YOUR NAME: _______________________________

CSC 280: Computer Models and Limitations Spring 2018
Instructor: Lane A. Hemaspaandra Grad TA: Andrew Read-McFarland

Instructions: Make sure to write your name on the top-right-corner name line of each sheet! We’ll grade only the work you put on these stapled test sheets (you may solve a problem only on the front and back of the sheet it is on, since we’ll separate the sheets during grading). Any blank paper we pass around (you may not bring in scrap paper) is for you to use as scrap paper and will be neither collected nor graded. Do four of the six problems. If you do more, we’ll grade all that you’ve done, and will count the four highest scores. (Since you might make mistakes that you don’t realize, I’d advise you to do more than four.) The order in which problems and multiple-choice answers are listed may differ from person to person. Before starting the test, read all problems and problem parts so that you can attack first whichever things offer you the best path to a strong score. Throughout this exam you may use, without proving it, the fact that $A_{TM}$ is undecidable. Closed book. Closed notes (except for at most one 8.5-by-11 2-sided handwritten self-prepared notes sheets, or two 8.5-by-11 1-sided such sheets). Closed computer, closed slide rule, closed phone, etc. (no electronics may even be in sight). This is an in-class midterm exam being given during the 4/18 class period.

No questions may be asked during the exam. (If you are convinced that something is severely wrong in a way that makes the problem impossible to solve, explain on your test sheet what is severely wrong. But for example minor typos where it is clear enough what was intended do not mean you can skip doing a problem.)

This will be viewed as 100-point exam, but you can get up to 8 extra credit points since each problem is worth 27 points, so it is possible to score a 108/100 on the exam.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Total (= sum of 4 highest scores)
Question 1. [27 points] Prove that $L_1 = \{\langle M', x, M'' \rangle \mid M' \text{ accepts } x \text{ and } M'' \text{ does not accept } x \}$ is undecidable. You must solve this using our template/contradiction approach, and you are required to key your answer on $A_{TM}$, that is, you’ll start by assuming $L_1$ is decidable and then will use a hypothetical total TM (decider) for $L_1$ to help you build a decider for $A_{TM}$. Of course, your answer should, following that construction, include the appropriate arguments, as per the template method.

Answer:

Note: On this and the next two problems, we don’t explicitly include in our template proof (not in the construction of $S$ and not in the argument of correctness) the case where the input to $S$ is syntactically flawed. (What we’d do is immediately reject on this case. And that would also come in in one of the usually two subcases within the “what happens if $\langle M, x \rangle$ is not in $A_{TM}$” part of the argument that comes after the construction. However, in our (“undecidability”) template-method proofs, to be able to focus on the really interesting things, we’ve said it is fine to not explicitly mention/discuss that case.)

Note: And on this and the next two problems, in the part that is in the funny font used within the template, I’ll write just in ascii LaTeX source notation things like subscripts and primes, e.g., I’ll write $L_1$ as meaning $L_{1_1}$.

Our decider for $A_{TM}$ assuming that the above set $L_1$ is decidable by machine $R$ is this machine/program $S$:

$S$: On input $\langle M, w \rangle$ do the following:

Build a machine $M'$ that on each input $y$ ignores $y$ and
immediately rejects.
Simulate $R(\langle M, w, M' \rangle)$, and
if $R$ accepts we accept and if $R$ rejects we reject (note that $R$ will always accept or reject as $R$ is a decider).
End of the program $S$

Why is $S$ a decider for $A_{TM}$? Well, certainly building the TM $M'$ won’t make us run forever. And note that $L(M') = \emptyset$. And keeping in mind that $R$ is a decider for $L_1$, the following things hold. If $\langle M, x \rangle$ belongs to $A_{TM}$, then since $M$ accepts $x$ and since $M'$ rejects $x$, $\langle M, x, M' \rangle \in L_1$, so $R$ accepts on that input, so $S$ accepts on $\langle M, x \rangle$. And if $\langle M, x \rangle$ does not belong to $A_{TM}$, then since $M$ does not accept $x$, $\langle M, x, M' \rangle \notin L_1$, so $R$ rejects on that input, so $S$ rejects on $\langle M, x \rangle$. Thus $S$ is a decider for $A_{TM}$.
YOUR NAME: 

**Question 2.** [27 points] Prove that $L_2 = \{\langle M_1, M_2, M_3, M_4 \rangle \mid L(M_1) = L(M_2) = L(M_3) \text{ and } L(M_1) \neq L(M_4)\}$ is undecidable. You must solve this using our template/contradiction approach, and you are **required** to key your answer on $A_{TM}$, that is, you’ll start by assuming $L_2$ is decidable and then will use a hypothetical total TM (decider) for $L_2$ to help you build a decider for $A_{TM}$. Of course, your answer should, following that construction, include the appropriate arguments, as per the template method.

