

Notes on LR Parsing

Geoffrey Matthews

Department of Computer Science
Western Washington University

November 26, 2018

Readings

- ▶ http://www.cs.rochester.edu/~nelson/courses/csc_173/grammars/cfg.html
- ▶ http://en.wikipedia.org/wiki/Context-free_grammar
- ▶ http://en.wikipedia.org/wiki/Context-free_language
- ▶ <http://en.wikipedia.org/wiki/Parsing>
- ▶ http://en.wikipedia.org/wiki/Pushdown_automata
- ▶ http://en.wikipedia.org/wiki/LR_parser
- ▶ <https://parasol.tamu.edu/~rwerger/Courses/434/lec12-sum.pdf>

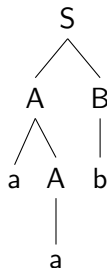
Bottom up parsing of CFGs

- ▶ We start with the input and attempt to build the parse tree.
- ▶ If we begin with the input and attempt to build the tree above it, we are doing **bottom-up** parsing.
- ▶ Equivalently, we try to construct a rightmost derivation from right to left, scanning the input left to right.

$$S \rightarrow AB$$

$$A \rightarrow aA \mid a$$

$$B \rightarrow b$$



$$S \xRightarrow{S \rightarrow AB} AB \xRightarrow{B \rightarrow b} Ab \xRightarrow{A \rightarrow aA} aAb \xRightarrow{A \rightarrow a} aab$$

$LR(k)$ grammars

- ▶ $LR(k)$ means we find a rightmost derivation by scanning the input left to right, and have to lookahead at most k symbols.

LR parsing: Shift and Reduce

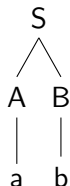
Shift: move character from input to stack

Reduce: if stack holds RHS of a rule, replace with LHS

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$



Stack	Input	Rule
\$	ab\$	shift
\$a	b\$	$A \rightarrow a$
\$A	b\$	shift
\$Ab	\$	$B \rightarrow b$
\$AB	\$	$S \rightarrow AB$
\$S	\$	accept

$S \xRightarrow{S \rightarrow AB} AB \xRightarrow{B \rightarrow b} Ab \xRightarrow{A \rightarrow a} ab$

- Note: At all times, stack+input=derivation string

LR parsing: Shift and Reduce

$$S \rightarrow ASB \mid c$$

$$A \rightarrow a$$

$$B \rightarrow b$$

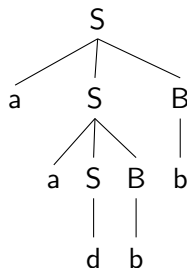

Stack	Input	Rule
\$	aacbb \$	shift
\$ a	acbb \$	$A \rightarrow a$
\$ A	acbb \$	shift
\$ Aa	cbb \$	$A \rightarrow a$
\$ AA	cbb \$	shift
\$ AAc	bb \$	$S \rightarrow c$
\$ AAS	bb \$	shift
\$ AASb	b \$	$B \rightarrow b$
\$ AASB	b \$	$S \rightarrow ASB$
\$ AS	b \$	shift
\$ ASb	\$	$B \rightarrow b$
\$ ASB	\$	$S \rightarrow ASB$
\$ S	\$	accept

$$\begin{aligned}
 S &\xRightarrow{S \rightarrow ASB} ASB \xRightarrow{B \rightarrow b} ASb \xRightarrow{S \rightarrow ASB} AASBb \xRightarrow{B \rightarrow b} AASbb \xRightarrow{S \rightarrow c} \\
 &AAcbb \xRightarrow{A \rightarrow a} Aacbb \xRightarrow{A \rightarrow a} aacbb
 \end{aligned}$$

Another LR parse

$$S \rightarrow aSB \mid d$$

$$B \rightarrow b$$



Stack	Input	Rule
\$	aadbb\$	shift
\$a	adbb\$	shift
\$aa	dbb\$	shift
\$aad	bb\$	$S \rightarrow d$
\$aaS	bb\$	shift
\$aaSb	b\$	$B \rightarrow b$
\$aaSB	b\$	$S \rightarrow aSB$
\$aS	b\$	shift
\$aSb	\$	$B \rightarrow b$
\$aSB	\$	$S \rightarrow aSB$
\$S	\$	accept

$$S \xRightarrow{S \rightarrow aSB} aSB \xRightarrow{B \rightarrow b} aSb \xRightarrow{S \rightarrow aSB} aaSBb \xRightarrow{B \rightarrow b} aaSbb \xRightarrow{S \rightarrow d} aadbb$$

