Midterm I
15-453: Formal Languages, Automata, and Computability

Lenore Blum, Asa Frank, Aashish Jindia, and Andrew Smith

February 11, 2014

Instructions:

1. Once the exam begins, **write your name on each sheet.**
2. This is a closed-book examination.
3. Do all your work on the attached sheets. Prove your answers for problems 5-10. For problems 1 through 4, proofs are not required.
4. There are **9** questions for a total of **100** points, plus one extra credit problem for a bonus 15 points.
5. You have 80 minutes to answer the questions.
6. Read over the whole exam. Pace yourself. Check your work. Good luck!

Last name: _____ First name: _____

Signature: _____ Andrew ID: _____

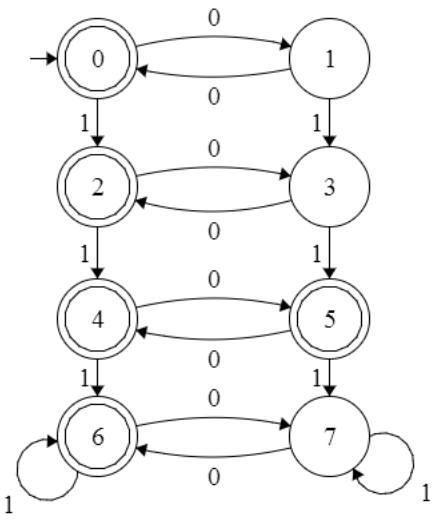
Question	Points	Score
1	5	
2	5	
3	5	
4	5	
5	10	
6	15	
7	15	
8	20	
9	20	
10	0	
Total	100	

1. (5 points) Give regular expressions and DFAs for the following language in $\{0, 1\}$:

   $\{w \mid w$ contains an even number of 0s, or w contains exactly two 1s$\}$

Regular expression: (1 01 0) 1 [0 10 10

DFA:



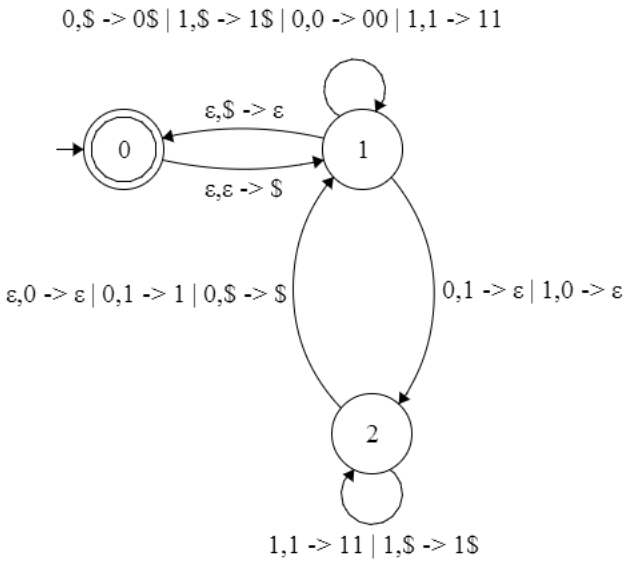
2. (5 points) Give a CFG and a PDA for the following language in $\{0, 1\}$:

$\{w \mid w$ is not the empty string, and w starts and ends with the same symbol$\}$

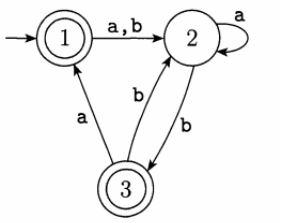
CFG:

$$S \to 0 \mid 1 \mid 0T0 \mid 1T1$$
$$T \to 0T \mid 1T \mid \varepsilon$$

PDA:



3. (5 points) Give a regular expression for the language recognized by the following DFA:



(Ra) (R ["), where R abbreviates (a [b)(a (bb)) b.

