# CS154 Midterm Examination

May 4, 2010, 2:15 - 3:30PM

**Directions**: Answer all 7 questions on this paper. The exam is open book and open notes. Any materials may be used.

SOLUTIONS

Name: _____

I acknowledge and accept the Honor Code.

Joe Solutions

(signed) _____

| **Problem** | **Points** |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| **Total** | |

**Problem 1** (20 pts.) Let L be the language mentioned in the third Challenge-Problem set: the set of strings over alphabet {0, 1, 2} that do not have two consecutive identical symbols. That is, strings of L are any string in {0,1,2}* such that there is no occurrence of 00, no occurrence of 11, and no occurrence of 22. In the space below, design a DFA (transition table or transition diagram -- your choice) that accepts L.

| | ***0*** | ***1*** | ***2*** |
|---|---|---|---|
| ->*A | *B* | *C* | *D* |
| *B | *E* | *C* | *D* |
| *C | *B* | *E* | *D* |
| *D | *B* | *C* | *E* |
| *E* | *E* | *E* | *E* |

For each of your states, give a brief description of what strings get you to that state.

| ***State*** | ***Purpose*** |
|---|---|
| *A* | *Start state, empty string read so far* |
| *B* | *No consecutive symbol pair, 0 is last symbol* |
| *C* | *No consecutive symbol pair, 1 is last symbol* |
| *D* | *No consecutive symbol pair, 2 is last symbol* |
| *E* | *Dead state, some pair seen previously* |
| | |

***Comment**: Points were deducted for errors to the state diagram, such as missing initial state or dead state. Another common error was describing the states B, C, and D by only the last input symbol, without stating that the entire string read had to be free of consecutive symbol pairs.*

**Problem 2** (15 pts.) Below is the transition table of a DFA.

| | **0** | **1** |
|---|---|---|
| ->A | E | B |
| * B | D | A |
| C | G | A |
| * D | G | E |
| E | A | D |
| F | B | E |
| * G | B | A |

Find the distinguishable states by filling out the table below. However, place X's only for those pairs that are distinguishable by the basis step. For those discovered to be distinguishable during the induction, place numbers 1, 2, ... indicating the order in which you discovered these pairs to be distinguishable. *Note* that many different orders are correct. *Also note*: cells with # are not to be filled in.

| | G | F | E | D | C | B |
|---|---|---|---|---|---|---|
| A | X | 1 | | X | 2 | X |
| B | | X | X | | X | # |
| C | X | | 3 | X | # | # |
| D | | X | X | # | # | # |
| E | X | 4 | # | # | # | # |
| F | X | # | # | # | # | # |

Which sets of states are mutually equivalent?

*{A,E}, {B,D,G}, {C,F}*

In the space below, draw the transition table for the minimum-state equivalent DFA.

| | **0** | **1** |
|---|---|---|
| ->A | A | B |
| *B | B | A |

***Comment**: Points were allloted as follows:*

*5 for the table of indistinguishable pairs (-2 if pairs that are distinguishable other than by the basis step are marked by X).*
*4 for the sets of equivalent states*
*6 for the transition table (-3 if you did not eliminate the unreachable state corresponding to {C, F})*

**Problem 3** (10 pts.) We wish to construct a DFA B from a given DFA A such that B accepts input string w if and only if A accepts w but does not accept www (Note there are three w's, not two). The construction is similar to that of a problem on the second Challenge-Problem set, but it involves keeping track of what all states of A do, not just the accepting states. Let A have states $q_1$, $q_2$,..., $q_n$, where $q_1$ is the start state. The states of B are vectors of n states of A. The start state of B is $[q_1, q_2,..., q_n]$, that is, the states of A in order. Let $\delta_A$ the transition function of A. Then the transition function $\delta_B$ is defined by

$$\delta_B([p_1, p_2,..., p_n], a) = [\delta_A(p_1, a), \delta_A(p_2, a),..., \delta_A(p_n, a)]$$

That is, B simulates A from each of its states, each reading the same input. To complete the construction, all that needs to be done is to define the final states of B. That's your job. In the space below, state the condition under which state $[p_1, p_2,..., p_n]$ should be a final state of B.

*Let $p_1$ be $q_i$. Let $p_j$ be $q_j$. Then $p_1$ must be accepting and $p_j$ not accepting.*

***Comment**: If you got the point that $p_1$ needs to be an accepting state but little else, you got 3/10. You also got 3/10 if you simply repeated the solution to the similar challenge problem.*

**Problem 4** (20 pts.) A *wiggle string* is defined to be a nonempty string of 0's and 1's such that each 0 is followed by a 1, and each 1 is followed by a 0. A *0-wiggle string* is a wiggle string that begins and ends in 0, and a *1-wiggle string* is a wiggle string that begins and ends with 1. For example, 01010 is a 0-wiggle string, 1 is a 1-wiggle string, 0101 is a wiggle string that is neither a 0-wiggle string nor a 1-wiggle string, and 010010 is not a wiggle string. The language of the grammar G consisting of productions

S -> SAS | 0
A -> ASA | 1

(S is the start symbol) is the set of 0-wiggle strings, as you will prove. In this proof, you can assume the true fact that the concatenation of two wiggle strings is a wiggle string, as long as the first ends in a symbol different from the symbol by which the second begins. You do not have to state this fact in the proof. Give your proof by answering the following very specific questions.

