

CMSC 330: Organization of Programming Languages

Practice Problem 3 Solutions

Recursive Descent Parser

- For terminals, create function `match(a)`
 - If lookahead is `a` it consumes the lookahead by advancing the lookahead to the next token, and returns
- For each nonterminal `N`, create a function `parse_N`
 - Let $N \rightarrow \beta_1 \mid \dots \mid \beta_k$ be the productions of `N`
 - For each production $N \rightarrow \beta_i$ (where $\beta_i = \alpha_1 \dots \alpha_n$)
 - If the lookahead is in `First(β_i)`
 - Call `parse_ α_1 (; ... ; parse_ α_n (`

CMSC 330

3

Left Recursion Elimination Algorithm

- Given grammar
 - $A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \dots \mid A\alpha_n \mid \beta$
- 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

5

Refresher

- Recursive descent parser
- First sets
- Rewriting grammar
 - Eliminating left recursion
 - Left factoring

CMSC 330

2

First Sets

- Definition
 - `First(γ)`, for any terminal or nonterminal γ , is the set of initial terminals of all strings that γ may expand to
- For a terminal `a`
 - `First(a) = { a }`
- For a nonterminal `N`
 - If $N \rightarrow \epsilon$, then add `ϵ` to `First(N)`
 - If $N \rightarrow \alpha_1 \alpha_2 \dots \alpha_n$, then add
 - `First(α_i)` if $\epsilon \notin \text{First}(\alpha_i)$
 - Else add `{ First(α_1) - ϵ } \cup First($\alpha_2 \dots \alpha_n$)`

CMSC 330

4

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

6

Problem 1a

Consider the grammar

```
S → ABd | aBc
A → ε
B → b | c
```

Compute First Sets

```
First(b) = {b}
First(c) = {c}
First(B) = First(b) ∪ First(c)
          = {b,c}
```

```
First(ε) = {ε}
First(A) = First(ε) = {ε}

First(ABd) = {First(A) - ε} ∪ First(Bd)
            = {ε - ε} ∪ First(Bd)
            = First(Bd) = First(B) = {b, c}

First(aBc) = {a}

First(S) = First(ABd) ∪ First(aBc)
          = {b, c} ∪ {a}
          = {a,b,c}
```

CMSC 330

7

Problem 1b

Grammar

```
S → ABd | aBc
A → ε
B → b | c
```

First sets for RHS

```
First(b) = {b}
First(c) = {c}

First(ABd) = {b, c}
First(aBc) = {a}
```

Grammar is predictive if...

```
First(ABd) ∩ First(aBc) = ∅
First(b) ∩ First(c) = ∅
And not left recursive...
```

Verify...

```
First(ABd) ∩ First(aBc)
= {b, c} ∩ {a}
= ∅

First(b) ∩ First(c)
= {b} ∩ {c}
= ∅
```

No overlap, grammar is predictive

CMSC 330

8

Problem 1c

Grammar

```
S → ABd | aBc
A → ε
B → b | c
```

Recursive descent parser

```
parse_B() {
  if (lookahead == "b")
    match("b"); // B → b
  else if (lookahead == "c")
    match("c"); // B → c
  else error();
}

parse_A() { return; } // A → ε

parse_S() {
  // S → ABd
  if ((lookahead == "b" ||
       lookahead == "c")) {
    parse_A(); parse_B(); match("d");
  }
  // S → aBc
  else if (lookahead == "a") {
    match("a"); parse_B(); match("c");
  }
  else error();
}
```

CMSC 330

9

Problem 1d

Grammar

```
S → ABd | aBc
A → ε
B → b | c
```

Lookahead in red

Parse "bd"

```
parse_S()
parse_A()
parse_B()
match("b")
match("d")
```

Remaining

```
"bd"
"bd"
"bd"
"bd"
"d"
..."
```

Parse "acc"

```
parse_S()
match("a")
parse_B()
match("c")
match("c")
```

Remaining

```
"acc"
"acc"
"cc"
"cc"
"cc"
..."
```

CMSC 330

10

Problem 2a

Consider the grammar

```
S → AS | b
A → SA | a
```

Compute First Sets

```
First(S) = ∅ for now
First(A) = ∅ for now
First(b) = {b}
First(a) = {a}
First(A) = First(SA) ∪ First(a)
          = First(S) ∪ {a}
          = {a} for now
```

```
First(S) = First(AS) ∪ First(b)
          = First(AS) ∪ {b}
          = First(A) ∪ {b}
          = {a} ∪ {b} = {a,b}

First(SA) = First(S)
          = {a,b}
First(A) = First(SA) ∪ First(a)
          = First(S) ∪ {a}
          = {a,b} ∪ {a} = {a,b}

First(S) = First(AS) ∪ First(b)
          = First(AS) ∪ {b}
          = First(A) ∪ {b}
          = {a,b} ∪ {b} = {a,b}
```

CMSC 330

11

Problem 2b

Grammar

```
S → AS | b
A → SA | a
```

First sets for RHS

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

First(A) = {a,b}
First(S) = {a,b}
```

Grammar is predictive if...