LR parsing arithmetic

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid a$$

$$E \Rightarrow E + T$$

$$\Rightarrow E + T * F$$

$$\Rightarrow E + T * a$$

$$\Rightarrow E + F * a$$

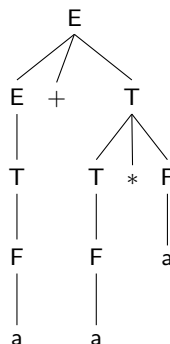
$$\Rightarrow E + a * a$$

$$\Rightarrow T + a * a$$

$$\Rightarrow F + a * a$$

$$\Rightarrow a + a * a$$

Stack	Input	Rule
\$	a+a*a\$	shift
\$a	+a*a\$	$F \rightarrow a$
\$F	+a*a\$	$T \rightarrow F$
\$T	+a*a\$	$E \rightarrow T$
\$E	+a*a\$	shift
\$E+	a*a\$	shift
\$E+a	*a\$	$F \rightarrow a$
\$E+F	*a\$	$T \rightarrow F$
\$E+T	*a\$	shift
\$E+T*	a\$	shift
\$E+T*a	\$	$F \rightarrow a$
\$E+T*F	\$	$T \rightarrow T*F$
\$E+T	\$	$E \rightarrow E+T$
\$E	\$	accept



LR(1) parsing

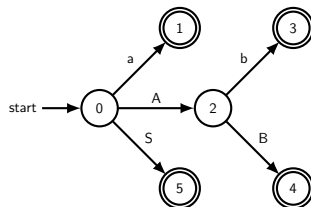
- ▶ The trick is to know when to shift and when to reduce.
- ▶ Hopefully by looking at **only one** symbol of the input.
- ▶ Everything on the stack has already been examined.
- ▶ We can use the entire stack to determine actions.
- ▶ We do this by using a DFA to keep track of stack state.
- ▶ We note each time a RHS appears on top of the stack.
- ▶ If a RHS is on top of the stack, a reduction is *possible*.
 - ▶ We can then choose whether to shift or reduce.
 - ▶ Otherwise you must shift.

LR(1) parsing

$$S \rightarrow AB$$
$$A \rightarrow a$$
$$B \rightarrow b$$

$$S \xRightarrow{S \rightarrow AB} AB \xRightarrow{B \rightarrow b} Ab \xRightarrow{A \rightarrow a} ab$$

Stack	Input	Rule
\$	ab\$	shift
\$a	b\$	$A \rightarrow a$
\$A	b\$	shift
\$Ab	\$	$B \rightarrow b$
\$AB	\$	$S \rightarrow AB$
\$S	\$	accept

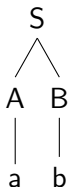


We will store the state of the DFA on the stack, too.

LR(1) parsing

$$S \rightarrow AB$$

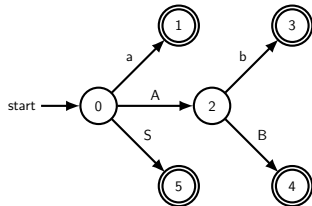
$$A \rightarrow a$$

$$B \rightarrow b$$


$$S \xRightarrow{S \rightarrow AB} AB \xRightarrow{B \rightarrow b} Ab \xRightarrow{A \rightarrow a} ab$$

	a	b	A	B	S	\$
0	1		2		5	
1		$A \rightarrow a$				
2		3		4		
3						$B \rightarrow b$
4						$S \rightarrow AB$
5						accept

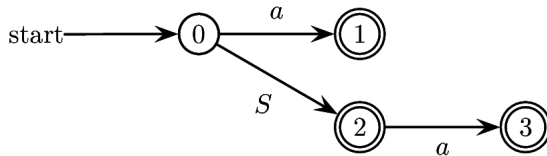
Stack	Input	Rule
0	ab\$	shift
0 a 1	b\$	$A \rightarrow a$
0 A 2	b\$	shift
0 A 2 b 3	\$	$B \rightarrow b$
0 A 2 B 4	\$	$S \rightarrow AB$
0 S 5	\$	accept



