

## CMSC 330: Organization of Programming Languages

### Parser Examples 2

## Left Recursion Elimination Algorithm

- Given grammar
  - $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n \mid \beta$
- Rewrite grammar as
  - $A \rightarrow \beta L$
  - $L \rightarrow \alpha_1 L \mid \alpha_2 L \mid \dots \mid \alpha_n L \mid \epsilon$
- Repeat as necessary

CMSC 330

2

## Left Factoring Algorithm

- Given grammar
  - $A \rightarrow x\alpha_1 \mid x\alpha_2 \mid \dots \mid x\alpha_n \mid \beta$
- Rewrite grammar as
  - $A \rightarrow xL \mid \beta$
  - $L \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$
- Repeat as necessary

CMSC 330

3

## Example 1 – Left Factoring

Consider the grammar

$$\begin{aligned} S &\rightarrow T + S \mid T \\ T &\rightarrow U^* T \mid U \\ U &\rightarrow (S) \mid V \\ V &\rightarrow 0 \mid 1 \mid \dots \mid 9 \end{aligned}$$

Common prefix requires applying left factoring

$$\begin{aligned} S &\rightarrow T + S \mid T \\ T &\rightarrow U^* T \mid U \end{aligned}$$

Left factor S

$$\begin{aligned} S &\rightarrow T + S \mid T \\ &\downarrow \\ S &\rightarrow T L \\ L &\rightarrow + S \mid \epsilon \end{aligned}$$

Left factor T

$$\begin{aligned} T &\rightarrow U^* T \mid U \\ &\downarrow \\ T &\rightarrow U M \\ M &\rightarrow^* T \mid \epsilon \end{aligned}$$

CMSC 330

4

## Example 3 – Computing First Sets

Consider the grammar

$$\begin{aligned} S &\rightarrow AS \mid b \\ A &\rightarrow SA \mid a \end{aligned}$$

### Compute First Sets

$$\begin{aligned} \text{First}(S) &= \emptyset \text{ for now} \\ \text{First}(A) &= \emptyset \text{ for now} \\ \text{First}(b) &= \{b\} \\ \text{First}(a) &= \{a\} \\ \text{First}(A) &= \text{First}(SA) \cup \text{First}(a) \\ &= \text{First}(S) \cup \{a\} \\ &= \{a\} \text{ for now} \end{aligned}$$

$$\begin{aligned} \text{First}(S) &= \text{First}(AS) \cup \text{First}(b) \\ &= \text{First}(AS) \cup \{b\} \\ &= \text{First}(A) \cup \{b\} \\ &= \{a\} \cup \{b\} = \{a,b\} \\ \text{First}(SA) &= \text{First}(S) \\ &= \{a,b\} \\ \text{First}(A) &= \text{First}(SA) \cup \text{First}(a) \\ &= \text{First}(S) \cup \{a\} \\ &= \{a,b\} \cup \{a\} = \{a,b\} \\ \text{First}(S) &= \text{First}(AS) \cup \text{First}(b) \\ &= \text{First}(AS) \cup \{b\} \\ &= \text{First}(A) \cup \{b\} \\ &= \{a,b\} \cup \{b\} = \{a,b\} \end{aligned}$$

CMSC 330

5

## Example 3 – Using First Sets

Grammar

$$\begin{aligned} S &\rightarrow AS \mid b \\ A &\rightarrow SA \mid a \end{aligned}$$

### First sets for RHS

$$\begin{aligned} \text{First}(a) &= \{a\} \\ \text{First}(b) &= \{b\} \\ \text{First}(A) &= \{a,b\} \\ \text{First}(S) &= \{a,b\} \end{aligned}$$

Grammar is predictive if...

$$\begin{aligned} \text{First}(AS) \cap \text{First}(b) &= \emptyset \\ \text{First}(SA) \cap \text{First}(a) &= \emptyset \end{aligned}$$

And not left recursive

Verify...

$$\begin{aligned} \text{First}(AS) \cap \text{First}(b) &= \{a,b\} \cap \{b\} \\ &= \{b\} \\ \text{First}(SA) \cap \text{First}(a) &= \{a,b\} \cap \{a\} \\ &= \{a\} \end{aligned}$$

Overlap! Grammar is not predictive

CMSC 330

6

### Example 3 – Ambiguous Grammars

Grammar  
 $S \rightarrow AS \mid b$   
 $A \rightarrow SA \mid a$

c) Can the grammar be parsed by a backtracking parser?  
 No, due to (mutual) left recursion,  
 where  $S \rightarrow AS$  derives  $S \rightarrow SAS$

d) Is the grammar ambiguous?  
 Yes. 2 left-most derivations of "abab"  
 e) Are all ambiguous grammars non-parseable by predictive parsers?  
 Yes. RHS guaranteed to conflict  
 f) Are all non-ambiguous grammars parseable by predictive parsers?  
 No. Consider  $S \rightarrow aab \mid aac$

CMSC 330

7

### Example 4 – Computing First Sets

Consider the grammar  
 $S \rightarrow (L) \mid a$   
 $L \rightarrow L,S \mid S$

