Midterm II
15-453: Formal Languages, Automata, and Computability

Lenore Blum, Asa Frank, Aashish Jindia, and Andrew Smith

April 8, 2014

Instructions:

1. Once the exam begins, **write your name on each sheet.**
2. This is a closed-book examination.
3. Do all your work on the attached sheets. Justify your answers.
4. There are **10** questions for a total of **150** points. 100 points will guarantee an A.
5. You have 80 minutes to answer the questions.
6. Read over the whole exam. Pace yourself. Check your work. Good luck!

Last name: ____________________ First name: ____________________

Signature: ____________________ Andrew ID: ____________________

Question	**Points**	**Score**
1-5	50	
1a	10	
2a	10	
3a	10	
4a	10	
5a	10	
6	10	
7	10	
8	10	
9	10	
10	10	
Total	150	

1. Recall the following definitions:

   - $A_{TM} = \{\langle M, w\rangle \mid M \text{ is a Turing machine and } w \in L(M)\}$
   - $E_{TM} = \{\langle M\rangle \mid M \text{ is a Turing machine and } L(M) = \emptyset\}$
   - $EQ_{TM} = \{\langle M, N\rangle \mid M \text{ and } N \text{ are Turing machines and } L(M) = L(N)\}$
   - $EQ_{DFA} = \{\langle M, N\rangle \mid M \text{ and } N \text{ are DFAs and } L(M) = L(N)\}$
   - $EQ_{PDA} = \{\langle M, N\rangle \mid M \text{ and } N \text{ are PDAs and } L(M) = L(N)\}$

   Additionally, define NoLeft to be the language of pairs $\langle M, w\rangle$ where M is a Turing machine, and when M is run on input w it never moves the tape head left during computation.

   Finally, recall that A_{TM} is Turing-recognizable but not Turing co-recognizable.

   Fill in the following table. Justify your answers for part a. (5 points for each correct cell; -5 points for each incorrect cell; 10 points for each justification)

	Language	**Turing Recognizable?**	**Turing co-Recognizable?**
0	A_{TM}	a. Yes	b. No
1	E_{TM}	a. No	a. Yes
2	EQ_{TM}	a. No	b. No
3	EQ_{DFA}	a. Yes	b. Yes
4	EQ_{PDA}	a. No	b. Yes
5	NoLeft	a. Yes	b. Yes

1a. (10 points) Justify.

Solution: We will show that $\neg A_{TM} \leq_m E_{TM}$.
Given an input $\langle M, w\rangle$ for $\neg A_{TM}$, we create a new machine $T_{\langle M,w\rangle}$ that does the following:
$T_{\langle M,w\rangle}$ ="On input x,
1. Return $M(w)$"
Now, if $w \notin L(M)$ then $T_{\langle M,w\rangle}$ will not accept any strings and $L(T_{\langle M,w\rangle}) = \emptyset$.
Hence, $T_{\langle M,w\rangle} \in E_{TM}$. If $w \in L(M)$ then $T_{\langle M,w\rangle}$ will accept all strings and $T_{\langle M,w\rangle} \notin E_{TM}$
We have successfully shown that $\neg A_{TM} \leq_m E_{TM}$. If E_{TM} was Turing-recognizable, then would $\neg A_{TM}$ be too. But we know that this is not the case. Hence, E_{TM} is not Turing recognizable.

2a. (10 points) Justify.
Solution: We will show that $E_{TM} \leq_m EQ_{TM}$.
Let $M_\emptyset$ be the Turing machine that rejects all inputs. So, $L(M_\emptyset) = \emptyset$.
Given an input $\langle M\rangle$ for E_{TM}, we will construct the input $\langle M, M_\emptyset\rangle$ for EQ_{TM}.
Now, if $M \in E_{TM}$, we have that $L(M) = \emptyset = L(M_\emptyset)$. Therefore $\langle M, M_\emptyset\rangle \in EQ_{TM}$. If $M \notin E_{TM}$ then $L(M) \neq L(M_\emptyset)$ and $\langle M, M_\emptyset\rangle \notin EQ_{TM}$.
Hence, $E_{TM} \leq_m EQ_{TM}$. So if EQ_{TM} was Turing recognizable, then E_{TM} would be too. But we have shown that it is not. So, EQ_{TM} is not Turing Recognizable.

3a. (10 points) Justify.
Solution: We can design a decider for EQ_{DFA} as follows:
$M =$"On input $\langle M_1, M_2 \rangle$:
1. Minimize M_1 and M_2 using the DFA minimization algorithm.
2. Compare states and transitions of minimized DFAs and check for equivalence.
3. Accept if the minimized DFAs are equivalent and reject otherwise."
This machine decides EQ_{DFA}. So, EQ_{DFA} is decidable and hence Turing-Recognizable.

