CS 373: Intro to Theory of Computation
Spring 2010 MockMidterm II, March, 2010

INSTRUCTIONS (read carefully)

- Print your name and netID here and netID at the top of each other page.

NAME:

NETID:

- It is wise to skim all problems and point values first, to best plan your time. If you get stuck on a problem, move on and come back to it later.
- Points may be deducted for solutions which are correct but excessively complicated, hard to understand, hard to read, or poorly explained.
- 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 apparent bugs or unclear questions to the attention of the proctors.

Problem 1: True/False (10 points)

Completely write out "True" if the statement is necessarily true. Otherwise, completely write "False". Other answers (e.g. "T") will receive credit only if your intent is unambiguous. For example, "$x + y > x$" has answer "False" assuming that y could be 0 or negative. But "If x and y are natural numbers, then $x + y \geq x$" has answer "True". You do not need to explain or prove your answers.

1. Let M be a DFA with n states such that $L(M)$ is infinite. Then $L(M)$ contains a string of length at most $2n - 1$.

Solution:

True.

2. Let $L_w = \{\langle M \rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$, where w is some fixed string. Then there is an enumerator for L_w.

Solution:

True.

3. The set of undecidable languages is countable.

Solution:

False.

4. There is a bijection between the set of Turing-recognizable languages and the set of decidable languages.

Solution:

True.

5. If L is a non-regular language over Σ^*, and h is a homomorphism, then $h(L)$ must also be non-regular. Is this statement correct?

Solution:

No. Suppose that h mapped all characters to the empty string. Then $h(L)$ would be regular no matter what L is.

6. Suppose all the words in language L are no more than 1024 characters long. Then L must be regular. Is this statement correct?

Solution:

Yes. There's only a finite set of strings with ≤ 1024 characters. So L is finite and therefore regular.

7. A minimum size NFA for a regular language L, always has strictly fewer states then the minimum size DFA for the language L. True or False?

Solution:

False, consider Σ^*.

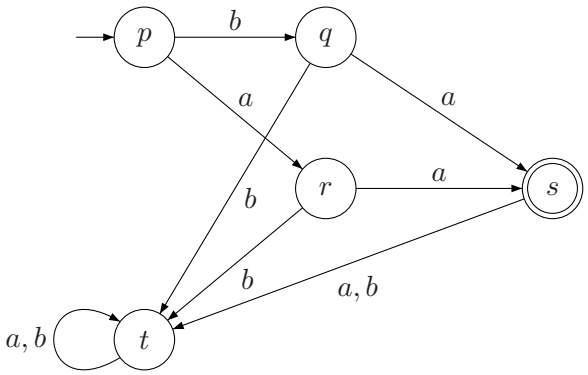
8. Let L_i be a regular language, for $i = 1, \ldots, \infty$. Is the language $\bigcup_{i=1}^{\infty} L_i$ always regular? True or false?

Solution:

False, let $L_i = \{a^i b^i\}$.

Problem 2: Minimization (20 points)

Minimize this DFA:



Solution:

We start with the partition that separates final states from non-final states:

$$P_0 = \{\overbrace{(p, q, r, t)}^{A}, \overbrace{(s)}^{B}\}$$

Partite set B is a singleton and can't be refined anymore. In A, by reading a and b, states q and r transit to (B, A) while p and t transit to (A, A). Therefore we need to p, t from q, r:

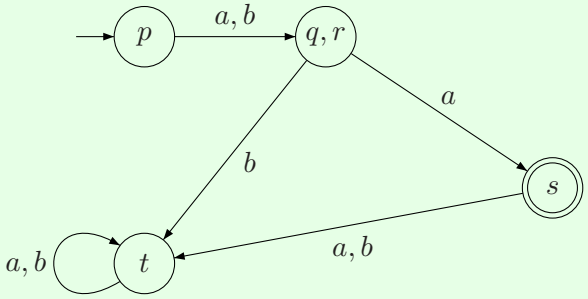
$$P_1 = \{\overbrace{(p, t)}^{A}, \overbrace{(q, r)}^{C}, \overbrace{(s)}^{B}\}$$

Now both states in C on reading a and b go to (B, A) so we don't need to split C. B is a singleton so no split will happen for B. In A, upon reading a, state p transits to C while t transits to A. Therefore we need to separate p from t:

$$P_1 = \{\overbrace{(p)}^{A}, \overbrace{(t)}^{D}, \overbrace{(q, r)}^{C}, \overbrace{(s)}^{B}\}$$

The only non-singleton partite set is C. On reading a and b, all states in C go to (B, D), therefore we don't need to split C and as such we are done.