Left recursion: $S \rightarrow Sa \mid a$

Stack	Input
0	a a a a \$
0 a 1	a a a \$
0 S 2	a a a \$
0 S 2 a 3	a a \$
0 S 2	a a \$
0 S 2 a 3	a \$
0 S 2	a \$
0 S 2 a 3	\$
0 S 2	\$

	a	$\$$	S
0	1		2
1	$S \rightarrow a$		
2	3	accept	
3	$S \rightarrow Sa$	$S \rightarrow Sa$	



Right recursion: $S \rightarrow aS \mid a$

Stack

0

0 a 1

0 a 1 a 1

0 a 1 a 1 a 1

0 a 1 a 1 a 1 a 1

0 a 1 a 1 a 1 S 2

0 a 1 a 1 S 2

0 a 1 S 2

0 S 3

Input

a a a a \$

a a a \$

a a \$

a \$

\$

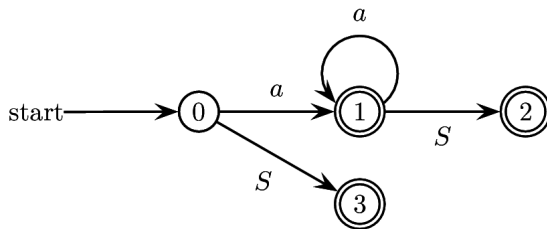
\$

\$

\$

\$

	a	$\$$	S
0	1		3
1	1	$S \rightarrow a$	2
2		$S \rightarrow aS$	
3		accept	



Middle recursion: $S \rightarrow aSa \mid bSb \mid c$

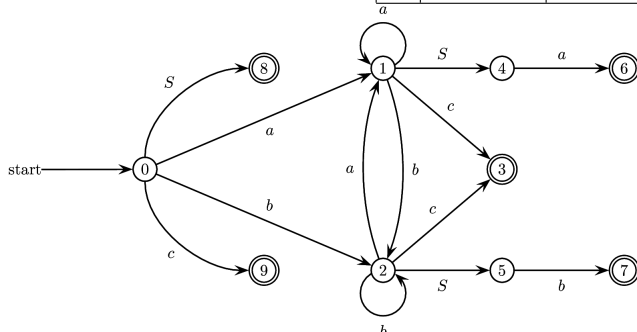
Stack

0
0 a 1
0 a 1 b 2
0 a 1 b 2 c 3
0 a 1 b 2 S 5
0 a 1 b 2 S 5 b 7
0 a 1 S 4
0 a 1 S 4 a 6
0 S 8

Input

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

	a	b	c	\$	S
0	1	2	9		8
1	1	2	3		4
2	1	2	3		5
3	$S \rightarrow c$	$S \rightarrow c$			
4	6				
5		7			
6	$S \rightarrow aSa$	$S \rightarrow aSa$		$S \rightarrow aSa$	
7	$S \rightarrow bSb$	$S \rightarrow bSb$		$S \rightarrow bSb$	
8				accept	
9				$S \rightarrow c$	

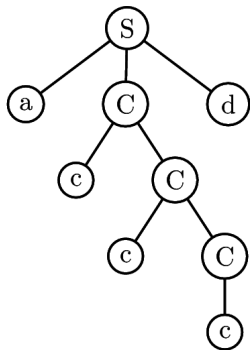


LR(1) parsing, a more complex example

$S \rightarrow aCd \mid bCD$

$C \rightarrow cC \mid c$

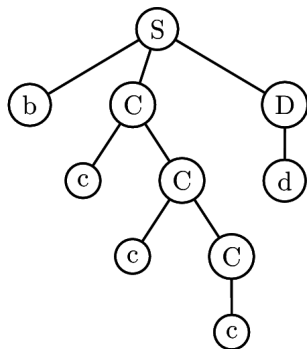
$D \rightarrow d$



Stack	Input	Rule
\$	acccd\$	shift
\$a	cccd\$	shift
\$ac	ccd\$	shift
\$acc	cd\$	shift
\$accc	d\$	$C \rightarrow c$
\$accC	d\$	$C \rightarrow cC$
\$acC	d\$	$C \rightarrow cC$
\$aC	d\$	shift
\$aCd	\$	$S \rightarrow aCd$
\$S	\$	accept

