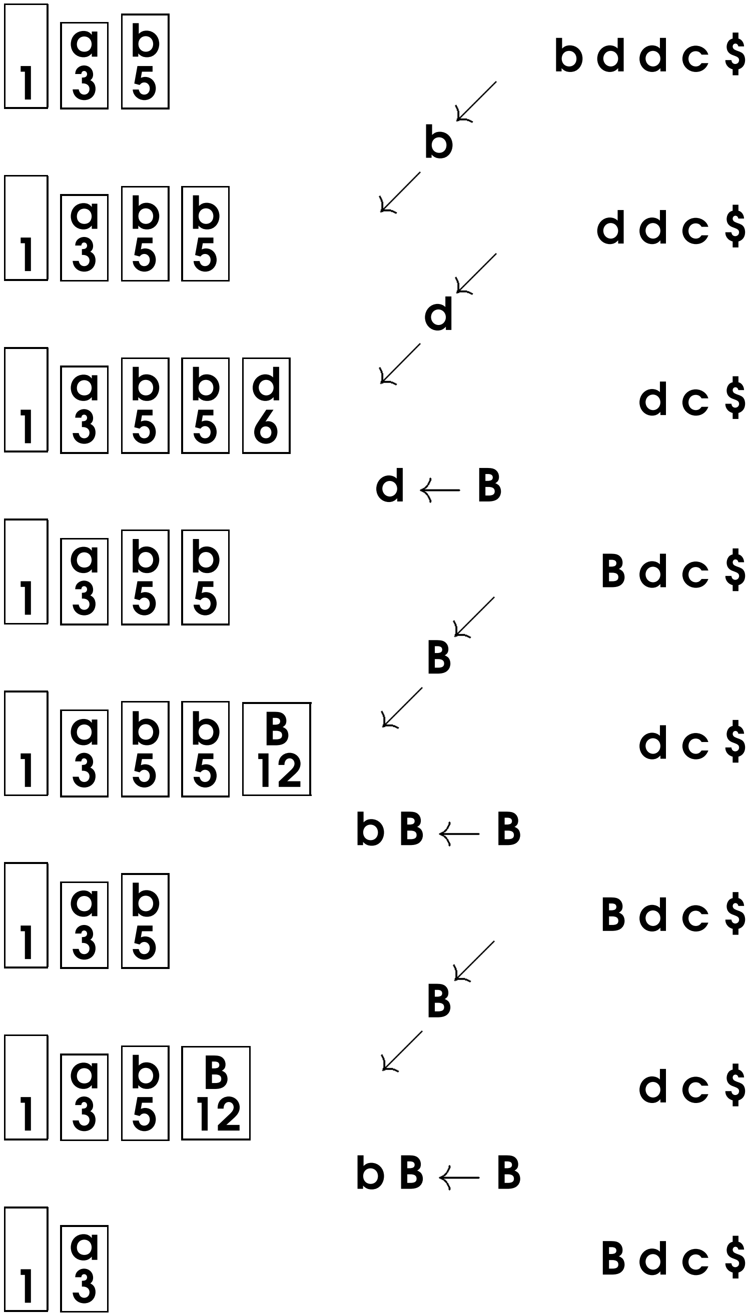
Table for our example

State	a	b	c	d	q	$	A	B	C	Q
1	Goto State 3	Goto State 5	Reduce by rule $\boxed{6}$	Goto State 6		Reduce by rule $\boxed{6}$	Goto State 2	Goto State 4		
2			Goto State 8	Reduce by rule $\boxed{3}$		Reduce by rule $\boxed{3}$			Goto State 7	
3		Goto State 5		Goto State 6				Goto State 9		
4					Goto State 11					Goto State 10
5		Goto State 5		Goto State 6				Goto State 12		
6	Reduce by rule $\boxed{8}$	• •	• • •	• • •	• • •	• • •	• • •	• • •	• •	Reduce by rule $\boxed{8}$
7						Goto State 13				
8	Reduce by rule $\boxed{2}$	• •	• • •	• • •	• • •	• • •	• • •	• • •	• •	Reduce by rule $\boxed{2}$
9			Goto State 8	Reduce by rule $\boxed{3}$		Reduce by rule $\boxed{3}$			Goto State 14	
10	Reduce by rule $\boxed{5}$	• •	• • •	• • •	• • •	• • •	• • •	• • •	• •	Reduce by rule $\boxed{5}$
11	Reduce by rule $\boxed{9}$	• •	• • •	• • •	• • •	• • •	• • •	• • •	• •	Reduce by rule $\boxed{9}$
12	Reduce by rule $\boxed{7}$	• •	• • •	• • •	• • •	• • •	• • •	• • •	• •	Reduce by rule $\boxed{7}$
13	☺	• •	• • •	• • •	• • •	• • •	• • •	• • •	• •	☺
14				Goto State 15						
15	Reduce by rule $\boxed{4}$	• •	• • •	• • •	• • •	• • •	• • •	• • •	• •	Reduce by rule $\boxed{4}$

