

Section 3: LR Parsing

Aaron Johnston & Nate Yazdani

Announcements

- Scanner is due tonight
 - Be sure to test, push, and tag!
- Every person has 4 late days
 - Up to 2 can be used per assignment
 - **Submitting project components a day late uses a late day from each partner!**

Get Your LR Jargon On

- Frontier
 - The upper “layer” of the current parse tree (held in the stack)

Get Your LR Jargon On

- Frontier
 - The upper “layer” of the current parse tree (held in the stack)
- Sentential Form
 - A string that can be generated at any point in a derivation (can be reached using any number of productions from the start symbol)

Get Your LR Jargon On

- Frontier
 - The upper “layer” of the current parse tree (held in the stack)
- Sentential Form
 - A string that can be generated at any point in a derivation (can be reached using any number of productions from the start symbol)
- Handle
 - An occurrence of the right side of a production in the frontier that is used in the rightmost derivation to arrive at the current string
 - Given the derivation ... \Rightarrow **aAbcde** \Rightarrow **abbcde**, using the production **A** \Rightarrow **b**:
 - The substring ‘**b**’ at index 1 would be a handle of **abbcde**

LR (0)



Left-to-Right

Only takes one pass,
performed from the left

Rightmost

At each point, finds the
derivation for the rightmost
handle (bottom-up)

No Lookahead

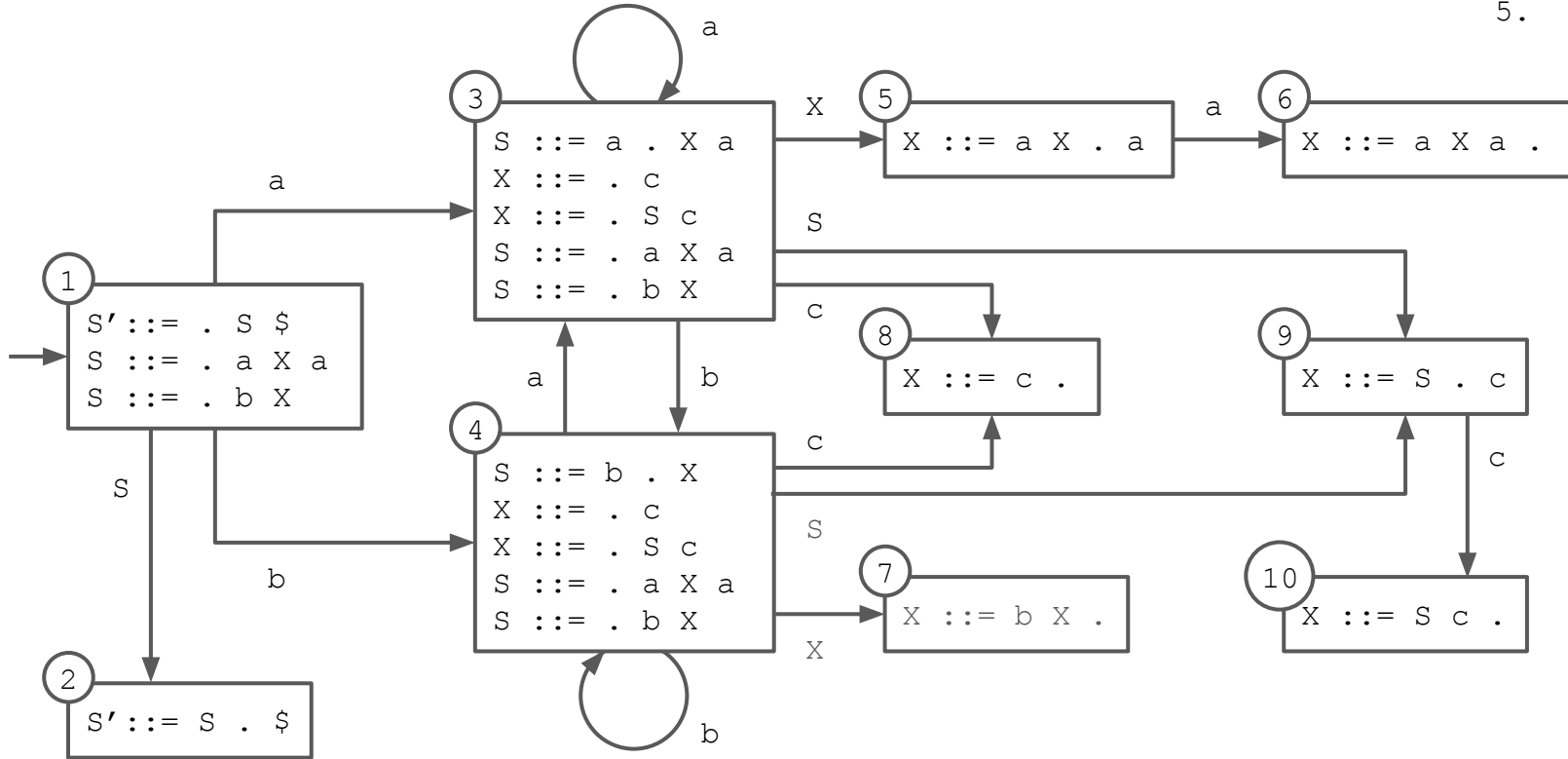
After shifting a single token
from the input, has enough
information to proceed

