

# CMSC 330: Organization of Programming Languages

## Parser Examples 2

### Example 3 – Computing First Sets

Consider the grammar

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

Compute First Sets

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

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

CMSC 330

2

### Example 3 – Using First Sets

Grammar

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

Grammar is predictive if...

$\text{First}(AS) \cap \text{First}(b) = \emptyset$   
 $\text{First}(SA) \cap \text{First}(a) = \emptyset$   
 And not left recursive

First sets for RHS

$\text{First}(a) = \{a\}$   
 $\text{First}(b) = \{b\}$   
 $\text{First}(A) = \{a,b\}$   
 $\text{First}(S) = \{a,b\}$

Verify...

$\text{First}(AS) \cap \text{First}(b)$   
 $= \{a, b\} \cap \{b\}$   
 $= \{b\}$   
 $\text{First}(SA) \cap \text{First}(a)$   
 $= \{a, b\} \cap \{a\}$   
 $= \{a\}$

Overlap! Grammar is not predictive

CMSC 330

3

### 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

4

### Example 4 – Computing First Sets

Consider the grammar

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

Compute First Sets

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

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

CMSC 330

5

### Example 4 – Using First Sets

Grammar

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

First sets for RHS

$\text{First}(a) = \{a\}$   
 $\text{First}((L)) = \{(,)$   
 $\text{First}(S) = \{(, a)$   
 $\text{First}(L,S) = \{(, a)$

Grammar is predictive if...

$\text{First}((L)) \cap \text{First}(a) = \emptyset$   
 $\text{First}(L,S) \cap \text{First}(S) = \emptyset$   
 And not left recursive

Verify...

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

Overlap! Grammar is not predictive

CMSC 330

6

## 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

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

CMSC 330

7

## Example 4 – Recomputing First Sets

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

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

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

Verify...

$First(L) \cap First(a)$   
 $= \{ (, a \} \cap \{ a \}$   
 $= \emptyset$   
 $First(S, M) \cap First(\epsilon)$   
 $= \{ , \} \cap \{ \epsilon \}$   
 $= \emptyset$

No overlap, grammar is predictive

CMSC 330

8

## Example 4 – Recursive Descent Parser

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

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

CMSC 330

9

## Example 4 – Parsing Input

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

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,a)"
match(",")	"a,a)"
parse_S()	"a)"
match("a")	"a)"
parse_M()	")"
match(")")	"")"
	""

CMSC 330

10

## 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

11

## 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  $First(E+T) \cap First(T) = \{ (, a \}$

- 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

12

## Example 5 – Recomputing First Sets

Consider the grammar

```
E → TL
L → + TL | ε
T → a | ( E )
```

f) Compute First Sets

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

```
First(T) = First(a) ∪ First(( E ))
           = { a } ∪ { ( } = { (, a }
First(E) = First(TL)
           = First(T) = { (, a }
First(+ TL) = { + }
First(ε) = { ε }
First(L) = First(+ TL) ∪ First(ε)
           = { + } ∪ { ε } = { +, ε }
```

CMSC 330

13

## Example 5 – Using First Sets

Consider the grammar

```
E → TL
L → + TL | ε
T → a | ( E )
```

```
First(a) = { a }
First(( E )) = { ( }
First(+ TL) = { + }
First(ε) = { ε }
```

g) Grammar is predictive if...

```
First(+ TL) ∩ First(ε) = ∅
First(a) ∩ First(( E )) = ∅
And not left recursive
```

Verify...

```
First(+ TL) ∩ First(ε)
= { + } ∩ { ε }
= ∅
First(a) ∩ First(( E ))
= { a } ∩ { ( }
= ∅
```

No overlap, grammar is predictive

CMSC 330

14

## Example 5 – Recursive Descent Parser

Grammar

```
E → TL
L → + TL | ε
T → a | ( 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

15

## Example 5 – Parsing Input

Grammar

```
E → TL
L → + TL | ε
T → a | ( E )
```

Lookahead in red

Parse "a+a+a"

```
parse_E()
parse_T()
  match("a")
parse_L()
  match("+")
parse_T()
  match("a")
parse_L()
  match("+")
parse_T()
  match("a")
parse_L()
  match("a")
parse_T()
  match("a")
parse_L()
  ""
```

Remaining

```
"a+a+a"
"a+a+a"
"a+a+a"
"+a+a"
"a+a"
"a+a"
"+a"
"a"
"a"
""
```

CMSC 330

16

## Example 5 – A Slightly Different Grammar

Consider the grammar

```
E → T + E | T
T → a | ( E )
```

Vs. previous grammar

```
E → E + T | T
T → a | ( E )
```

a) Can the grammar be parsed by a predictive parser?

No, since  $\text{First}(T+E) \cap \text{First}(T) \neq \emptyset$

b) Would grammar accept same language?

Yes – all sums of a's

c) 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

17