6.045 Midterm Solutions

Your Name: ____________________________

April 5, 2012

Some Tips:

• **Don’t be a novelist.** Every part of the free-answer questions (except possibly the extra credits) can be answered in a few sentences or less. *If you’re writing paragraph after paragraph, you’re wasting your time!* Your goal is just to show us that you “get it,” not to impress us with your writing skills or attention to detail.

• **Budget your time.** Don’t spend more than about 20 minutes on one problem unless you’ve already finished the others.

• **Try all the problems.** Don’t give up on anything as “obviously too hard,” without at least thinking about it first.

• **Go over your responses in the true/false/open section.** You’d be amazed at the number of careless mistakes people make.

*Good luck, and may the power of Turing be with you!*

<table>
<thead>
<tr>
<th>Problem</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1 (T/F/O)</td>
<td>/ 30</td>
</tr>
<tr>
<td>Problem 2</td>
<td>/ 30</td>
</tr>
<tr>
<td>Problem 3</td>
<td>/ 20</td>
</tr>
<tr>
<td>Problem 4</td>
<td>/ 20</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>/ 100</td>
</tr>
</tbody>
</table>
Problem 1 [30 points, 1.5 points per statement, rounded up to the nearest integer]: (Proven) True, (Proven) False, or Open—Circle One

(a) T F [O] : P = NP.
(b) T F [O] : PSPACE = EXPSPACE.
(c) T F [O] : PSPACE \ P has the same cardinality as the integers (where A \ B means the set of elements in A but not in B).
(d) T F [O] : \{L \subseteq \{0,1\}^* : L is context-free but not regular\} has the same cardinality as the integers.
(e) T F [O] : n^{log n} \in \Omega (n^{2 log n}).
(f) T F [O] : n^{log n} \in n^{\Theta (2 log n)}.
(g) T F [O] : The gate set \{XOR, AND\} is universal (assuming constants are available free).
(h) T F [O] : If for all k, there exists an oracle A such that M^A() runs for \geq k steps, then there exists an oracle A such that M^A() runs forever. (Where M is an oracle Turing machine.)
(i) T F [O] : If for all k, there exists an input x such that M(x) runs for \geq k steps, then there exists an input x such that M(x) runs forever.
(j) T F [O] : L = \{\langle M \rangle : M is a DFA that accepts at least one input\} is decidable.
(k) T F [O] : L = \{x : x ends with 0 and P = NP, or x ends with 1 and P \neq NP\} is decidable.
(l) T F [O] : L = \{0^n110^n : n \geq 1\} is regular.
(m) T F [O] : L = \{0^n110^n : n \geq 1\} is context-free.
(n) T F [O] : L = \{10^n1 : n \geq 1\} is regular.
(o) T F [O] : Every language decidable by a nondeterministic pushdown automaton is also decidable by a deterministic pushdown automaton.
(p) T F [O] : The regular expressions ((0|1)(0|1))^* and (00|01|10|11)^* generate the same language.
(q) T F [O] : The regular expressions 0^*(10^*1)^*0^* and 0^*(10^*10^*)^*0^* generate the same language.
(r) T F [O] : The regular expression 0^*(10^*10^*)^*0^* generates the same language as the context-free grammar (Init \rightarrow S, S \rightarrow 1S1, S \rightarrow 0S, S \rightarrow S0, S \rightarrow ).
(s) T F [O] : L = \{0^n1^2n0^n : n \geq 1\} is context-free.
(t) T F [O] : For every computable function f, there exists a constant c_f such that K(f(x)) \leq K(x) + c_f for all x, where K is Kolmogorov complexity.
Problem 2 [30 points]: DFAs and NDFAs with lots of states
Given a positive integer \( n \), let an “\( n \)-language” be a subset \( L \) of \( \{0,1\}^n \). (For example, the set of all 100-bit strings that are palindromes would be a 100-language.)

(a) [5 points] How many \( n \)-languages are there?

(b) [5 points] Show that every \( n \)-language is decidable by a DFA with \( O(2^n) \) states.

(c) [5 points] Given a positive integer \( k \), give an upper bound on the number of different DFAs with \( k \) states over the alphabet \( \{0,1\} \). (You can assume, for simplicity, that there is exactly one “Accept” state.)

(d) [5 points] Using (a) and (c), show that there exists an \( n \)-language that is not decidable by any DFA with \( o(2^n/n) \) states.

(e) [5 points] Give an upper bound on the number of different NDFAs with \( k \) states (again, assuming the alphabet \( \{0,1\} \) and a single “Accept” state).

(f) [5 points] Show that there exists an \( n \)-language that is not decidable by any N DFA with \( o(2^{n/2}) \) states.