4. (5 points) Convert the following context-free grammar G to Chomsky normal form: G = (f A; B g; f 0; 1g; R; A), where R is defined by

A ! ABA j B j "
B ! 00 j 1

A ! CA j ZZ j 1 j "
C ! AB
B ! ZZ j 1
Z ! 0

5. (10 points) Let B be the language of all palindromes over $\{0;1\}$ containing an equal number of 0s and 1s. Show that B is not context-free.

   Note: This proves that context-free languages are NOT closed under intersection!

Idea: We choose a palindrome with $2p$ 1's and $2p$ 0's such that the 1's are all at the middle. We pump it by deleting the repeated strings. Because there are so many 1s, the deleted strings cannot have 0s from both side, so they either take 0s from one side and leave the string asymmetric or take only 1s and leave the string with an unequal amount of 0s and 1s.

AFSOC L is context-free and let p be its pumping length. Let $s = 0^p1^{2p}0^p \in L$. By the pumping lemma, we may choose u, v, x, y, z such that $s = uvxyz$, $|vy| > 0$ or $|yy| > 0$, $|vxy| \leq p$, and $uxz \in L$. If v and y both consist entirely of 1's, then $uxz = 0^p1^{2p-|vy|}0^p$ does not have the same number of 0s and 1s. Otherwise, v or y contains a nonzero amount of symbols m from one of the sets of 0s and some amount n from the set of 1s, but none from the other set of 0s since $|vxy| \leq p$. We have four cases: the set of 0s intersecting vy can be 0 or 1, and the substring containing 0s and 1s can by v or y. These correspond to uxz is of the form $0^{p-m}1^{2p-n-|y|}0^p$, $0^{p-m-|v|}1^{2p-n}0^p$, or $0^p1^{2p-n-|v|}0^{p-m}$, $0^p1^{2p-n}0^{p-m-|y|}$. None of these is a palindrome since $m > 0$. In any case $uxz \notin L$.

6. (15 points) Let $\Sigma = \{0, 1, +, =\}$ and define

$$\mathsf{Add} = \{x{=}y{+}z \mid x, y, z \text{ are binary integers, and } x = y + z\}.$$

Show that Add is not regular.

Idea: there must be only one $-$ and one $+$ in any string in Add, so the repeated string must be in one of the binary numbers. But of course changing the sum term makes the equation wrong, and changing the numbers added changes their sum, also making it wrong.

AFSOC Add is regular and let p be its pumping length. Let $s = 1^{p+1}{-}10^{p}{+}1^{p}$. Clearly this equation is true, so $s \in L$. Now, $|s| \geq p$, so by the pumping lemma, we can write $s = xyz$ with $|xy| \leq p$, $|y| > 0$, and $xz \in L$. Since the first p characters of s are all 1 and $|xy| \leq p$, y must be made of 1s only, so $xz = 1^{p-|y|}{-}10^{p}{+}1^{p}$. Since $|y| > 0$, xz has a different string to the left of its equals sign than s but is otherwise identical. Since binary expansions are unique, the equation given by xz cannot be true (or both strings would be binary expansions of the number to which their common right-hand side evaluates). Thus $xz \notin L$, contradicting the pumping lemma and proving that L is not regular.

7. (15 points) Let G be a context-free grammar in Chomksy normal form that contains b variables. Show that if G generates some string with a derivation having at least 2^b substitions, then $L(G)$ is infinite.

   (Recall that a derivation is a sequence of substitions starting at the start variable and ending at a string of terminals).

IDEA: We will want to use the pigeonhole principle. But on what? The derivation length alone isn't good enough, because knowing a variable is substituted twice in the derivation doesn't tell us it actually derives itself. Instead, we use the derivation length to bound the depth of the parse tree. Then we can use the pigeonhole principle on any path down the parse tree to find a variable that derives a string including itself. Repeating this substitution gives us arbitrarily long strings, so the language is infinite.

