CS373, Spring 2010. Midterm 2

INSTRUCTIONS (read carefully)

- Fill in your name, netid, and discussion section time below. Also write your netid on the other pages (in case they get separated).

NAME:	
NETID:	**DISC:**

- There are 6 problems. Make sure you have a complete exam.
- The point value of each problem is indicated next to the problem, as well as in the table below.
- Points may be deducted for solutions which are correct but excessively complicated, hard to understand, or poorly explained. Please keep your solutions ***short*** and crisp.
- The "I DON'T KNOW" rule does apply. If you do not know the answer to a problem, you can simply write "I DON'T KNOW" and you will get 20% of credit for that problem. The number of points you get in this manner cannot exceed 10 points across the whole exam.
- The exam is designed for one hour and thirty minutes, but you have the two full hours to finish it.
- It is wise to skim all problems and point values first, to best plan your time.
- This is a closed book exam. No notes of any kind are allowed. Do all work in the space provided, using the backs of sheets if necessary. See the proctor if you need more paper.
- Please bring any apparent bugs to the attention of the proctors.
- After the midterm is over, discuss its contents with other CS 373 students **only** after verifying that they have also taken the exam (e.g. they aren't about to take the conflict exam). *There are many people taking the conflict exam— so please make sure before you discuss!*
- We indicate next to each problem how much time we suggest you spend on it. We also suggest you spend the last 25 minutes of the exam *reviewing* your answers.

Problem	Possible	Score
1	10	
2	15	
3(a)	13	
3(b)	12	
4(a)	10	
4(b)	15	
5	15	
6	10	
Total	100	

Problem 1: Yea or Nay (10 points)

[10 minutes]

The answers to these problems should be just choosing True or False; no other explanation is necessary.

(A) All finite languages are regular.

☐ True ☐ False

Solution:

True. Append the members of finite set using $\cup$ to obtain a regular expression for that set.

(B) If L is regular and Turing-recognizable, then L is Turing-decidable.

☐ True ☐ False

Solution:

True. Membership problem for regular laguages is decidable.

(C) If L is regular and $L' \subseteq L$, then L' is regular.

☐ True ☐ False

Solution:

False. For example every language on alphabet Σ is a subset of Σ^* which is regular.

(D) If L is Turing-decidable and L' is regular, then $L \cap L'$ is regular.

☐ True ☐ False

Solution:

False. Pick $L = \{0^n 1^n : n \geq 0\}$ and $L' = \{0, 1\}^*$.

(E) The language $L = \{\langle D \rangle \mid D \textit{ is a DFA and there exists a TM } M \textit{ such that } L(M) = L(D)\}$ is Turing-decidable.

☐ True ☐ False

Solution:

True. Every DFA can be simulated by a TM, so the decider of L just needs to check that $\langle D \rangle$ is valid encoded DFA.

(F) The language containing all English words written in all websites in the world wide web is regular.

☐ True ☐ False

Solution:

True. It is a finite set and therefore regular.

(G) If $L \subseteq \Sigma^*$ is Turing-recognizable and $\Sigma^* \setminus L$ is Turing-recognizable, then there is a TM that decides L.

☐ True ☐ False

Solution:

True. The decider for L on input w simulates recognizers of L and $\overline{L}$ on w and accepts if the former accepts and rejects if the later rejects (and since w is exactly in one of L or $\overline{L}$, exactly one of those cases happen).

(H) If L_1 reduces to L_2 and L_2 is undecidable, then L_1 is undecidable.

☐ True ☐ False

Solution:

False. For example pick $L_1 = \emptyset$ and $L_2 = A_{TM}$.

(I) There is a TM that takes in input TM $\langle M \rangle$ and converts it to a TM $\langle M' \rangle$ such that for any w, M' accepts w iff M halts on w.

☐ True ☐ False

Solution:

True. For example this conversion:

Algorithm $M'(w)$
1. Simulate $M(w)$
2. **return true**

(J) Every Turing machine is a decider of some language.

☐ True ☐ False

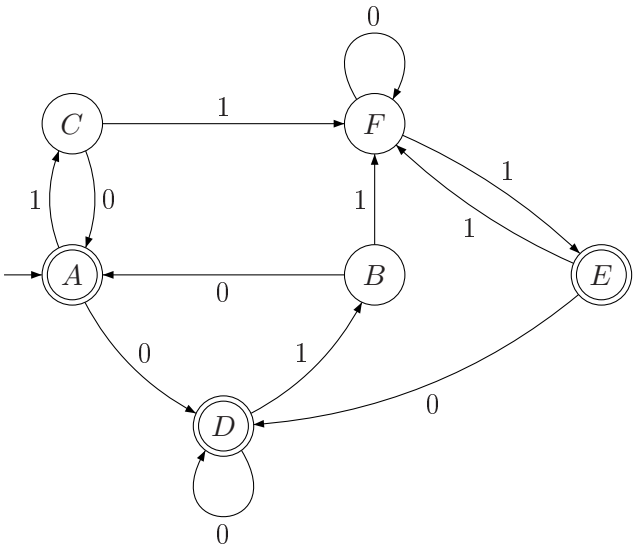
Solution:

False. If there is an input w that makes the TM never halt, then that TM can not decide any set by definition.