**Answer:**

Our decider for $A_{TM}$ assuming that the above set $L_2$ is decidable by machine $R$ is this machine/program $S$:

- **S:**
  - On input $\langle M, w \rangle$ do the following.
  - Build a machine $M'$ that, on input $y$, ignores $y$ and rejects.
  - Build the machine $M''$ that, on input $y$, ignores $y$ and immersively simulates $M$ on $w$.
  - Simulate $R(\langle M', M', M', M'' \rangle)$ and
    - if $R$ accepts we accept and if $R$ rejects we reject (note that $R$ will always accept or reject as $R$ is a decider).

End of the program $S$

Why is $S$ is a decider for $A_{TM}$? Well, certainly building the TMs $M'$ and $M''$ won’t make us run forever. Note that $L(M') = \emptyset$. Note that $L(M'')$ equals $\emptyset$ if $\langle M, w \rangle \notin A_{TM}$ and equals $\Sigma^*$ if $\langle M, w \rangle \in A_{TM}$. And keeping in mind that $R$ is a decider for $L_2$, the following things hold. If $\langle M, x \rangle$ belongs to $A_{TM}$, then, since $\emptyset = \emptyset = \emptyset$ and $\emptyset \neq \Sigma^*$, we have that $\langle M', M', M', M'' \rangle$ is in $L_2$, and so $R$ accepts on $\langle M', M', M', M'' \rangle$, and so $S$ accepts on $\langle M, x \rangle$. And if $\langle M, x \rangle$ does not belong to $A_{TM}$, then since in that case $L(M') = \emptyset \neq \Sigma^* = L(M'')$, we have that $\langle M', M', M', M'' \rangle$ is not in $L_2$, and so $R$ rejects on $\langle M', M', M', M'' \rangle$, and so $S$ rejects on $\langle M, x \rangle$. Thus $S$ is a decider for $A_{TM}$. 

**Question 3.** [27 points]

Prove that $L_3 = \{\langle M \rangle \mid L(M) = \{\epsilon, 0, 1, 00, 11, 0000, 1111, \ldots \} \}$ is undecidable.

You must solve this using our template/contradiction approach, and you are **required** to key your answer on $A_{TM}$, that is, you’ll start by assuming $L_3$ is decidable and then will use a hypothetical total TM (decider) for $L_3$ to help you build a decider for $A_{TM}$. Of course, your answer should, following that construction, include the appropriate arguments, as per the template method.

**Answer:**

Our decider for $A_{TM}$ assuming that the above set $L_3$ is decidable by machine $R$ is this machine/program $S$:

$S$: 

On input $<M,w>$ build the machine $M'$ that itself, when its input is $y$, does the following:

- If $y$ is not one of $\epsilon, 0, 1, 00, 11, 0000, 1111, \ldots$, then reject
- else immersively simulate $M$ on $w$.

Simulate $R(<M'>)$, and if $R$ accepts we accept and if $R$ rejects we reject (note that $R$ will always accept or reject as $R$ is a decider).

End of the program $S$.

Why is $S$ is a decider for $A_{TM}$? Well, certainly building the TM $M'$ won’t make us run forever. Note that $L(M')$ equals $\emptyset$ if $\langle M, w \rangle \notin A_{TM}$ and equals $\{\epsilon, 0, 1, 00, 11, 0000, 1111, \ldots \}$ if $\langle M, w \rangle \in A_{TM}$. And keeping in mind that $R$ is a decider for $L_3$, the following things hold. If $\langle M, x \rangle$ belongs to $A_{TM}$, then, since $L(M') = \{\epsilon, 0, 1, 00, 11, 0000, 1111, \ldots \}$, we have that $\langle M' \rangle$ is in $L_3$, and so $R$ accepts on $\langle M' \rangle$, and so $S$ accepts on $\langle M, x \rangle$. And if $\langle M, x \rangle$ does not belong to $A_{TM}$, then since in that case $L(M') = \emptyset \neq \{\epsilon, 0, 1, 00, 11, 0000, 1111, \ldots \}$, we have that $\langle M' \rangle$ is not in $L_3$, and so $R$ rejects on $\langle M' \rangle$, and so $S$ rejects on $\langle M, x \rangle$. Thus $S$ is a decider for $A_{TM}$. 
Question 4. [27 points] Prove that if $A$ is Turing-recognizable and $\overline{A}$ is Turing-recognizable, then $A$ is decidable. (In solving this you may, if you wish, use as settled truth—and without having to yourself provide a proof of this fact—the fact that a set $B$ is Turing-recognizable if and only if there exists an enumerating Turing machine (aka an enumerator) $E$ such that the language enumerated by $E$ is $B$.)