From lecture, we know that if G is in Chomsky normal form, a string of length x must have any derivation of length at most 2^x 1. Thus if there is a derivation of length 2^b, G must derive a string of length $b+1$. Then the parse tree must have depth at least $b+1$, since any substitution in Chomsky normal form only increases the length by 1. Thus there is a path down the parse tree of length $b+1$. By pigeonhole, this path must contain some variable A twice. Then A derives a string of the form uAv for $u;v\ 2\ \Sigma$, and A also derives some string $x\ 2\ \Sigma$. Then we see A yields $u^i x v^i$ for any $i\ 2\ N$, by substituting uAv for $A\ i$ times and then substituting x for A, as in the proof of the pumping lemma for regular languages. Since A is in the tree below the start variable S, S derives some aAb. Then $au^i x v^i b$ is in the string for any i. There are infinitely many such strings, so $L(G)$ is infinite.

8. (20 points) Consider the language $F = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and either } i \neq 1 \text{ or } j = k\}$.

   (a) Show that F is not regular.

IDEA: Since the pumping lemma won't work here, we'll reason directly about a DFA for F. In fact we can use the pigeonhole principle in the same way as in the proof of the pumping lemma.

AFSOC that F is regular and let D be a DFA with k states that recognizes it. Let $s = ab^k c^k$, which is clearly in L. While processing the b^k substring of s, D makes k transitions and thus is in $k+1$ states. By the pigeonhole principle, some two of them must be the same, say state q after processing b^i and b^j for $i < j$. Therefore, after processing ab^j, D is in state q, from which point processing $b^{k-j}c^k$ takes it to some accept state q_a (since $ab^i b^{k-j} c^k = s \in L$). But we also have that processing ab^j takes D to q, from which point processing $b^{k-j}c^k$ takes D to q_a, so D accepts $ab^{j+k-j}c^k$. But $j > i$, so $k = j + k - j > i + j - j$, and therefore $ab^{j+n-j}c^k \notin L$, contradicting the assumption that D recognizes L. Therefore L is not regular.

   (b) Show that the pumping lemma for regular languages applies to F. In other words, show that there is p such that for any $w \in F$ with $|w| \geq p$, there are x, y, z with $xyz = w$, $|y| > 0$, and $|xy| \leq p$ such that for any $i \geq 0$, $xy^i z \in F$.
   Note: (a) and (b) together show that the converse of the pumping lemma is false.

IDEA: If w has some number of as, we can pump the first few of them, being careful not to reach a string with only one a. Otherwise, the relative amounts of bs and cs don't matter, so we can simply pump whichever one of them exists.

Choose $p = 2$. Let $s = a^i b^j c^k \in L$ with $|s| \geq p$. Choose $x = \varepsilon$, y to be the first character of s or first two (aa) if $i = 2$, and z to be the rest of s. Clearly $s = xyz$, $|xy| \leq p$, and $|y| > 0$; we will show that $xy^n z \in L$ for each $n \in \mathbb{N}$.

- **Case $i = 0$:** If $j = 0$, then $s = c^k$, $y = c$, $z = c^{k-1}$, and $xy^n z = \varepsilon c^n c^{k-1} = c^{k+n-1}$. Otherwise, by similar reasoning (we will use this implicitly throughout), $y = b$ and $xy^n z = b^{j+n-1}c^k$. Both of these are in L since they begin with a^0 and $0 \neq 1$.
- **Case $i = 1$:** In this case we must have $j = k$ since $i = 1$. $xy^n z$ becomes $a^n b^j c^k$, which has the form $a^i b^j c^k$ where $j = k$, so it is in L for any n.
- **Case $i = 2$:** Here $y = a^2$ and $xy^n z = a^{2n} b^j c^k$, which is in L for any n since $2n \neq 1$.
- **Case $i > 2$:** Here $xy^n z = a^{n+i-1} b^j c^k$. For any n, $n + i - 1 - i + 1 > 1$, so $n + i - 1 \neq 1$ and $xy^n z \in L$.

