Theory of Computation

Midterm Exam.

Fall' 2002 (YEN)

Name: ....

I.D.#: ....

1. (30 pts) True or false (mark **O** for 'true'; **X** for 'false'). (Score=Max{0, Right-$\frac{1}{2}$Wrong}.)

(1) **X** If L_1 is regular and $L_2 \subseteq L_1$, then L_2 is regular as well.
Solution: counterexample—
$L_1 = \{a\}^*$ and $L_2 = \{a^p \mid p \text{ } is \text{ } prime\}$
L_1 is regular and $L_2 \subseteq L_1$, but L_2 is not regular

(2) **X** If L_1 is regular and L_2 is not regular, then $L_1 \cup L_2$ is not regular.
Solution: counterexample—
$L_1 = \{a\}^*$ and $L_2 = \{a^p \mid p \text{ } is \text{ } prime\}$
L_1 is regular and L_2 is not regular, but $L_1 \cup L_2$ is regular

(3) **O** If L_1 is regular and $L_1 \cup L_2$ is not regular, then L_2 is not regular.
Solution: we know that regular language is closed under union, i.e.,

$$L_1 \text{ } is \text{ } regular \ \wedge \ L_2 \text{ } is \text{ } regular \implies L_1 \cup L_2 \text{ } is \text{ } regular$$

Now the right hand side of the proposition above is not *true*, so L_1 or L_2 is not regular. With the fact that L_1 is regular, L_2 is not regular.

(4) **X** If L_1 is regular and L_2 is not regular, then $L_1 \cap L_2$ is not regular.
Solution: counterexample—
$L_1 = \phi$ and $L_2 = \{a^p \mid p \text{ } is \text{ } prime\}$
$L_1 \cap L_2 = \phi$ is regular

(5) **X** If L_1 and $L_1 - L_2$ are regular, then L_2 must be regular.
Solution: counterexample—
$L_1 = \phi$ and $L_2 = \{a^p \mid p \text{ } is \text{ } prime\}$
$L_1 - L_2 = L_1 \cap \overline{L_2} = \phi \cap \overline{L_2} = \phi$ is regular,
but L_2 is not regular.

(6) **X** $\{a^i b^{i+j} c^j | i, j \geq 0\}$ is regular.

(7) **X** $\{(a^n b)^n | n \geq 1\}$ is context free.

(8) **X** $\{(a^n b)^m | m, n \geq 1\}$ is context free.

(9) **O** $\{a^m b^n c^p d^q | m+n=p+q\}$ is context free.

(10) **O** $\{wcw^R cw | w \in \{a,b\}^*\}$ ($\subseteq \{a,b,c\}^*$) can be represented as the intersection of two context free languages.

Solution: It can be represented as the intersection of the following two context-free languages—

$$L_1 = \{\omega c x c \omega | \omega, x \in \{a,b\}^*\}$$
$$L_2 = \{\omega c \omega^R c x | \omega, x \in \{a,b\}^*\}$$

(11) **O** We define L_1/L_2=$\{x \mid \exists y \in L_2$, such that $xy \in L_1\}$.
If both L_1 and L_2 are regular, so is L_1/L_2.

(12) **X** If L_1/L_2 and L_1 are context free, then L_2 must be regular.

(13) **X** $\{a^m b^n | m,n \geq 1\}/\{b^n | n \geq 1\}$=$\{a^m | m \geq 1\}$.

Solution: It should be
$$\{a^m b^* | m \geq 1\}$$

(14) **X** If L_1 is regular and L_2 is not regular, then L_1L_2 (the concatenation of L_1 and L_2) cannot be regular.

(15) **O** If L_1 is context-free and L_2 is regular, then $L_1 - L_2$ is context free.

Solution: See problem 2.(f)

(16) **X** If L_1 is regular and L_2 is context free, then $L_1 - L_2$ is context free.

Solution: See problem 2.(f)

(17) **O** If L is context free, then L^R (=$\{x^R | x \in L\}$) is also context free.

Solution: we can always find a CFG to produce L^R: by *replacing the string on right-hand-side of every production rule with its reverse*, for example,
if L is produced by the grammar G:

$$A \to 01B$$
$$B \to 2\#$$

then L^R can be produced by G^R:

$$A \to B10$$
$$B \to \#2$$

which is made by replacing $01B$ on the RHS of first production rule of G with $B10$, and replacing $2\#$ with $\#2$ in the second rule.