Answer sketch:

Briefly, given such a set $A$, since $A$ is Turing-recognizable and $\overline{A}$ is Turing-recognizable, there are TMs $M$ and $M'$ such that $L(M) = A$ and $L(M') = \overline{A}$. Our decider for $A$ works as follows. On input $y$ (that is a member of $\Sigma^*$ of course), our decider for $A$ will in a dovetailed fashion simulate both $M$ running on $y$ and $M'$ running on $y$. (By dovetailed, we here mean run $M$ on $y$ for 1 step, then run $M'$ on $y$ for 1 step, then run $M$ on $y$ for 2 steps, then run $M'$ on $y$ for 2 steps, and so on.) Except as soon as, in that dovetailing, we see one of $M$ or $M'$ accept $y$, then we immediately accept if we saw $M$ accept $y$ and we immediately reject if we saw $M'$ accept $y$.

Note that since each $y$ is either in $A$ or in the complement of $A$, one or the other of $M$ and $M'$ must eventually accept $y$, and if it is $M$ then indeed that proves that $y$ is in $A$ and if it is $M'$ then that indeed proves that $y$ is not in $A$. So the machine we just defined is a decider, and indeed is a decider for $A$.

The proof I just gave does this in terms of Turing machines. However, there is also an equally good proof that uses an argument involving enumerating Turing machines.
Question 5. [27 points] (2 pts per correct part, and 1 free point) True/False. Answer True or False (do not show your work—this is True/False; don’t write just T or F as those can sometimes be hard to distinguish) in the provided space just before each problem. Be careful; some of these questions are perhaps hard. Recall that $A \leq_m B$ (this is known as many-one polynomial-time reductions, aka polynomial-time mapping reductions) if and only if there exists a polynomial-time function $f$ such that, for each $x$: $x \in A \iff f(x) \in B$. 

1. False [2 pts] For each $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$, if $A$ and $B$ are undecidable then $A \cup B$ is undecidable. (True/False)

2. True [2 pts] $\{x \in \{0,1\}^* \mid |x| \equiv 0 \pmod{2}\} \leq_m \{x \in \{0,1\}^* \mid |x| \equiv 1 \pmod{2}\}$. (True/False)

3. True [2 pts] For each $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$, if $A$ and $B$ are decidable sets, then $A \cap B$ is decidable. (True/False)

4. True [2 pts] For each $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$, if $A$ and $B$ are Turing-recognizable sets, then $A \cap B$ is Turing-recognizable. (True/False)

5. True [2 pts] For each $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$, if $A \in P - \{\emptyset, \Sigma^*\}$ and $B \in P - \{\emptyset, \Sigma^*\}$ then $A \leq_m B$. (True/False)

6. True [2 pts] For each $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$, if $A$ and $B$ are both NP-complete then $B \leq_m A$. (True/False)

7. False [2 pts] For each $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$, if $A \in \text{NP}$ and $A \leq_m B$, then $B \in \text{NP}$. (True/False)

8. True [2 pts] For each $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$, if $A \in \text{NP}$ and $B \leq_m A$, then $B \in \text{NP}$. (True/False)

9. False [2 pts] For each $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$, if $A$ and $B$ are Turing-recognizable sets, then $A \Delta^B$ is Turing-recognizable. (True/False)

10. True [2 pts] For each $A \subseteq \Sigma^*$, if $A$ is decidable, then $\overline{A}$ is decidable. (True/False)

11. False [2 pts] For each $A \subseteq \Sigma^*$, if $A$ is Turing-recognizable, then $\overline{A}$ is Turing-recognizable. (True/False)

12. True [2 pts] For each $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$, if $A$ and $B$ are decidable sets, then $A \Delta B$ is decidable. (True/False)

13. False [2 pts] For each $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$, if $A$ and $B$ are Turing-recognizable sets, then $\overline{A} \cup B \cup \overline{A} \Delta M$ is Turing-recognizable. (True/False)
Answer:

The answers are included above. Some quick comments on some of the harder of the answers:

1. Let $Y$ be any undecidable problem, e.g., $A_{TM}$. Let $A = \{0x \mid x \in Y\} \cup \{\epsilon\} \cup \{1x \mid x \in \Sigma^*\}$ and $B = \{1x \mid x \in Y\} \cup \{\epsilon\} \cup \{0x \mid x \in \Sigma^*\}$. $A$ and $B$ are each easily seen to be undecidable (as can easily be proven by the template method), yet $A \cup B = \Sigma^*$ and $\Sigma^*$ is not undecidable.

