6.045 Midterm

Your Name: Scott Aaronson

March 29, 2016

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>Points</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1 (T/F/O)</td>
<td>20 / 20</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>30 / 30</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>25 / 25</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>30 / 25</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>105 / 100</strong></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 [20 points, 1 point per statement]: (Proven) True, (Proven) False, or Open—Circle One

(a) T F O The value of \( BB(42) \) is independent of ZF set theory.

(b) T F O \( 2\sqrt{n} = \Omega(n\log n) \).

(c) T F O \( 3^n = o(2.99^n) \).

(d) T F O \( P = \text{PSPACE} \) implies \( \text{EXP} = \text{EXPSPACE} \).

(e) T F O \( \text{EXP} = \text{EXPSPACE} \) implies \( P = \text{PSPACE} \).

(f) T F O \( \{\text{MAJ}_3, \text{NOT}\} \) is a universal set of Boolean gates (where \( \text{MAJ}_3 \) denotes 3-bit majority).

(g) T F O There is a language \( L \) such that \( L \in \text{TIME}(f(n)) \) implies \( L \in \text{TIME}(\log f(n)) \) for all \( f \).

(h) T F O The set of non-context-free languages has cardinality \( \aleph_0 \).

(i) T F O \( \text{EXPSPACE} \) has cardinality \( \aleph_0 \).

(j) T F O The set of languages that are in \( \text{EXP} \) but not \( \text{PSPACE} \) has cardinality \( \aleph_0 \).

(k) T F O If \( L \) is recognized by an N DFA, then so is \( \overline{L} \).

(l) T F O If \( L \) is recognized by an NPDA, then so is \( \overline{L} \).

(m) T F O If \( L \) is recognized by a DPDA, then so is \( \overline{L} \).

(n) T F O If \( L \) is any uncomputable language, then \( \text{HALT} \leq_T L \).

(o) T F O \( L = \{ \langle M \rangle : M(\langle M \rangle) \text{ halts} \} \) is Turing-equivalent to \( \text{HALT} \).

(p) T F O \( L = \{ \langle M \rangle : M(x) \text{ accepts at least 100 inputs } x \} \) is Turing-equivalent to \( \text{HALT} \).

(q) T F O For all large enough \( n \), there's a Boolean function \( f : \{0,1\}^n \to \{0,1\} \) with no circuit of \( \sqrt{\sqrt{n}} \) \( \text{NAND} \) gates.

(r) T F O: Every circuit containing \( \text{NOT} \) gates computes a non-monotone Boolean function.

(s) T F O: The regular expression \( (0|1)^* 1^* \) generates all binary strings.

(t) T F O: Every regular expression that generates infinitely many strings contains a * symbol.
Problem 2 [30 points]: Regular and Context-Free Languages
Consider the language $L = \{x^ay^bz^c : c = a + b\}$.

(a) [5 points] Prove that $L$ is not regular.
(b) [5 points] Prove that $L$ is context-free, by giving a CFG for it.
(c) [5 points] Does your CFG have a unique derivation for every string $w \in L$? If not, is there a different CFG that does?
(d) [5 points] Give a deterministic or nondeterministic pushdown automaton for $L$.
(e) [5 points] Consider the modified language $L' = \{x^ay^bz^c : c \equiv a + b \pmod{2}\}$. Give a DFA or NDFA for $L'$.
(f) [5 points] Give a regular expression for $L'$.

For parts (b), (c), (d), (e), and (f), you don't need to prove your answer.

(a) Even if we consider inputs of the form $x^a y^c$ only (with $b = 0$), we know from class that no DFA can decide if $a = c$. This can be proved using either the Pumping Lemma or the Myhill Lemma.

(b) $S \rightarrow xSZ$.
    $S \rightarrow T$
    $T \rightarrow yTZ$
    $T \rightarrow \lambda$.

(c) Yes, it has unique derivation $S$.

(d) $\text{push } \#$ $\rightarrow x$ $\text{push } x$ $\rightarrow \varepsilon$ $\text{pop } \#$ $\rightarrow \varepsilon$.