The minimized DFA looks like this (by merging both r and q into a new "superstate"):



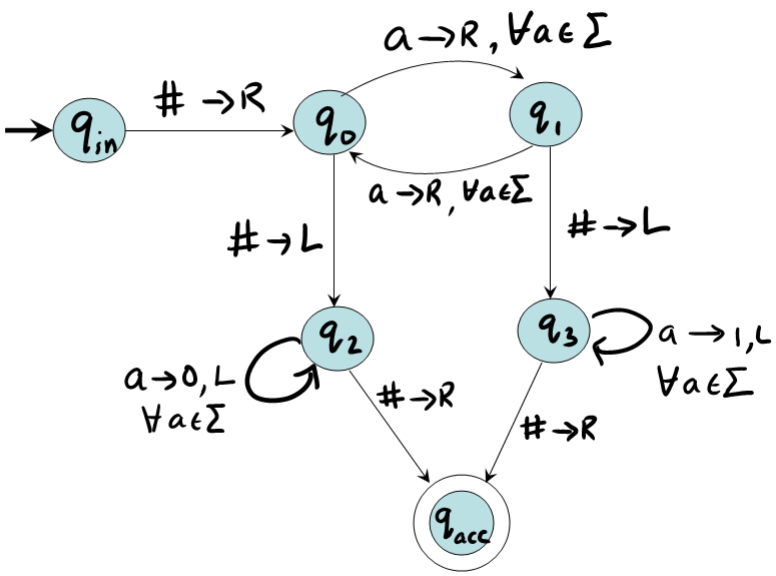
Problem 3: TM design (10 points)

Give the state diagram of a TM M that does the following on input $\#w$ where $w \in \{0,1\}^*$. Let $n = |w|$. If n is even, then M converts $\#w$ to $\#0^n$. If n is odd, then M converts $\#w$ to $\#1^n$. Assume that ϵ is an even length string.

The TM should enter the accept state after the conversion. We don't care where you leave the head at the end of the conversion. The TM should enter the reject state if the input string is not in the right format. However, your state diagram does not need to explicitly show the reject state or the transitions into it.

Solution:

The Turing machine's description is:



In the diagram, the initial state is q_{in}, and the accept state is q_{acc}; the reject state is not shown, and we assume that all transitions from states that are not depicted go to the reject state. We are assuming that the blank tape-symbol is #.

Intuitively, the TM first reads the tape content w, moving right, alternating between states q_0 and q_1 in order to determine whether $|w|$ is even or odd. If $|w|$ is even, it ends in state q_0, moves left rewriting every letter in w with a 0, till it reaches the first symbol on the tape, and then accepts and halts. If $|w|$ is odd, it does the same except that it rewrites letters in w with 1's.

Problem 4: UTM (10 points)

Show L is TM-recognizable:

$$L = \{\langle M \rangle \mid M \text{ accepts some string } w \text{ in at most } |w| \text{ steps.}\}$$

Solution:

Here is code of a recognizer for L:

Algorithm *isInL*($\langle M \rangle$)
1. $\mathcal{N} \leftarrow$ some enumerator of all strings
2. **for** i$\leftarrow$0 **to** ∞
3. **do** $w \leftarrow \mathcal{N}(i)$
4. simulate M on w for at most $|w|$ steps ($*$ simulation is carried out using a call to UTM $*$)
5. **if** the previous simulation accepted
6. **then return true**

We are checking all the strings and since for every string w, we need to run the simulation M on w for at most $|w|$ steps, in finite amount of time each single check is complete. Therefore if $\langle M \rangle \in L$, we will figure this out in finite amount of time (since $\mathcal{N}$ will generate that witness w in finite amount of time). Moreover it is obvious that if we return true, then $\langle M \rangle \in L$, therefore $isinL$ is actually a recognizer for L.

Problem 5: Decidability (10 points)

Show that
$EQINT_{DFA} = \{\langle A, B, C\rangle \mid A, B, C$ are DFAs over the same alphabet Σ, and $L(A) = L(B) \cap L(C)\}$ is decidable.

This question does not require detail at the level of tuple notation. Rather, keep your proof short by exploiting theorems and constructions we've seen in class.

Solution:

We know that for any two DFAs A and B over an alphabet Σ, we can build a DFA D whose language is the intersection of $L(A)$ and $L(B)$. We also know that we can build a TM that can complement an input DFA, and we know that we can build a TM that checks whether the language of a DFA is empty (this TM can, for example, do a depth-first search from the initial state of the DFA, and explore whether any final state is reachable).