2. The reduction that postpends one bit to its input does that reduction.

5. The two exclusions are because (within our current setting) every set in P other than $\emptyset$ fails to $\leq_p$-reduce to $\emptyset$, and every set in P other than $\Sigma^*$ fails to $\leq_p$-reduce to $\Sigma^*$. (Do you see why?) Aside from those tricky special cases, however, we do have that $A \leq_p B$, namely, on input $x$ our poly-time reduction function itself in poly time solves the questions “$x \in A$?” and if the answer is Yes maps to a fixed element of $B$ and if the answer is No maps to a fixed element of $B$ (such elements exist as $B$ is neither $\emptyset$ nor $\Sigma^*$).

6. Follows from the definition of NP-completeness.

7. Let $Y$ be any undecidable problem, e.g., $A_{TM}$. $\emptyset \leq_p Y$, and $\emptyset \in P$, yet $Y$ is not in NP.

9. Let $A$ be any set that is Turing-recognizable but not decidable and let $B$ be $\Sigma^*$.

11. Every set (e.g., $A_{TM}$) that is Turing-recognizable but not decidable is a counterexample.

13. For $A = \Sigma^*$ and $B = \emptyset$ the specified set is $\emptyset \cup \emptyset \cup \overline{A_{TM}} = \overline{A_{TM}}$, which is not Turing-recognizable.
Question 6. [27 points]. For this question, you may use (without needing to include here a proof of this fact) the fact that SAT is NP-complete. Prove that the set $L_6 = \{ f \mid (each variable appearing in $f$ appears the same number of times$) \ AND \ f \in SAT\}$ is NP-complete. (Both $x$ and $\overline{x}$ are viewed as instances (appearances) of the variable $x$, and so the formula $(x \lor \overline{x} \lor \overline{x}) \land y$ has the variable $x$ appearing three times and the variable $y$ appearing one time.)

Note that we are NOT necessarily assuming that $f$ is in conjunctive normal form (CNF). Also, although on undecidability proofs we usually don’t make you explicitly handle the case where the input is syntactic garbage, in this problem’s solution’s parts (hint: that was a plural) you should make sure to handle the case where the input is not a syntactically legal formula.

Answer sketch:

Clearly, $L_6$ is in NP. Why? Our NP machine for it can first deterministically check that its input is a syntactically legal formula in which all variables appear the same number of times as each other (if any of that fails, then reject). Then it simply guesses every possible assignment of the variables, and each path accepts exactly if its guessed assignment satisfies the formula.

On the other hand, SAT $\leq^p_m L_6$, by the following reduction. If our input is $F$, check whether $F$ is a syntactically legal formula and if not then output any fixed element of $\overline{L_6}$, e.g., $x_1 \lor x_1 \lor x_2$. Otherwise, for each variable in the formula, count how many times the variable appears. Let $k$ be the largest such count. Select some (not wildly long) variable name that does not appear in $F$; below we will denote that variable as $new$. And we’ll use $x_1, \ldots, x_j$ to denote the variables that appear in $F$.

Now, output the following formula (which clearly can be computed in polynomial time, as is required as we are building a $\leq^p_m$ reduction):

$(F) \land$

$(new \lor \ldots \lor new \lor$

$x_1 \lor \ldots \lor x_1 \lor$

$\ldots \ldots$

$x_j \lor \ldots \lor x_j)$, except let us specify the exact number of terms that are on each of those lines making up the second conjunct. Let us have “new” appear exactly $k + 1$ times in that second conjunct’s first line. And on the rest of the second conjunct’s lines, each variable $x_i$ will appear on its line exactly $k + 1 - n_i$ times, where $n_i$ is the number of times the variable $x_i$ appears in $F$. (If one’s logic allows 0-variable formulas—for example by allowing True and False as atoms and so True \lor True \lor False would in such a logic be a legal formula with 0 variables—then we can as a special case have our reduction output $F$ when the input $F$ is a syntactically legal 0-variable formula; but we’ll below ignore that special case, though we just mentioned how to alter the construction to handle it.)

Note that in this new formula, every variable appears exactly $k + 1$ times. If $F$ was
satisfiable, so is the new formula, via setting the $x_i$ values to whatever is any satisfying assignment of $F$, and setting new to True. If $F$ is not satisfiable, then if $F$ was syntactic junk then we mapped to an element of $\overline{L}_6$ and if $F$ was syntactically valid, then we mapped to something that belongs to $L_6$ (why?—because if the output formula can be satisfied, then the assignment that does that must make each conjunct true, and so the first conjunct can be made true, which means that $F$ in fact is satisfiable). So we indeed have established that SAT $\leq_p L_6$. (Note: One can shorthand the formula and make it more rigorous using a math notation but it would convey less as an answer unless you know that particular notation. If one is very careful (to avoid having things that appear exactly $k$ times generating syntactic junk in the output, such as side-by-side OR symbols; and one can handle that for example by using as a shorthand a more math-y notation along with invoking an existing convention about empty disjuncts), one can use $k$ rather than $k + 1$ as the goal-number of times each variable appears, and can build a proof around that.)