Name: ______________________________________
Student ID: ______________________________________

CSE 322 Spring 2010: Take-Home Final Exam

SOLUTIONS

Total: 150 points, 8 questions

Due: Before 4:30pm, Monday, June 7, 2010

Where: CSE Front Desk

Instructions:

1. Write your name and student ID on the first sheet and your last name on all sheets.
2. Write or mark your answers in the space provided. If you need more space, make sure you write down the question number and your name on any additional sheets, and staple these to the exam.
3. If you don't know the answer to a question, don't omit it - do the best you can. You may still get partial credit for whatever you wrote down.
4. Collaboration policy is the same as for the homework assignments.

Name: ____
Student ID: ____

1. **(25 points: 5 each) Circle True (T) or False (F) below. Very briefly justify your answers** (e.g., by contradiction or an example/counter-example, by citing a theorem or result we proved in class, or by *briefly* sketching a construction).

   a. For any two sets A and B, if A is <u>uncountably infinite</u> and B is <u>countably infinite</u>, then $A \cap B$ is countably infinite ……………………………….......................… T F
   Why/Why not?

   FALSE. $A \cap B$ can be empty which is finite. E.g.,
   A=[-2,-1] (set of all real numbers between -2 and -1) and
   B= N, the set of natural numbers.

   b. If R is any <u>regular</u> language and L is any <u>context free</u> language, then $L \circ R$ is context-free …………………..……………………….…………………….. T F
   Why/Why not?

   TRUE. Every regular language is a CFL and CFLs are closed under concatenation as you showed in your homework #5.

   c. The language $\{a^m b^n c^n d^m \mid m, n \geq 0\}$ over $\Sigma = \{a,b,c,d\}$ is <u>not context free</u>... T F
   Why/Why not?

   FALSE. Here's a CFG for the language:
   $S \rightarrow aSd \mid B$
   $B \rightarrow bBc \mid \varepsilon$

Name: ______________________________________
Student ID: ______________________________________

1. (cont.)

d. For any two languages A and B, if A $\subseteq$ B, then A is reducible to B…...… T F
Why/Why not?

FALSE. A hard problem (language) can be a subset of an easy problem and not be reducible to the easy problem. E.g., let A = A_{TM} and B = Σ^*. Then, A $\subseteq$ B but A is not reducible to B (because Σ^* is decidable and A_{TM} is not).

e. If language A is reducible to language B and B is undecidable, then A must be undecidable………………………………....…………………….…….… T F
Why/Why not?

FALSE. The reduction only proves A $\leq$ B, and B's undecidability does not tell us anything about A's hardness. E.g., A = $\varnothing$ (the empty language) is decidable (via a TM that rejects all strings). A can be reduced to B = A_{TM} simply by not using B's decider at all in the reduction and just rejecting all strings.

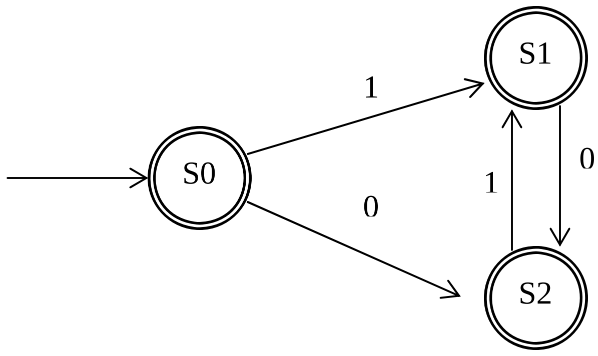
Name: ______________________________________

Student ID: ______________________________________

2. (18 points: 6 each)

Let L = {w | $w \in \{0,1\}^*$ and w contains neither 00 nor 11 as a substring}. Give:

(a) The state diagram of a finite automaton (DFA or NFA) accepting L.



(b) A regular expression denoting L.

The regular expression for the language L is $(1 \cup \varepsilon)\ (01)^*\ (0 \cup \varepsilon)$.

(c) A context-free grammar generating L.

$S \rightarrow A\ T\ B$
$T \rightarrow 01T \mid \varepsilon$
$A \rightarrow 1 \mid \varepsilon$
$B \rightarrow 0 \mid \varepsilon$

You can either follow the constructions given in the lectures/book for converting one of these forms to the other or you can just give a direct answer for each part.

Name: ____
Student ID: ____

3. (15 points)
A language is prefix-closed if the prefix of any string in the language is also in the language. Show that every *infinite prefix-closed context free language* contains an *infinite regular* subset. (Hint: Go over the proof of the pumping lemma for context free languages and see what it implies if the language is also prefix-closed).