Using the table

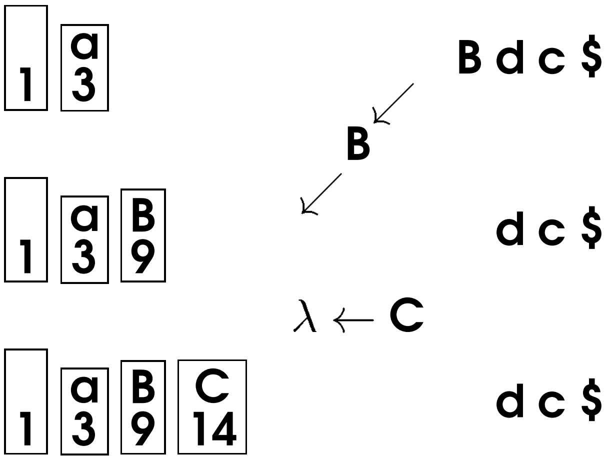
State	a	b	c	d	q	$	A	B	C	Q
1	3	5	$\boxed{6}$	6		$\boxed{6}$	2	4		
2			8	$\boxed{3}$		$\boxed{3}$			7	
3		5		6				9		
4					11					10
5		5		6				12		
6	$\boxed{8}$	. . . .								$\boxed{8}$
7						13				
8	$\boxed{2}$	. . . .								$\boxed{2}$
9			8	$\boxed{3}$		$\boxed{3}$			14	
10	$\boxed{5}$	. . . .								$\boxed{5}$
11	$\boxed{9}$	. . . .								$\boxed{9}$
12	$\boxed{7}$	. . . .								$\boxed{7}$
13	☺	. . . .								☺
14				15						
15	$\boxed{4}$	. . . .								$\boxed{4}$





Using the table (cont'd)

State	a	b	c	d	q	$	A	B	C	Q
1	3	5	$\boxed{6}$	6		$\boxed{6}$	2	4		
2			8	$\boxed{3}$		$\boxed{3}$			7	
3		5		6				9		
4					11					10
5		5		6				12		
6	$\boxed{8}$	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	$\boxed{8}$
7						13				
8	$\boxed{2}$	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	$\boxed{2}$
9			8	$\boxed{3}$		$\boxed{3}$			14	
10	$\boxed{5}$	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	$\boxed{5}$
11	$\boxed{9}$	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	$\boxed{9}$
12	$\boxed{7}$	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	$\boxed{7}$
13	☺	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	☺
14				15						
15	$\boxed{4}$	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	$\boxed{4}$



