Solution Sketches: Midterm I
2018/3/5 (last revised 2018/3/9/1027pm)
CSC 280: Computer Models and Limitations Spring 2018
Instructor: Lane Hemaspaandra Grad TA: Andrew Read-McFarland

Instructions: Make sure to write your name on the top-right-corner name line of each sheet! We’ll grade only the work you put on these stapled test sheets (you may solve a problem only on the front and back of the sheet it is on, since we’ll separate the sheets during grading). Any blank paper we pass around (you may not bring in scrap paper) is for you to use as scrap paper and will be neither collected nor graded. Do four of the six problems. If you do more, we’ll grade all that you’ve done, and will count the four highest scores. (Since you might make mistakes that you don’t realize, I’d advise you to do more than four.) The order in which problems and multiple-choice answers are listed may differ from person to person; before starting the test, read all problems and problem parts so that you can attack first whichever things offer you the best path to a strong score. Throughout this exam you may use, without proving it, the claim that \( \{0^i1^i \mid i \geq 0\} \) is nonregular (but if you for example want to use the fact that some other language is nonregular, you’ll need to prove that other language nonregular). On “prove or disprove,” if you make the wrong choice as to proving or disproving you’ll get no credit at all, so choose carefully. CFL and CFG stand respectively for context-free language and context-free grammar. Don’t miss problems on the back of a page, if any. Closed book. Closed notes (except for a most one 8.5-by-11 2-sided handwritten self-prepared notes sheets, or two 8.5-by-11 1-sided such sheets). Closed computer, closed slide rule, closed phone, etc. (NO electronics may even be in sight). This is an in-class midterm exam being given during the 3/5 class period.

No questions may be asked during the exam. (If you are convinced that something is severely wrong in a way that makes the problem impossible to solve, explain, on your test sheet, what is severely wrong. But for example minor typos where it is clear enough what was intended do not mean you can skip doing a problem.)

This will be viewed as 100-point exam, but you can get up to 8 extra credit points since each problem is worth 27 points, so it is possible to score a 108/100 on the exam.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Total (= sum of 4 highest scores)
Question 1. [27 points] Draw (or, if you prefer, you may specify it by specifying each of the 5 parts of the 5-tuple; but for this problem, a drawing will yield full credit if correct; but please be very careful not to botch or forget one or more of the needed arrows, or to forget to as per our conventions denote in your drawing the start state and the final states) a 7-state DFA accepting the following language, over the alphabet \{0, 1\}: All strings that when interpreted as binary natural numbers are multiples of 7 or are one greater than a multiple of 7. (What is the natural number related to a string for this problem? Ignore leading 0's. Treat the empty string as if it is interpreted as the value 0. So 00000 represents 0 and the empty string represents 0 and 000101 represents 5 and 101 represents 5 and 01000 represents 8 and 1001 represents 9. So our DFA will for example accept the inputs 00000 and 01000 and \(\epsilon\) but will reject the inputs 000101 and 101 and 1001.)

Answer:

By now, I think you all know well how to do this. If you have any questions, please see Read or me during our office hours, or ask at your tutorials or workshops, so we can go over it with you to ensure that you can solve this and similar problems.

But, so that you have it, here is a brief solution sketch; I’ll briefly sketch each of the 5 parts of the 5-tuple. Consider a 7-state DFA whose states we’ll interpret as “remainders” namely with those remainders being the values 0,1,2,3,4,5,6. Our start state will be the state related to remainder 0. Our accepting (aka final) states will be the states related to remainders 0 and 1. Our alphabet is \{0, 1\}. And our transition function will be as follows. If we are in the state corresponding to remainder \(i\) and our input is the character 0, then we transition to the state corresponding to remainder

\[2i \pmod{7}.\]

If we are in the state corresponding to remainder \(i\) and our input is the character 0, then we transition to the state corresponding to remainder

\[2i + 1 \pmod{7}.\]

For example, if we are in the state corresponding to remainder 5, and the input is 1, then since \((2 \times 5) + 1 = 11\) and 11 \((\pmod{7})\) is 4, we transition to the state corresponding to remainder 4.
Question 2. [27 points]

1. [7 points] Draw (so you do not have to, and indeed, should not, give it as a 5-tuple; for this problem, you need merely draw an appropriate DFA) a 5-state DFA (your DFA must be fully specified—i.e., no “implicit sink” convention) accepting the language, over the alphabet \(\{0, 1\}\), \(\{x \in \{0, 1\}^* \mid |x| \equiv 4 \pmod{5}\}\). (So, all strings over that alphabet, such that the length of the string is itself congruent to four, mod five; e.g., the DFA should accept 010110101 as it is of length 9 but should reject 000111 as it is of length 6.)