Problem 1 (On Worksheet)

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$

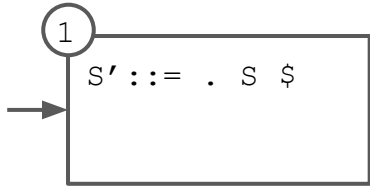
Completed State Diagram

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$



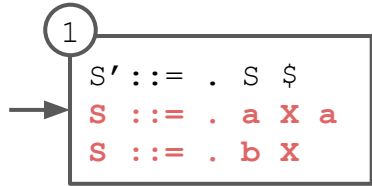
State Diagram Construction

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$



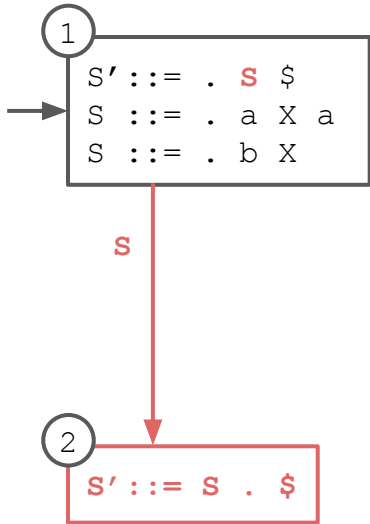
State Diagram Construction

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$



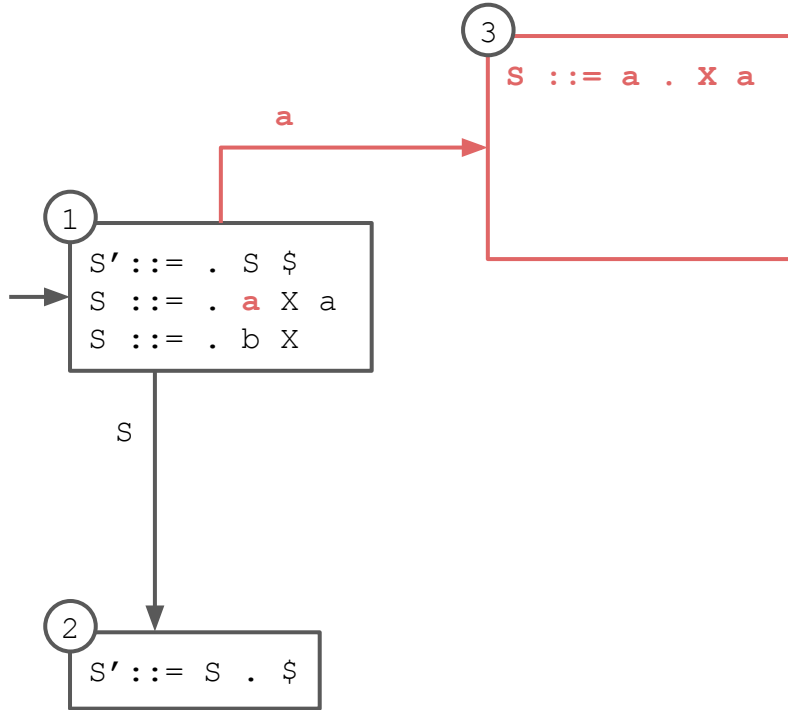
State Diagram Construction

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$



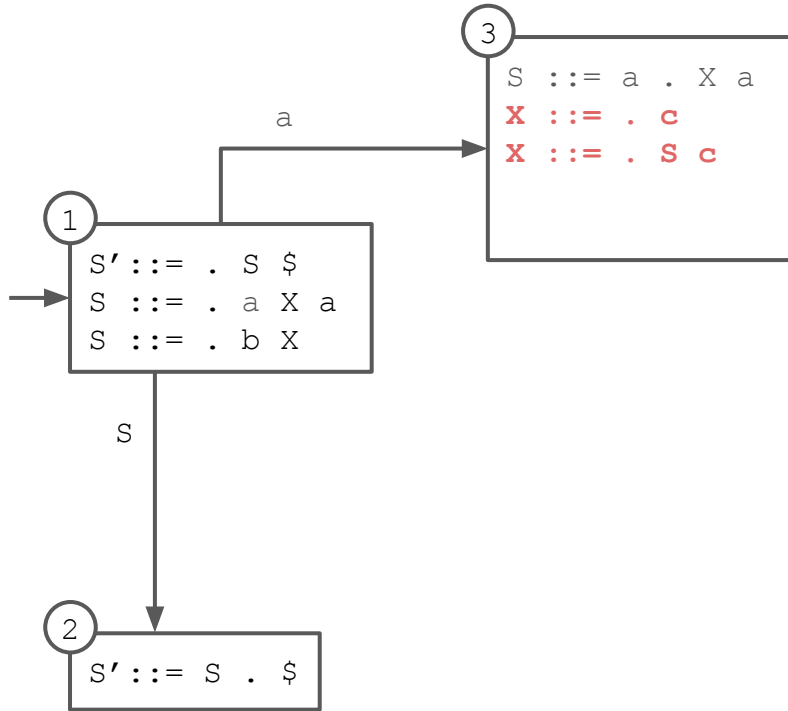
State Diagram Construction

