Harvard University
Computer Science 121
Midterm — October 23, 2012

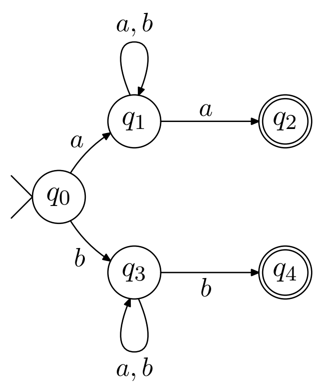
This is a closed-book examination. You may use any result from lecture, Sipser, problem sets, or section, as long as you quote it clearly. The alphabet is $\Sigma = \{a, b\}$ except where otherwise stated. You have 80 minutes. The problems total 75 points. Use pen and write your name on all bluebooks you use. Good luck!

PROBLEM 1 (1+1+1+1+1 points)

For each of the following, say whether it is a string, language, or neither. No justification necessary.
(A) Σ^* (B) ε (C) $\emptyset$ (D) $\{\emptyset\}$ (E) $\{\varepsilon\}$
(A) Language (B) String (C) Language (D) Neither (E) Language

PROBLEM 2 (3+4+3+5 points)

Consider the following NFA N:



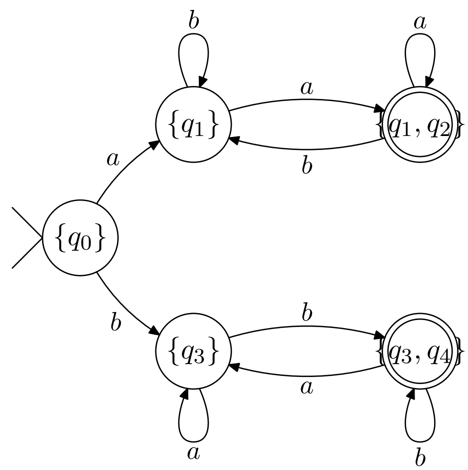
(A) Which of the following strings are accepted by N? bb, $abaa$, abb.
(B) Write out the formal 5-tuple for N.
(C) Describe in English the language $L(N)$.
(D) Convert the NFA to a DFA using the Subset Construction. (You may simply draw the state diagram of the DFA, omitting unreachable states.)

(A) Yes, Yes, No.
(B) $(\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, \delta, q_0, \{q_2, q_4\})$, with δ given by

	a	b	ε
q_0	$\{q_1\}$	$\{q_3\}$	$\emptyset$
q_1	$\{q_1, q_2\}$	$\{q_1\}$	$\emptyset$
q_2	$\emptyset$	$\emptyset$	$\emptyset$
q_3	$\{q_3\}$	$\{q_3, q_4\}$	$\emptyset$
q_4	$\emptyset$	$\emptyset$	$\emptyset$

(C) All strings over $\Sigma = \{a, b\}$ of length at least two where the first and last characters are the same.

(D)



PROBLEM 3 (5 points)

For a string $w \in \{a, b\}^*$, define $\overline{w}$ to be the string where all a's are replaced with b's and vice-versa. For example, $\overline{aabab} = bbaba$. Give a context-free grammar for the language $\{w \in \{a, b\}^* : w^R = \overline{w}\}$.

Solution: Denote w as $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_n$, where $\sigma_i \in \Sigma = \{a, b\}$. Then $w^R = \overline{w}$ if and only if $\sigma_i = \overline{\sigma_{n-i+1}}$ for every i. We can design a CFG as follows. $G = (\{S\}, \{a, b\}, \{S \to aSb, S \to bSa, S \to \epsilon\}, S)$. The main idea is that, in each step, if G generates σ in the first position, then it must generate $\overline{\sigma}$ in the last position.

PROBLEM 4 (4+4+4+4 points)

TRUE or FALSE? Justify your answers in one or two sentences.
(A) If L_1 is regular and $L_2 \subseteq L_1$, then L_2 is regular
(B) $bababbba \in L(((aba \cup ba)^*b)^*)$
(C) There exists a language L such that L^* is uncountable
(D) The grammar $S \to SS|a$ is ambiguous

(A) FALSE: Let $L_1 = \Sigma^*$ and $L_2 = \{a^n b^n : n \in \mathbb{N}\}$. We proved in class that L_2 is non-regular, and $L_2 \subseteq L_1$, so the answer must be FALSE.