Answer:
2. [20 points] Prove that for each regular set $A \subseteq \{0, 1\}^*$ it holds that

$$L_A = \{x \mid (\exists y \in \{0, 1\}^*)[|y| = |x| \text{ and } xy \in A]\}$$

is regular. Do so by letting $M = (Q, \{0, 1\}, \delta, q_0, F)$ be a DFA accepting $A$ and then specify clearly—via specifying its components, $(Q', \{0, 1\}, \delta', q'_0, F')$—an NFA accepting $L_A$. You do not have to here include a proof of the correctness of your construction, but it should in fact be correct. (You do NOT have to use our “uniform” method to solve this. You can if you wish. Or you can do it by a problem-specific approach. Whichever way you use, though, please be careful to ensure that you construction is correct.)

**Answer:**

Let $S$ be a state name that is not already used in $Q$.

- $Q' = (Q \times Q) \cup \{S\}$.
- $q'_0 = S$.
- $F' = \{(q, q) \mid q \in Q\}$.

We define $\delta'$ by

$$\delta'(S, \epsilon) = \{(q_0, q) \mid q \in F\},$$

and for all $c \in \{0, 1\}$,

$$\delta'(S, c) = \emptyset,$$

and for all $q_1, q_2 \in Q$ and $c \in \{0, 1\}$,

$$\delta'((q_1, q_2), c) = \{(\delta(q_1, c), q) \mid (\exists b \in \{0, 1\})[\delta(q, b) = q_2]\},$$

and for all $q_1, q_2 \in Q$,

$$\delta'((q_1, q_2), \epsilon) = \emptyset.$$
Question 3. [27 points] Let \( L = \{a^ib^j,c^k \mid i,j,k \in \{0,1,2,3,4,\ldots \} \text{ and } i < j \text{ and } i < k \} \). Either use the pumping lemma for context-free languages to prove that \( L \) is not a CFL (for a maximum of 27 points) OR (for a maximum of 22 points) use the pumping lemma for regular languages to prove that \( L \) is not a regular set. Do only one of these two options and you must start by explicitly stating which you are attempting. (Warning/FYI: The “it isn’t a CFL” proof is quite difficult. One of the proofreaders suggested that with “only” a 5-point gap in maximum score, doing the 22-point version might be a very attractive option.)

You MUST write here either “it isn’t a CFL” or “it isn’t a regular set” to let us know what you will be trying to prove; if you write one thing and then do the other, or leave this blank, you will get 0 points, so do not leave this blank:

Answer Sketch: [Note: In class and the slides I tried to present things via proof by contradiction using the pumping lemma. A different approach, which some of you may have seen in books or perhaps even in WSs, is under-the-hood the same, but phrases it all as an adversary argument, against a “demon,” and my two answers below are phrased—somewhat telegraphically as that idiom usually builds into the idiom somewhat tacitly the connection to the Pumping Lemma—in that idiom, so that you’ll have seen it at least once. Although both approaches are valid, the one I used in class and in the slides is the clearer and more explicit of the two approaches. Also, below, I’m using in the CFL case different variable names than you’ll be used to for the five parts and the key string, namely, I’m using the names that are quite commonly used by people who learned this in the Before Sipser era.]

And your answer would start by filling in the “You MUST write here” line with whichever of the two phrases is the one that captures your choice of which result you will be proving among the two.

If you chose to prove that the language is not regular:

Suppose the demon chooses \( n \) as the so-called pumping length (what we typically call \( p \) in 280/480). Then we will choose the string \( s = a^{n+1}b^{n+2}c^{n+2} \); note that for this choice, certainly we have that \( |s| \geq n \) and that \( s \in L \). The demon now breaks \( s \) into \( uvw \), where \( |v| \geq 1 \) and \( |uv| \leq n \). Since \( |uv| \leq n \), clearly \( uv \) consists of only \( a \)'s. And since in addition \( |v| \geq 1 \), we know that \( v \) is a nonempty string of \( a \)'s. So we get that for any \( i \geq 0 \), \( uv^iw = a^{n+1+(i-1)|v|}b^{n+2}c^{n+2} \). If we pick any \( i \geq 3 \), the resulting pumped string will have more \( a \)'s than \( b \)'s or \( c \)'s since \( |v| > 0 \), and will therefore not be in \( L \). Hence, \( L \) does not satisfy the pumping lemma and is thus not regular. (Yes, this is sort of an overkill proof. Even \( i = 2 \) would ensure that the number of \( a \)'s is greater than or equal to the numbers of \( b \)'s and \( c \)'s, and that would be enough to underpin a fine proof since the definition of \( L \) has “\(<\)” in the two key places. But the \( i = 3 \) choice is quite legal.)
If you chose to prove that the language is not a CFL:

Suppose the demon chooses \( n \geq 0 \). We choose the string \( z = a^n b^{n+1} c^{n+1} \) and note that \( z \in L \) and that \( |z| > n \). The demon now breaks \( z \) into \( u, v, w, x, y \) such that \( z = uvwxy \), \(|vwx| \leq n\) and \( |vx| \geq 1\). We will show that this string cannot be pumped.

If either \( v \) or \( x \) consists of two or more kinds of characters, then the string \( uvwxxxy \) will not have the form of strings in our set. So, we can assume that \( v \) and \( x \) each consist of at most one type of character.

Let us consider the possibilities when \( |v| > 0 \). If \( v \) consists only of \( a \)'s then \( uvvvwxxxy \) will either have more \( a \)'s than \( b \)'s or more \( a \)'s than \( c \)'s, since \( x \) may contain \( b \)'s or \( c \)'s, but not both. If \( v \) consists of only \( b \)'s or only \( c \)'s, then the string \( uwy \) will not have strictly more \( b \)'s and \( c \)'s than \( a \)'s.

Now let us consider what happens if \( v = \epsilon \). In this case \( |x| \geq 1 \), since otherwise the condition that \( |vx| > 0 \) will be violated. We also know that \( x \) consists of exactly one type of character. By the same arguments as above, we can see that \( z \) cannot be pumped. Since we have exhausted all possible ways that the demon could have legally broken up \( z \), we can conclude that \( z \) cannot be pumped and that the language is not context free.
Question 4. [27 points].

1. [10 points] Prove or disprove (and you must start by explicitly stating which you are doing): Every subset of a regular set is regular (i.e., for each regular set $R$ and each set $S \subseteq R$, it holds that $S$ is regular).

Answer:

We will disprove the claim. Consider regular set $\{0, 1\}^*$. This set has as a subset $\{0^i1^i \mid i \geq 0\}$, which we know not to be regular.

2. [17 points] Prove that for each regular set $A \subseteq \{0, 1\}^*$ and each set $B \subseteq \{0, 1\}^*$, it holds that

$$L_{A,B} = \{x \mid (\exists y \in B)[xyyy \in A]\}$$

is regular. In particular, do so by letting $M = (Q, \{0, 1\}, \delta, q_0, F)$ be a DFA accepting $A$ and then specify clearly—via specifying its five components—either a DFA or an NFA accepting $L_{A,B}$, and explicitly say which of those two options you are giving. You do not have to here include a proof of the correctness of your construction, but it should in fact be correct.

Answer:

Let $M' = (Q, \{0, 1\}, \delta, q_0, F')$ be a DFA with the set of states, alphabet, delta function and starting state being the same as that of the DFA $M$. Define $F' = \{q \mid q \in Q \land (\exists y \in B)[\delta(q, yyy) \in F]\}$. Then, the DFA $M'$ accepts the language $L_{A,B}$. 
Question 5. [27 points]

1. [17 points] Give a context-free grammar (by giving all four parts of the four-tuple that defines the CFG), for the set \{0^i1^j2^k \mid i, j, k \geq 0 \text{ and } i + j = k\}

Answer:

The variable set will be \{S, T\}. The start state is S. The alphabet is \{0, 1, 2\}. And our set productions is \{S \rightarrow 0S2, S \rightarrow T, T \rightarrow 1T2, T \rightarrow \epsilon\}. (If you gave all four parts of the four-tuple, but for the productions said something more informal such as: “Our productions are \(S \rightarrow 0S2 \mid T\) and \(T \rightarrow 1T2 \mid \epsilon\),” that too is ok here.)

2. [10 points] Show that the set of context-free languages is closed under union, i.e., that if \(A\) and \(B\) are context-free languages, then \(A \cup B\) is also a context-free language. For this problem, you may not use the equivalence between PDAs (pushdown automata) and CFLs/CFGs; rather, you should show this via creating/manipulating grammars for \(A\) and \(B\). In particular, assume that you have a context-free grammar for \(A\), and a context-free grammar for \(B\). (You in your solution may assume that no nonterminal symbol of either grammar is a symbol of any kind of the other grammar.) From those, explain how to construct a context-free grammar for \(A \cup B\), i.e., give that new grammar. (You do not have to include a proof of its correctness, but it indeed must be correct.)

