Programming Languages - Lab 2: Grammars, Parse Trees, & Automata

Name: Answer Key

1. **[5pts]** Describe the language denoted by the regular expression: 0(0|1)*0
   It generates the set of strings that begin and end with a 0, with zero or more 0s and/or 1s in between.

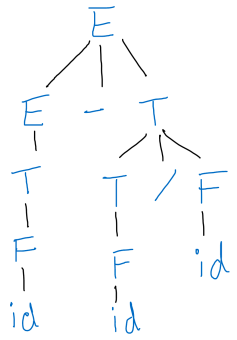
2. **[5pts]** Write a regular expression for the language that includes all strings of 0's and 1's with an even number of 0's followed by an odd number of 1's.
   (00)*1(11)*

3. **[12pts]** Consider the following grammar consisting of terminals {-, /, (,), id}, the set of non-terminals {E, T, F}, the start symbol E, and the following rules:
   $E \rightarrow T$
   $E \rightarrow E - T$
   $T \rightarrow F$
   $T \rightarrow T / F$
   $F \rightarrow id$
   $F \rightarrow (E)$
   a. Construct a leftmost derivation for the same string.
   b. Construct a rightmost derivation for the same string.
   c. Construct a parse tree for the string: id – id / id

   (a) Leftmost:
   $E \rightarrow E - T$
   $\rightarrow T - T$
   $\rightarrow F - T$
   $\rightarrow id - T$
   $\rightarrow id - T / F$
   $\rightarrow id - F / F$
   $\rightarrow id - id / F$
   $\rightarrow id - id / id$

   (b) Rightmost:
   $E \rightarrow E - T$
   $\rightarrow E - T / F$
   $\rightarrow E - T / id$
   $\rightarrow E - F / id$
   $\rightarrow E - id / id$
   $\rightarrow T - id / id$
   $\rightarrow F - id / id$
   $\rightarrow id - id / id$

   (c) Parse Tree

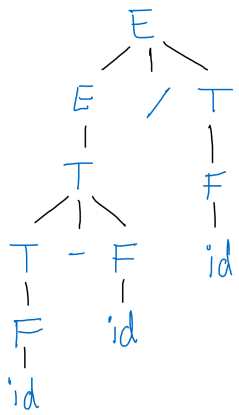


4. **[8pts]** Replace the productions $E \rightarrow E - T$ and $T \rightarrow T / F$ from the grammar described in Problem 3 with the productions $E \rightarrow E / T$ and $T \rightarrow T - F$ respectively, so that the resulting rules are as follows:

   $E \rightarrow T$
   $E \rightarrow E / T$
   $T \rightarrow F$
   $T \rightarrow T - F$
   $F \rightarrow id$
   $F \rightarrow (E)$

   a. Construct a new parse tree for the string: id – id / id
   b. What is the difference? Is the new grammar ambiguous? Defend.
   It is not ambiguous, but it gives subtraction higher precedence than division.

   (a) Parse Tree



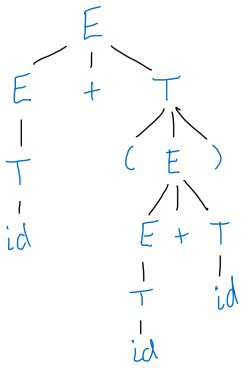
5. **[12pts]** Consider the CFG consisting of terminals {+, (,), id}, non-terminals {E, T}, start symbol E, and production rules:

   E → E + T | T
   T → (E) | id

   Give the rightmost and leftmost derivations for id + (id + id). Is this grammar ambiguous? Defend!

   Rightmost:
   E → E + T
   → E + (E)
   → E + (E + T)
   → E + (E + id)
   → E + (T + id)
   → E + (id + id)
   → T + (id + id)
   → id + (id + id)

   Leftmost:
   E → E + T
   → T + T
   → id + T
   → id + (E)
   → id + (E + T)
   → id + (T + T)
   → id + (id + T)
   → id + (id + id)



   It is not ambiguous. Different derivations produce just one parse tree.