$S \Rightarrow aCd \Rightarrow acCd \Rightarrow accCd \Rightarrow acccd$

LR(1) parsing



Stack	Input	Rule
\$	bcccd\$	shift
\$b	cccd\$	shift
\$bc	ccd\$	shift
\$bcc	cd\$	shift
\$bccc	d\$	$C \rightarrow c$
\$bccC	d\$	$C \rightarrow cC$
\$bcC	d\$	$C \rightarrow cC$
\$bC	d\$	shift
\$bCd	\$	$D \rightarrow d$
\$bCD	\$	$S \rightarrow bCD$
\$S	\$	accept

$S \Rightarrow bCD \Rightarrow bCd \Rightarrow bcCd \Rightarrow bccCd \Rightarrow bcccd$

LR(1) parsing

$$S \rightarrow aCd \mid bCD$$

$$C \rightarrow cC \mid c$$

$$D \rightarrow d$$

- ▶ $S \Rightarrow aCd \Rightarrow acCd \Rightarrow accCd \Rightarrow acccd$
- ▶ $S \Rightarrow bCD \Rightarrow bCd \Rightarrow bcCd \Rightarrow bccCd \Rightarrow bcccd$
- ▶ At any point, the derivation string must look like one of these:

aCd ac^+Cd ac^+d bCD bCd bc^+Cd bc^+d

- ▶ Whenever we see one of these, we have to know which rule to apply at what point in the shifting of the string.

LR(1) parsing

$$S \rightarrow aCd \mid bCD$$

$$C \rightarrow cC \mid c$$

$$D \rightarrow d$$

Stack	Input	Rule	Peek
\$aCd	\$	$S \rightarrow aCd$	
\$ac ⁺ C	d\$	$C \rightarrow cC$	
\$ac ⁺	d\$	$C \rightarrow c$	d
\$bCD	\$	$S \rightarrow bCD$	
\$bCd	\$	$D \rightarrow d$	
\$bc ⁺ C	d\$	$C \rightarrow cC$	
\$bc ⁺	d\$	$C \rightarrow c$	d

Stack	Input	Rule
\$	bcccd\$	shift
\$b	cccd\$	shift
\$bc	ccd\$	shift
\$bcc	cd\$	shift
\$bccc	d\$	$C \rightarrow c$
\$bccC	d\$	$C \rightarrow cC$
\$bcC	d\$	$C \rightarrow cC$
\$bC	d\$	shift
\$bCd	\$	$D \rightarrow d$
\$bCD	\$	$S \rightarrow bCD$
\$S	\$	accept

Stack	Input	Rule
\$	acccd\$	shift
\$a	cccd\$	shift
\$ac	ccd\$	shift
\$acc	cd\$	shift
\$accc	d\$	$C \rightarrow c$
\$accC	d\$	$C \rightarrow cC$
\$acC	d\$	$C \rightarrow cC$
\$aC	d\$	shift
\$aCd	\$	$S \rightarrow aCd$
\$S	\$	accept

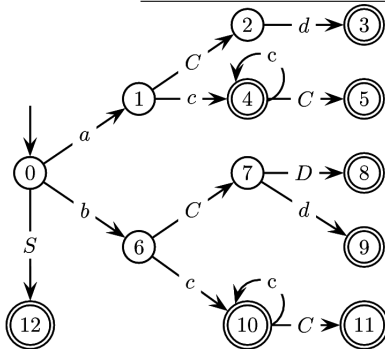
DFA for LR parsing

$$S \rightarrow aCd \mid bCD$$

$$C \rightarrow cC \mid c$$

$$D \rightarrow d$$

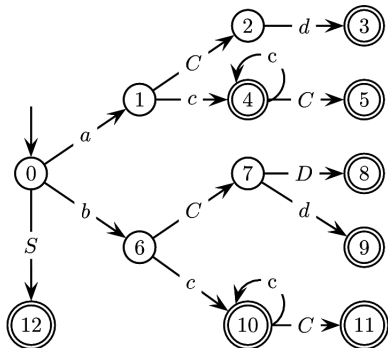
Stack	Input	Rule	Peek
\$aCd	\$	$S \rightarrow aCd$	
\$ac^+C	d\$	$C \rightarrow cC$	
\$ac^+	d\$	$C \rightarrow c$	d
\$bCD	\$	$S \rightarrow bCD$	
\$bCd	\$	$D \rightarrow d$	
\$bc^+C	d\$	$C \rightarrow cC$	
\$bc^+	d\$	$C \rightarrow c$	d