(B) FALSE: We see that because the regular expression is of the form $((X)^*b)^*$, where $X = aba \cup ba$, we know that any string in L is either ε or it ends in a b.

(C) FALSE: Every language is countable (shown in class) and L^* is still a language.

(D) TRUE: The string aaa has two different parse trees:

```
       S               S
     /   \           /   \
    S     S         S     S
    |   /   \     /   \   |
    | S     S   S       S |
    | |     |   |       | |
    a a     a   a       a a
```

PROBLEM 5 (6+6+6+6 points)

For each of the following languages, say whether it is regular, context-free, both, or neither. Briefly justify your answers.
(A) $L = \{a^i b^j : i - j = 2012\}$
(B) $L = \{a^i b^j : i + j = 2012\}$
(C) $L = \{a^n b^n a^n : n \geq 0\}$
(D) $L = \{w : w$ contains both *obama* and *romney* as substrings$\}$, over alphabet $\Sigma = \{a, b, \ldots, z\}$

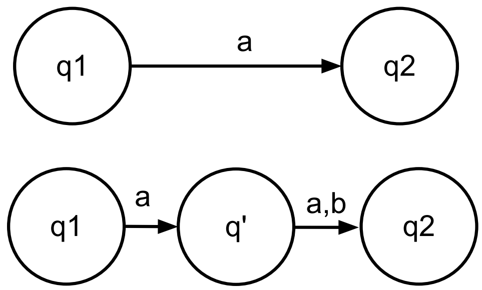
(A) CONTEXT-FREE, NOT REGULAR. This is very similar to $\{a^n b^n : n \geq 0\}$. In fact, it is $\{a^{n+2012} b^n : n \geq 0\}$. We can use this to show that it is nonregular: Consider $L_1 = \{b^{2012}\}$ and $L_2 = \{a^n b^n : n < 2012\}$. Both L_1 and L_2 are finite and hence regular. But $L \circ L_1 \cup L_2 = \{a^n b^n : n \geq 0\}$, which is a known non-regular language. Because the regular languages are closed under concatenation and union, then it must be the case that L is not regular. L is, however, context free. We can devise a grammar $S \rightarrow a^{2012}T, T \rightarrow aTb, T \rightarrow \varepsilon$ to produce L. Because L can be generated by a context-free grammar, L is context-free.
(B) BOTH. All strings in this language are 2012 characters long. Because the language has a largest string, that means that the language is finite and hence regular. And if it is regular, then it is also context-free.
(C) NEITHER. It suffices to show that L is not context-free, as this implies that it is also nonregular. We can prove that L is not context-free by an application of the CF Pumping Lemma, similar to the proof that $\{a^n b^n c^n\}$ is not context-free from lecture. Let p be the CF pumping length for L, and consider the string $s = a^p b^p a^p$. By the CF pumping lemma, s can be partitioned into $s = uvxyz$ where v or y is nonempty and $uv^i xy^i z \in L$ for all $i \in \mathcal{N}$. If either v or y contain two different symbols, then $uv^2 xy^2 z$ is not of the form $a^* b^* a^*$, and hence is not in L. If v and y each contain only a single symbol, then $uv^2 xy^2 z$ will not have an equal number of symbols in all three segments (the initial a's, the b's, and the final a's), as we will be pumping only one or two of the three segments. This is a contradiction, so L cannot be CF.
(D) BOTH. L is regular because it is the intersection of the two regular languages $L(\Sigma^* obama \Sigma^*)$ and $L(\Sigma^* romney \Sigma^*)$, and the class of regular languages is closed under intersection. L is context free because every regular language is context free.

PROBLEM 6 (10 points)

For a string $w \in \{a, b\}^*$, define $\text{odd}(w)$ to be the string consisting of the symbols in odd-numbered positions in w. That is, if $w = w_1 w_2 w_3 \cdots w_n$, then $\text{odd}(w) = w_1 w_3 w_5 \cdots w_m$, where m is either n or $n - 1$ depending on whether n is odd or even.