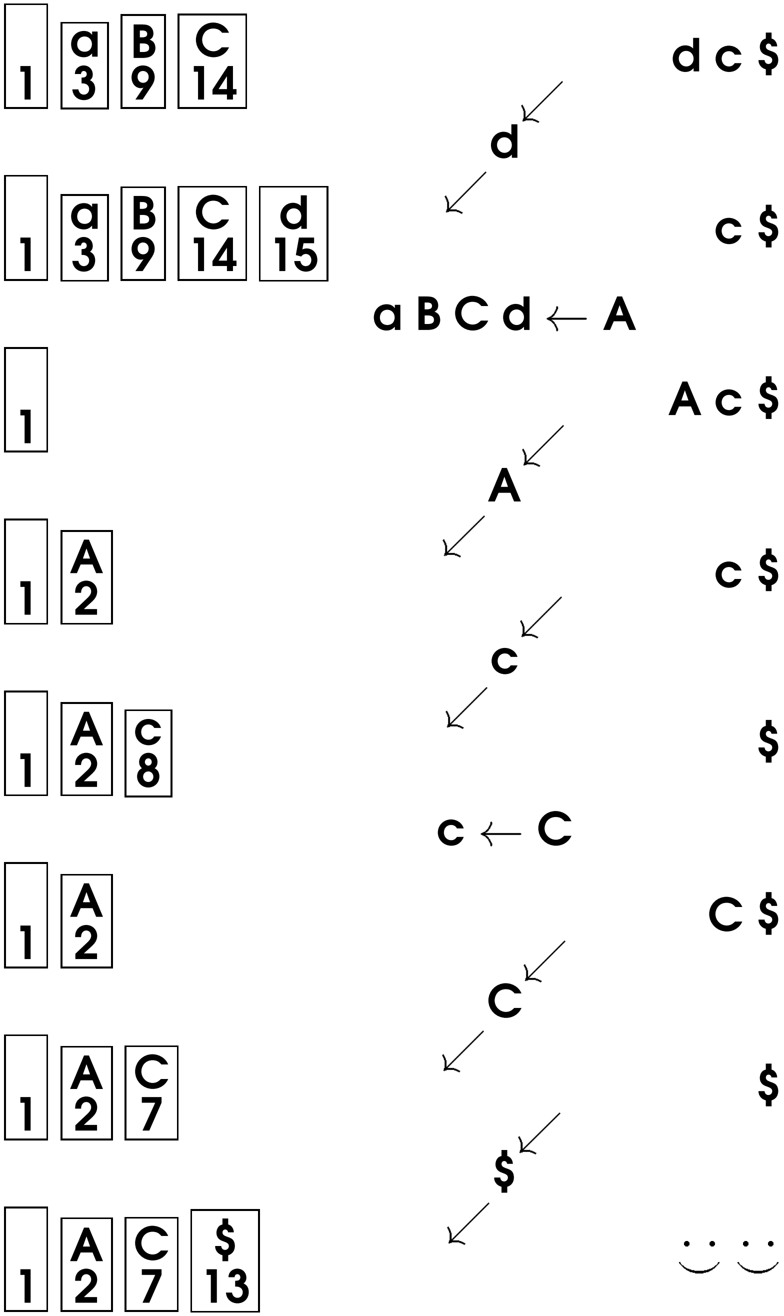
$\boxed{1}\ \boxed{a\ 3}$ B d c $

B

$\boxed{1}\ \boxed{a\ 3}\ \boxed{B\ 9}$ d c $

$\lambda \leftarrow C$

$\boxed{1}\ \boxed{a\ 3}\ \boxed{B\ 9}\ \boxed{C\ 14}$ d c $



$\boxed{1}\ \boxed{a\ 3}\ \boxed{B\ 9}\ \boxed{C\ 14}$ d c $

d

$\boxed{1}\ \boxed{a\ 3}\ \boxed{B\ 9}\ \boxed{C\ 14}\ \boxed{d\ 15}$ c $

a B C d ← A

$\boxed{1}$ A c $

A

$\boxed{1}\ \boxed{A\ 2}$ c $

c

$\boxed{1}\ \boxed{A\ 2}\ \boxed{c\ 8}$ $

c ← C

$\boxed{1}\ \boxed{A\ 2}$ C $

C

$\boxed{1}\ \boxed{A\ 2}\ \boxed{C\ 7}$ $

$

$\boxed{1}\ \boxed{A\ 2}\ \boxed{C\ 7}\ \boxed{\$\ 13}$ ☺ ☺

Set of items construction for our expression grammar

$\boxed{1}$	S	$\rightarrow$	E \$
$\boxed{2}$	E	$\rightarrow$	E + T
$\boxed{3}$		|	T
$\boxed{4}$	T	$\rightarrow$	T $*$ F
$\boxed{5}$		|	F
$\boxed{6}$	F	$\rightarrow$	(E)
$\boxed{7}$		|	a

	First	Follow
S	$\{(,a\}$	$\{\ \}$
E	$\{(,a\}$	$\{+,),\$\}$
T	$\{(,a\}$	$\{*,+,),\$\}$
F	$\{(,a\}$	$\{*,+,),\$\}$

(1)	S	$\rightarrow$	• E \$	Goto State 2
	E	$\rightarrow$	• E + T	Goto State 2
		|	• T	Goto State 3
	T	$\rightarrow$	• T $*$ F	Goto State 3
		|	• F	Goto State 4
	F	$\rightarrow$	• (E)	Goto State 5
		|	• a	Goto State 6

(2)	S	$\rightarrow$	E • \$	Goto State 7
	E	$\rightarrow$	E • + T	Goto State 8

(3)	E	$\rightarrow$	T •	Reduce by rule $\boxed{3}$
	T	$\rightarrow$	T • $*$ F	Goto State 9

The above shift/reduce conflict is resolved by noting that $* \notin Follow(E)$.

(4)	T	$\rightarrow$	F •	Reduce by rule $\boxed{5}$

(5)	F	$\rightarrow$	(• E)	Goto State 10
	E	$\rightarrow$	• E + T	Goto State 10
		|	• T	Goto State 3
	T	$\rightarrow$	• T $*$ F	Goto State 3
		|	• F	Goto State 4
	F	$\rightarrow$	• (E)	Goto State 5
		|	• a	Goto State 6

(6)	F	$\rightarrow$	a •	Reduce by rule $\boxed{7}$

Set of items construction for our expression grammar

$\boxed{1}\ \mathbf{S} \to \mathbf{E}\ \$$

$\boxed{2}\ \mathbf{E} \to \mathbf{E} + \mathbf{T}$

$\boxed{3}\ \quad\ \mid\ \mathbf{T}$

$\boxed{4}\ \mathbf{T} \to \mathbf{T} * \mathbf{F}$

$\boxed{5}\ \quad\ \mid\ \mathbf{F}$

$\boxed{6}\ \mathbf{F} \to (\ \mathbf{E}\ )$

$\boxed{7}\ \quad\ \mid\ \mathbf{a}$

	First	**Follow**
S	$\{\,(,a\,\}$	$\{\ \}$
E	$\{\,(,a\,\}$	$\{+,),\$\}$
T	$\{\,(,a\,\}$	$\{*,+,),\$\}$
F	$\{\,(,a\,\}$	$\{*,+,),\$\}$

(7) $\mathbf{S} \to \mathbf{E}\ \$\ \bullet$ ☺

(8)	$\mathbf{E} \to \mathbf{E} + \bullet\ \mathbf{T}$	Goto State	**11**
	$\mathbf{T} \to \bullet\ \mathbf{T} * \mathbf{F}$	Goto State	**11**
	$\mid\ \bullet\ \mathbf{F}$	Goto State	**4**
	$\mathbf{F} \to \bullet\ (\ \mathbf{E}\ )$	Goto State	**5**
	$\mid\ \bullet\ \mathbf{a}$	Goto State	**6**

(9)	$\mathbf{T} \to \mathbf{T} * \bullet\ \mathbf{F}$	Goto State	**12**
	$\mathbf{F} \to \bullet\ (\ \mathbf{E}\ )$	Goto State	**5**
	$\mid\ \bullet\ \mathbf{a}$	Goto State	**6**

(10)	$\mathbf{E} \to \mathbf{E}\ \bullet\ + \mathbf{T}$	Goto State	**8**
	$\mathbf{F} \to (\ \mathbf{E}\ \bullet\ )$	Goto State	**13**

(11)	$\mathbf{E} \to \mathbf{E} + \mathbf{T}\ \bullet$	Reduce by rule	$\boxed{2}$
	$\mathbf{T} \to \mathbf{T}\ \bullet\ * \mathbf{F}$	Goto State	**9**

The above shift/reduce conflict is resolved by noting that $* \notin Follow(E)$.