Problem 2: Minimization (15 points)

[15 minutes]

Minimize the following DFA and draw the result DFA. You must use the partition-based minimization algorithm to minimize the DFA, and show the steps of the minimization, including the partitions at every stage.



Solution:

We start with the partition that groups together the non-final states together and the final states together. So

$$P_0 = \{\{A, D, E\}, \{B, C, F\}\}$$

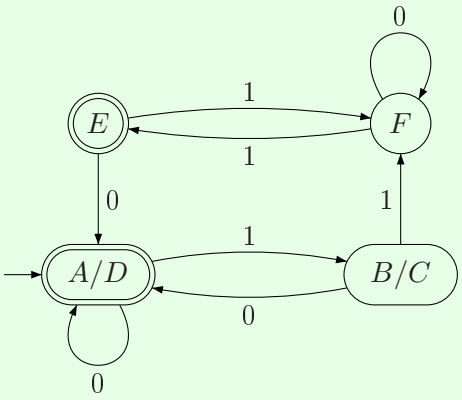
We say $S_1 = \{A, D, E\}$ and $S_2 = \{B, C, F\}$. By reading 0 or 1, each state in S_1 goes to (S_1, S_2). But for states in S_2, B and C go to (S_1, S_2) and F goes to (S_2, S_1). Hence we get a new partition of states which is a refinement of P_0:

$$P_1 = \{\{A, D, E\}, \{B, C\}, \{F\}\}$$

Now examine the states in S_1 again, A and D remain in the same partition, while D and E are differentiated by 1. Then we form a new partition

$$P_2 = \{\{A, D\}, \{E\}, \{B, C\}, \{F\}\}$$

Now we are done since iterating to refine P_2 gives the same partition. So the partition P_2 gives the minimal automaton as follows:



Problem 3: Non-regularity (13+12 points)

[20 minutes]

a) Prove that $L_{odd-sq} = \{0^{(2n+1)^2} | n \geq 0\}$ is non-regular, from first principles, using Myhill-Nerode Theorem or the Pumping Lemma.

You cannot assume the non-regularity of any language to solve this problem.

Solution:

We use MNT to prove this claim.

Let $S = L_{odd-sq}$, obviously S is an infinite set. Let $x, y \in S$ and $x \neq y$. Then $x = 0^{(2i+1)^2}$, $y = 0^{(2j+1)^2}$. Without loss of generality, let $i < j$. Now we choose our witness $z = 0^{8(i+1)}$, then $xz = 0^{(2i+1)^2}0^{8(i+1)} = 0^{(2i+3)^2} \in L_{sq}$. But $yz = 0^{(2j+1)^2}0^{8(i+1)} = 0^{4j^2+4j+8i+9}$. Since $(2j+1)^2 < 4j^2 + 4j + 8i + 9 < 4j^2 + 12j + 9 = (2j+3)^2$, $yz \notin L_{sq}$. Hence by MNT L_{odd-sq} is non-regular.

b) Use the above result to prove that $L_s = \{0^{n^2+n} | n \geq 0\}$ is non-regular by closure properties.

You are allowed to assume the non-regularity of only one language, namely L_{odd-sq}, and further you are required to use closure properties of regular languages only to solve this problem (you are not allowed to use the Myhill-Nerode Theorem or the Pumping Lemma).

Solution:

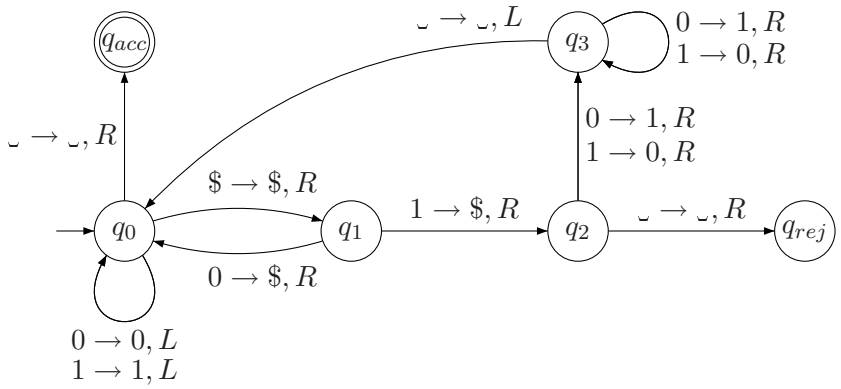
We prove this claim by the closure of regular language under homomorphism and concatenation.