4a. (10 points) Justify.
Solution: We will show that $\neg A_{TM} \leq_m EQ_{PDA}$.
Let $P_\emptyset$ be the PDA that rejects all inputs.
Given an input $\langle M, w \rangle$ for $\neg A_{TM}$, we construct a PDA $P_{M,w}$ which accepts only those strings that are accepting configuration histories of M run on input w. We know from class that we can create such a PDA. We then construct input $\langle P_{M,w}, P_\emptyset \rangle$ for EQ_{PDA}.
Now, if $w \notin L(M)$ then there are no accepting configuration histories of M run on w and $L(P_{M,w}) = \emptyset = L(P_\emptyset)$. So, $\langle P_{M,w}, P_\emptyset \rangle \in EQ_{PDA}$.
If $w \in L(M)$ then there is at least one string accepted by $P_{M,w}$ and $L(P_{M.w}) \neq L(P_\emptyset)$. So, $\langle P_{M,w}, P_\emptyset \rangle \notin EQ_{PDA}$.
Hence, $\neg A_{TM} \leq_m EQ_{PDA}$. Therefore since $\neg A_{TM}$ is not turing recognizable, we have that EQ_{PDA} cannot be Turing recognizable either.

5a. (10 points) Justify.
Solution: We have the following algorithm to decide NoLeft:
Given $\langle M, w \rangle$, run TM M on w for $|w| + n + 1$ steps, where $n =$number of states of M. If during the run, TM ever goes left then reject. If not then accept.

After $|w|$ steps, if the turing machine head never moves left, then we would have reached the end of input and beginning of blank cells. At this point, if we will only be following blank transitions. So, if after $n + 1$ steps, we never move left, then by pigeonhole principle we would have looped back to some state following only blank transitions without moving left and hence will never move left.

6 (10 points) Let DoubleSat be the language of CNF formulas with at least two satisfying assignments. Show that DoubleSat is NP-complete.

Solution:
DoubleSat $\in$ NP: A polynomial verifier for DoubleSat would take as input the CNF and two satisfying assignments, both of polynomial length. To verify it, the verifier would just test the assignments against the CNF and check their validity and that they are different. All of this is possible in polynomial time. Hence, DoubleSat $\in NP$.

DoubleSat is NP-Hard: We will prove this by reducing 3-SAT to DoubleSat.
Given an input 3-CNF ϕ for a 3-SAT problem, we create input ϕ' for DoubleSat as:
$\phi' = \phi \wedge (x \vee x \vee \neg x)$, where x is a new variable that does not appear in ϕ.
Now, if $\phi \in 3 - SAT$, we have that there is at least one satisfying assignment for ϕ. For this assignment, we then have at least 2 satisfying assignments for ϕ', one with $x = TRUE$ and one with $x = FALSE$. Hence, $\phi' \in$ DoubleSat
If $\phi' \in$ DoubleSat, then we have at least two satisfying assignment for ϕ'. Since $\phi' = \phi \wedge (x \vee x \vee \neg x)$, we have that both ϕ and $(x \vee x \vee \neg x)$ are satisfied by these assignments. Also since x appears only in the final clause, removing it from the satisfying assignments of ϕ' will yield at least one satisfying assignment for ϕ. So, $\phi \in$ 3-SAT
Hence, we have successfully reduced 3-SAT to DoubleSat. Moreover this reduction was in polynomial time. So, DoubleSat is NP-Hard.
We have hence shown that DoubleSat is NP-Complete.

7 (10 points) If $A \leq_m B$ and B is regular, must A be regular? Explain your answer.

Solution: No, A may not be regular.
Consider the sets $A = \{0^n 1^n | \forall n \in \mathbb{N}\}$ and $B = \{1\}$
Consider a function f defined as:

$$f(s) = \begin{cases} 1 & \text{if s is of the form } 0^n 1^n \\ 0 & \text{otherwise} \end{cases}$$

This function is computable because a Turing machine can easily check if the input is of the given form and print out the corresponding value accordingly.
Now, if $s \in A$, we have $f(s) = 1 \in B$ and if $s \notin A$, $f(s) = 0 \notin B$.
Therefore, we have that $A \leq_m B$, B is regular and A is not regular.

8 (10 points) Define

$$\textsc{Halts} = \{\langle M, w\rangle \mid M \text{ is a Turing machine which halts on input } w.\}$$
$$\textsc{Total} = \{\langle M\rangle \mid M \text{ is a Turing machine which halts on all input.}\}$$

If we have an oracle for Halts, can we use this to decide Total?