(g) [Extra credit, 10 points] Show that every \( n \)-language is decidable by an N DFA with \( O(2^n/2) \) states.

Solutions:

(a) There are \( 2^n \) strings of length \( n \), each of which may either be or not be in the language. Thus, there are a total of \( 2^{2^n} \) \( n \)-languages.

(b) We can make a binary tree which goes through all the possible characters at each place, so that the leaves represent all the \( 2^n \) length-\( n \) strings. Those that are in \( L \) are accept states and those that aren’t, reject states.

(c) For each state, there are \( k \) choices of states to go to when reading a 0 and \( k \) choices of states to go to when reading a 1. Thus, we have \( (k^2)^k = k^{2k} \) as an upper bound on the number of DFAs. (Note that this is not an exact count, depending on whether you care about equivalence across permutations of start state, accept states, etc., but for the purposes of an upper bound, this is good enough.)

(d) There are at least as many DFAs as regular languages, so \( k^{2k} \geq 2^{2^n} \). Taking the log of both sides gets us that \( 2k \log k \geq 2^n \). Dividing, we obtain \( k = \Omega \left( \frac{2^n}{n} \right) \).

(e) There are \( 2^k \) of each of 0-transitions, 1-transitions, and \( \varepsilon \)-transitions, each of which can go between any of \( k \) states. This leads to a total of \( (8^k)^k = 8^{k^2} \) total NDFAs.

(f) There are at least as many NDFAs as regular languages, so \( 8^{k^2} \geq 2^{2^n} \). Taking the log of both sides gets us that \( 2k^2 \geq 2^n \). Dividing, we obtain \( k = \Omega \left( \frac{2^n}{2} \right) \).

(g) We can make two binary trees, one for each \( \frac{n}{2} \)-bit string. The leaves of the two binary trees “face” each other with every pair of them having an \( \varepsilon \)-transition. The root of one binary tree is the accept state and the root of the other is an accept state. For the tree with the start state, the arrows flow from the root to the leaves and then, following the \( \varepsilon \)-transitions, to the leaves of the other tree and towards the root of that tree. If a language is in the string, the NDFA will follow the appropriate \( \varepsilon \) transition to get the right suffix.
Problem 3 [20 points]: Turing machines, and how many squares they visit

Let $FINSQUARE = \{\langle M \rangle : \text{the total number of tape squares } M() \text{ ever visits is finite}\}$. Also, let $HALT = \{\langle M \rangle : M() \text{ halts}\}$. Here, as usual, $\langle M \rangle$ is an encoding of a Turing machine.

(a) [5 points] Is $FINSQUARE \subseteq HALT$? (If yes, explain why; if no, describe an element of $FINSQUARE \setminus HALT$.)

(b) [5 points] Is $HALT \subseteq FINSQUARE$? (If yes, explain why; if no, describe an element of $HALT \setminus FINSQUARE$.)

c) [5 points] Is $FINSQUARE \leq_T HALT$? (If yes, describe a reduction; if no, explain why not.)

d) [5 points] Is $HALT \leq_T FINSQUARE$? (If yes, describe a reduction; if no, explain why not.)

e) [Extra credit, 5 points] Let $FINLEFT = \{\langle M \rangle : \text{the maximum distance to the left of its initial location that } M()'s \text{ tape head ever moves is finite}\}$. Show that $FINLEFT \leq_T SUPERHALT$. (For this part and the next part, you can use the fact that $SUPERHALT$ is Turing-equivalent to the language $FINHALT = \{\langle M \rangle : M(x) \text{ halts for at most finitely many inputs } x\}$.)

(f) [Extra credit, 10 points] Show that $SUPERHALT \leq_T FINLEFT$.

Solutions:

(a) No. Consider a Turing machine that switches between two states, in one moving the tapehead right and in the other moving it left without writing anything to the tape. The machine will never halt, but will only ever visit two different squares.

(b) Yes. If a Turing machine halts, then it runs for $k$ steps. The largest number of cells a Turing machine can visit in $k$ steps is $k$.

c) Yes. Given $M$, we can decide if $\langle M \rangle \in FINSQUARE$ by creating a new machine, $M'$, that simulates $M$ but keeps track of each configuration $M$ has ever been in, and halts if it ever enters the same one (which means it’s in a loop). Then we can ask the oracle if $\langle M' \rangle \in HALT$; if (and only if) it is, then $\langle M \rangle \in FINSQUARE$.