Prove first that every 0-wiggle string is in L(G). To do so, we need to prove a more general statement inductively: (1) if w is a 0-wiggle string then S =>* w, and (2) if w is a 1-wiggle string, then A =>* w.

(a) On what is your induction?

*The length of w.*

(b) What is the basis case?

*|w| = 1.*

***Comment**: The induction is always on some parameter, and the basis case is always an integer or set of integers. Many people confused what was needed in (a), (b), (e), and (f).*

(c) Prove the basis.

*(1) w = 0, and S => 0. (2) w = 1, and A => 1.*

(d) For the inductive part, first show that if w is a 0-wiggle string, then S =>* w.

*Assume |w| = n > 1 and assume the IH for strings shorter than length n. If w is a 0-wiggle string, then w = 01x for some 0-wiggle string x. By the IH, S =>* x. Thus, S => SAS => 0AS => 01S =>* 01x = w.*

The second part of the induction is that if w is a 1-wiggle string, then A =>* w. You do not have to provide this part, since its proof is essentially like that of (d), with 0 and 1 interchanged.

Conversely, to show that every string in L(G) is a 0-wiggle string, we shall show that (1) if S =>* w, then w is a 0-wiggle string, and (2) if A =>* w, then w is a 1-wiggle string.

(e) On what is your induction?

*Number of steps in the derivation.*

(f) What is the basis case?

*One step.*

(g) Prove the basis.

*The only one-step derivations are S => 0 and A => 1. The resulting strings are 0- and 1-wiggle strings, respectively.*

(h) For the inductive part, first show that if S =>* w, then w is a 0-wiggle string of length n.

*Assume the derivation is multistep, and assume the IH for shorter derivations. Then the derivation begins S => SAS =>* w. We can break w = xyz, such that in this derivation S =>* x, A =>* y, and S =>* z, all by shorter derivations. By the IH, x and z are 0-wiggle strings, and y is a 1-wiggle string. Thus, by the properties of wiggle strings stated in the opening of the problem, xyz = w is a 0-wiggle string.*

***Comment**: Many people made a mistake in this part of confusing it with (d), where we were allowed to choose the derivation. Either you wound up giving another proof of (d), or you made some assumption about the derivation which you could not justify. I am happy to talk with anyone about why certain arguments are not really inductive proofs or why a particular proof assumes something without proof.*

To complete the inductive part, you also need to show that if A =>* w, then w is a 1-wiggle string. You do not need to provide this part, since it is essentially (h), with 0 and 1 interchanged.

**Problem 5** (15 pts.) The grammar from Problem 4:

S -> SAS | 0
A -> ASA | 1

(S is the start symbol) is ambiguous. Give an example of a string that has two or more parse trees.

*01010*

Show two of its parse trees in the space below.

I can't draw parse trees in MS Word, but I'll show two leftmost derivations:

*S => SAS => SASAS => 0ASAS => 01SAS => 010AS => 0101S => 01010*

*S => SAS => 0AS => 0ASAS => 01SAS => 010AS => 0101S => 01010*

Give an unambiguous grammar for the set of 0-wiggle strings. You need not justify your answer.

*S -> 01S | 0 works and is pretty clearly unambiguous.*

**Problem 6** (10 pts.) Let L be the language $\{a^i \mid i \text{ is a perfect square}\}$. That is, L contains the strings a, aaaa, aaaaaaaaa, aaaaaaaaaaaaaaaa, and so on. We want to prove that L is not context free. (*Hint*: Notice that as strings in L get larger, the difference in length between one string and the next longer string grows without limit.)

Suppose n is the pumping-lemma constant for L. What string z do you choose to apply the pumping lemma?

*$z = n^2$ a's. (Sorry – MSWord does not allow double superscripts.)*

Suppose "the adversary" picks a way to break z = uvwxy that satisfies the constraints of the pumping lemma. We hope to prove that there must be some string in L whose length is not a perfect square. In terms of u, v, w, x, and y, what string would you choose?

*uvvwxxy*

Complete the proof that your chosen string is not in L.

*$|uvvwxxy|$ is greater than $n^2$, but no greater than $n^2 + n$. But the next perfect square after $n^2$ is $n^2 + 2n + 1$. Thus, uvvwxxy does not have a length that is a perfect square and so is not in L.*

**Problem 7** (10 pts.) Suppose we use the construction of a CFG G whose language is N(P) for a given PDA P, as given in the text or in the class slides. Let P have s states. Suppose also that $\delta(q, a, X) = \{(p, \beta)\}$, while $\delta(q, \varepsilon, X)$ is empty. Let β have length k. As a function

of s and k, how many productions of grammar G have variable [qXp] on the left?

*$s^{k-1}$ if k > 0, and 1 if k = 0.*

***Comment**: I messed up on the statement of this problem. I should have added that δ(q, b, X) was also empty for b ≠ a. So the answer above is really a lower bound. The explanation for this formula is that for k = 0, there is only one production, [qXp] -> ε. For k = 1, there is also one production, [qXp] -> a[pXp]. But each time k increases by 1, there is an additional state in the family of productions, and this state can take on any of s values.* **If anyone feels that they could have gotten this problem if they had the clarification about δ(q, b, X) being empty for b ≠ a, please talk to me for an adjustment.***

***Error Codes**: X1 = you used k in place of k-1 (-4). X2 = you forgot that k=0 is a special case (-1).*