Also, note that $L(A) \subseteq L(B)$ iff $L(A) \cap \overline{L(B)} = \emptyset$.

So, we can build a Turing machine that takes as input $\langle A, B, C\rangle$, and first checks whether these are all DFAs (over the same alphabet Σ). Now, if they are, we must first check if $L(A) \subseteq L(B) \cap L(C)$. We can first build a DFA D that accepts $L(B) \cap L(C)$. Now we need to check if $L(A) \subseteq L(D)$, which is the same as checking if $L(A) \cap \overline{L(D)} = \emptyset$. We can do this by first complementing D to get a DFA E, and then building a DFA F that accepts the intersection of the langauges of A and E, and then finally checking if the language of F is empty. We do a similar check to verify whether $L(B) \cap L(C) \subseteq L(A)$.

Hence, the decider for $EQINT_{DFA}$ works as follows.

1. Input is $\langle A, B, C\rangle$.
2. Check if A, B and C are DFAs over the same alphabet Σ. If not, reject.
3. Build the automaton D accepting $L(B) \cap L(C)$.
4. Complement D to get DFA E.
5. Build the DFA F that accepts $L(A) \cap L(E)$.
6. Check if $L(F) = \emptyset$. If it is not, then reject (as $L(A) \not\subseteq L(B) \cap L(C)$).
7. Complement A to obtain the DFA G.
8. Construct DFA H accepting $L(G) \cap L(D)$.
9. Check if $L(H) = \emptyset$. If it is, accept, else reject (as $L(B) \cap L(C) \not\subseteq L(A)$).

Problem 6: Reduction (20 points)

Prove that $L = \{\langle M, w\rangle \mid M \text{ accepts } w \text{ in more than 3 steps}\}$ is undecidable.
(You may use the fact that $A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$ is undecidable.)

Solution:

Let's assume that L is decidable (assume D_L is a decider for L) and we will build a decider for A_{TM} as follows:

Algorithm $D_{A_{TM}}(x)$
1. **if** x is not of the form $\langle M, w\rangle$ for some TM M
2. **then return false**
3. **else** $\langle M, w\rangle \leftarrow x$
4. $\langle N\rangle \leftarrow$Write down the code of TM N ($*$ details of N follows $*$)
5. Simulate D_L on $\langle N, w\rangle$ till it halts
6. Accept if the previous simulation accepts, reject otherwise.

where the code of N is:

Algorithm $N(z)$
1. Move the head 2 cells to the right and then 2 cells to the left.
2. $M(z)$

Checking that $D_{A_{TM}}$ decides A_{TM}: Obviously $L(M) = L(N)$ and if N accepts something, it will accept it after at least 4 dummy moves. Therefore line marked "5" above accepts iff $\langle M, w\rangle \in A_{TM}$. $D_{A_{TM}}$ accepts iff we pass line "2" (which means x is of format $\langle M, w\rangle$) and line "5" accepts which precisely happens when $x = \langle M, w\rangle \in A_{TM}$. So $L(D_{A_{TM}}) = A_{TM}$. Moreover all lines of $D_{A_{TM}}$ will finish in finite time (including line "5" which simulates a decider D_L and is guaranteed to finish in finite time). Therefore $D_{A_{TM}}$ decides A_{TM}.

Finally since we know A_{TM} has no decider, we conclude that L can't have any decider.

Problem 7: Non-regularity (20 points)

(a) Prove that the following language L is not regular ($\Sigma = \{0, 1\}$):

$$L = \{0^n 10^m \mid n < m\}$$

Your proof should use MNT (or the pumping lemma).

Solution:

Consider the infinite set of strings $\{0^i : i \geq 0\}$. Any two strings of this set are distinguishable: Pick $x = 0^i$ and $y = 0^j$ (w.l.g. assume $i < j$). Note that $z = 10^{i+1}$ is a witness string (since $i < i+1$ but $i+1 \leq j$), that is $xz \in L$ but $yz \notin L$. So L has an infinite number of suffix languages and therefore is infinite.

(b) Assume $A = \{0^n 10^n \mid n \geq 0\}$ is non-regular. Use this together with closure properties to give another proof for non-regularity of L.

Solution:

Assume L is regular. Since $\{0\}$ is regular then $L_1 = \{0\}L$ is also regular. Then L_1^R is also regular. And $L_1 \cap L_1^R$ is also regular. But note that $L_1 \cap L_1^R = A$ which we know is not regular. Therefore our assumption about regularity of L is false.