Name: ______________________________

Student ID: ______________________________

# CS181 Winter 2010 - Final

Due Friday, March 12, 2010

You are limited to 10 hours (not necessarily contiguous) to take this exam. It must be turned in during discussion by 4:00 P.M. on Friday, March 12, 2010. This exam is open-book and open-notes, but any materials not used in this course are prohibited. The exam is to be taken individually without any outside help, except possibly from the professor or the TA, within the time limits set forth. There are a total of 5 questions and the exam is worth 175 points. Place your name and UID on every page of your solutions. **Staple this cover sheet to your solutions. Please use separate pages for each question.**

Please **handwrite** the following honor code agreement and sign and date in the spaces provided.
**Honor Code Agreement**: I promise and pledge my honor that, for the purpose of this exam, I am not to collaborate with any person except for the professor or the TA, nor will I refer to any material except for the class textbook and my own class notes. I will abide by the CS181 Honor Code.

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

Signature: ______________________________

Date: ______________________________

| **Question** | **1** | **2** | **3** | **4** | **5** | Total |
|---|---|---|---|---|---|---|
| **Points** | | | | | | |

1. For each of the following questions, give a rigorous proof for your answer.

   (a) (**15 pts.**) Let $w$ be an arbitrary fixed string. Let $L_w = \{\langle M \rangle \mid M \text{ is a Turing Machine that accepts } w\}$. Prove that for every string $w$, $L_w$ is Turing-Recognizable but not decidable.

   (b) (**10 pts.**) Let $A$ and $B$ be arbitrary Turing-Recognizable languages such that $\overline{A \cup B}$ is also Turing-Recognizable. Prove that $A$ and $B$ must both be decidable.

   (c) (**15 pts.**) Define an $n$-PDA to be exactly like an ordinary PDA except that it is only allowed to have at most $n$ symbols on the stack. Let $REG$ be the class of regular languages, $CF$ be the class of context free languages, and $THOUSAND$ be the class of languages recognized by 1000-PDAs. Exactly one of the following three statements is true. Decide which one is true and give a rigorous proof for why it is true (you need not prove why the other two statements are false):

      i. $CF = THOUSAND$

      ii. $REG \subset THOUSAND \subset CF$

      iii. $REG = THOUSAND$

      Note that $A \subset B$ means that $A \subseteq B$ but $A \neq B$.

2. This problem is inspired by the Büchi automaton question from the midterm. Define a MalBüchi Turing Machine $M$ to be one that also has *non-halting accept states*, and we say that $M$ accepts a string if it halts and accepts OR if it never halts but visits an accepting state infinitely often. It rejects if it halts and rejects OR if it never halts and only visits accepting states finitely many times. Unlike Büchi automata though, **the inputs are still ordinary finite length strings**. The set of strings that a MalBüchi Turing Machine $M$ accepts is the language recognized by $M$. Call a language *Mal-Recognizable* if there is some MalBüchi Turing machine that recognizes it.

   (a) (**15 pts.**) Recall that we showed that $\overline{A_{TM}}$ is not Turing Recognizable. In contrast, prove that $\overline{A_{TM}}$ for ordinary Turing Machines is Mal-Recognizable.
   (For the purpose of this problem,
   $\overline{A_{TM}} = \{\langle M, w\rangle \mid M \text{ is a Turing Machine and } M \text{ does not accept } w\}$.)

   (b) (**15 pts.**) Use your knowledge of different kinds of infinities to show that there must exist a language that is not recognized by any MalBüchi Turing Machine.

   (c) (**15 pts.**) Use diagonalization to construct an explicit language $L$ that is not recognized by any MalBüchi Turing Machine.
   (Note that by an explicit language $L$, we mean a language that you can define. An example of this is the language of Self-Hating Turing Machines:
   $SHT = \{\langle M\rangle \mid M \text{ is a Turing Machine and } M(\langle M\rangle) \text{ does not accept}\}$.)

3. We consider a new machine called a *Numerical Automaton*. A Numerical Automaton $N$ is similar to an NFA except that $\epsilon$-transitions are not allowed and, in addition, it has a counter at its disposal. The counter begins at 1, and the Numerical Automaton can either add 1 to the counter, subtract 1 from the counter, or double the counter (i.e. multiply by 2). It must perform exactly one of these three actions with each transition taken, and the Numerical Automaton is not allowed to look at the current value of the counter. Moreover, the accepting conditions for Numerical Automata (see below) are entirely determined by the counter, and hence we make no distinction between accepting versus non-accepting states.