```
First(AS) ∩ First(b) = ∅
First(SA) ∩ First(a) = ∅
And not left recursive
```

Verify...

```
First(AS) ∩ First(b)
= {a, b} ∩ {b}
= {b}

First(SA) ∩ First(a)
= {a, b} ∩ {a}
= {a}
```

Overlap! Grammar is not predictive

CMSC 330

12

Problem 2c to 2f

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

13

Problem 3a

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

14

Problem 3b

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

15

Problem 3c to 3d

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

16

Problem 3e to 3f

Grammar

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

- e) Compute First Sets

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

- f) Grammar is predictive if...

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

Verify...

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

No overlap, grammar is predictive

CMSC 330

17

Problem 3g

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

18

Problem 3h

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

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

19

Problem 4a

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

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

20

Problem 4b to 4e

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

- 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
 $E \rightarrow E + T | T$
 $T \rightarrow a | (E)$
 \downarrow
 $E \rightarrow TL$
 $L \rightarrow + TL | \epsilon$
 $T \rightarrow a | (E)$

CMSC 330

21

Problem 4f

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

- f) Compute First Sets
 First(a) = { a }
 First((E)) = { (}

First(T) = First(a) \cup First((E))
 = { a } \cup { (} = { (, a }
 First(E) = First(TL)
 = First(T) = { (, a }
 First(+ TL) = { + }
 First(ϵ) = { ϵ }
 First(L) = First(+ TL) \cup First(ϵ)
 = { + } \cup { ϵ } = { +, ϵ }

CMSC 330

22

Problem 4g

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

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

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

No overlap, grammar is predictive

CMSC 330

23

Problem 4h

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

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 -> epsilon
  }
}

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

CMSC 330

24

Problem 4i

Grammar

```
E → TL
L → +TL | ε
T → a | (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

25

Problem 5

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

26

Problem 6a to 6b

- Rewrite the following grammars so they can be parsed by a predictive parser by eliminating left recursion and applying left factoring where necessary.

a) $S \rightarrow S + a | b$
 \downarrow
 $S \rightarrow bL$
 $L \rightarrow +aL | \epsilon$

b) $S \rightarrow S + a | S + b | c$
 \downarrow
 $S \rightarrow cL$
 $L \rightarrow +aL | +bL | \epsilon$
 \downarrow
 $S \rightarrow cL$
 $L \rightarrow +M | \epsilon$
 $M \rightarrow aL | bL$

CMSC 330

27

Problem 6c to 6g

c) $S \rightarrow S + a | S + b | \epsilon$
 \downarrow
 $S \rightarrow L$
 $L \rightarrow +aL | +bL | \epsilon$
 \downarrow
 $S \rightarrow L$
 $L \rightarrow +M | \epsilon$
 $M \rightarrow aL | bL$

d) $S \rightarrow abc | ac$
 \downarrow
 $S \rightarrow aL$
 $L \rightarrow bc | c$

e) $S \rightarrow abc | abbb$
 \downarrow
 $S \rightarrow abL$
 $L \rightarrow c | b$

f) $S \rightarrow abc | ab$
 \downarrow
 $S \rightarrow abL$
 $L \rightarrow c | \epsilon$

g) $S \rightarrow aa | ab | ac$
 \downarrow
 $S \rightarrow aL$
 $L \rightarrow a | b | c$

CMSC 330

28

Problem 6h to 6k

h) $S \rightarrow aa | ab | a$
 \downarrow
 $S \rightarrow aL$
 $L \rightarrow a | b | \epsilon$

i) $S \rightarrow aa | ab | \epsilon$
 \downarrow
 $S \rightarrow aL | \epsilon$
 $L \rightarrow a | b$

j) $S \rightarrow aSc | aSb | b$
 \downarrow
 $S \rightarrow aSL | b$
 $L \rightarrow c | b$

k) $S \rightarrow aSc | aSb | a$
 \downarrow
 $S \rightarrow aL$
 $L \rightarrow Sc | Sb | \epsilon$
 \downarrow
 $S \rightarrow aL$
 $L \rightarrow SM | \epsilon$
 $M \rightarrow c | b$

CMSC 330

29

Problem 6l

l) $S \rightarrow aSc | aS | a$
 \downarrow
 $S \rightarrow aL$
 $L \rightarrow Sc | S | \epsilon$
 \downarrow
 $S \rightarrow aL$
 $L \rightarrow SM | \epsilon$
 $M \rightarrow c | \epsilon$

CMSC 330

30