(12) $\mathbf{T} \to \mathbf{T} * \mathbf{F}\ \bullet$ Reduce by rule $\boxed{4}$

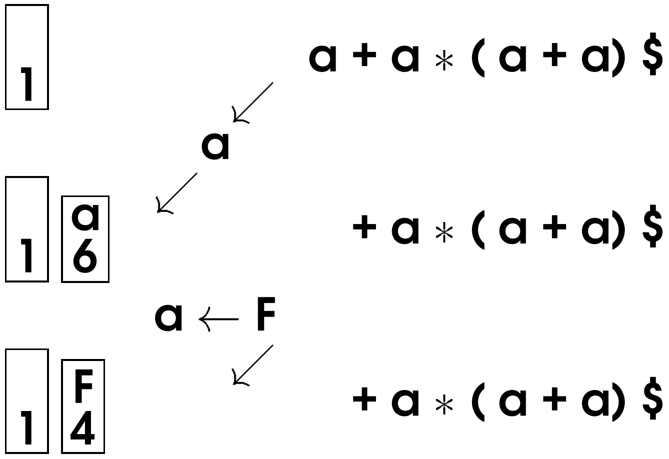
(13) $\mathbf{F} \to (\ \mathbf{E}\ )\ \bullet$ Reduce by rule $\boxed{6}$