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$



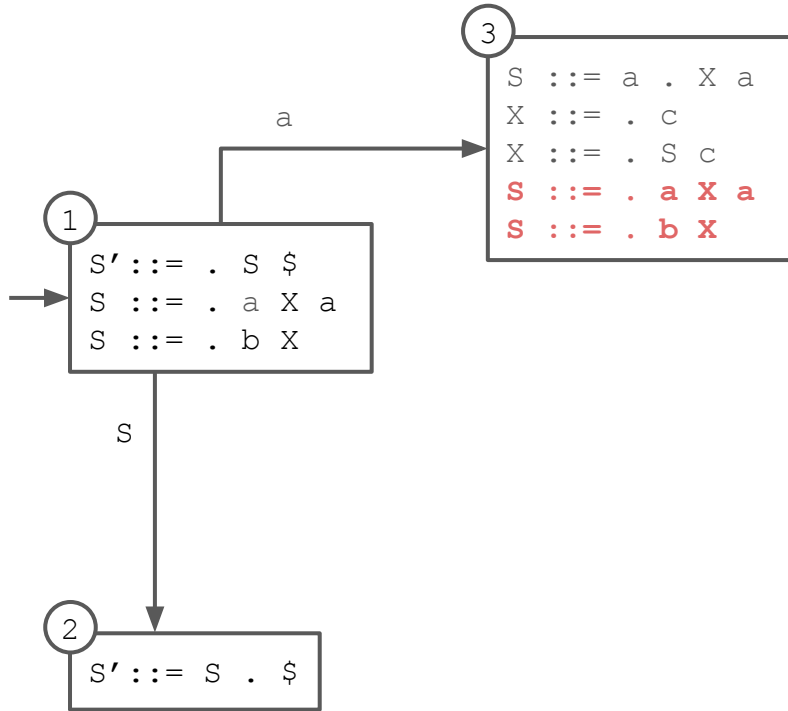
State Diagram Construction

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$



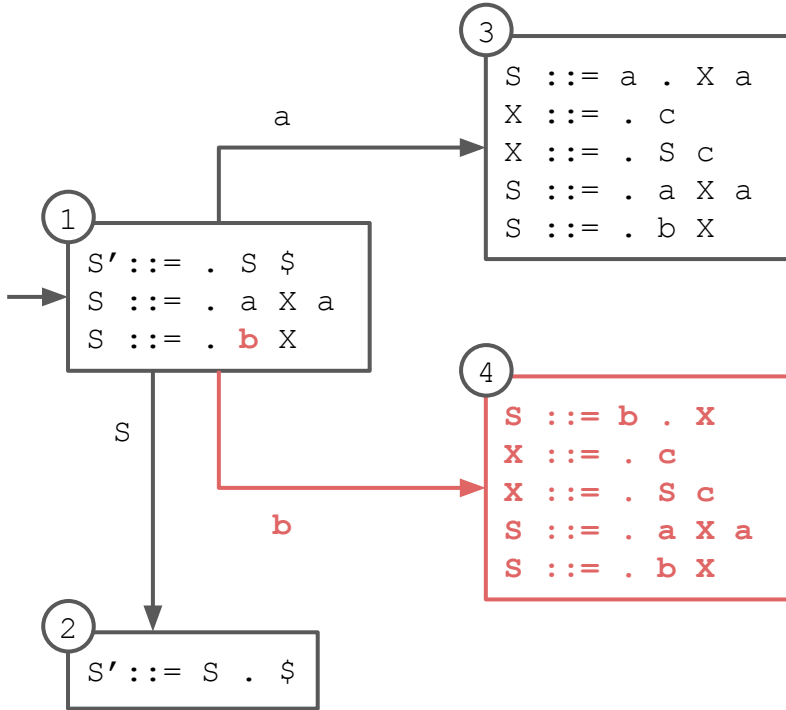
State Diagram Construction

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$