We see that in any case x, y, and z as constructed satisfy the constraints of the pumping lemma for $p = 2$.

9. (20 points) If A and B are languages over the same alphabet Σ, define the equal concatenation of A and B to be

$$EC(A;B) = \{xy \mid x \in A; y \in B; |x| = |y|\}.$$

Show that if A and B are regular, then $EC(A;B)$ is context-free.

IDEA: our PDA will follow along the DFAs for A and B, only reaching a final state if it traverses both in order. This gives concatenation. To make sure the strings are even length, we push onto the stack for every transition in the DFA for A, and we pop off the stack for every transition in B. We accept only if it reaches a final state with an empty stack.

Let D_A be a DFA that recognizes B and D_B, B. We construct a PDA P that recognizes $EC(A;B)$ as follows. Its state set is the union of those of D_A and D_B (WLOG these are disjoint), plus a unique start state and a unique accept state. The start state simply pushes a \$ onto the stack and "-transitions to the start state of D_A. P's transition function behaves as those of D_A and D_B on those states. For each transition within its copy of D_A, P pushes a 0 onto the stack; for those within D_B, it pops from the stack. Each accept state of D_A makes an "-transition to the start state of D_B (without reading or changing the stack), and the accept states of D_B "-transition with a \$ atop the stack to the accept state, which makes no transitions. We now show that $L(P) = EC(A;B)$

( ): Let $s \in L(P)$. Then some execution path of P on s must behave as followoms. P pushes a \$ and reads, say, n characters of s, pushing n 0s onto the stack, before accepting in D_A and transitioning to the start state of D_B. Then it reads precisely enough characters to leave a \$ on the stack upon reaching an accept state in D_B. Since it pops a character with each transition in D_B, it must read exactly n characters here, too. For p to accept, this must be the end of s. Therefore, if $s = s_1 \cdots s_k$, where each $s_i \in \Sigma$, we have $s_1 \cdots s_n \in A$, $s_{n+1} \cdots k \in B$, and these strings have equal size: $k = 2n$. Therefore $s \in EC(A;B)$.

( ): The converse argument follows the same reasoning: for $s \in EC(A;B$, we can write $s = xy$ with $|x| = |y|$, and then s causes P to behave as described above and to accept. Therefore $L(P) = EC(A;B)$, completing the proof.

10. (15 points extra credit) For any language A over Σ, consider the language of strings obtained by deleting a single character from any string in A:

$$\text{Delete}(A) = \{xz \mid x, z \in \Sigma \text{ and } xyz \in A \text{ for some } y \in \Sigma\}$$

Show that if A is regular, then $\text{Delete}(A)$ is regular.

IDEA: Our NFA has a second set of states, to record if it has already guessed a deleted character. From any state until it has already guessed, it has the option to choose to travel an extra transtion, effectively guessing as to what the deleted character is. When it does so it moves into the second set of states, so that it cannot guess again.

Let L be a regular language and D a DFA that recognizes it; we will construct an NFA N that recognizes $\text{Delete}(A)$.

Say $D = (Q, \Sigma, \delta, q_0, F)$. Then we put

$$N = (Q \cup Q', \Sigma, \delta', q_0, F')$$

where

$$Q' = \{q' \mid q \in Q\}$$
$$F' = \{q' \mid q \in F\}$$

and δ' is given by

$$\delta'(q, c) = \{\delta(q, c)\}$$
$$\delta'(q', c) = \{\delta(q, c)'\}$$
$$\delta'(q, \varepsilon) = \{\delta(q, a)' \mid a \in \Sigma\}$$
$$\delta'(q', \varepsilon) = \emptyset$$

for $q \in Q$, $q' \in Q'$ and $c \in \Sigma$.

We now show that $L(N) = \text{Delete}(A)$.