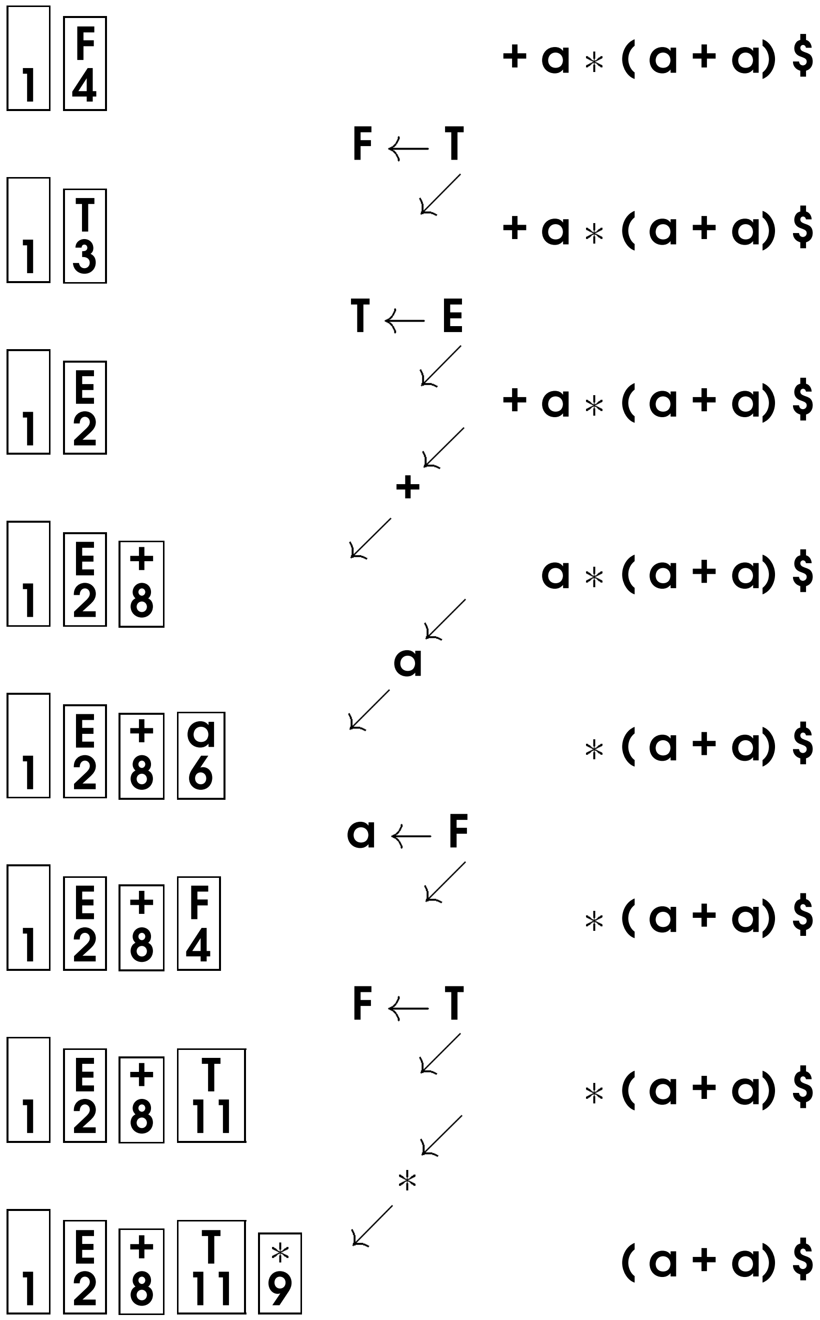
The resulting parse table

<table>
<tr><th>State</th><th>a</th><th>+</th><th>*</th><th>(</th><th>)</th><th>$</th><th>E</th><th>T</th><th>F</th></tr>
<tr><td>1</td><td>Goto State 6</td><td></td><td></td><td>Goto State 5</td><td></td><td></td><td>Goto State 2</td><td>Goto State 3</td><td>Goto State 4</td></tr>
<tr><td>2</td><td></td><td>Goto State 8</td><td></td><td></td><td></td><td>Goto State 7</td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td>Reduce by rule $\boxed{3}$</td><td>Goto State 9</td><td></td><td>Reduce by rule $\boxed{3}$</td><td>Reduce by rule $\boxed{3}$</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>Reduce by rule $\boxed{5}$</td><td colspan="7">. . . . .</td><td>Reduce by rule $\boxed{5}$</td></tr>
<tr><td>5</td><td>Goto State 6</td><td></td><td></td><td>Goto State 5</td><td></td><td></td><td>Goto State 10</td><td>Goto State 3</td><td>Goto State 4</td></tr>
<tr><td>6</td><td>Reduce by rule $\boxed{7}$</td><td colspan="7">. . . . .</td><td>Reduce by rule $\boxed{7}$</td></tr>
<tr><td>7</td><td>☺</td><td colspan="7">. . . . .</td><td>☺</td></tr>
<tr><td>8</td><td>Goto State 6</td><td></td><td></td><td>Goto State 5</td><td></td><td></td><td></td><td>Goto State 11</td><td>Goto State 4</td></tr>
<tr><td>9</td><td>Goto State 6</td><td></td><td></td><td>Goto State 5</td><td></td><td></td><td></td><td></td><td>Goto State 12</td></tr>
<tr><td>10</td><td></td><td>Goto State 8</td><td></td><td></td><td>Goto State 13</td><td></td><td></td><td></td><td></td></tr>
<tr><td>11</td><td></td><td>Reduce by rule $\boxed{2}$</td><td>Goto State 9</td><td></td><td>Reduce by rule $\boxed{2}$</td><td>Reduce by rule $\boxed{2}$</td><td></td><td></td><td></td></tr>
<tr><td>12</td><td>Reduce by rule $\boxed{4}$</td><td colspan="7">. . . . .</td><td>Reduce by rule $\boxed{4}$</td></tr>
<tr><td>13</td><td>Reduce by rule $\boxed{6}$</td><td colspan="7">. . . . .</td><td>Reduce by rule $\boxed{6}$</td></tr>
</table>

Using the table

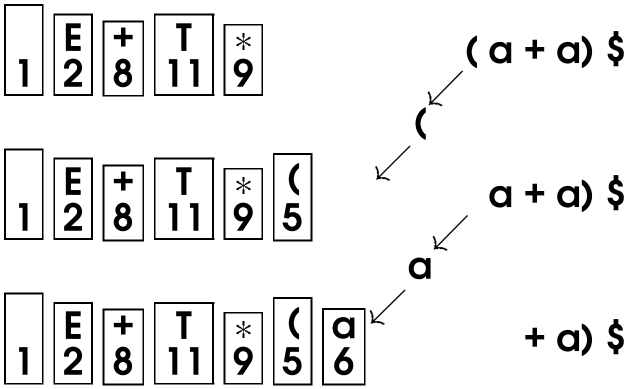
State	a	+	∗	(	)	$	E	T	F
1	6			5			2	3	4
2		8				7			
3		$\boxed{3}$	9		$\boxed{3}$	$\boxed{3}$			
4	$\boxed{5}$	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	$\boxed{5}$
5	6			5			10	3	4
6	$\boxed{7}$	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	$\boxed{7}$
7	☺	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	☺
8	6			5				11	4
9	6			5					12
10		8			13				
11		$\boxed{2}$	9		$\boxed{2}$	$\boxed{2}$			
12	$\boxed{4}$	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	$\boxed{4}$
13	$\boxed{6}$	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	$\boxed{6}$





Using the table

State	a	+	*	(	)	$	E	T	F
1	6			5			2	3	4
2		8				7			
3		$\boxed{3}$	9		$\boxed{3}$	$\boxed{3}$			
4	$\boxed{5}$	· ·	· ·	· ·	· ·	· ·	· ·	· ·	$\boxed{5}$
5	6			5			10	3	4
6	$\boxed{7}$	· ·	· ·	· ·	· ·	· ·	· ·	· ·	$\boxed{7}$
7	☺	· ·	· ·	· ·	· ·	· ·	· ·	· ·	☺
8	6			5				11	4
9	6			5					12
10		8			13				
11		$\boxed{2}$	9		$\boxed{2}$	$\boxed{2}$			
12	$\boxed{4}$	· ·	· ·	· ·	· ·	· ·	· ·	· ·	$\boxed{4}$
13	$\boxed{6}$	· ·	· ·	· ·	· ·	· ·	· ·	· ·	$\boxed{6}$

