

LL Parsing

LL(k) Grammars

- Subset of CFG's
- Permits deterministic left-to-right recognition with a look ahead of k symbols
- Builds the parse tree top-down
- If the correct production can be deduced from the partially constructed tree and the next k symbols in the unscanned string, for every possible step, then the grammar is said to be LL(k)
- If a parse table can be constructed for the grammar, then it is LL(k), if it can't, it is not LL(k)

CS 5300 - SJAllan

2

Properties of LL(k) Grammars

- Each LL(k) grammar is unambiguous
- An LL(k) has no left-recursion
 - Why?
- Problems with LL(k) parsing
 - Left recursion
 - Order of alternatives in grammar
 - Failure

CS 5300 - SJAllan

3

Removing Recursion

- Group A-productions as:
 $A \rightarrow A\alpha_1 | A\alpha_2 | \dots | A\alpha_m | \beta_1 | \beta_2 | \dots | \beta_n$
- Replace the A-productions by
 $A \rightarrow \beta_1 A' | \beta_2 A' | \dots | \beta_n A'$
 $A' \rightarrow \alpha_1 A' | \alpha_2 A' | \dots | \alpha_m A' |$

CS 5300 - SJAllan

4

Removing Recursion – Example 1

$Z \rightarrow E$	$Z \rightarrow E$
$E \rightarrow T E+T E-T$	$E \rightarrow TE'$
$T \rightarrow F T*F T/F$	$E' \rightarrow +TE' -TE' \epsilon$
$F \rightarrow a (E)$	$T \rightarrow FT'$
Grammar	$T' \rightarrow *FT' /FT' \epsilon$
	$F \rightarrow a (E)$
	Recursion removed

CS 5300 - SJAllan

5

Removing Recursion – Example 2

$S \rightarrow Aa b$	$S \rightarrow Aa b$	$S \rightarrow Aa b$
$A \rightarrow Ac Sd e$	$A \rightarrow Ac Aad bd e$	$A \rightarrow bdA' eA'$
Grammar	Removing recursion	$A' \rightarrow cA' adA' \epsilon$
		Removing immediate recursion

CS 5300 - SJAllan

6

Left Factoring

- Like factoring in mathematics
 - Take common parts of productions and form a new nonterminal
- Allows us to defer, until later, which alternative to take

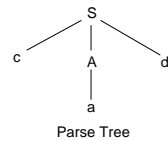
CS 5300 - SJAllan

7

Order of Alternatives

- The order in which alternatives are considered can affect the language accepted
- The string cabcd may not be accepted

$S \rightarrow cAd$
 $A \rightarrow a \mid ab$
 Grammar



CS 5300 - SJAllan

8

Failure

- We failure is reported, we have very little idea where the error actually occurred

CS 5300 - SJAllan

9

Deterministic LL(1) Parsers

- LL(1) grammars:
 - Table size grows exponentially with k
 - If the grammar is not LL(1), then it usually is not LL(k) for any k
 - Like LR parsers, we can parse LL grammars using tables

CS 5300 - SJAllan

10

LL(1) Parsing Algorithm

Algorithm 5.2 LL(1) Parse Algorithm
 { Input: A string ω and a parsing table M for grammar G . }
 { Output: If ω is in $L(G)$, a leftmost derivation of ω ; otherwise, an error indication. }

Initially, the parser is in a configuration in which it has S on the stack with S , the start symbol of G on top, and ω in the input buffer.

Set ip to point to the first symbol of ω .

repeat
 Let X be the top stack symbol and a the symbol pointed to by ip .
 if $X \in V_t$ or $\$$
 if $X = a$
 Pop X from the stack and advance ip .
 else
 error()
 end if
 else
 if $M[X, a] = X \rightarrow Y_1 Y_2 \dots Y_k$ { X is a nonterminal }
 Pop X from the stack
 Push Y_1, Y_2, \dots, Y_k onto the stack, with Y_1 on top
 Output the production $X \rightarrow Y_1 Y_2 \dots Y_k$
 else
 error()
 end if
 until $X = \$$ { stack is empty }

