

# CMSC 430 Practice Problems

1. Describe the languages denoted by the following regular expressions:

- (a)  $0(0|1)^*0$
- (b)  $((\text{epsilon}|0)1^*)^*$
- (c)  $(0|1)^*0(0|1)(0|1)$

2. Write regular expressions for the following languages.

- (a) All strings of 0's and 1's that do not contain the substring 011.
- (b) All strings of 0's and 1's that do not contain the subsequence 011.

Substring is any contiguous sequence of characters found in a string. Subsequence is any sequence of characters (contiguous or non-contiguous) found in a string. For instance, the string "110010" contains the substrings "110" (first 3 characters), "100" (2nd, 3rd, 4th characters), and "010" (last 3 characters). It contains the subsequences "111" (1st, 2nd, 5th characters) and "000" (3rd, 4th, 6th characters), though neither are substrings.

3. For the regular expressions  $(a|b)^*$  and  $(a^*|b^*)^*$

- (a) Construct an NFA using Thompson's construction algorithm
- (b) Show the sequence of moves in parsing "ababab"
- (c) Convert the NFA to a DFA using subset construction algorithm
- (d) Minimize the DFA using the partitioning algorithm

4. Write a grammar for the following languages. Are the grammars ambiguous?:

- (a) All strings of 0's and 1's that have the same number of 0's and 1's.
- (b) All strings of 0's and 1's that have more 0's than 1's.
- (c) All balanced pairs of left and right parentheses (e.g., "()", "()()", "(()())").

5. Consider the grammar

$S ::= ( L ) | a$   
 $L ::= L , S | S$

For each of the following strings "(a,a)" and "(a,(a,a))"

- (a) Find the parse tree
- (b) Find the leftmost derivation

(c) Find the rightmost derivation

6. Consider the grammar

$S ::= aSbS | bSaS | \text{epsilon}$

- (a) Show grammar is ambiguous by constructing two leftmost derivations for "abab"
- (b) Show grammar is ambiguous by constructing two rightmost derivations for "abab"

7. Consider the grammar

$S ::= ( L ) | a$

$L ::= L , S | S$

- (a) Eliminate left recursion from the grammar
- (b) Compute FIRST and FOLLOW for each non-terminal
- (c) Build a non-backtracking recursive descent parser, given the following code:

```

tok; // current token

match(x) { // matches token
    if (tok != x) // if wrong token
        error(); // exit with error
    tok = getToken(); // get new token
}

parser() {
    tok = getToken(); // initialize
    S(); // start symbol
    match("$"); // match EOF
}

```

(d) Show an example parse of the string "(a,a)"

8. Consider the grammar

$S ::= AS | b$

$A ::= SA | a$

- (a) Compute FIRST and FOLLOW for each non-terminal
- (b) Construct the set of LR(0) items for the grammar
- (c) Construct the set of LR(1) items for the grammar
- (d) Construct the LR(1) action/goto tables for the grammar
- (e) List any shift/reduce or reduce/reduce conflicts. What is the effect if we always shift for a shift/reduce conflict? What is the effect if we always reduce for a shift/reduce conflict?
- (f) Show an example parse of the string "abab"

9. Consider the grammar

$S ::= Aa | bAc | Bc | bBa$

$A ::= d$

$B ::= d$

- (a) Show that the grammar is not LALR(1)