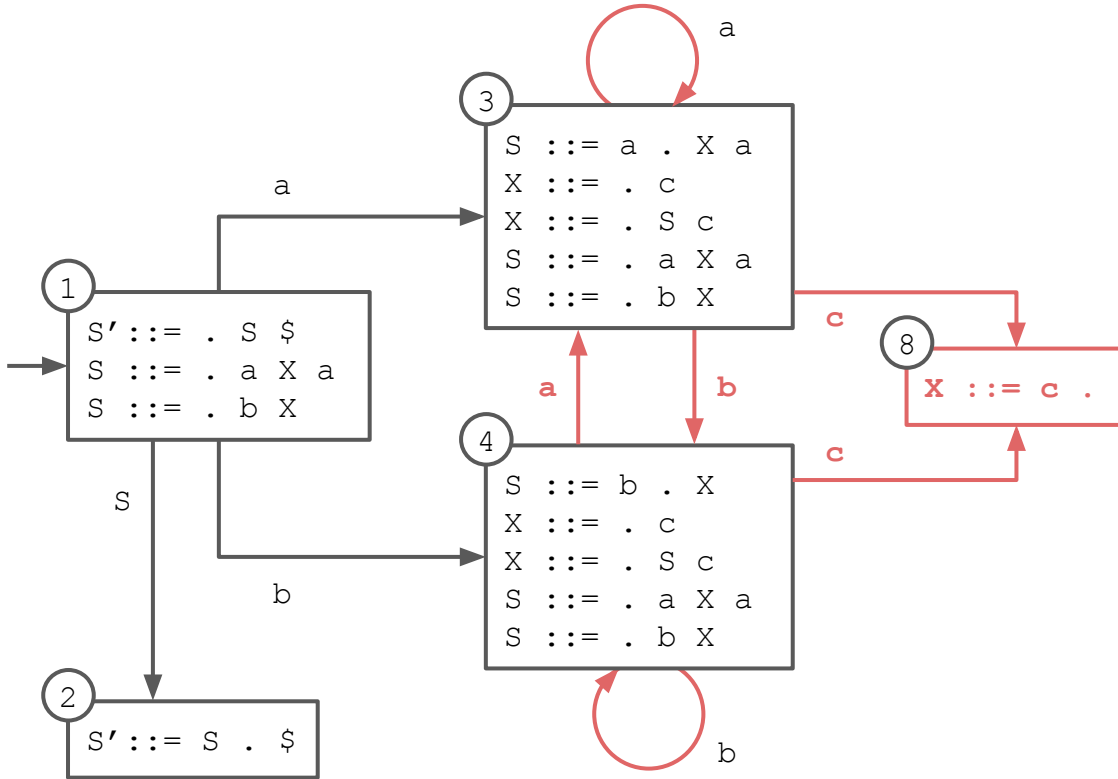
State Diagram Construction

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$



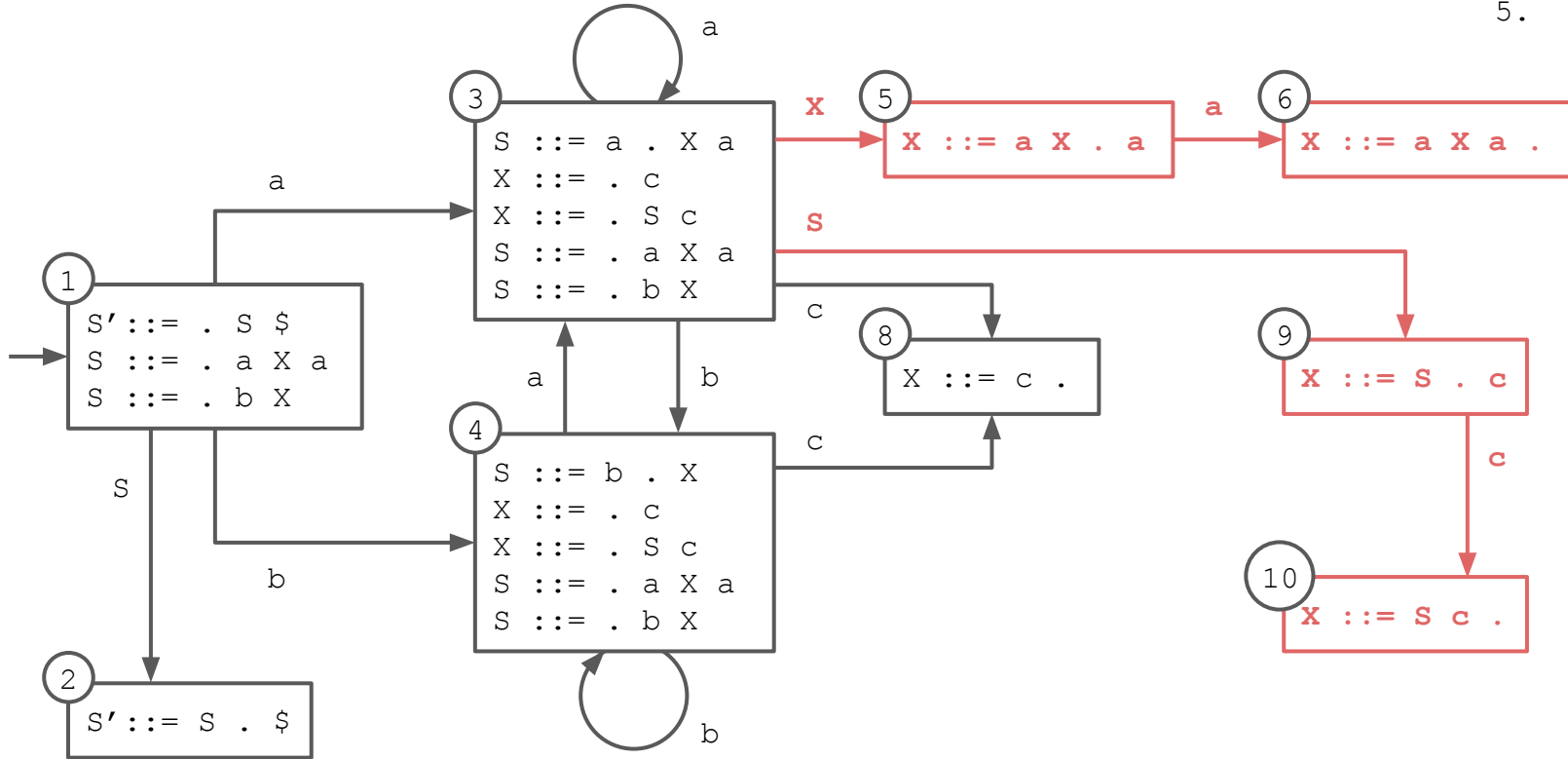
State Diagram Construction

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$



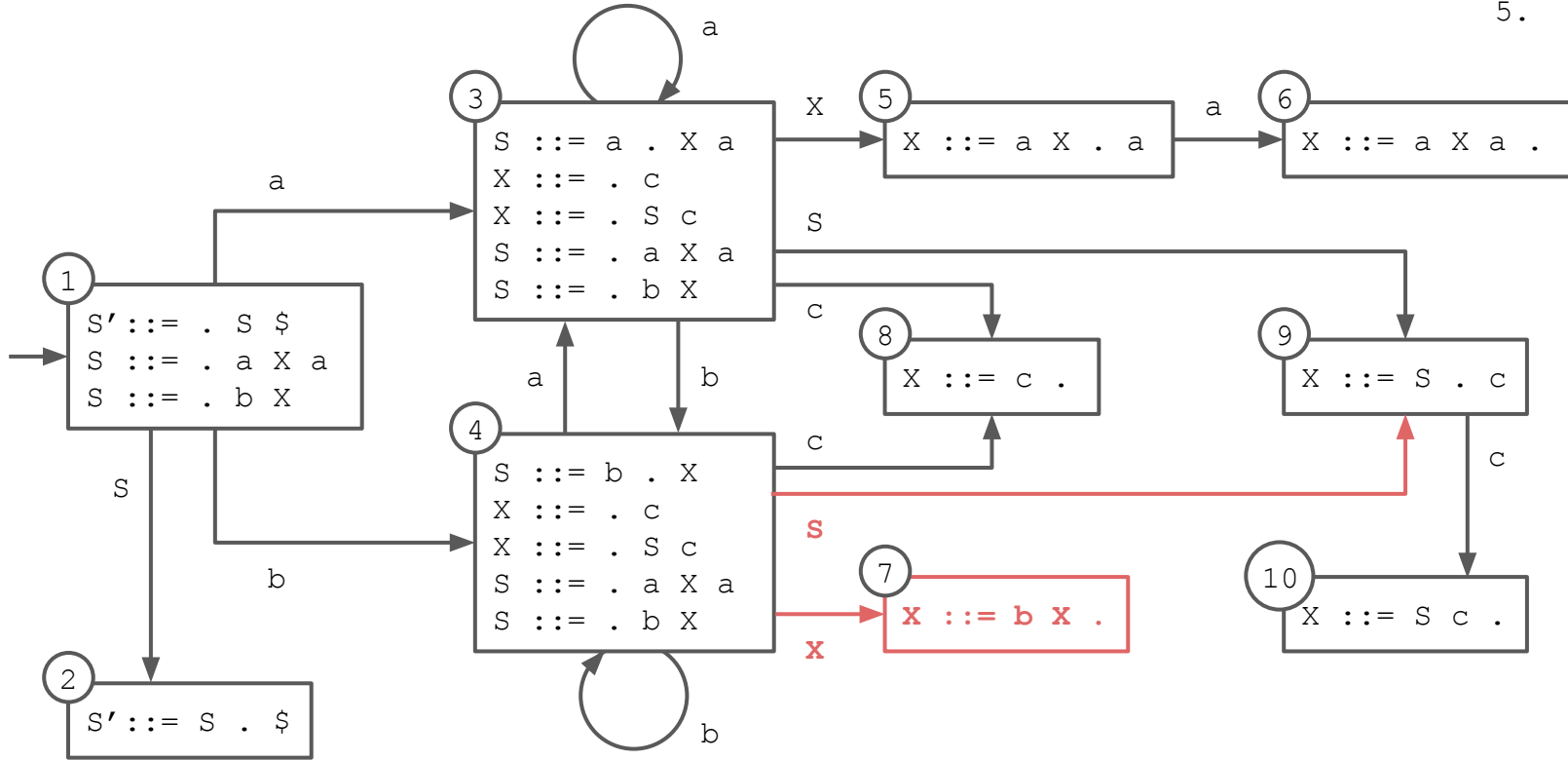
State Diagram Construction