CS 5300 - SJAllan

11

LL(1) Parsing - Example

- $E \rightarrow TE^*$
- $E^* \rightarrow \epsilon$
- $E^* \rightarrow \epsilon$
- $T \rightarrow +T$
- $T \rightarrow -T$
- $T \rightarrow FT^*$
- $T \rightarrow \epsilon$
- $T^* \rightarrow FT^*$
- $F^* \rightarrow +F$
- $F^* \rightarrow -F$
- $F \rightarrow a$
- $F \rightarrow (E)$

Nonterminal	Input Symbol							
	a	+	-	*	/	()	\$
E	1					1		
E*		2	2				3	3
T		4	5					
T*	5					6		
T**		7	7	8	8		7	7
F*				9	10			
F	11					12		

$\{a^*(a+a), E\}$ $\{(a+a), (E)T^*E^*\}$ $\{+(a), +TE^*T^*E^*\}$ $\{(, E^*)\}$
 $\{a^*(a+a), TE^*\}$ $\{(a+a), E)T^*E^*\}$ $\{(a), TE^*T^*E^*\}$ $\{(, \epsilon)\}$
 $\{a^*(a+a), FT^*E^*\}$ $\{(a+a), TE^*T^*E^*\}$ $\{(a), FT^*E^*T^*E^*\}$
 $\{a^*(a+a), aT^*E^*\}$ $\{(a+a), FT^*E^*T^*E^*\}$ $\{(a), aT^*E^*T^*E^*\}$
 $\{(a+a), T^*E^*\}$ $\{(a+a), aT^*E^*T^*E^*\}$ $\{(), T^*E^*T^*E^*\}$
 $\{(a+a), FT^*E^*\}$ $\{(a+a), T^*E^*T^*E^*\}$ $\{(), E^*T^*E^*\}$
 $\{(a+a), FT^*E^*\}$ $\{(a+a), E^*T^*E^*\}$ $\{(), T^*E^*\}$
 $\{(a+a), FT^*E^*\}$ $\{(a+a), TE^*T^*E^*\}$ $\{(, T^*E^*)\}$

CS 5300 - SJAllan

12

Selector Table – Example 1

1. $S \rightarrow AS$
2. $A \rightarrow B$
3. $A \rightarrow C$
4. $B \rightarrow aC$
5. $C \rightarrow aa$

Grammar

Nonterminal	Input Sybol	
	a	\$
S	1	
A	2/3	
B	4	
C	5	

Selector Table

- This grammar is not LL(1)
 - Look at the conflict in the table

CS 5300 - SJAllan

13

Selector Table – Example 2

1. $S \rightarrow AB$
2. $A \rightarrow XY$
3. $A \rightarrow ZW$
4. $X \rightarrow xX$
5. $X \rightarrow \epsilon$
6. $Y \rightarrow yY$
7. $Y \rightarrow \epsilon$
8. $Z \rightarrow zZ$
9. $Z \rightarrow \epsilon$
10. $W \rightarrow wW$
11. $W \rightarrow c$
12. $B \rightarrow bB$
13. $B \rightarrow d$

Grammar

	First	Follow
S	z,w,c,x,y,b,d	\$
A	z,w,c,x,y	b,d
X	x, ϵ	y,b,d
Y	y, ϵ	b,d
Z	z, ϵ	w,c
W	w,c	b,d
B	b,c	\$

First and Follow sets

NT	Input Symbol							
	b	c	d	w	x	y	z	\$
S	1	1	1	1	1	1	1	
A	2	3	2	3	2	2	3	
X	5		5		4	5		
Y	7		7			6		
Z		9		9			8	
W		11		10				
B	12		13					

Selector Table

CS 5300 - SJAllan

14

Algorithm for Building Selector Table

```

Algorithm 5.3 LL(1) Selector Table Construction
{ Input: Grammar G. }
{ Output: Parsing table M. }

for each production  $A \rightarrow \alpha$  of the grammar do
  for  $a \in \text{First}(\alpha)$  do
    Add  $A \rightarrow \alpha$  to  $M[A, a]$ .
  end for
  if  $\epsilon \in \text{First}(\alpha)$ 
    Add  $A \rightarrow \alpha$  to  $M[A, b]$  for each  $b \in \text{Follow}(A)$ 
  end if
end for
Make each undefined entry of M be error.
    
```

CS 5300 - SJAllan

15