Let L be an infinite prefix-closed context free language. Since L is a CFL, the pumping lemma holds. Let p be the pumping length and let s be a string in L longer than p. Then s can be split into uvxyz such that $uv^ixy^iz \in L$ for all $i \geq 0$ and $|vy| \geq 1$. Since L is prefix-closed, all prefixes of s are also in L, therefore, $uv^i \in L$ for all $i \geq 0$. Thus, the regular language $uv^* \subseteq L$. If $v \neq \varepsilon$, uv^* is an infinite regular subset of L, which proves the statement. If $v = \varepsilon$, then $y \neq \varepsilon$ (since we know $|vy| \geq 1$). In this case, uxy^* is an infinite regular subset of L.

Name: ______________________________________
Student ID: ______________________________________

4. (12 points)
A *k*-PDA is a pushdown automaton with *k* stacks. Show that 2-PDAs are more powerful than conventional PDAs with only 1 stack. (Hint: Give a language that you can show is recognizable by a 2-PDA but not by 1-PDAs. An implementation level description is sufficient – no need to formally define the 2-PDA).

The language L = {$a^n b^n c^n$ | n ≥ 0} is not context free as proved in class. Therefore, it cannot be recognized by a 1-PDA. L can however be recognized by the following 2-PDA: Push all initial a's on stack #1. Push the following b's on stack #2. Reject if there is an "a" following any "b". When a "c" is read, pop one a and one b for each c. Reject if an a or b is read. If the input ends and both stacks are empty, accept, otherwise reject.

Alternate solution:
The language L = {$0^n 1^n 0^n$ | n ≥ 0} is not context free as proved in class (see lecture slides). Therefore, it cannot be recognized by a 1-PDA. L is however decidable as shown in class (see lecture slides), i.e., L is accepted by a TM. Problem 5 proves that a 2-PDA can simulate an arbitary TM. Therefore, L can be recognized by a 2-PDA.

Name: ______________________________________
Student ID: ______________________________________

5. (15 points)
Show that a 2-PDA is in fact as powerful as a Turing machine (TM). Describe how a 2-PDA can simulate an arbitrary TM. How is a particular configuration uq_iv of the TM, where $u, v \in \Gamma^*$, represented by the 2-PDA using its two stacks? How does the 2-PDA simulate the TM transitions $\delta(q_i,a) = (q_j,b,L)$ and $\delta(q_i,a) = (q_j,c,R)$? An implementation level description is sufficient – no need to formally define the 2-PDA.

Our Turing machine simulator has 2 stacks, and in each step, it can read (pop) either of the stacks, or push a character onto either of them. It can also read the input. Informally, the 2-PDA does the following:

i. **It pushes the TM start state q_0 onto stack1. It then reads all the input, pushing it onto stack1, until the input is exhausted.**
ii. **It pops stack1, pushing each popped element onto stack2, until it reaches the marker indicating the TM state.**
iii. **At all times, the Turing machine configuration TqaS (where T is the string to the left of the TM head, *q* is the current TM state, *a* is the symbol under the head and S is the string to the head's right) is represented by the 2-PDA configuration: Tq on stack1 (with q at the top of stack1), and aS in the reverse order on stack2 (i.e., with the symbol *a* on top).**
iv. **For each step of the TM, the 2-PDA pops the top of stack1 (which is the TM state), and the top of stack2 (which is the symbol currently under the simulated machine's head).**
v. **This information is all we need to make a TM transition. The 2-PDA simulates $\delta(q,a) = (q', b, L)$ by (i) pushing b onto stack2 (ii) popping stack1 and pushing the symbol obtained onto stack2 (iii) pushing q' onto stack 1. It simulates $\delta(q,a) = (q', b, R)$ by pushing b and q', in that order, onto the top of stack 1. Thus the correct resulting TM configuration is represented in the format of step iii, above.**

One small detail: If at any time at step iv, the 2-PDA finds that stack2 is empty, it first pushes a blank onto stack2, and then performs step iv.

Name: ______________________________
Student ID: ______________________________

6. (20 points: 10 each)

a. Let L_1 and L_2 be any two decidable languages, decided by TMs M_1 and M_2 respectively. Construct a decider TM M for the language $L_1 - L_2$. Give a high level description, starting with M = "On input w: …".

On input w, our decider M does the following:

1. **Run M_1 on input w. The computation is guaranteed to halt, since M_1 is a decider for the language L_1.**
2. **Run M_2 on input w. Again, the computation is guaranteed to halt.**
3. **If M_1 accepted, and M_2 rejected, then accept the string w, else reject.**

b. Let L_1 and L_2 be any two Turing-recognizable languages, recognized by TMs M_1 and M_2 respectively. Prove or Disprove: $L_1 - L_2$ is Turing-recognizable. Either sketch a proof or give a specific counterexample.

L_1-L_2 is not necessarily Turing-recognizable.

Counterexample: Consider $L_1 = \Sigma^*$, and $L_2 = A_{TM}$. We know that L_2 is Turing-recognizable but not decidable. Now L_1-L_2 is the *complement of* the language L_2. If we have a recognizer for a language and its complement, then we have a decider for the language. This is a contradiction, since A_{TM} is undecidable. Hence we cannot always build a recognizer for the language L_1-L_2.