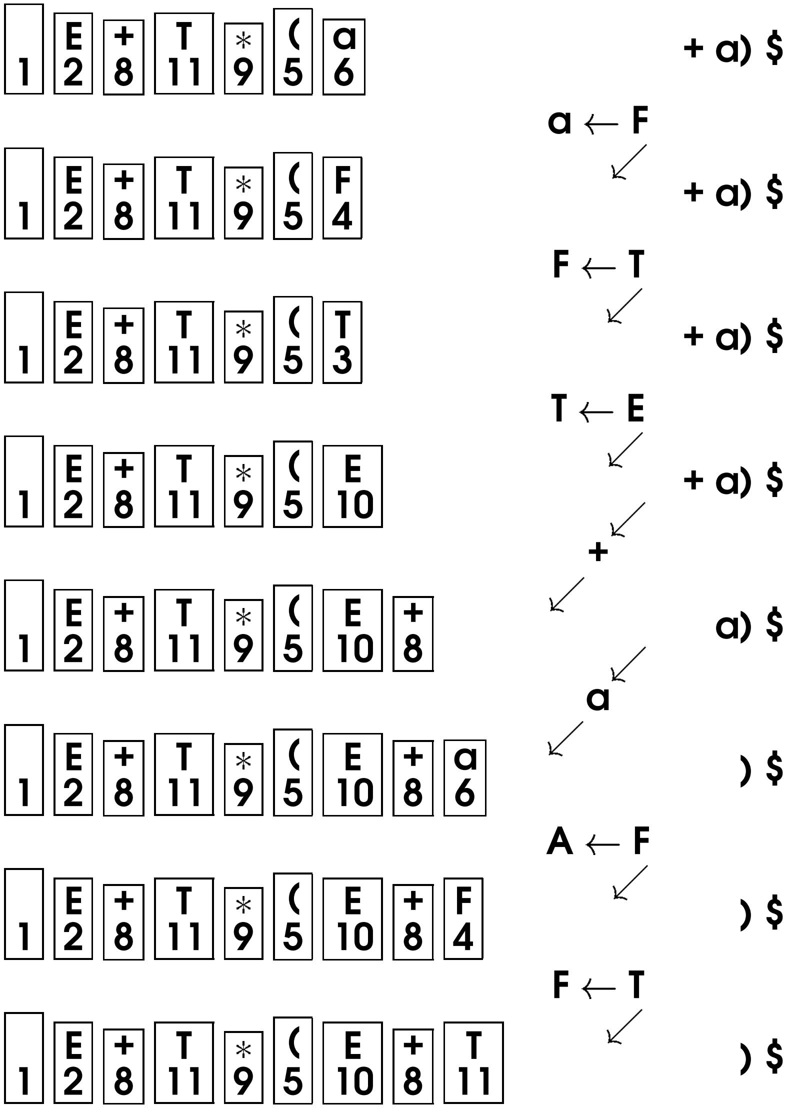

1 E 2 + 8 T 11 * 9   (a + a) $

(

1 E 2 + 8 T 11 * 9 (5   a + a) $

a

1 E 2 + 8 T 11 * 9 (5 a 6   + a) $


1 E 2 + 8 T 11 * 9 (5 a 6   + a) $

a ← F

1 E 2 + 8 T 11 * 9 (5 F 4   + a) $

F ← T

1 E 2 + 8 T 11 * 9 (5 T 3   + a) $

T ← E

1 E 2 + 8 T 11 * 9 (5 E 10   + a) $

+

1 E 2 + 8 T 11 * 9 (5 E 10 + 8   a) $

a

1 E 2 + 8 T 11 * 9 (5 E 10 + 8 a 6   ) $

A ← F

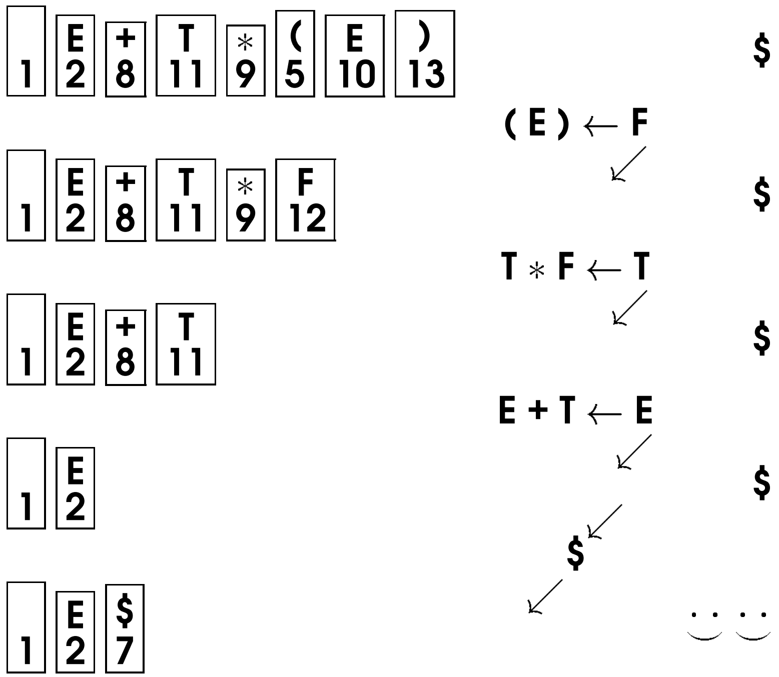
1 E 2 + 8 T 11 * 9 (5 E 10 + 8 F 4   ) $

F ← T

1 E 2 + 8 T 11 * 9 (5 E 10 + 8 T 11   ) $

Using the table

State	a	+	*	(	)	$	E	T	F
1	6			5			2	3	4
2		8				7			
3		$\boxed{3}$	9		$\boxed{3}$	$\boxed{3}$			
4	$\boxed{5}$	· ·	· ·	· ·	· ·	· ·	· ·	· ·	$\boxed{5}$
5	6			5			10	3	4
6	$\boxed{7}$	· ·	· ·	· ·	· ·	· ·	· ·	· ·	$\boxed{7}$
7	☺	· ·	· ·	· ·	· ·	· ·	· ·	· ·	☺
8	6			5				11	4
9	6			5					12
10		8			13				
11		$\boxed{2}$	9		$\boxed{2}$	$\boxed{2}$			
12	$\boxed{4}$	· ·	· ·	· ·	· ·	· ·	· ·	· ·	$\boxed{4}$
13	$\boxed{6}$	· ·	· ·	· ·	· ·	· ·	· ·	· ·	$\boxed{6}$



$\boxed{1}\ \boxed{E\ 2}\ \boxed{+\ 8}\ \boxed{T\ 11}\ \boxed{*\ 9}\ \boxed{(\ 5}\ \boxed{E\ 10}\ \boxed{)\ 13}$ $