Show that if L is regular, then so is $\{w \in \{a,b\}^* : \mathrm{odd}(w) \in L\}$.

Solution: Suppose L is regular; then there is a DFA M recognizing L. We construct a new DFA M' as follows: For every old transition $\delta(q_1, \sigma) = q_2$, we create a new state q', delete the old transition, and create the transitions $\delta(q_1, \sigma) = q'$ and $\delta(q', \sigma') = q_2$ for every $\sigma' \in \Sigma$. For example:



Also, if q_2 is an accept state, we make q' an accept state. This makes sure that we accept both odd and even length strings.

Now we justify that this DFA M' recognizes $L' = \{w : \mathit{odd}(w) \in L\}$. First, if $\mathit{odd}(w) \in L$, then there was an accepting path in M on every odd letter in w, so M' must have an accepting path on w (since the odd letters go through states corresponding to M, and the even letters can be anything). Second, if w is accepted by M', then the odd letters of w must have followed transitions on the states corresponding to M, so $\mathit{odd}(w)$ would be accepted by M, so $\mathit{odd}(w) \in L$ and thus $w \in L'$.

Since M' is a DFA recognizing L', L' must be regular.

(Note: many students attempted to show that $\{odd(w) : w \in L\}$ is regular. Unfortunately, while this is still true, it is trickier to do.)

PROBLEM 7 (CHALLENGE! 1 points)

Prove that if R is a regular expression, then $L(R)$ satisfies the pumping lemma with a pumping length equal to $|R|+1$, where $|R|$ is the size (or length) of R. (You can omit the $|xy| \leq p$ condition.)

Solution: Intuitively, if a regular expression R matches a string w with $|w| > |R|$, then a Kleene star must have been used to do the matching. We can simply let y be the (nonempty) part of w that matches the Kleene star. Then y^n still matches the Kleene star.

Formally, we prove this by structural induction on R.

Base case: If $R = \emptyset$, $R = \varepsilon$, or $R = \sigma$ for some $\sigma \in \Sigma$, then, since $L(R)$ contains no strings of length at least $|R| + 1$, the result is vacuously true.

Induction step: Let R be a regular expression with $|R| > 1$. Suppose the result holds for regular expressions smaller than R. By the definition of regular expressions, we can write R in one of the following ways. In each case, R_1 and R_2 are regular expressions that are smaller than R and, using the induction hypothesis, we can show that the result holds for R.

$R = R_1 \cup R_2$: Let $w \in L(R)$ with $|w| > |R|$. Then either $w \in L(R_1)$ and $|w| > |R_1|$ or $w \in L(R_2)$ and $|w| > |R_2|$. In the first case, we can write $w = xyz$ with $y \neq \varepsilon$ and $xy^n z \in L(R_1) \subset L(R)$ for all $n \geq 0$. The other case is similar. Either way, the result holds for R.

$R = R_1 \cdot R_2$: Let $w \in L(R)$ with $|w| > |R|$. Then we can write $w = w_1 w_2$ with $w_1 \in L(R_1)$ and $w_2 \in L(R_2)$. Either $|w_1| > |R_1|$ or $|w_2| > |R_2|$. In the first case, let $w_1 = xyz$ with $y \neq \varepsilon$ and $xy^n z \in L(R_1)$ for all $n \geq 0$; then $w = xy(zw_2)$ and $xy^n zw_2 \in L(R_1) \cdot L(R_2) = L(R)$ for all $n \geq 0$. The second case is similar and, either way, the result holds for R.

$R = R_1^*$: If $w \in L(R)$ and $w \neq \varepsilon$, then we can write $w = w_1 w_2 \cdots w_k$ where $w_i \in L(R_1) - \{\varepsilon\}$ for all i. Set $x = \varepsilon$, $y = w_1$, and $z = w_2 \cdots w_k$. Since $y \in L(R_1) - \{\varepsilon\}$, $y^n \in L(R_1 *)$ for all $n \geq 0$. Thus $xy^n z = w_1^n w_2 \cdots w_k \in L(R_1^*) = L(R)$ and, hence, the result holds for R.

THE END

REMEMBER TO PUT YOUR NAME ON YOUR WORK.