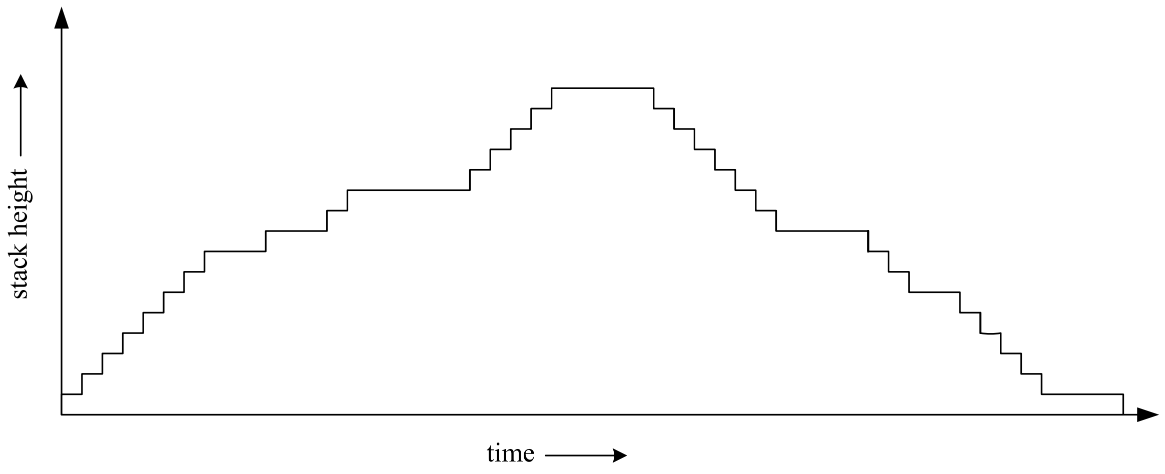
   Formally, a Numerical Automaton $N$ is a 4-tuple $(Q, \Sigma, \delta, q_0)$, where $Q$ is a finite set of states, $\Sigma$ is the input alphabet, $q_0$ is the initial state, and $\delta : Q \times \Sigma \to \mathcal{P}(Q) \times \{+1, -1, \times 2\}$ is the transition function. For example, if $N$ is at state $p$, moves to state $q$ while consuming symbol $a$, and doubles the counter, this can be represented as $\delta(p, a) = (\{q\}, \times 2)$. The accepting conditions for Numerical Automata are as follows. The moment the counter reaches 0, the machine accepts if and only if there is no more input to be consumed. If the entire input has been consumed before the counter ever reaches 0, the machine automatically rejects. The set of strings accepted by $N$ is the language of $N$. We say that a Numerical Automaton $N$ recognizes language $L$ if $L = L(N)$. We call a language *Numerical* if it is recognized by some Numerical automaton.

   (a) (**5 pts.**) Prove that there exists a context-free language $L$ which is not Numerical.
   (*Hint: A singleton language should suffice.*)

   (b) (**15 pts.**) Prove that there exists a Numerical language $L$ which is not context-free.

4. Call a CFG a *Straightline Grammar* if every rule fits one of the following three forms: $C \to wD$, $C \to Dw$, or $C \to \epsilon$, where $w \in \Sigma^*$ and $C, D \in V$. We say a context-free language $L$ is *Straightline* if there is some Straightline Grammar $G$ such that $L = L(G)$.

   Now, consider a restricted form of PDAs. We say that a PDA $P$ is *Hilly* if, once $P$ pops something off the stack, it is never allowed to push something onto the stack again. Every transition either pushes a symbol, pops a symbol, or consumes an input symbol without modifying the stack. Hence, the stack diagram can look as follows:



   A *Hilly* language is one which has a Hilly PDA that accepts it.

   (a) (**20 pts.**) Prove that every Straightline language is Hilly.
   (*Hint: Try using a state for each variable. You might also need some extra states.*)

   (b) (**5 pts.**) Suppose we have a grammar in which every rule fits one of the following two forms: $C \to w_1 D w_2$ or $C \to \epsilon$, where $w_1, w_2 \in \Sigma^*$ and $C, D \in V$. Prove that there exists a Straightline Grammar that is equivalent. That is, given a grammar $G$ in which every rule fits one of the two forms, construct a Straightline Grammar $G'$ such that $L(G) = L(G')$.

   (c) (**15 pts.**) Prove that every Hilly language is Straightline.
   (*Hint: Do not forget to consider transitions which leave the stack unmodified.*)

5. We explore another use of the Recursion Theorem for this problem. We say that a language $L$ is *rich* if there is a Turing Machine $D$, which we will call the *Detective*, that computes a function and always halts such that for any Turing Machine $M$ which has $L(M) \subseteq L$, it must be the case that $D(\langle M \rangle)$ is in $L$ but not in $L(M)$.

   (a) (**10 pts.**) Prove that a rich language is not recursively enumerable.
   (Note: You do not need the Recursion Theorem for this part.)

   For this next part, if you really want to challenge yourself, do not look at the last page as it contains a really big hint.

   (b) (**20 pts.**) In this part, we use the Recursion Theorem as a way to obtain a string in a language. Suppose we have some language $L$ which satisfies the property that $\overline{L}$ is rich. Since $\overline{L}$ is rich, there is a Turing Machine $D$ (the Detective) where for any Turing Machine $M$ we have $L(M) \subseteq \overline{L} \Rightarrow D(\langle M \rangle)$ is in $\overline{L}$ but not in $L(M)$. If the *only thing you are given* is the Detective $D$, show how to get a string in the language $L$.

   Note that it is easy to get a string in $\overline{L}$, since we can construct a Turing Machine $E$ such that $L(E) = \emptyset$. Hence, we have $L(E) = \emptyset \subseteq \overline{L} \Rightarrow D(\langle E \rangle)$ is in $\overline{L}$ but not in the empty set, which implies that $D(\langle E \rangle)$ is in $\overline{L}$.

   The challenge for you in this problem is to show how to obtain a string in $L$ if the only thing you are given is $D$.

As a (very big) hint to Problem 5, Part b), consider the following Turing Machine $T$ which uses the Recursion Theorem to obtain its own description and the Detective $D$ for $\overline{L}$:

```
Construct T = "On input $x$:
                  Obtain, via the Recursion Theorem, own description $\langle T \rangle$
                  If $x = D(\langle T \rangle)$:
                      accept
                  Otherwise:
                      reject"
```