Answer:

Let \(A\) and \(B\) be context-free languages. Let \(A\) be generated by the CFG \(G_A = (V_A, \Sigma_A, R_A, S_A)\) and let \(G_B = (V_B, \Sigma_B, R_B, S_B)\) be a context-free grammar that generates \(B\). We without loss of generality assume that the sets of nonterminals of the two grammars are completely disjoint (if they were not, then appropriately rename them so that they are).

We now create the new grammar \(G = (V_A \cup V_B \cup \{S\}, \Sigma_A \cup \Sigma_B, R_A \cup R_B \cup \{S \rightarrow S_A\} \cup \{S \rightarrow S_B\}, S)\), where \(S\) is a new symbol that did not appear anywhere in either of the grammars.

We can easily see that every string that has a derivation in \(G_A\) has a derivation in \(G\), and similarly for \(G_B\). Also, the strings that have derivations in \(G\) are precisely those that had derivations in \(G_A\) or \(G_B\). Hence, the language of the new grammar \(G\) is \(A \cup B\), and so CFLs are closed under union.
Question 6. [27 points] Multiple choice. Answer (do not show your work—this is multiple choice) in the provided area at the bottom of the page.

Note: The order of the parts in this question or its answers may vary from test to test.

1. [5.4 points] \( \{0^{280k} \mid k \in \{1, 2, 3, \ldots, 2004\}\} \) is:
   (A) regular and a CFL, (B) not regular but is a CFL, (C) not a CFL but is regular, 
   (D) neither regular nor a CFL.
   **Answer:**
   (A) is correct.
   All finite sets are regular (and thus are CFLs).

2. [5.4 points] \( \{0^i \mid i \in \{0, 1, 2, 3, 4, \ldots\}\} \) is:
   (A) regular and a CFL, (B) not regular but is a CFL, (C) not a CFL but is regular, 
   (D) neither regular nor a CFL.
   **Answer:**
   (D) is correct.
   We can use the pumping lemma for CFLs to show that it is not a CFL. (And so it also is not regular.)

3. [5.4 points] The set of all strings generated by the CFG whose start symbol is \( S \), whose variable set is \( \{S\} \), whose alphabet \( \Sigma \) is \( \{a, b\} \), and whose (two) rules (aka productions) are \( S \to aS \) and \( S \to bb \) is:
   (A) regular and a CFL, (B) not regular but is a CFL, (C) not a CFL but is regular, 
   (D) neither regular nor a CFL.
   **Answer:**
   (A) is correct.
   The language is \( a^*bb \) and is regular. All regular sets are CFLs (alternately, it is clearly a CFL since it is generated by a CFG).

4. [5.4 points] You have seen a construction (sometimes called the “ripping” construction, since Sipser at one point symbolically dubs the state that is being ripped out at a given stage \( q_{rip} \) that converts a DFA into a so-called GNFA, and then repeatedly “rips” out a state (and does some rather massive relabeling of edges, in which as you saw in your SIP reading the formula \( (R_1)(R_2)^*(R_3) \cup (R_4) \) plays a big role). The overall goal of that entire construction is to show that for each DFA there is a FOO that describes the same language as the DFA accepts. Which of the following is the correct value of FOO:
(A) inherently ambiguous context-free grammar, (B) NFA, (C) regular expression, (D) Turing machine.

**Answer:**

(C) is correct.

5. [5.4 points] \( \{a^i b^j a^i b^j | i \in \{0, 1, 2, 3, 4, \ldots \} \text{ and } j \in \{0, 1, 2, 3, 4, \ldots \} \} \) is:

(A) regular and a CFL, (B) not regular but is a CFL, (C) not a CFL but is regular, (D) neither regular nor a CFL.

**Answer:**

(B) is correct.

If we assume that this language is regular, and intersect it with the regular expression \( a^n b^n \), we should get a regular language. However, the intersection is exactly the set \( \{a^n b^n | n \geq 0 \} \), which we know is not regular. The language is a CFL since it is the language of the CFG whose start symbol is \( S \), whose variable set is \( \{S, A\} \), whose alphabet \( \Sigma \) is \( \{a, b\} \), and whose productions are

\[ S \rightarrow aSb \mid A, \text{ and} \]
\[ A \rightarrow bAa \mid \epsilon. \]

Answers go here (A/B/C/D):

6.1 __A__
6.2 __D__
6.3 __A__
6.4 __C__
6.5 __B__