d) Yes. Given $M$, we can decide whether $\langle M \rangle \in HALT$ by creating a new machine, $M'$, that works as before — it simulates $M$ but keeps track of each configuration $M$ has ever been in. If it ever enters the same configuration, then we know it will never halt. At this point, $M'$ stops simulating $M$ and instead just moves its tape head to the left forever. This ensures that whenever $M$ loops, it also visits an infinite number of squares, so we can ask the oracle if $\langle M' \rangle \in FINSQUARE$; if (and only if) it is, then $\langle M \rangle \in HALT$.

e) Given $M$, we can decide whether $\langle M \rangle \in FINLEFT$ by creating a new machine, $M'(k)$ that, on input $k \in \mathbb{N}$, simulates $M()$ until it moves $k$ squares to the left of its initial position, and halts when $M$ does so. (If $M()$ never moves $k$ squares to the left of its initial position, then $M'(k)$ just loops.) Then, $\langle M \rangle \in FINLEFT$ if and only if there are infinitely many $k$ for which $M'(k)$ halts. That question can be answered by an oracle for $FINHALT$.

(f) Given $M$, we can decide whether $\langle M \rangle \in FINHALT$ by creating a new machine $M'$ that, on an empty input, dovetails over $M(x_1), M(x_2), M(x_3), \ldots$ (where $x_1, x_2, x_3, \ldots$ is a lexicographical ordering of inputs), only using the part of the tape that is to the right of its start location. If and when at least $k$ of the $M(x_i)$ runs have halted, $M'$ suspends the dovetailing, moves $k$ squares to the left of its start position, and then goes back and resumes the dovetailing. Then $M'(i)$ will go infinitely far to the left if and only if $M(x_i)$ halts for infinitely many $i$’s (i.e. if $\langle M \rangle \in FINHALT$).
Problem 4 [20 points]: Proofs of $1 + 1 = 3$

For this problem, you can assume that the formal system $F$ is sound, that the axioms and deduction rules of $F$ can all be given explicitly, and that $F$ is capable of expressing any ordinary mathematical reasoning—for example, about Turing machines and proofs. In other words, $F$ is simply one of the usual systems to which Gödel’s Incompleteness Theorem applies, such as the standard axiom systems for arithmetic or for set theory.

Let $L = \{x \mid x$ encodes a proof, in $F$, that $1 + 1 = 3\}$.

For (a)-(d), give brief explanations for your answer.

(a) [5 points] Is $L$ the empty language?
(b) [5 points] Is your answer to (a) provable in $F$?
(c) [5 points] Is $L$ decidable?
(d) [5 points] Is your answer to (c) provable in $F$?

(e) [Extra credit, 10 points] Describe a Turing machine $M$, such that the statement

“$M$ runs in $O(2^n)$ time, where $n$ is the length of $M$’s input”

is independent of $F$ (i.e., neither provable nor disprovable in $F$).

(f) [Extra credit, 5 points] Describe a language $L \subseteq \{0, 1\}^*$, such that the statement “$L \in TIME(2^n)$” is independent of $F$.

Solutions:

(a) Yes. If it weren’t empty then $F$ would not be sound, which contradicts our assumption.

(b) No. If we could prove that $L$ is empty in $F$, we would have proved that $F$ is consistent in $F$, which is a violation of Gödel’s Second Incompleteness Theorem.

(c) Yes. $L$ is empty, and the empty language (indeed, any finite language), is clearly decidable.

(d) Yes. An algorithm to decide $L$ (whether or not it was empty) is just an algorithm that checks whether the axioms and inference rules of $F$ were being correctly applied to prove that $1 + 1 = 3$; the validity of such an algorithm can be proven in $F$.

(e) On input $x \in \{0, 1\}^n$, $M$ first checks whether $x$ encodes an $F$-proof that $1 + 1 = 3$ (this can clearly be done in $O(2^n)$ time for any standard axiom system $F$). If $x$ does not encode such a proof, then $M$ halts; otherwise, $M(x)$ kills time for an additional $3^n$ steps. If $F$ is consistent, then $M$ runs in $O(2^n)$ time, whereas if $F$ is inconsistent, it runs for $\Omega(3^n)$ time (since the proof that $1 + 1 = 3$ can be padded out to arbitrary lengths). Since $F$’s consistency is independent of $F$ (since we’re assuming that $F$ is sound), it follows that whether $M$ runs in $O(2^n)$ time is independent of $F$ as well.

(f) Just take $L = \{(x, \langle M \rangle, 0^n) \mid x$ encodes an $F$-proof that $1 + 1 = 3 \text{ and } M()$ halts in $\leq 3^n \text{ steps.}\}$. If $F$ is consistent, then $L$ is the empty language (here in $TIME(2^n)$). If $F$ is inconsistent, then $L$ is not in $TIME(2^n)$ by the Time hierarchy theorem.