(18) **O** $\{xxxx | x \in \{0,1\}^*\}$ can be accepted by a nondeterministic linear bounded

automaton (LBA).

(19) **O** Let $L(\subseteq \Sigma^*)$ be a regular language.
Then $Enlarge(L)$=$\{x \in \Sigma^* \mid$ for some $y \in \Sigma^*, xy \in L\}$ is also regular.

(20) **O** Every regular language can be accepted by some NFA with exactly 15 final states.
Solution: Every regular language can be accepted by some NFA, which might has an arbitrary number of final states. We can turn it into an NFA with exactly one final state, and then an NFA with exactly 15 final states:

(a) Introduce a new state into the NFA, and add λ transitions from every final state to it.

(b) Replace the NFA's set of final state with the set that consists of only the newly introduced state. That is, the newly introduced state is now the only one designated as the final state. The resulted NFA is equivalent to the original one, but has exactly one final state.

(c) Introduce another 14 states into the NFA, and add λ transitions from the only one final state to them. Then designate the 14 new states as the final states as well. Now the NFA is equivalent ot the original one, but has exactly 15 final states.

(21) **O** For every n, $L_n = \{a^i b^i \mid i \leq n\}$ is regular.
Solution: It's regular because it's finite.

(22) **X** Nondeterministic and deterministic versions of PDAs are equivalent.
Solution: Example 2.10 on p.105 of the textbook tells us that it's not the case.

(23) **X** If a language L satisfies the conditions stated in the pumping lemma for CFLs, then L is context free.
Solution: Pumping lemma is a necessary condition, but not a sufficient one!

(24) **X** Every infinite set of strings over a single letter alphabet Σ (=$\{a\}$) contains an infinite context free subset.

(25) **X** $L_1 = L_2$ if and only if $L_1^* = L_2^*$.
Solution: counterexample—
let $L_1 = \{a, aa\}$, $L_2 = \{a^p | p$ is 1 or is prime$\}$
$L_1 \neq L_2$, but $L_1^* = L_2^* = a^*$

(26) **O** For any languages L_1, L_2 and L_3, $L_1(L_2 \cap L_3) \subseteq (L_1L_2) \cap (L_1L_3)$
Solution: see the next problem

(27) **X** For any languages L_1, L_2 and L_3, $(L_1L_2) \cap (L_1L_3) \subseteq L_1(L_2 \cap L_3)$.
Solution: counterexample—
let $L_1 = \{a, aa\}$, $L_2 = \{a^i | i \ is \ odd\}$, $L_3 = \{a^i | i \ is \ even\}$;
$(L_1L_2) \cap (L_1L_3) = \{a\}$, $L_1(L_2 \cap L_3) = \phi$

(28) **O** Every non-regular language is infinite.

Solution: because for every finite language, we can construct a finite state automaton to accept it.

(29) **X** A context-free language is inherently ambiguous iff there exists an ambiguous context-free grammar generating the language.

Solution: by definition, ***inherently ambiguous*** context-free languages are languages that can only be generated by ambiguous grammars.

(30) **X** The intersection of two non-context-free languages cannot be context-free.

Solution: counterexample—
The intersection of the following two non-context-free languages

$$\{a^i b^i c^i \mid i \geq 0\}$$

and

$$\{a^i b^j c^i \mid 1 < i < j < 2i\}$$

is ϕ, which is context-free.

2. (30 pts) Answer the following questions.

   (a) (3 pts) Let $FLIP(A) = \{ww^R \mid w \in A\}$. ($w^R$ denotes the reversal of w.) Is there a regular language X such that $FLIP(X)$ is not regular? Why?
   Solution:
   If $X = \{10^i1 \mid i \geq 0\}$, then $FLIP(X) = \{10^i110^i1 \mid \imath \geq 0\}$, which is not regular.