Solution: We from from class that Halt $\in \Sigma_1^0$ and Total $\in \Pi_2^0$. We also have from class that Total is in fact m-complete in Π_2^0. Hence we cannot have that Total $\in \Delta_2^0$ as $\Delta_2^0 \neq \Pi_2^0$.
Therefore, Total cannot be decided even with an oracle for Halt because it is not in Δ_2^0.

9 (10 points) Recall that a Turing machine is minimal if there is no equivalent Turing machine with a shorter description, and that $\textsc{Min}_{TM}$ is the language of minimal Turing machines. Show that any infinite subset of $\textsc{Min}_{TM}$ is not Turing-recognizable.

Solution: Assume for sake of contradiction that there exists some infinite subset of $\textsc{Min}_{TM}$ that is Turing-Recognizable. Let this subset be S.
Since S is Turing-recognizable, we have that there exists some enumerator E that enumerates S. Consider the following turing machine M:
$M =$ "On input x:
1. Obtain own source code $\langle M\rangle$.
2. Use E to enumerate elements of S until you find a TM T output by E such that $|\langle M\rangle| < |\langle T\rangle|$.
3. Output T(x)."

We have that step 1 is possible by the recursion theorem. In step 2 will will eventually find such a TM T because S is infinite and hence its elements' lengths are not bounded. Hence, we have that the above Turing machine M effectively simulates T on all it's inputs. i.e., M accepts, rejects and loops on exactly those inputs that are accepted, rejected and looped on respectively by T. Therefore M is equivalent to T. But we have that $|\langle M\rangle| < |\langle T\rangle|$. This is a contradiction as T is supposed to be minimal. Hence, $\textsc{Min}_{TM}$ has no infinite subset that is Turing recognizable.

10 (10 points) Define HALFCLIQUE to be the language

$$\{\langle G\rangle \mid G \text{ is an undirected graph containing a clique with at least half its vertices}\}.$$

Show that HALFCLIQUE is NP-complete.

Solution:
HalfClique $\in$ NP: A possible verifier for HALFCLIQUE could take as input the Graph and set of vertices that form a half-clique. It would then just check if the set has size at least half the number of vertices and if the set of vertices forms a clique. Each of these is possible in polynomial time. Hence HALFCLIQUE $\in NP$.

HalfClique is NP-Hard: We will show this by reducing CLIQUE to HALFCLIQUE. Given an input $\langle G, k\rangle$ for CLIQUE we create an input for HALFCLIQUE based on the following cases:

1. If $k = \frac{|G|}{2}$, our new input is $\langle G\rangle$.

In this case if $\langle G\rangle \in$ HALFCLIQUE, then we have a clique of size at least $|G|/2 = k$. So, a clique of size k exists in G and $\langle G, k\rangle \in$ CLIQUE. If $\langle G\rangle \notin$ HALFCLIQUE then the largest clique in G is of size $< |G|/2 = k$. So $\langle G, k\rangle \notin$ CLIQUE

2. If $k < \frac{|G|}{2}$, then construct graph G' by adding adding a complete graph of $|G| - 2k$ vertices to G and connecting each of these new vertices to every vertex in G. Our new input is now $\langle G'\rangle$

In this case, if $\langle G'\rangle \in$ HALFCLIQUE, then there exists a clique of size $\geq |G| - k$. We know that at most$|G|-2k$ of these vertices came from the complete graph we added, we know that the original graph G must have had a clique of size $\geq |G|-k-(|G|-2k) = k$. So, $\langle G, k\rangle \in$ CLIQUE. If $\langle G'\rangle \notin$ HALFCLIQUE then the largest clique in G' has size $< |G| - k$. Since the added complete graph would have increased the size of largest clique in G by $|G| - 2k$, we have that the largest clique in G is of size $< k$. Hence, $\langle G, k\rangle \notin$ CLIQUE

3. If $k > \frac{|G|}{2}$, then construct graph G' by adding $2k - |G|$ new isolated vertices to G. Our new input is now $\langle G'\rangle$.

In this case, if $\langle G'\rangle \in$ HALFCLIQUE, then there exists a clique of size $\geq k$. We know that none of the new vertices could have contributed to this clique as they were all isolated. So this clique exists wholly within the original graph G. So, $\langle G, k\rangle \in$ CLIQUE. If $\langle G'\rangle \notin$ HALFCLIQUE, then the largest clique in G' has size $< k$. This would hence also be the largest clique in G and therefore, $\langle G, k\rangle \notin$ CLIQUE.
We have hence successfully reduced CLIQUE to HALFCLIQUE. So, HALFCLIQUE is NP-Hard.

So, HALFCLIQUE is NP-Complete.