($\subseteq$): Let $s \in L(N)$. Then by definition we can write $s = s_1 \cdots s_n$ where each $s_i \in \Sigma_\varepsilon$, and we have $r_0 \cdots r_n$, where $r_0 = q_0$, $r_n \in F'$, and each $r_{i+1} \in \delta'(r_i, s_{i+1})$. Since $r_n \in F' \subseteq Q'$, choose the smallest k such that $r_k \in Q'$. Since we never have an element of Q in an output of δ on a state in Q', $r_i \in Q$ for every $i < k$. Therefore, we must have $s_i \neq \varepsilon$ for $i > k$, since no $\delta'(r_j, s_{j+1})$ can be empty for $0 \leq j < n$ (for it contains r_{j+1}). Note that, for $q \in Q$, $(q, c) \in Q' \iff c = \varepsilon$. We have $r_{k-1} \in Q$ and $r_k \in Q'$, so this means $s_k = \varepsilon$, and similarly, since $r_0 = q_0 \in Q$, we must have $s_i \neq \varepsilon$ for $i < k$ by the minimality of k.

Now, $q_{k-1} \in \delta'(q_{k-1}, \varepsilon)$, so, considering δ', there must be some $a \in \Sigma$ such that $\delta(q_{k-1}, a) = q_k$. Further, we have $r_{i+1} \in \delta'(r_i, s_i) = \{\delta(r_i, s_i)\}$ and thus $r_{i+1} = \delta(r_i, s_i)$ for $0 \leq i < k - 1$. Similarly, for each $i \geq k$, we have $r_i \in Q'$, say $r_i = (r_i')'$, and then $r_{i+1}' = \delta(r_i, s_{i+1})$. Now let $s' = s_0 \cdots s_{k-1} a s_k \cdots s_n \in \Sigma$ and consider $r_0 \cdots r_{k-1}' r_k \cdots r_n'$. We have every $r_i \in Q$, $r_0 = q_0$, $r_n' \in F$, and

$$\delta(r_i, s_{i+1}) = r_{i+1}; \qquad 0 \leq i < k - 1$$
$$\delta(r_{k-1}, a) = r_k$$
$$\delta(r_i', s_{i+1}) = r_{i+1}'; \qquad k \leq i < n$$

Therefore, by definition, D accepts s', so $s' \in L$. Further, we have $s \in \text{Delete}(L)$ with $x = s_0 \cdots s_{k-1}$, $y = a$, $z = s_{k+1} \cdots s_n$. Therefore, N accepts only strings in $\text{Delete}(A)$.

($\supseteq$): Let $s \in \text{Delete}(A)$ and choose x, y, z with $x, z \in \Sigma^*$, $y \in \Sigma$, $xz = s$, and $s' = xyz \in A$. Then D accepts xyz, so we have $s' = s'_0 \cdots s'_n$ and a sequence of states $q_0 \cdots q_n$ with $q_n \in F$ and $\delta(q_i, s'_{i+1}) = q_{i+1}$ for $0 \le i < n$. Let $k = |x|$ (so $s_k = y$) and write $s = s'_0 \cdots s'_{k-1} s'_{k+1} \cdots s'_n$ and consider $r_0 \cdots r_n = q_0 \cdots q_{k-1} q'_k \cdots q'_n$. We have $r_0 = q_0 \in Q$, and $q_n \in F'$ so $r_n = q'_n \in F'$. Now, for each $0 \le i < k - 1$,

$$r_{i+1} = q_{i+1} \in \{q_{i+1}\} = \delta(q_i, s_i);$$

for $i = k - 1$,

$$r_{i+1} = q'_k \in \{\delta(q_i, a) \mid a \in \Sigma\} = \delta(q_i, \varepsilon) = \delta(r_i, s_{i+1})$$

because $\delta(q_i, y) = q_k$; and for $i \ge k < n$,

$$r_{i+1} = q'_{i+1} \in \{q'_{i+1}\} = \delta(q'_i, s_i) = \delta(r_i, s_i).$$

Thus, by definition, N accepts s. Therefore N accepts every string in $\text{Delete}(A)$, concluding the proof.