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$



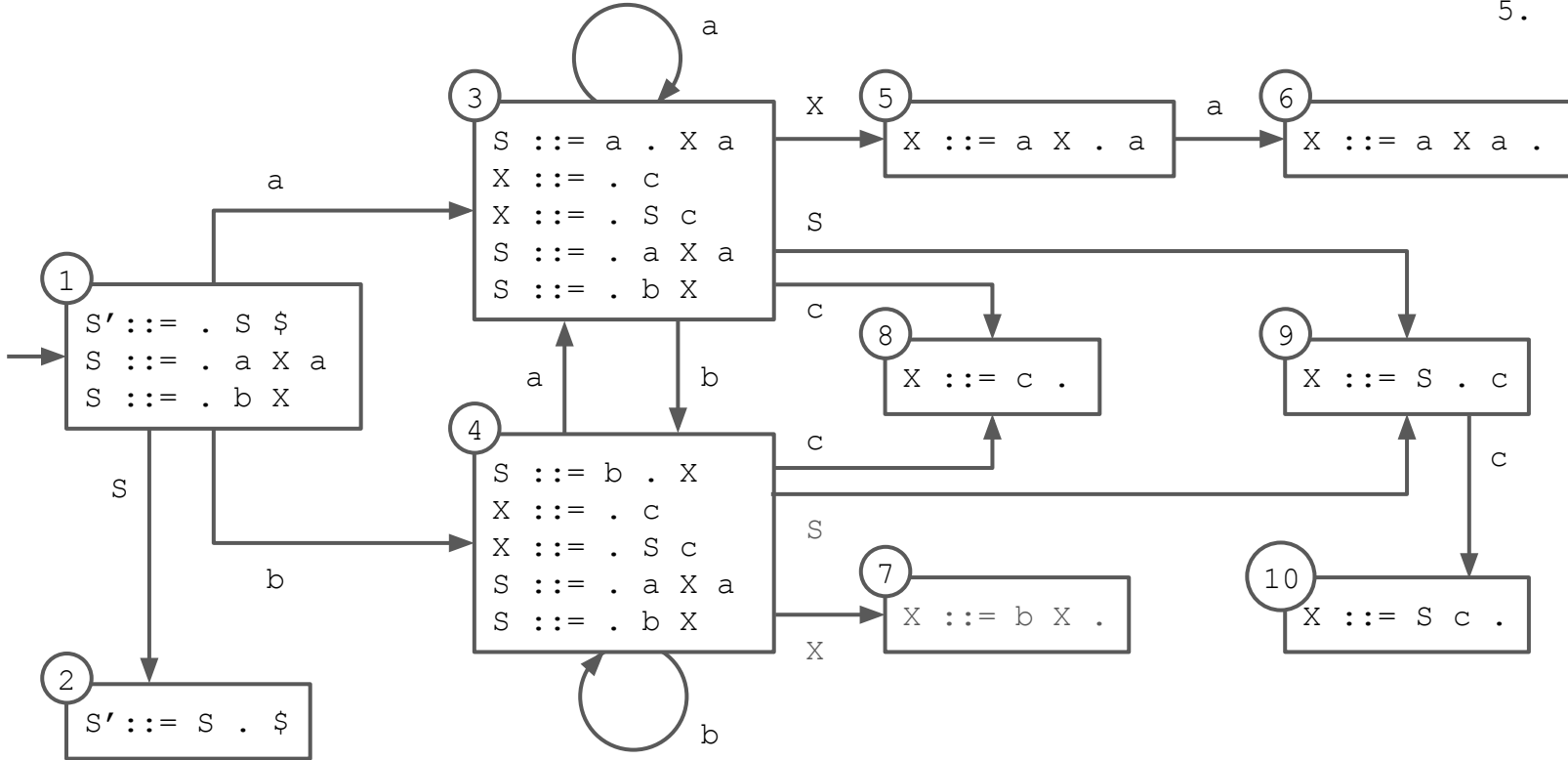
State Diagram Construction

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$



State Diagram Construction

1. $S' ::= S \$$
2. $S ::= a X a$
3. $S ::= b X$
4. $X ::= c$
5. $X ::= S c$



Converted to Table

s# means “shift and enter state #”