   (b) (3 pts) Write down the set of strings expressed by the regular expression: $(\lambda \cup 0 \cup 10)(\lambda \cup 1)$. ($\lambda$ denotes the empty string, and 0 and 1 are two symbols.)
   Solution: $\{\lambda, 0, 1, 01, 10, 101\}$

   (c) (3 pts) Define *configurations* for finite automata and pushdown automata.
   Solution:
   for FSA: <u>current state</u> & <u>rest of the input string</u>
   for PDA: <u>current state</u> & <u>rest of the input string</u> & <u>current stack content</u>

   (d) (3 pts) Eliminate the λ rule in the following CFG (i.e., write down an equivalent one without the λ rule).
   $S \to AAA$
   $A \to a \mid b \mid \lambda$
   Solution:
   $S \to aaa \mid aab \mid aba \mid abb \mid baa \mid bab \mid bba \mid bbb \mid aa \mid ab \mid ba \mid bb \mid a \mid b \mid \lambda$
   or
   $S \to AAA \mid AA \mid A \mid \lambda$
   $A \to a \mid b$

   (e) (4 pts) List four operations under which the context-free languages are closed.
   Solution: Union, concatenation, star, reverse

   (f) (8 pts) Suppose L is context-free and R is regular.

   (1) Is $L - R$ context-free? Why?
   Solution:
   It is known that regular language is closed under complement and that CFL is closed under intersection with regular languages.
   $L - R = L \cap \overline{R}$, so $L - R$ is context-free

   (2) Is $R - L$ context-free? Why?
   Solution:
   It is known that CFL is <u>NOT</u> closed under complement and that CFL is closed under intersection with regular languages.
   $R - L = R \cap \overline{L}$, so $R - L$ is not necessarily context-free

   (g) (6 pts)

   (1) State Myhill-Nerode theorem.
   Solution:
   A language L over alphabet Σ is non-regular $\Leftrightarrow$ there is an infinite subset of Σ^* such that for each pair of string x and y in it, exactly one of xz and yz is in L for some $z \in \Sigma^*$.

(2) Instead of using the pumping lemma, use Myhill-Nerode theorem to show $\{ww \mid w \in \Sigma^*\}$ to be non-regular.
Solution:
$L = \{ww \mid w \in \Sigma^*\}$.
A non-regularity proof by Myhill-Nerode requires finding an infinite set S of strings and a string for each pair of strings of S.

Consider the set of string $S = \{a^i \mid i \in N\}$. Let a^m and a^n be arbitrary two different members of S, where $m, n \in N$ and $m \neq n$. Select ba^mb as a string to be appended to a^m and a^n, then a^mba^mb is in L while a^nba^mb is not. Since a^m and a^n are an arbitrary pair of strings of S, S satisfies the conditions of Myhill-Nerode theorem. Hence L is non-regular.

3. (10 pts) Let $L = \{a^ib^jc^i \mid i \leq j \leq 2i\}$. Prove that L is NOT context-free.
Solution:
Assume that L is a CFL and obtain a contradiction.
Let p be the pumping length for L that is guaranteed to exist by the pumping lemma. Select the string $s = a^pb^pc^p$. Clearly s is a member of L and of length at least p.
Condition 3 says that $|vxy| \leq p$, which implies the following five cases:

a. $|vxy|$ lies entirely within a^p (or c^p): pumping s up or down will make the number of a not eqaul to the number of c, and therefore not a member of L. Condition 1 is violated and a contradiction occurs.

b. v (y) contains only characters of a (c), and y (v) contains characters of a and b (b and c): pumping s up will make the string of the form $a^xb^ya^zb^uc^v$ ($a^xb^yc^zb^uc^v$), and therefore not a member of L. Condition 1 is violated and a contradiction occurs.

c. v (y) contains only characters of a (c), and y (v) contains only characters of b: pumping s up or down will make the number of a not eqaul to the number of c, and therefore not a member of L. Condition 1 is violated and a contradiction occurs.

d. v (y) contains characters of a and b (b and c), and y (v) contains only characters of b: similar to case **b.**.

e. $|vxy|$ lies entirely within b^p: pumping s up will make the number of b more than two times the number of a, and therefore not a member of L. That violates condition 1 of the lemma and is thus a contradiction.

4. (10 pts) Consider the infinite 2-dimensional grid $G = \{(m, n) \mid m \text{ and } n \text{ are integers}\}$. Every point in G has 4 neighbors, $North, South, East$, and $West$. Starting at the origin $(0, 0)$, a string of commands N, S, E, W generates a path in G. For instance the string NESW generates a path clockwise around a unit square touching the origin. Say that a path is *closed* if it starts at the origin and ends at the origin. Let C be the set of all strings over $\Sigma = \{N, S, E, W\}$ that generate a closed path.