(E) ← F

$\boxed{1}\ \boxed{E\ 2}\ \boxed{+\ 8}\ \boxed{T\ 11}\ \boxed{*\ 9}\ \boxed{F\ 12}$ $

T * F ← T

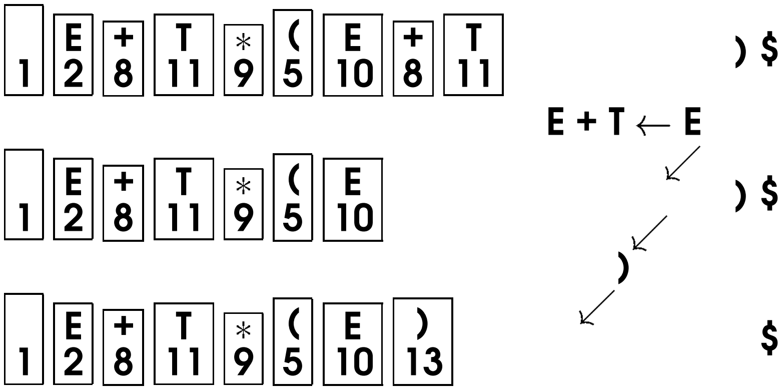
$\boxed{1}\ \boxed{E\ 2}\ \boxed{+\ 8}\ \boxed{T\ 11}$ $

E + T ← E

$\boxed{1}\ \boxed{E\ 2}$ $

$

$\boxed{1}\ \boxed{E\ 2}\ \boxed{\$\ 7}$ ☺ ☺



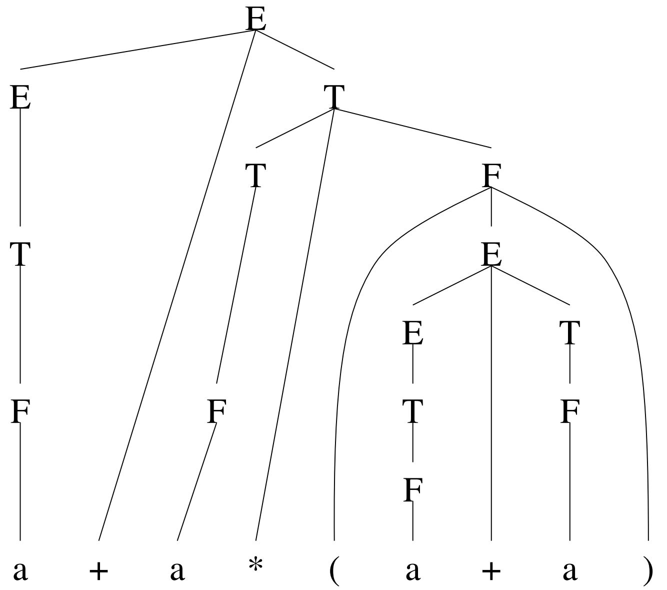
$\boxed{1}\ \boxed{E\ 2}\ \boxed{+\ 8}\ \boxed{T\ 11}\ \boxed{*\ 9}\ \boxed{(\ 5}\ \boxed{E\ 10}\ \boxed{+\ 8}\ \boxed{T\ 11}$) $

E + T ← E

$\boxed{1}\ \boxed{E\ 2}\ \boxed{+\ 8}\ \boxed{T\ 11}\ \boxed{*\ 9}\ \boxed{(\ 5}\ \boxed{E\ 10}$) $

)

$\boxed{1}\ \boxed{E\ 2}\ \boxed{+\ 8}\ \boxed{T\ 11}\ \boxed{*\ 9}\ \boxed{(\ 5}\ \boxed{E\ 10}\ \boxed{)\ 13}$ $



Summary of LR table construction methods

LR(0): If the table contains no conflicts, then the grammar is unambiguous and each state clearly indicates precise shifts and reduces.

SLR(k): Where conflicts exist, this method analyzes the grammar to obtain sets of at the k symbols that can follow each nonterminal. For an item containing

$$\boxed{\begin{array}{llcl} (1) & A & \rightarrow & B\ C\ \bullet \\ & D & \rightarrow & C\ \bullet\ F \\ & G & \rightarrow & C\ \bullet \end{array}}$$

if the k symbols that can follow A are disjoint from each of the strings of k symbols derivable from F, then the shift/reduce conflict is resolved. If the k symbols that can follow A are different from those that can follow G, then the reduce/reduce conflict is resolved.

LR(k): While the SLR method analyzes the grammar for follow information, the LR(k) method begins with a more elaborate set of items that already incorporates follow information. For example, given

$$\boxed{(3)\quad A \rightarrow \{\ \bullet\ E\,\}}$$

$$\boxed{(4)\quad B \rightarrow (\ \bullet\ E\,)}$$

the SLR method would assume that "}" or ")" could follow an E in any context. The LR(k) method carries into each state the relevant follow set. Thus, the table constructed by LR can have many more states than the table constructed by SLR.

LALR(k): is a compromise between SLR and LR. The table is the same size as SLR, but conflict resolution is sharper.

The methods described above are successful only for unambiguous grammars. Earley's algorithm (1, 16) can construct parses (and derivations) for ambiguous grammars. Note that LR parsing is more powerful that LL parsing.

What happens when LR(k) constructions fail?

If table construction reveals an inadequate state, one of the following must hold:

The grammar is ambiguous.

If the language is not itself inherently ambiguous, then perhaps the grammar can be modified to generate the same language, but unambiguously.

This is a task for human intelligence, as it's provably undecidable (*i.e.*, there is no mechanical process to decide) that a grammar is ambiguous.