Compute First Sets  
 $First(S) = \emptyset$  for now  
 $First(L) = \emptyset$  for now  
 $First((L)) = \{ ( \}$   
 $First(a) = \{ a \}$

$First(S) = First((L)) \cup First(a)$   
 $= \{ ( \} \cup \{ a \} = \{ (, a \}$   
 $First(L) = First(L,S) \cup First(S)$   
 $= First(L) \cup First(S)$   
 $= \emptyset \cup First(S)$  for now  
 $= \{ (, a \}$  for now  
 $First(L,S) = First(L)$   
 $= \{ (, a \}$  for now  
 $First(L) = First(L,S) \cup First(S)$   
 $= First(L) \cup First(S)$   
 $= \{ (, a \} \cup \{ (, a \} = \{ (, a \}$   
 $First(L,S) = First(L)$   
 $= \{ (, a \}$

CMSC 330

8

### Example 4 – Using First Sets

Grammar  
 $S \rightarrow (L) \mid a$   
 $L \rightarrow L,S \mid S$

First sets for RHS  
 $First(a) = \{ a \}$   
 $First((L)) = \{ ( \}$   
 $First(S) = \{ (, a \}$   
 $First(L,S) = \{ (, a \}$

Grammar is predictive if...  
 $First((L)) \cap First(a) = \emptyset$   
 $First(L,S) \cap First(S) = \emptyset$   
 And not left recursive

Verify...  
 $First((L)) \cap First(a)$   
 $= \{ ( \} \cap \{ a \}$   
 $= \emptyset$   
 $First(L,S) \cap First(S)$   
 $= \{ (, a \} \cap \{ (, a \}$   
 $= \{ (, a \}$

Overlap! Grammar is not predictive

CMSC 330

9

### Example 4 – Rewriting the Grammar

Grammar  
 $S \rightarrow (L) \mid a$   
 $L \rightarrow L,S \mid S$

c) Can the grammar be parsed by a backtracking parser?  
 No, due to left recursion,  
 $L \rightarrow L,S$

d) Rewrite grammar using the rule for eliminating left recursion  
 $S \rightarrow (L) \mid a$   
 $L \rightarrow L,S \mid S$   
 $\downarrow$   
 $S \rightarrow (L) \mid a$   
 $L \rightarrow SM$   
 $M \rightarrow , SM \mid \epsilon$

CMSC 330

10

### Example 4 – Recomputing First Sets

Grammar  
 $S \rightarrow (L) \mid a$   
 $L \rightarrow SM$   
 $M \rightarrow , SM \mid \epsilon$

e) Compute First Sets  
 $First(S) = \{ (, a \}$   
 $First(L) = \{ (, a \}$   
 $First(M) = \{ \epsilon, , \}$   
 $First(a) = \{ a \}$

f) Grammar is predictive if...  
 $First((L)) \cap First(a) = \emptyset$   
 $First(, SM) \cap First(\epsilon) = \emptyset$   
 And not left recursive

Verify...  
 $First((L)) \cap First(a)$   
 $= \{ ( \} \cap \{ a \}$   
 $= \emptyset$   
 $First(, SM) \cap First(\epsilon)$   
 $= \{ , \} \cap \{ \epsilon \}$   
 $= \emptyset$

No overlap, grammar is predictive

CMSC 330

11

### Example 4 – Recursive Descent Parser

Grammar  
 $S \rightarrow (L) \mid a$   
 $L \rightarrow SM$   
 $M \rightarrow , SM \mid \epsilon$

g) Parser  

```

parse_S() {
    if (lookahead == "(") {
        match("("); parse_L();
        match(")"); // S -> (L)
    }
    else if (lookahead == "a")
        match("a"); // S -> a
    else error();
}
    
```

```

parse_L() {
    if ((lookahead == "(") ||
        (lookahead == "a")) {
        parse_S(); // L -> SM
        parse_M();
    }
}

parse_M() {
    if (lookahead == ",") {
        match(","); parse_S();
        parse_M(); // M -> , SM
    }
    else return; // M -> ε
}
    
```

CMSC 330

12

## Example 4 – Parsing Input

Grammar  
 $S \rightarrow (L) \mid a$   
 $L \rightarrow SM$   
 $M \rightarrow , SM \mid \epsilon$

Lookahead in red

Parse "(a,a)"	Remaining
parse_S()	"(a,a)"
match("(")	"(a,a)"
parse_L()	"a,a)"
parse_S()	"a,a)"
match("a")	"a,a)"
parse_M()	",a)"
match(",")	",a)"
parse_S()	"a)"
match("a")	"a)"
parse_M()	")"
match(")")	"")"
	""

CMSC 330

13

## Example 5 – Computing First Sets

Consider the grammar  
 $E \rightarrow E + T \mid T$   
 $T \rightarrow a \mid ( E )$

Compute First Sets

First(E) =  $\emptyset$  for now  
 First(T) =  $\emptyset$  for now  
 First(a) = { a }  
 First(( E )) = { ( }