(a) Is C context-free? Give a convincing proof.

Solution:
C is the set of strings in which $\#(N) = \#(S)$ and $\#(E) = \#(W)$
Assume that C is a CFL and obtain a contradiction using the pumping lemma.
Let p be the pumping length for C that is guaranteed to exist by the pumping lemma. Select the string $s = N^p E^p S^p W^p$. Clearly s is a member of C and of length at least p.
By condition 3 of the pumping lemma ($|vxy| \leq p$), it is easily seen that v and y can only contain at most two kinds of commands next to each other (N and E, E and S, or S and W), and pumping s will cause $\#(N) \neq \#(S)$ or $\#(E) \neq \#(W)$, which violates the condition 1 and a contradiction occurs.

(b) Describe in English two CFLs A and B, such that $C = A \cap B$.
Solution:
A is a CFL in which every string has an equal number of N and S.
B is a CFL in which every string has an equal number of E and W.

5. (10 pts)

(1) Let B=$\{1^k y \mid y \in \{0,1\}^*$, y contains at *least* k 1s, for $k \geq 1\}$. Show that B is regular.
Solution:
$1^k y = 11^{k-1} y = 1z$, where $z = 1^{k-1} y$. $\Rightarrow$ z contains at least $(k-1) + k = 2k - 1$ 1s, where $k \geq 1$.
And since $k \geq 1$, $(2k - 1) \geq (2 \times 1 - 1) = 1$
B=$\{1^k y \mid y \in \{0,1\}^*$, y contains at least k 1s, for $k \geq 1\}$=$\{1z \mid z \in \{0,1\}^*$, z contains at least one 1$\}$, which can be easily checked by a FSA.

(2) Let C=$\{1^k y \mid y \in \{0,1\}^*$, y contains at *most* k 1s, for $k \geq 1\}$. Show that C is NOT regular.
Solution:
Assume that C is regular and obtain a contradiction. Let p be the pumping length for C that is guaranteed to exist by the pumping lemma. Select the string $s = 1^p 0 1^p$. Clearly s is a member of C and of length at least p.
Condition 3 of the pumping lemma says $|xy| \leq p$, which implies that y lies entirely within the front 1^p. Pumping s down will make the 1s in y outnumber those in the front 1^p. A contradiction occurs.

6. (10 pts) You are given two DFAs $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$. Suppose you want to construct a DFA $M = (Q, \Sigma, \delta, q, F)$ to accept $L(M_1) - L(M_2)$ (i.e., the set of strings that are in $L(M_1)$ but are not in $L(M_2)$). Suppose you use the *product construction*, so $Q = Q_1 \times Q_2$. Write down δ, q, and F precisely.
Solution:
$\delta((x, y), a) = (\delta_1(x, a), \delta_2(y, a))$, where $(x, y) \in (Q_1 \times Q_2)$, and $a \in \Sigma$
$q = (q_1, q_2)$
$F = F_1 \times (Q_2 - F_2)$

7. (10 pts) A Deterministic Counter Automaton (DCA) is a deterministic pushdown automaton whose stack alphabet consists of just one symbol. Just like a PDA, at every transition a DCA can push or pop a symbol from the stack, and can test for the stack being empty.

(a) Give a non-regular language accepted by a DCA.
Solution: $L = \{a^i b^i \mid i \in N\}$

(b) Is the language $L = \{w\#w^R \mid w \in \{0,1\}^*\}$ accepted by a DCA? Prove it. (Hint: How many configurations are possible for a DCA when it reaches $\#$?)

Solution:

There are $2^{|w|}$ possibilities for a string of length $|w|$. And it implies that there must be $2^{|w|}$ different configurations for the machine to remember the string it reads so far.

However, from the structure of DCA, we know that the DCA can only offer at most $|Q| \times |w|$ different configurations (even if it changes states and pushes or pops the stack every time it reads a character), where $|Q|$ is the number of states the DCA has, which is finite. $2^{|w|}$ will outnumber $|Q| \times |w|$ when $|w|$ grows long enough. With pigeon hole principle, we know that it's impossible for a DCA to accept L.