UVa - cs302: Theory of Computation Spring 2008

Exam 2

Due: 8 April 2008

Your Name:

UVa Email Id:

Honor Policy. For this exam, you must **work alone**. You may consult the single (two-sided) page of notes you brought, but may not look at any other materials or aid or accept aid from other students.

Directions. Answer all four questions, including all sub-parts. You may use the backs of pages for your scratch work, but we will only grade answers that are written in the answer boxes, or that are found following clearly marked arrows from these boxes. The box for each answer is designed to be big enough to easily fit a full credit, correct answer. If you feel like you need more space to write your answer, then either your answer is incorrect, inelegant, or you are providing more detail than needed for full credit.

Question	Target	Score
1	5-10-10	
2	10-15	
3	5-5-10	
4	10-10-10	
Total	100	

Problem 1: Short Answers. (25) For each question, provide a correct, clear, precise, and concise answer from the perspective of a theoretical computer scientist.

a. [5] What is a *language*?

b. [10] Explain the essence of the *Church-Turing Thesis* in a way that would be understandable to a typical fifth grader.

c. [10] What does it mean for a language to **not** be *Turing-recognizable*?

Problem 2: Turing Machines. (25)

a. [10] Provide an implementation-level description of a Turing machine that decides the language $A = 0^n 1^m 0^{n/m}$ where $n \geq 0$ and $m \geq 1$ (that is, the input is n zeros, followed by m ones, followed by n divided by m zeros). To be in the language, n must be divisible by m. For example, strings in the language include 00100, 00110, and 00001100; but the string 000110 is not in the language.

b. [15] Define a *cyclical Turing machine* as a Turing machine with a one-way infinite tape (as in our standard Turing machine definition), except that the left edge behavior is defined differently. Instead of staying in the leftmost square, if a cyclical Turing machine would move left past the left edge of the tape, the head moves to the rightmost non-blank square.

Prove that the class of languages that can be recognized by a cyclical Turing machine is identical to the class of languages that can be recognized by a regular Turing machine.

Problem 3: Problematic Proofs. (20)

Each of these "proofs" claims to prove a conjecture. Some of the conjectures are false, others may be true, but *all* the proofs are bad. For each proof, explain clearly why the provided proof does not satisfactorily prove the given conjecture. Your answer should identify a fundamental flaw in the proof (for example, by explaining what it actually proves instead of the conjecture), not a minor issue like lack of sufficient detail.

a. [5] **Conjecture:** *The set of languages that can be recognized by a deterministic push-down automaton is equivalent to the set of languages that can be recognized by a Turing machine.*

Proof by Simulation. We prove the conjecture by showing how to simulate any DPDA with a TM.

We can simulate the stack by using the TM's tape, writing a mark after the end of the input and then writing the stack from left to right on the TM tape. The top of the stack is represented by the leftmost square on the tape after the mark.

To simulate a *push*, we move the tape head to the first blank, and then move right to left copying the contents of each square into the square to its right. Then, we put the pushed symbol in the leftmost square.

To simulate a *pop*, we read the top of the stack, and then move left to right, copying the symbol in each square into the square to its left and overwriting the rightmost non-blank square with a blank.

To simulate the DPDA processing the input, we start by adding a dot to the first input symbol, and then mark each successive input symbol with a dot as the input is processed from left to right.

Thus, we can simulate a DPDA with a TM. This proves a DPDA is equivalent to a TM.

b. [5] **Conjecture:** Define $H(L)$ as the set of even-length strings in L. That is,

$$H(L) = \{w | w \in L \text{ and } |w| = 2k \text{ for some } k \geq 0\}$$

$H(L)$ is closed for context-free languages. (Note: this is based on the original uncorrected Exam 1 comments.)

Proof: Since L is a CFL, there exists a DPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ that recognizes L. We show how to construct a DPDA $M_H = (Q_H, \Sigma, \Gamma, \delta_H, q_{0H}, F_H)$ that recognizes $H(L)$. The basic idea is to make two copies the states of Q, corresponding to having seen an odd or even number of input symbols so far, and duplicating all the transition rules.

$Q_H = Q \times \{even, odd\}$ — states in M_H keep track of the state in M, and whether an even or odd number of symbols have been seen so far.
$q_{OH} = (q_0, even)$
$F_H = \{(q_f, even) | q_f \in F\}$ δ_H is defined by:

$$\delta_H((q, even), x, \gamma) = ((q_r, odd), \beta) \text{ if } \delta(q, x, \gamma) \rightarrow (q_r, \beta)$$
$$\delta_H((q, odd), x, \gamma) = ((q_r, even), \beta) \text{ if } \delta(q, x, \gamma) \rightarrow (q_r, \beta)$$

c. [10] **Conjecture:** The "Busy Bee" language, defined below, is undecidable:

> $L_{BusyBee} = \{\langle M, w, k \rangle\}$ where M describes a Turing machine, and k is the number of different FSM states M enters before halting on input w. (Note that q_{Accept} and q_{Reject} are counted as states for the number of different states count.)

Proof by Reduction. We prove that $L_{BusyBee}$ is undecidable by reducing $HALT_{TM}$ to it.

Construct $M_{BusyBee}$ given a machine M_{HALT} that decides $HALT_{TM}$:

$M_{BusyBee}$ =

```
Simulate $M_{HALT}$ on input $\langle M, w \rangle$.
If $M_{HALT}$ accepts, simulate $M$ on $w$. On a second tape, list the states of
    $M$, and add a mark on each state that is entered. Count the number of marked states on the second tape. If the number matches $k$,
    **accept**.
Otherwise, **reject** (without simulating $M$ on $w$ since it does not halt).
```

Since building $M_{BusyBee}$ requires M_{HALT}, which we know does not exist, this proves that $L_{BusyBee}$ is undecidable.

Problem 4: Decidability. (30) For each part, state clearly whether the described language is *decidable* or *undecidable*. Support your answer with a convincing proof. You may use any of the theorems we have proven in class or in the book, but you may not use Rice's theorem in your proof unless you also include a proof of Rice's theorem.

a. [10] $LONGER_{DFA} = \{\langle A, B \rangle\}$ where A and B are descriptions of DFAs and the longest string that is accepted by A is longer than the longest string that is accepted by B.

b. [10] $NOTSUB_{TM} = \{\langle A, B \rangle\}$ where A and B are descriptions of Turing machines and there is some string w which is accepted by A that is not accepted by B (that is, the language accepted by A is not a subset of the language accepted by B).

c. [10] $L_{BusyBee} = \{\langle M, w, k \rangle\}$ where M describes a Turing machine, and k is the number of different FSM states M enters before halting on input w. (Note that q_{Accept} and q_{Reject} are counted as states for the number of different states count.) (The same language as in Problem 3c.)

End of Exam