(e) 

(f) $(xx)^*(yy)^*(zz)^*$ $\mid (xx)^*x(yy)^*y(zz)^*$ $\mid (xx)^*x(yy)^*x(zz)^*$ $\mid (xx)^*(yy)^*y(zz)^*$ $\mid (xx)^*(yy)^*x(zz)^*$.
(Blank page for continuing your answers to Problem 2)
(d) The conversion from $M \leftrightarrow M'$ constitutes a reduction from $\text{HALT}$ to $L_F$. I.e., if $(M) \in \text{HALT}$ then $(M') \in L_F$ and if $(M) \in \text{HALT}$ then $(M') \notin L_F$. Here $L_F$ is uncomputable.

Problem 3 [25 points]: Provable Loopers

Let $F$ be a formal system powerful enough to reason about arithmetic and Turing machines, such as ZF set theory. You can assume that $F$ is sound. Now consider the language that consists of all encodings of Turing machines that not only run forever, but provably run forever:

$$L_F = \{(M) : F \text{ proves that } M(\ ) \text{ runs forever}\}.$$

(Here, as usual, $M(\ )$ means $M$ run on a blank input.)

(a) [5 points] Prove that $L_F \leq_T \text{HALT}$.

(b) [5 points] Give an example of an input on which $L_F$ differs from $\text{HALT}$ (where, for concreteness, $\text{HALT} = \{(M) : M(\ ) \text{ halts}\}$). In other words: verbally describe a Turing machine $Z$ that runs forever on a blank input, but that $F$ can’t prove runs forever—and for which we therefore have $(Z) \in \text{HALT}$ but $(Z) \notin L_F$.

(c) [10 points] Given any Turing machine $M$, explain how to build a new machine $M'$ with the following interesting properties:

- If $M(\ )$ halts, then $M'(\ )$ runs forever, and moreover $F$ proves that $M'(\ )$ runs forever.
- If $M(\ )$ runs forever, then $M'(\ )$ might or might not run forever—but at any rate, $F$ doesn’t prove that $M'(\ )$ runs forever.

[Hint: You’ll want your machine $Z$ from part (b), and the step-by-step simulation of $M$, as two ingredients in building $M'$.]

(d) [5 points] Assuming part (c), show that $L_F$ is uncomputable, and indeed, is Turing-equivalent to $\text{HALT}$.

(i) We can construct a TM $M'$ that enumerates all the theorems of $F$, halting iff it finds a proof that $M(\ )$ runs forever. Then by deciding whether $M'(\ )$ halts, we also decide whether $F$ proves that $M(\ )$ runs forever.

(ii) Let $Z$ be a machine that enumerates all the theorems of $F$, halting iff it finds a proof of $\theta \equiv 0$, then $Z(\ )$ runs forever (since $F$ is sound), but $F$ can’t prove $Z(\ )$ runs forever since otherwise $F$ would prove its own consistency, violating the $2^{nd}$ incompleteness theorem. Hence $(Z) \in \text{HALT}$ but $(Z) \notin L_F$.

(iii) We alternate between simulating $M(\ )$ and simulating $Z(\ )$.

If $Z(\ )$ ever halts (finds an inconsistency in $F$), then we halt.

If $M(\ )$ ever halts, then we go into a trivial infinite loop.

Result: If $M(\ )$ halts, then $F$ can simulate $M'(\ )$ step-by-step until $M'(\ )$ halts and thereby prove that $M'(\ )$ runs forever.

If $M(\ )$ runs forever, then $M'(\ )$ runs forever iff $F$ is consistent (and $F$ proves this). So $F$ can’t prove $M'(\ )$ runs forever without proving its own consistency.
Problem 4 [25 points]: Resource-Bounded Kolmogorov Complexity

Given a string \( x \in \{0,1\}^* \), recall that \( K(x) \) is the number of bits in the shortest program \( P \) such that \( P(\cdot) = x \); that is, the shortest program that, when given a blank input, halts in finite time and outputs \( x \). (Here and throughout, we fix any reasonable programming language, and consider only programs written in that language.)

Now, given a function \( f \), let \( K_{\text{TIME}}(f(n)) (x) \) be the number of bits in the shortest program \( P \) such that \( P(\cdot) = x \) and \( P(\cdot) \) halts after at most \( f(|x|) \) time steps. Likewise, let \( K_{\text{SPACE}}(f(n)) (x) \) be the number of bits in the shortest program \( P \) such that \( P(\cdot) = x \) and \( P(\cdot) \) having used at most \( f(|x|) \) bits of memory. In what follows, we’ll fix \( f(n) = n^3 \) for the sake of concreteness.