Partial credit (5 points) if an explanation is given as to why the proof in (a) will not work (M_2 may not halt). (This is not a counterexample though)

Name: ______________________________________
Student ID: ______________________________________

7. (30 points: 15, 15 points) For the following proofs, you may use high-level descriptions of the required TMs.

a. Show that for any infinite language L, L is decidable iff some enumerator TM enumerates L in lexicographic order.

Proof.

(→) Let L be a decidable language. Then, there exists a decider TM D such that L(D) = L. We can use D to construct an enumerator E for L as follows:

```
E =     "Ignore the input.
           1. Repeat the following for i = 1, 2, 3, …
           2. Let $w_i$ be the *i*th string in the lexicographic ordering of
              strings in $\Sigma^*$.
           3. Run D on $w_i$. If D accepts, then print $w_i$."
```

Clearly, E prints all strings in L in lexicographic order.

(←) Let E be an enumerator that prints all strings in the language L in lexicographic order. Then, we can construct a decider D for L as follows:

```
D = "On input w:
        1. Run enumerator E and let $w_1$, $w_2$, $w_3$, … be the strings
           printed by E.
        2. If w = $w_i$ for some i, then ACCEPT w and halt.
        3. If at any time E prints a string $w_i$ that comes after w in
           the lexicographic ordering of strings in $\Sigma^*$, then
           REJECT w and halt."
```

Since E is guaranteed to print strings in lexicographic order, if it prints a string that comes after w in lex. order, then we can be sure that it will never print w, and therefore, w is not in L.
It is clear from D's description that D always halts and L(D) = L.

(continued on next page)

Name: ______________________________________
Student ID: ______________________________________

7. (cont.)

b. Show that every <u>infinite Turing-recognizable</u> language has an <u>infinite decidable subset</u>. (Hint: Use the result in (a) and the result you know regarding Turing-recognizable languages and enumerator TMs (Theorem 3.21 in the text)).

Let A be an infinite Turing-recognizable language. Then, there exists an enumerator E that enumerates all strings in A (in some order, possibly with repetitions). We construct another enumerator E' that prints a subset of A in lexicographic order:
"Ignore the input.

1. **Simulate E. When E prints its first string w_1, print w_1 and let *prev_w* = w_1.**
2. **Continue simulating E.**
3. **When E is ready to print a new string *w*, check to see if *w* is longer than *prev_w* (this ensures *w* occurs after *prev_w* in lex. order). If so, then print *w* and let *prev_w* = *w*, otherwise do not print *w*.**
4. **Go to 2."**

It is clear that E' as constructed above only prints strings in A, therefore its language is a subset of A. Since A is infinite, there will always be strings in A longer than the current *prev_w* – E will eventually print one of these and so will E' (and update *prev_w*). Therefore, the language of E' is also infinite. Finally, since E' only prints strings in lexicographic order, its language is decidable as proved in (a). Thus, the language of E' is an infinite decidable subset of A.

Name: ____
Student ID: ____

8. (15 points)
After proposing a toast to "Touring machines," an already toasted CS major from [name-deleted] university claims that the problem of figuring out whether a given TM accepts a finite number of strings is simple enough to be decidable. Having taken CSE 322, you know better, so you define the language $FINITE_{TM} = \{ \langle M \rangle \mid M$ is a TM and $L(M)$ is finite$\}$. Show that $FINITE_{TM}$ is undecidable by giving a reduction from a known undecidable language to $FINITE_{TM}$. For your reduction, you may use any of the languages shown to be undecidable in Section 5.1 in the textbook (up to Theorem 5.4 and its proof). (Hint: Use a reduction roughly along the lines of the one used in Theorem 5.2 and discussed in class.)

We use a construction similar to Theorem 5.2, and reduce A_{TM} to the problem $FINITE_{TM}$. We need to show that given a decider for $FINITE_{TM}$, we can use it to solve A_{TM}, making A_{TM} decidable (a contradiction). Therefore, $FINITE_{TM}$ must be undecidable. We can build the decider for A_{TM} as follows:
On input $< M, w >$,

(a) Construct a new machine M' to do the following: on any input x, it ignores the input and instead simulates M on w on a separate tape. If M accepts w then M' accepts x, otherwise *M'* rejects or loops on *x*.
(b) Feed the input $< M' >$ to the decider for $FINITE_{TM}$. If it rejects$< M' >$, then we accept the input $< M, w >$, else we reject.

Here is what is happening: If it is the case that M accepts w, then our constructed machine M' will accept any input x given to it. As a result, its language is Σ^*, an infinite language. Otherwise its language is ϕ (finite). Our decider for $FINITE_{TM}$ will answer the question of whether M''s language is finite or infinite, and we can therefore use this answer to decide whether M accepts or rejects w. Thus a decider for $FINITE_{TM}$ yields a decider for A_{TM} which is a contradiction.

[End of Exam]

Have a great summer!