

CMSC 430 Practice Problems

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

- (a) $0(0|1)^*0$
- (b) $((\epsilon|0)1^*)^*$
- (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.

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 "ababbab"
- (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

$S ::= aSbS | bSaS | \epsilon$

- (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) Build a non-backtracking recursive descent parser, given the following code:

```

tok;    // current token

match(x) {
    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
}

```

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

8. Consider the grammar

$S ::= AS | b$
 $A ::= SA | a$

- (a) Construct the set of LR(0) items for the grammar
- (b) Construct the set of LR(1) items for the grammar
- (c) Construct the LR(1) action/goto tables for the grammar
- (d) 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?

- (e) 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)

10. Consider the following grammar, which generates expressions formed by applying "+" to integer and floating point constants. When two integers are added, the result is integer, otherwise, it is a float.

$E \rightarrow E + T \mid T$
 $T \rightarrow \text{num} . \text{num} \mid \text{num}$

- (a) Give a syntax-directed definition to determine the type of each subexpression. Assign each symbol an attribute "type".
(b) Derive a parse tree for the expression "5 + 4 + 3.2 + 1" Annotate the parse tree, showing the computation of the "type" attribute for each grammar symbol.

11. Given the following C declarations:

```
typedef struct {
    int a, b;
} CELL, *PCELL;
CELL foo[100];
PCELL bar(int x, CELL y) { ... }
```

- (a) write the type expression for "foo"
(b) write the type expression for "bar"

12. Consider the following grammar, which generates expressions for a number of operators. Suppose the type of each operator is a range of integers.

$E \rightarrow E + E \mid E * E \mid \dots$

- (a) Write the type-checking rules which calculates the range for each subexpression.

13. Consider the following lexically nested C code:

```
int a, b;
int foo() { int a, c; }
int bar() { int a, d; /* HERE */ }
```

- (a) How can symbol tables represent the state of each scope at the point marked HERE? Draw a diagram.
(b) What symbols are visible/not visible at point HERE?

14. Translate the arithmetic expression

$a * - (b + c)$

into

- (a) a syntax tree
(b) postfix notation
(c) 3-address code
(d) stack code