## Programming Language Translation

## Practical 4 Handin

## <Caron, Litha, Luke>

### Task 1 Palindromes (6 marks)

1. Does grammar 1 describe palindromes? If not, why not?

No. Both two possible productions are non-terminating (Palin1 cannot be derived to terminal symbols).

Is it an LL(1) grammar? If not, why not?

Yes, it is an LL(1) grammar. The two alternative productions for Palin1 have first sets that are disjoint.

2. Does grammar 2 describe palindromes? If not, why not?

?

Is it an LL(1) grammar? If not, why not?

No. It is not an LL(1) grammar, based on rule 1. There are alternative productions for non-terminal Palin2. The sets of initial terminal symbols for that can be generated from each of the alternatives are not disjoint— “a” is in the first set of production alternatives 1 and 3; “b” is in the first set of production alternatives 2 and 4.

Palin2 = "a" Palin2 "a" . (1)

Palin2 = "b" Palin2 "b" . (2)

Palin2 = "a" . (3)

Palin2 = "b" . (4)

FIRST(Palin2 1) = {“a”}

FIRST (Palin2 2) = {“b”}

FIRST (Palin2 3) = {“a”}

FIRST (Palin2 4) = {“b”}

FIRST(Palin2 1) ∩ FIRST (Palin2 3) = {“a”} This violates Rule 1.

FIRST (Palin2 2) ∩ FIRST (Palin2 4) = {“b”} This violates Rule 1.

3. Does grammar 3 describe palindromes? If not, why not?

Grammar 3 describes a small set of palindromes (aaa, bbb, aba, bab, aa, bb).

Is it an LL(1) grammar? If not, why not?

No. It is not an LL(1) grammar. Grammar 3 violates rule 2. This is because Palin3 is nullable, and the terminals preceding and following Palin3 are the same i.e. the first sets and follow sets are not disjoint.

Test for rule 1:

Rewrite G3 as:

Palin3 = "a" X "a" . (1)

Palin3 = "b" Y "b" . (2)

X = Palin3 | E . (3,4)

Y = Palin3 | E . (5,6)

First (Palin3 1) = {“a”}

First (Palin3 2) = {“b”} Rule 1 passes as the first sets for Palin3 are distinct

First (Palin3) = {“a”,”b”}

First (X 3) = First (Palin3) = {“a”,”b”}

First (X4) = Empty

First (X) = {“a”,”b”}

First (Y5) = First (Palin3) = {“a”,”b”}

First (Y6) = Empty

First (Y) = {“a”,”b”}

Rule 1 passes as the first sets for X are distinct and the first sets for Y are distinct.

X and Y are both nullable.

Follow (X) = {“a”}. First(X) and Follow(X) are not distinct sets, as both contain “a”. Therefore rule 2 is violated.

Follow (Y) = {“b”}. First(Y) and Follow(Y) are not distinct sets, as both contain “b”. Therefore rule 2 is violated.

4. Does grammar 4 describe palindromes? If not, why not?

Yes.

Is it an LL(1) grammar? If not, why not?

No. Grammar 4 violates rules 1 and 2 of LL(1) grammars. The first sets of the alternatives for Palin4 are not distinct (violating rule 1). The first sets and follow sets of Palin4 (which is nullable) are not distinct (violating rule 2).

5. Can you find a better grammar to describe palindromes? If so, give it, if not, explain why not.

It would be better to use a context-sensitive grammar to describe palindromes. A context-sensitive grammar could recognise productions of the form anbncn, whereas the context-free grammar of COCOL productions can only recognise anban.

### Task 2 Expressions again [8 marks]

Is this an ambiguous grammar? (Hint: try to find an expression that can be parsed in more than one way). Give the parse trees (hand drawn figures are fine).

Yes. Expressions is an ambiguous grammar. Certain expressions can be parsed in more than one way to create two leftmost parse trees.

A graph on a notebook

Description automatically generated

Is expression.atg LL(1) compliant?

No, it is not LL(1) compliant. Because of Rule 2 (the first sets and follow sets of Expression are not distinct (respectively).

If not, why not and can you find a suitable grammar that is LL(1)? Hand in the .atg file of the new grammar.

Prove, by applying Rule 1 and Rule 2 (if applicable) that the new grammar is LL(1) compliant.

Grammar:

PRODUCTIONS

Expression = Ex {"^" Ex } EOF.

Ex = Term { ( "+" | "-" ) Term} .

Term = Factor { ( "\*" | "/" ) Factor } .

Factor = "a" | "b" | "c" .

END Expression.

Refactored:

Expression = Ex | B\* (1,2)

B = “^” B | e (3,4)

Exp = Term | C (5,6)

C = “+” Term | “-“ Term | e (7,8,9)

Term = Factor | D (10,11)

D = “\*” Factor | “/” Factor | e (12,13,14)

Factor = “a” | “b” | “c” (15,16,17)

Test rule 1

First sets

First (Factor15) = {a}

First(Factor16) = {b}

First(Factor17) = {c}

First (Factor) = {a,b,c} Rule 1 compliant

First(D12) = \*

First (D13) = /

First (D14) = Empty

First (D) = {\*,/} Rule 1 compliant

First (Term10) = First(Factor) = {a,b,c}

First (Term11) = First(D) = {\*,/}

First (Term) = {a,b,c,\*,/} Rule 1 compliant

First (C7) = “+”

First(C8) = “-“

First(C9) = Empty

First (C) = {+,-}

Rule 1 compliant

First (Exp5) = First (Term) = {a,b,c,\*,/}

First (Exp6) = First (C) = {+,-}

First (Exp) = {a,b,c,\*,/,+,-}

Rule 1 compliant

First (B3) = “^”

First (B4) = Empty

First (B) = {^}

Rule 1 compliant

First (Expression1) = First(Ex) = {a,b,c,\*,/,+,-}

First (B) = {^}

Rule 1 compliant

Test rule 2 nullable alternatives