Using the DFA in LR parsing

Stack	Input	Rule	Peek
\$aCd	\$	$S \rightarrow aCd$	
\$ac^+C	d\$	$C \rightarrow cC$	
\$ac^+	d\$	$C \rightarrow c$	d
\$bCD	\$	$S \rightarrow bCD$	
\$bCd	\$	$D \rightarrow d$	
\$bc^+C	d\$	$C \rightarrow cC$	
\$bc^+	d\$	$C \rightarrow c$	d

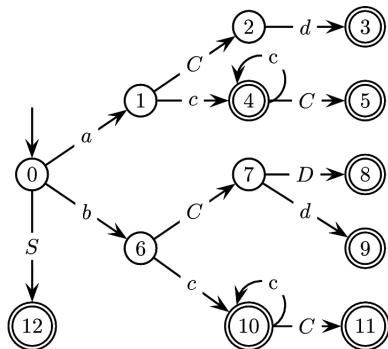
Stack	Input	Rule
0	acccd\$	shift
0 a 1	cccd\$	shift
0 a 1 c 4	ccd\$	shift
0 a 1 c 4 c 4	cd\$	shift
0 a 1 c 4 c 4 c 4	d\$	$C \rightarrow c$
0 a 1 c 4 c 4 C 5	d\$	$C \rightarrow cC$
0 a 1 c 4 C 5	d\$	$C \rightarrow cC$
0 a 1 C 2	d\$	shift
0 a 1 C 2 d 3	\$	$S \rightarrow aCd$
0 S 12	\$	accept



Using the DFA in LR parsing

Stack	Input	Rule	Peek
\$aCd	\$	$S \rightarrow aCd$	
\$ac^+C	d\$	$C \rightarrow cC$	
\$ac^+	d\$	$C \rightarrow c$	d
\$bCD	\$	$S \rightarrow bCD$	
\$bCd	\$	$D \rightarrow d$	
\$bc^+C	d\$	$C \rightarrow cC$	
\$bc^+	d\$	$C \rightarrow c$	d

Stack	Input	Rule
0	bcccd\$	shift
0 b 6	cccd\$	shift
0 b 6 c 10	ccd\$	shift
0 b 6 c 10 c 10	cd\$	shift
0 b 6 c 10 c 10 c 10	d\$	$C \rightarrow c$
0 b 6 c 10 c 10 C 11	d\$	$C \rightarrow cC$
0 b 6 c 10 C 11	d\$	$C \rightarrow cC$
0 b 6 C 7	d\$	shift
0 b 6 C 7 d 9	\$	$D \rightarrow d$
0 b 6 C 7 D 8	\$	$S \rightarrow bCD$
0 S 12	\$	accept



More examples in notes on repo.

LR(k) languages, Knuth's theorem

Theorem

$$\begin{aligned} LR(k) \text{ languages} &= LR(1) \text{ languages} \\ &= \text{deterministic context free languages} \end{aligned}$$

LR parsing exercises

Redo all the solved examples. Also, find DFAs and tables for the following languages, and trace some parses:

► $S \rightarrow a \mid b \mid c$

► $S \rightarrow aSa \mid b$

► $S \rightarrow ABC$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow c$

Lex and Yacc Style Parsers

- ▶ `http://epaperpress.com/lexandyacc/`
- ▶ `https://docs.racket-lang.org/parser-tools/index.html`

Non-LR language

Palindromes:

$$S \rightarrow \epsilon \mid a \mid b \mid aSa \mid bSb$$

- ▶ We never know where the middle is.
- ▶ We cannot know when to stop shifting and start reducing.

Stack	Input	Rule
\$	aababaa\$	Shift
\$ a	ababaa\$	Shift
\$ aa	babaa\$	Shift
\$ aab	abaa\$	Shift
\$ aaba	baa\$	$S \rightarrow a$???
\$ aabS	baa\$	Shift
\$ aabSb	aa\$	$S \rightarrow bSb$
\$ aaS	aa\$	Shift
\$ aaSa	a\$	$S \rightarrow aSa$
\$ aS	a\$	Shift
\$ aSa	\$	$S \rightarrow aSa$
\$ S	\$	Accept