- occurs when there is a transition on a terminal

r# means “reduce using production #”

- occurs when a state contains an item with the location at the end of the right-hand side

g# means “go to state #”

- occurs when there is a transition on a nonterminal

acc means “accept”

- occurs when the start symbol (S here) has been completed and there is no more input

STATE	ACTION				GOTO	
	a	b	c	\$	S	X
1	s3	s4			g2	
2				acc		
3	s3	s4	s8		g9	g5
4	s3	s4	s8		g9	g7
5	s6					
6	r2	r2	r2	r2		
7	r3	r3	r3	r3		
8	r4	r4	r4	r4		
9			s10			
10	r5	r5	r5	r5		

Parse Trace

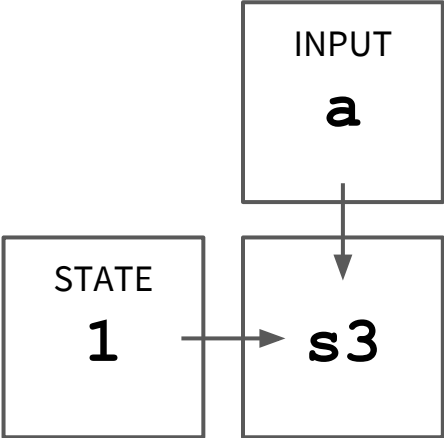
STACK

INPUT

\$ s₁

a b c c a \$

Parse Trace



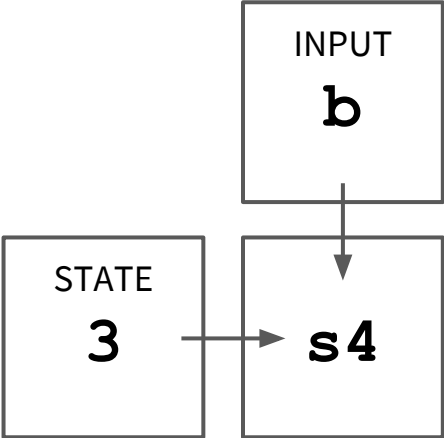
STACK

INPUT

\$ s₁
\$ s₁ a s₃

a b c c a \$
b c c a \$

Parse Trace



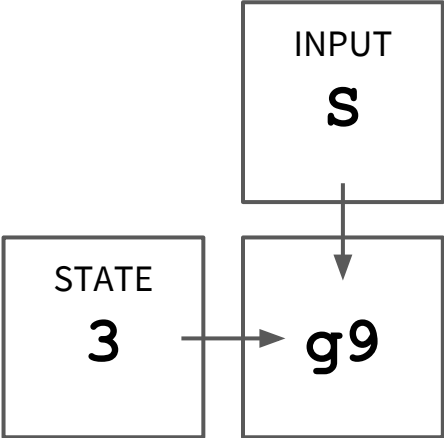
STACK

INPUT

\$ s₁
\$ s₁ a s₃
\$ s₁ a s₃ b s₄

a b c c a \$
b c c a \$
c c a \$

Parse Trace



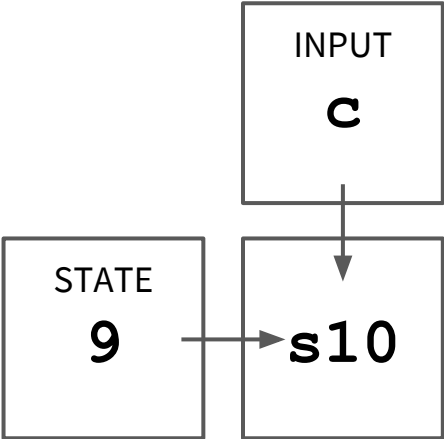
STACK

\$	s_1							
\$	s_1	a	s_3					
\$	s_1	a	s_3	b	s_4			
\$	s_1	a	s_3	b	s_4	c	s_8	
\$	s_1	a	s_3	b	s_4	X	s_7	
\$	s_1	a	s_3	S	s_9			

INPUT

a	b	c	c	a	\$
	b	c	c	a	\$
		c	c	a	\$
			c	a	\$
			c	a	\$
			c	a	\$

Parse Trace



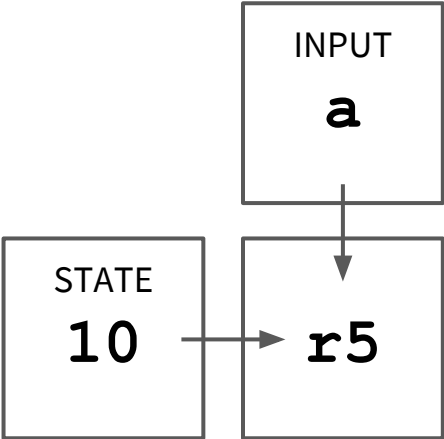
STACK

\$ s₁
 \$ s₁ a s₃
 \$ s₁ a s₃ b s₄
 \$ s₁ a s₃ b s₄ c s₈
 \$ s₁ a s₃ b s₄ X s₇
 \$ s₁ a s₃ S s₉
\$ s₁ a s₃ S s₉ c s₁₀