First(T) = First(a)  $\cup$  First(( E ))  
 = { a }  $\cup$  { ( } = { (, a }  
 First(E) = First(E+T)  $\cup$  First(T)  
 = First(E)  $\cup$  First(T)  
 =  $\emptyset$   $\cup$  First(T) for now  
 = { (, a } for now  
 First(E+T) = First(E)  
 = { (, a } for now  
 First(E) = First(E+T)  $\cup$  First(T)  
 = { (, a }  $\cup$  { (, a } = { (, a }

CMSC 330

14

## Example 5 – Rewriting the Grammar

Grammar  
 $E \rightarrow E + T \mid T$   
 $T \rightarrow a \mid ( E )$

- b) Is the grammar ambiguous?  
 No. Unique left-most derivations.
- c) Can the grammar be parsed by a predictive parser?  
 No, since  $\text{First}((E+T)) \cap \text{First}(T) = \{ (, a \}$

- d) Can the grammar be parsed by a backtracking parser?  
 No, due to left recursion, where  $E \rightarrow E + T$
- e) Rewrite the grammar using the rule for eliminating left recursion  
 $E \rightarrow E + T \mid T$   
 $T \rightarrow a \mid ( E )$   
 $\downarrow$   
 $E \rightarrow T L$   
 $L \rightarrow + T L \mid \epsilon$   
 $T \rightarrow a \mid ( E )$

CMSC 330

15

## Example 5 – Recomputing First Sets

Consider the grammar  
 $E \rightarrow T L$   
 $L \rightarrow + T L \mid \epsilon$   
 $T \rightarrow a \mid ( E )$

f) Compute First Sets

First(a) = { a }  
 First(( E )) = { ( }

First(T) = First(a)  $\cup$  First(( E ))  
 = { a }  $\cup$  { ( } = { (, a }  
 First(E) = First(T L)  
 = First(T) = { (, a }  
 First(+ T L) = { + }  
 First( $\epsilon$ ) = {  $\epsilon$  }  
 First(L) = First(+ T L)  $\cup$  First( $\epsilon$ )  
 = { + }  $\cup$  {  $\epsilon$  } = { +,  $\epsilon$  }

CMSC 330

16

## Example 5 – Using First Sets

Consider the grammar  
 $E \rightarrow T L$   
 $L \rightarrow + T L \mid \epsilon$   
 $T \rightarrow a \mid ( E )$

First(a) = { a }  
 First(( E )) = { ( }  
 First(+ T L) = { + }  
 First( $\epsilon$ ) = {  $\epsilon$  }

- g) Grammar is predictive if...  
 $\text{First}(+ T L) \cap \text{First}(\epsilon) = \emptyset$   
 $\text{First}(a) \cap \text{First}(( E )) = \emptyset$   
 And not left recursive

Verify...  
 $\text{First}(+ T L) \cap \text{First}(\epsilon)$   
 = { + }  $\cap$  {  $\epsilon$  }  
 =  $\emptyset$   
 $\text{First}(a) \cap \text{First}(( E ))$   
 = { a }  $\cap$  { ( }  
 =  $\emptyset$

No overlap, grammar is predictive

CMSC 330

17

## Example 5 – Recursive Descent Parser

Grammar  
 $E \rightarrow T L$   
 $L \rightarrow + T L \mid \epsilon$   
 $T \rightarrow a \mid ( E )$

Recursive descent parser

```

parse_E() {
    // E → TL
    if ((lookahead == "(") ||
        (lookahead == "a")) {
        parse_T(); parse_L();
    }
    else error();
}

parse_L() {
    if (lookahead == "+") { // L → +TL
        match("+"); parse_T(); parse_L();
    }
    else { // L → ε
        ;
    }
}

parse_T() {
    if (lookahead == "a") // T → a
        match("a");
    else if (lookahead == "(") { // T → (E)
        match("("); parse_E(); match(")");
    }
    else error();
}
    
```

CMSC 330

18

## Example 5 – Parsing Input

Grammar  
 $E \rightarrow T L$   
 $L \rightarrow + T L \mid \epsilon$   
 $T \rightarrow a \mid ( E )$

Lookahead in red

Parse "a+a+a"	Remaining
parse_E()	"a+a+a"
parse_T()	"a+a+a"
match("a")	"a+a+a"
parse_L()	"a+a"
match("+")	"a+a"
parse_T()	"a+a"
match("a")	"a+a"
parse_L()	"a"
match("+")	"a"
parse_T()	"a"
match("a")	"a"
parse_L()	""
	""

CMSC 330

19

## Example 5 – A Slightly Different Grammar

Consider the grammar  
 $E \rightarrow T + E \mid T$   
 $T \rightarrow a \mid ( E )$

Vs. previous grammar  
 $E \rightarrow E + T \mid T$   
 $T \rightarrow a \mid ( E )$

- Can the grammar be parsed by a predictive parser?  
 No, since  $\text{First}(T+E) \cap \text{First}(T) \neq \emptyset$
- Would grammar accept same language?  
 Yes – all sums of a's
- What is the difference between this grammar and the previous grammar rewritten to eliminate left recursion?  
 This grammar is right associative  
 Previous grammar is left associative

CMSC 330

20