UVa - cs302: Theory of Computation Spring 2008

Exam 2 Preview Comments

Score Distributions

Question	Target	0	1	2	3	4	5	6	7	8	9	10	11-14	15
1a	5	0	0	2	1	0	78							
1b	10	3	0	4	8	28	12	8	1	12	2	2	0	1
1c	10	3	0	2	7	7	9	10	19	3	2	19		
2a	10	1	0	1	0	0	1	7	17	12	11	31		
2b	15	0	0	0	1	1	4	3	2	3	3	15	20	29
3a	5	12	4	1	2	0	62							
3b	5	28	3	1	1	2	46							
3c	10	12	4	5	2	0	3	0	3	0	0	52		
4a	10	1	0	15	5	3	7	4	9	13	8	7	0	2
4b	10	2	0	9	2	2	5	22	14	10	3	5	3	4
4c	10	5	0	5	4	5	3	15	20	8	4	3	0	9

	< 50	50–59	60–69	70–79	80–89	90-106
Total	6	15	17	21	14	8

Note: Answers are not provided for questions 1b, 2b, 4b, and 4c. If you are not satisfied with your exam score, you may increase your score by turning in correct, well-written, and clear answers to as many of those questions as you wish. Revised answers must be turned in on paper at the beginning of class of **Tuesday, April 15**.

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*?

Answer: A set of strings.

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

Answer: No solution provided (now), to give opportunity for revised answers.

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

Answer: A language, L, is not Turing-recognizable if there exists no TM that can recognize L. This means all possible TMs must either:

- Accept some string that is not in L, or
- Fail to accept some string that is in L (either by rejecting it, or not halting on it).

Problem 2: Turing Machines. (25)

a. [10] Provide an implementation-level description of a Turing machine that decides the language $A = 0^n1^m0^{n/m}$ 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.

Answer: First, we observe that the revsere of A is the language $0^i1^j0^k$ where $k = ij$. This is easily recognized by a TM, as described in example 3.11. So, all we need to do is replace the initial input with its reverse, and then simulate the multiplication machine. We can reverse the input by first copying it: start by writing a # at the end of the input, then go left to right over the input, find the first unmarked symbol, marking it, and finding the first blank on the tape and overwriting it with that symbol. Once all the input is marked (the # is reached), then copy the copy back into the input space but start from the end of the copy (replacing each symbol with a blank instead of marking it), but putting the symbols on the tape in reverse order. Then, simulate the multiplication machine.

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.

Answer: No solution provided (now), to give opportunity for revised answers.

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 pushdown 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. . . . (unnecessary details elided) . . . Thus, we can simulate a DPDA with a TM. This proves a DPDA is equivalent to a TM.

Answer: The proof only covers one direction: it shows a DPDA can be simulated by a TM. To show the sets of languages they recognize are equivalent, we need to show *both* directions (in this case, this would be impossible, since it is not possible for a DPDA to simulate a TM). Hence, what is actually proven here is that a DPDA is no more powerful than a TM (which is not surprising, since the set of languages that can be recognized by a DPDA is a subset of the Turing-recognizeable languages).

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. . . . (unnecessary details elided) **Answer:** As pointed out in the Exam 1 corrected comments, one cannot assume a DPDA exists for any CFL. There is a *nondeterministic* PDA that can recognize every CFL, but some CFLs cannot be recognized by a DPDA.

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.

Answer:

The proof does the reduction in the wrong direction. It shows that you can decide $L_{BusyBee}$ if you can decide $HALT_{TM}$ (that is, that $L_{BusyBee}$ is not *harder* than $HALT_{TM}$), but it does

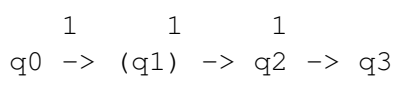
not show the conjecture which is that $L_{BusyBee}$ is undecidable. To show that, we would need a reduction that shows how to solve $HALT_{TM}$ if you were given a machine that can solve $L_{BusyBee}$.

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.

Answer:

This is very similar to the $BIGGER_{DFA}$ problem from PS5 that we discussed in Class 19 (and the PS5 comments). The only difference is at the end instead of counting the number of strings in each language, we need to find which language has the longest string. As we enumerate the strings (up to the number of states in the bigger DFA), we just keep track of the longest strings accepted by A and B. At the end, accept if the longest string accepted by A is longer than the longest string accepted by B. Note that it is not correct to claim that all you have to do is check for cycles, and then count the states. The number of states gives you the *maximum* possible length of an accepted string for a DFA with no cycles, but it does not give you the length of the longest accepted string. For example, here is a machine with 4 states where the longest accepted string is only 2 symbols:



where $q1$ is the only accepting state.

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).

Answer: No solution provided (now), to give opportunity for revised answers.

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.)

Answer: No solution provided (now), to give opportunity for revised answers.