(a) [5 points] Sort the following complexities in order from least to greatest (not necessarily via strict inequalities): \( K_{\text{TIME}}(n^2) (x) \), \( K_{\text{TIME}}(2n^2) (x) \), \( K_{\text{SPACE}}(n^2) (x) \), \( K(n) \). Give a sentence or so of justification for why the stated inequalities hold.

(b) [5 points] We saw in class that \( K(x) \) is uncomputable. By contrast, explain how to compute \( K_{\text{TIME}}(n^2) (x) \) in \( 2^{n^2} O(1) \) time.

(c) [5 points] Likewise, explain how to compute \( K_{\text{SPACE}}(n^2) (x) \) in \( O(n^2) \) space.

(d) [10 points] Show that more running time can yield better data compression: more concretely, there exists a positive integer \( n \), and a string \( x \in \{0,1\}^n \), such that \( K_{\text{TIME}}(n^2) (x) < K_{\text{TIME}}(n^2) (x) \). [Hint: This is equivalent to ruling out the possibility that \( K_{\text{TIME}}(n^2) (x) = K_{\text{TIME}}(n^2) (x) \) for every \( x \). Supposing that equality held, can you use part (b) to derive a contradiction, analogously to how we proved in class that \( K(x) \) is uncomputable? Note that \( 5^n \) is just meant to be a generous upper bound on the running time needed.]

(e) [5 points, extra credit] Show that actually, there exists a positive integer \( n \), and a string \( x \in \{0,1\}^n \), such that \( K_{\text{TIME}}(n^2 + 1) (x) < K_{\text{TIME}}(n^2) (x) \).

\( k \) (c)

\( k \) (b)

\( k \) (a)

\( k \) (d)

\( k \) (e)

\( k \) (f)

\( k \) (g)

\( k \) (h)

\( k \) (i)

\( k \) (j)

\( k \) (k)

\( k \) (l)

\( k \) (m)

\( k \) (n)

\( k \) (o)

\( k \) (p)

\( k \) (q)

\( k \) (r)

\( k \) (s)

\( k \) (t)

\( k \) (u)

\( k \) (v)

\( k \) (w)

\( k \) (x)

\( k \) (y)

\( k \) (z)
Suppose by contradiction that $\text{KTime}(n^2)(x) = \text{KTime}(\eta^3)(x)$ for all $x$. Now consider the following program, call it $Q$, that finds a string $x \in \{0,1\}^n$ s.t. $\text{KTime}(n^2)(x) > 101$ (since $101 < \log(n) + O(1)$ and $Q$ takes $\leq 101$ time), until it finds an $x \in \{0,1\}^n$ s.t. $p(\gamma) \neq x$ for all $p$ of length $\leq n/2$. This $Q$ clearly takes $2^n 2^{n/2} n^{O(1)} = 2^{3n/2} n^{O(1)}$ time and it clearly finds an $x \in \{0,1\}^n$ such that $\text{KTime}(n^2)(x) > \frac{n}{2}$. By assumption, then, $\text{KTime}(n^2)(x) \leq \log(n) + O(1)$ (since $101 < \log(n) + O(1)$ and $Q$ takes $\leq 101$ time), which is less than $\frac{n}{2}$ for all sufficiently large $n$'s! This is a contradiction, so we cannot have had $\text{KTime}(n^2)(x) = \text{KTime}(\eta^3)(x)$ for all $x$.

Suppose w.o.c. that $\text{KTime}(\eta^3)(x) = \text{KTime}(\eta^2)(x)$ for all $x$. Then in part (1), we can simply modify $Q$ so that it searches for an $x \in \{0,1\}^n$ with $\text{KTime}(\eta^3)(x) > 3\log\log n$. Clearly such an $x$ exists, and it takes $\eta^{2+O(1)} 2^{3\log\log n} 2^{3\log\log n} \eta^{2\log n} = \eta^{2+O(1)}$ time to find, much less than $\eta^{2+O(1)}$. But if (for example) $\eta$ is a power of 2, then $\text{KTime}(\eta^3)(x) = \log\log n + O(1)$, which is less than $3\log\log n$ for all sufficiently large $n$. So then this can't be equal to $\text{KTime}(n^2)(x)$. 