INPUT

a b c c a \$
 b c c a \$
 c c a \$
 c a \$
 c a \$
 c a \$
a \$

Parse Trace



STACK

```

$ s1
$ s1 a s3
$ s1 a s3 b s4
$ s1 a s3 b s4 c s8
$ s1 a s3 b s4 X s7
$ s1 a s3 S s9
$ s1 a s3 S s9 c s10
$ s1 a s3 X
  
```

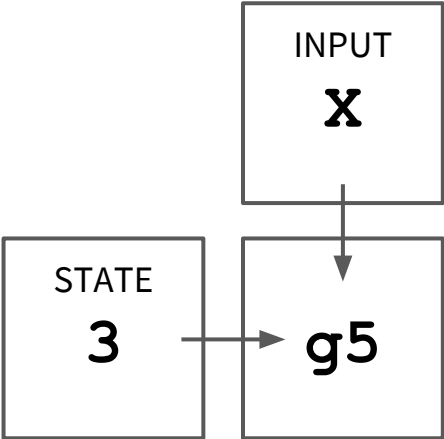
INPUT

```

a b c c a $
  b c c a $
    c c a $
      c a $
        c a $
          a $
            a $
  
```

5. X ::= S c

Parse Trace



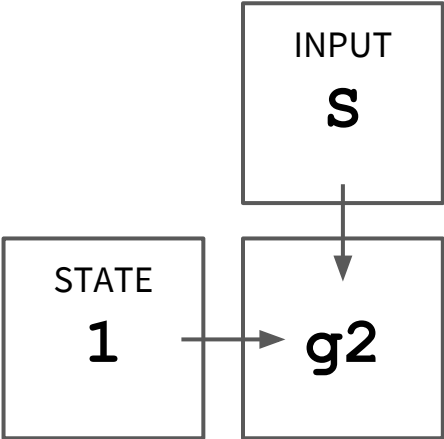
STACK

\$	s ₁							
\$	s ₁	a	s ₃					
\$	s ₁	a	s ₃	b	s ₄			
\$	s ₁	a	s ₃	b	s ₄	c	s ₈	
\$	s ₁	a	s ₃	b	s ₄	X	s ₇	
\$	s ₁	a	s ₃	S	s ₉			
\$	s ₁	a	s ₃	S	s ₉	c	s ₁₀	
\$	s ₁	a	s ₃	X	s ₅			

INPUT

a	b	c	c	a	\$
	b	c	c	a	\$
		c	c	a	\$
			c	a	\$
			c	a	\$
				a	\$
				a	\$

Parse Trace

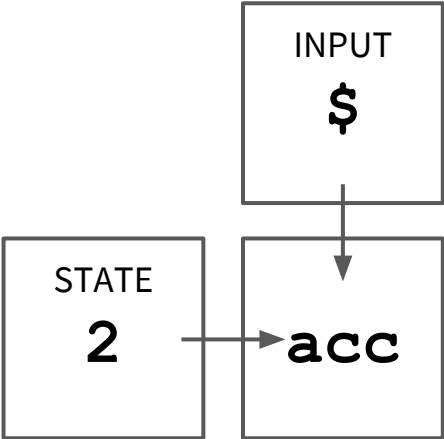


STACK

INPUT

STACK										INPUT					
\$	s ₁									a	b	c	c	a	\$
\$	s ₁	a	s ₃								b	c	c	a	\$
\$	s ₁	a	s ₃	b	s ₄							c	c	a	\$
\$	s ₁	a	s ₃	b	s ₄	c	s ₈						c	a	\$
\$	s ₁	a	s ₃	b	s ₄	X	s ₇						c	a	\$
\$	s ₁	a	s ₃	S	s ₉								c	a	\$
\$	s ₁	a	s ₃	S	s ₉	c	s ₁₀							a	\$
\$	s ₁	a	s ₃	X	s ₅									a	\$
\$	s ₁	a	s ₃	X	s ₅	a	s ₆								\$
\$	s ₁	S	s ₂												\$

Parse Trace



STACK

```

$ s1
$ s1 a s3
$ s1 a s3 b s4
$ s1 a s3 b s4 c s8
$ s1 a s3 b s4 X s7
$ s1 a s3 S s9
$ s1 a s3 S s9 c s10
$ s1 a s3 X s5
$ s1 a s3 X s5 a s6
$ s1 S s2
accept!
  
```

INPUT

```

a b c c a $
  b c c a $
    c c a $
      c a $
        c a $
          a $
            a $
              $
                $
                  $
  
```