A method that works well is to identify the inadequate states, and then work into and out of the state to generate a string that has more than one derivation. The conflicts (identified, for example, by YACC) are helpful in this process.

Underfueled table construction

1. **Generally, SLR is more powerful than LR(0); LALR is more powerful than SLR; LR is the most powerful (canonical) bottom-up parsing method.**
2. **Canonical LR parsers must form their reductions on top-of-stack. For some grammars (an example follows), no bounded amount of lookahead (bounded at table construction time) suffices to disambiguate some state.**

A good exercise is to attempt adding nested procedures into the ANSI C grammar. `foo(,,,...,) {` **becomes problematic: One can't tell whether** `foo` **is a procedure definition or invocation until the arbitrarily distant opening brace is seen.**

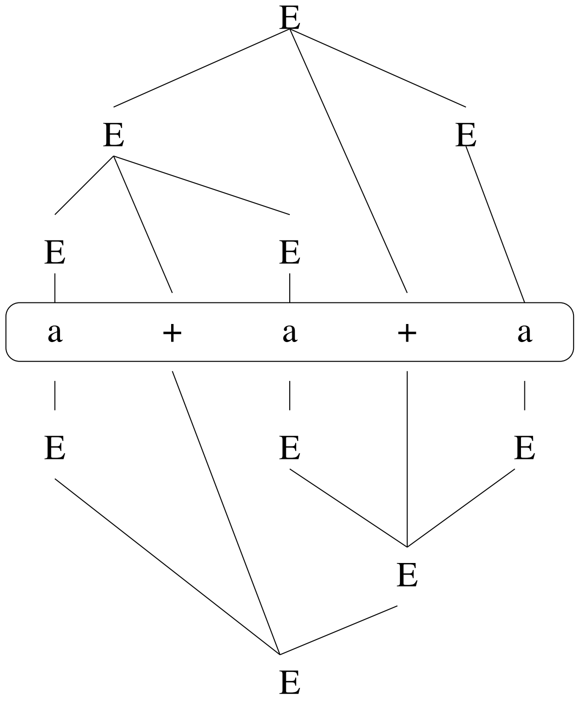
Identifying the cause of ambiguity

$$\begin{array}{lcl} \mathbf{E} & \rightarrow & \mathbf{E + E} \\ & | & \mathbf{a} \end{array}$$

YACC **finds a shift/reduce conflict in the following state:**

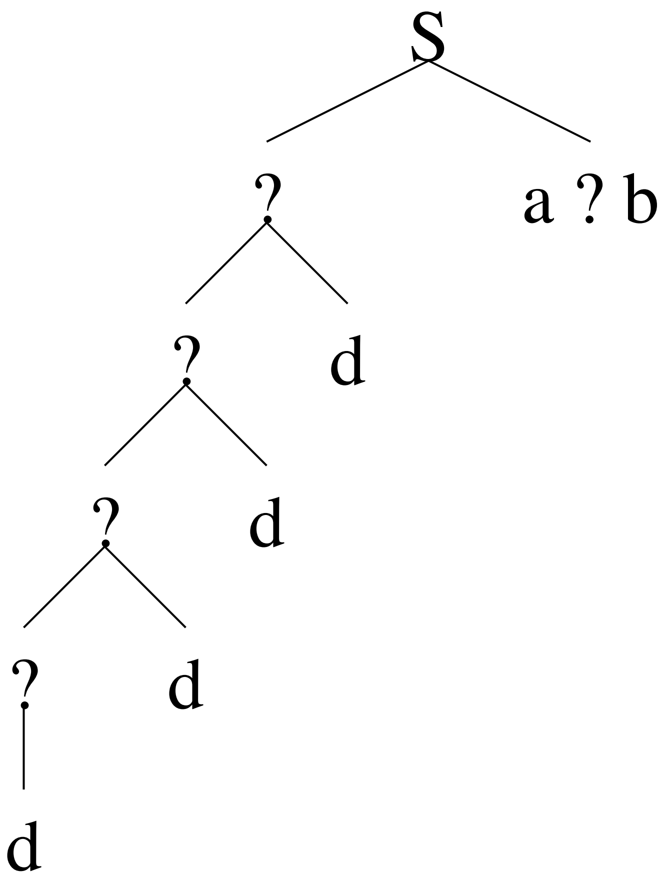
(4)	**E** $\rightarrow$ **E • + E**	Goto State	**3**
	E $\rightarrow$ **E + E •**	Reduce by rule	$\boxed{1}$

Lining up the "dots" shows we can reach this state with the prefix . . . E + E, and one rule shows how to continue this string to . . . E + E + E. . . . We can now easily construct two parses: one assumes state 4 shifts (bottom), one assumes state 4 reduces (top).



A grammar that is not LR(k) for any k

$$\begin{array}{lcl} \mathbf{S} & \rightarrow & \mathbf{A\ a} \\ & | & \mathbf{B\ b} \\ \mathbf{A} & \rightarrow & \mathbf{A\ d} \\ & | & \mathbf{d} \\ \mathbf{B} & \rightarrow & \mathbf{B\ d} \\ & | & \mathbf{d} \end{array}$$



In the above grammar, a reduction must occur for the first "d" in the input, but the lookahead necessary for deciding whether to reduce A→ d or B→ d could be arbitrarily large.

If the right-hand sides of the first rules for A and B were reversed, then the grammar is LR(1), but the stack grows arbitrarily large at parse time.

Often the grammar can be modified to become LR(k), since this problem usually pertains to *how* the language is structured by the grammar.