Assume L_s is regular. Define a homomorphism $h(0) = 0000$. Then by the closure properties under homomorphism and concatenation, $h(L_s) \circ 0 = \{0^{4n^2+4n+1} | n \geq 0\} = L_{odd-sq}$ is also regular, which contradicts the above result. Therefore L_s is non-regular.

Problem 4: TM notation and design (10+15 points)

[20 minutes]

4(a) Consider the TM described by the following diagram.



Answer the following questions for the above TM:

(a) What is $\delta(q_2, 0)$? And what is $\delta(q_1, 1)$?

(b) Is the tape alphabet Γ the same as input alphabet Σ, provided that $\Sigma = \{0, 1, \$\}$?

(c) Which of the following strings are accepted: \$11, \$110?

Solution:

(a) $(q_3, 1, R)$ and $(q_2, \$, R)$.

(b) No. T contains at least one more symbol $\sqcup$.

(c) \$11.

4(b) Let $L \subseteq \Sigma^*$ be some language over alphabet $\Sigma = \{0, 1\}$, such that $\epsilon \in L$, $0 \in L$ and $1 \notin L$.

Suppose that you're given a TM *Simp* that, when given a word $w \in \Sigma^*$ as an input, where $|w| > 1$, transforms it to an output containing two strings $w_1, w_2 \in \Sigma^*$ such that:

(a) $w \in L$ if and only if ($w_1 \in L$ and $w_2 \in L$)

(b) $|w_1| < |w|$ and $|w_2| < |w|$, where $|x|$ denotes the length of x

Intuitively, given a word w, *Simp* does not decide whether $w \in L$, but gives back two words of shorter length than w such that $w \in L$ if and only if the two words it outputs are in L.

Show that L is decidable. Give a high-level description of a decider D for language L, using *Simp* as part of D. You may assume that *Simp* always halts on any input $w \in \Sigma^*$.

Solution:

D has the following tapes:

(a) Q - input taps. Stores a list of words, is used as a FIFO queue

(b) W - stores current word to process, input tape of *Simp*

(c) $W1$, $W2$ - output tapes of *Simp*

The high-level algorithm is:

On input x:

1) if tape Q is empty - **accept**
2) clear W
3) copy the first word from Q to W
4) remove first word from Q
5) if word on W is 1 - **reject**, if it's 0 or ϵ - go to 1)
6) run *Simp*
7) append word on $W1$ to then end of queue Q
8) append word on $W2$ to then end of queue Q
9) go to 1)

Problem 5: Reductions (15 points)

[15 minutes]

a) Fix an encoding and ordering of Turing machines, $M_1, M_2, \ldots$.
Fix an encoding and ordering of all words $w_1, w_2, \ldots$. Assume these are computable, i.e. for any i, a TM can compute M_i and w_i, and also, given M compute the i such that $M = M_i$, and similarly, given w, compute i such that $w = w_i$.

Assume that the following language is not decidable:

$L_d = \{w \mid \exists i.\ w = w_i \textit{ and } M_i \textit{ does not accept } w_i\}$.

Prove that $A_{TM} = \{\langle M, w\rangle \mid M \textit{ is a TM that accepts } w\}$ is undecidable using a reduction.

(Note: You are not, of course, allowed to assume that A_{TM} is undecidable!)

Solution:

We reduce L_d (our known undecidable set) to A_{TM}, that is we build a decider D_{L_d} for L_d using an assumed decider $D_{A_{TM}}$ for A_{TM}. We know that given w, we can compute i such that $w_i = w$ and then by assumptions we can compute M_i. Now we need to check whether M_i accepts w_i or not, basically we want to know if $\langle M_i, w_i \rangle \in A_{TM}$? But this is a question that we can answer using $D_{A_{TM}}$. Here is the code:

Algorithm $D_{L_d}(w)$
1. Compute i such that $w_i = w$
2. Compute $\langle M_i \rangle$
3. **return** $\neg\, D_{A_{TM}}(\langle M_i, w_i \rangle)$

Problem 6: Reductions (10 points)

[10 minutes]

b) Using the fact that A_{TM} is undecidable, show that the following is undecidable using a reduction: $L = \{\langle M, w \rangle \mid M \textit{ is a TM that does not accept } w\}$.

Solution:

Now we reduce A_{TM} to L, that is we build a decider $D_{A_{TM}}$ for A_{TM} using a decider D_L for L. We just need to observe that $\langle M, w \rangle \in A_{TM} \iff \langle M, w \rangle \notin L$.

Algorithm $D_{A_{TM}}(\langle M, w \rangle)$
1. **return** $\neg\, D_L(\langle M, w \rangle)$