Questions 6 – 9 refer to the following context free grammar:

   A → B & A | B
   B → B @ C | C
   C → C * x | x | (A)

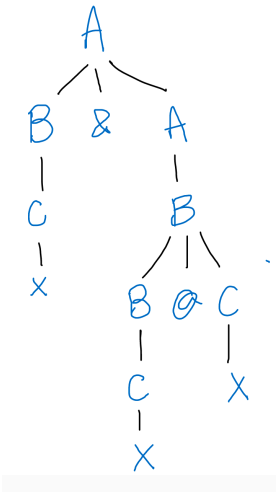
6. **[10pts]** Indicate True or False for each of the following statements.
   a. The & operator has higher precedence than the @ operator. False
   b. The & operator has higher precedence than the * operator. False
   c. The & operator associates to the left. False
   d. The * operator associates to the left. True
   e. The grammar is ambiguous. False

   Hint: binary + is left associative: id + id + id => (id + id) + id Also, try drawing the parse tree for id + id + id

7. **[5pts]** The abstract syntax tree for x*x&x@x@x is the same as which of the following?
   a. (((x*x)&x)@x)@x
   b. (x*x)&((x@x)@x) b.This one
   c. (x*x)&(x@(x@x))
   d. x*((x&x)@(x@x))

8. **[5pts]** Which of the following is a rightmost derivation? Discussed in pages 34-40 of Mead.
   a. A → B&A → B&B → B&B@C → B&B@x → B&C@x → B&x@x → C&x@x → x&x@x. a. This one
   b. A → B&A → C&A → C&B → C&B@C → C&C@C → C&C@x → C&x@x → x&x@x
   c. A → B&A → B&B → C&B → C&B@C → x&B@C → x&C@C → x&x@C → x&x@x
   d. A → B&A → C&A → x&A → x&B → x&B@C → x&C@C → x&x@C → x&x@x

9. **[5pts]** All derivations in the previous question correspond to the same parse tree. Draw that parse tree.



10. **[20pts (4each)]** Provide a regular expression for the regular grammars (a and b), a regular grammar *or* an FSA for the regular expression (c), and either a regular expression or a regular grammar for the FSAs (d and e).

a. $S \rightarrow aA$
$A \rightarrow bB$
$B \rightarrow aA \mid a$

(ab)+a

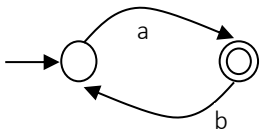
b. $S \rightarrow aA$
$A \rightarrow bS \mid \epsilon$

a(ba)*

$S \rightarrow aS \mid bS \mid \epsilon$

c. (a | b)*

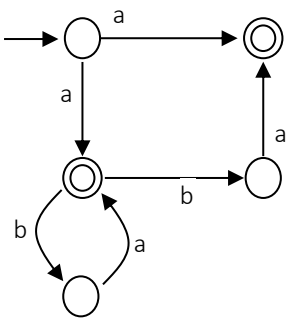
d.



a(ba)*

$S \rightarrow aA$
$A \rightarrow bS \mid \epsilon$

e.



a(ba)*

$S \rightarrow aA$
$A \rightarrow bS \mid \epsilon$

11. **[6pts]** Activity 1 on p. 37 of the Mead book.

Derivation 1:

Exp → Exp subT Exp
→ intT subT Exp
→ intT subT Exp subT Exp
→ intT subT intT subT Exp
→ intT subT intT subT intT
→ 10 – 4 – 3
→ 9

Derivation 2:

Exp → Exp subT Exp
→ Exp subT Exp subT Exp
→ Exp subT Exp subT intT
→ Exp subT intT subT intT
→ intT subT intT subT intT
→ 10 – 4 – 3
→ 3

12. **[6pts]** Activity 2 on p. 38 of the Mead book.

Two different parse trees for the string x,x,x