CS 373: Intro to Theory of Computation
Spring 2010 **HW 10**, April, 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 (14 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. If L_1 is context free and L_2 is not context free, then $L_1 L_2$ is not context free.

   False: ☐ **True**: ☐

Solution:

False. Pick $L_1 = \emptyset$ and $L_2 = \{a^n b^n c^n \mid n \geq 0\}$, we have $L_1.L_2 = \emptyset$.

2. If L_1 is context free and L_2 is not context free, then $L_1 L_2$ is context free.

   False: ☐ **True**: ☐

Solution:

False. Pick $L_1 = \{\epsilon\}$ and $L_2 = \{a^n b^n c^n \mid n \geq 0\}$, we have $L_1.L_2 = \{a^n b^n c^n \mid n \geq 0\}$.

3. Every context free language is not regular.

   False: ☐ **True**: ☐

Solution:

False. $\emptyset$ is regular.

4. If L_1 and L_2 are context free, then $L_1 \cap L_2$ is not context free.

   False: ☐ **True**: ☐

Solution:

False. For example pick $L_1 = L_2 = \emptyset$.

5. If L is not context free, then it is not regular.

   False: ☐ **True**: ☐

Solution:

True. All regular languages are context-free too.

6. Let $\Sigma = \{a, b\}$ and $L = \{a^n w a^n \mid n \geq 1, w \in \Sigma^*\}$. L is not regular but is context free.

   False: ☐ **True**: ☐

Solution:

False. L is regular with regular expression $a(a+b)^*a$.

7. A non-deterministic TM can decide languages that a regular TM cannot decide.

   False: ☐ **True**: ☐

Solution:

False. We know that TM's and NTM's are as powerful as each other.

8. For any $k > 1$, there is no language that is decided by a TM with k tapes, but is undecidable by any TM having $k-1$ (or less) tapes.

   False: ☐ **True**: ☐

Solution:

True. We know that a k-tape TM ($k \geq 1$) is as powerful as a 1-tape TM.

9. If a language L is context-free then $\overline{L}$ is TM decidable.

   False: ☐ **True**: ☐

Solution:

True. We know that the membership problem for CFL's is decidable.

10. The language $\overline{\text{A}_{\text{TM}}} = \left\{ \langle M, w \rangle \;\middle|\; M \text{ does not accept } w \right\}$ is TM recognizable.

    False: ☐ **True**: ☐

Solution:

False. We know that A_{TM} is recognizable and not decidable. Also we know that if L and $\overline{L}$ are both TM-recognizable, the L must be decidable. Therefore $\overline{A_{TM}}$ is not TM-recognizable.

11. It is possible for some undecidable language to be NP-Complete.

False: ☐ **True**: ☐

Solution:

False. Each NP-Complete set is in NP that means it has a polytime NTM that decides it, by definition.

12. Suppose L is TM recognizable but not TM decidable. Then any TM that recognizes L must fail to halt on an infinite number of strings.

False: ☐ **True**: ☐

Solution:

True. If not we can imagine a decider for L: first compare input to the elements of that finite set and if it is there reject. Otherwise simulate "recognizer" of L on the input and return what it returns.

Note that it just shows that such a decider exists, it is not a recipe of how to construct it since we don't have a representation of that finite set.

13. If VertexCover is in P, then HamiltonianPath is also in P.

False: ☐ **True**: ☐

Solution:

True. VertexCover is an NP-Complete problem and once we prove that an NP-Complete problem is in P, we have shown that NP =P. This would imply that all NP problem are in P, including the HamiltonianPath problem.

14. $3n^2 + 5n + 2 = O(\lg n + n^2/2)$

False: ☐ **True**: ☐

Solution:

True. For $n \geq 1$ we can see for example $3n^2 + 5n + 2 \leq 100(\lg n + n^2/2)$.

Problem 2: Classification (10 points)

For each language L described below, we have listed 2–3 language classes. Mark the most restrictive listed class to which L must belong. E.g. if L must always be context free and decidable (but not always regular) and we have listed "regular", "context-free" and "decidable", you must mark only "context-free".

(a) $L = \left\{ xw \;\middle|\; x, w \in \{\mathtt{a}, \mathtt{b}\}^* \text{ and } |x| = |w| \right\}$

☐ **Regular** ☐ **Context-free** ☐ **TM-Decidable**

Solution:

Regular with regular expression $((a+b)(a+b))^*$. Every even length string can be partitioned into two equal length parts.

(b) $L = \left\{ \mathtt{a}^i\mathtt{b}^j\mathtt{c}^k\mathtt{d}^m \;\middle|\; i+j+k+m \text{ is a multiple of } 13 \right\}$

☐ **Regular** ☐ **Context-free** ☐ **TM-Decidable**

Solution:

Regular. We just need to remember the remainder on 13 of the total number of characters visited in the string. To do this, we don't need more than 13 states in a DFA (corresponding to 13 possible remainders: $0, 1, \ldots, 12$). We start by the state representing remainder 0 and as we visit new character, we change state to the next remainder (and from state representing 12 to state representing 0).

(c) $L = \left\{ \langle w, M_1, M_2, \ldots, M_k \rangle \;\middle|\; \begin{array}{l} w \text{ is a string,} \\ k \text{ is an odd number larger than 2,} \\ \text{each } M_i \text{ is a TM,} \\ \text{and a majority of the } M_i\text{'s accept } w \end{array} \right\}$

☐ **TM-Decidable** ☐ **TM-Recognizable** ☐ **Not TM recognizable**

Solution:

TM-Recognizable: We start all simulations $M_i(w)$ in parallel and we accept once a majority of them halt and accept.

It is not TM-decidable since we can reduce A_{TM} to it: given $\langle M, w \rangle$ we can pass $\langle w, M, \ldots, M \rangle$ to a decider of L and report the result.

(d) $L = \{x_1 \# x_2 \# \ldots \# x_n \mid x_i \in \{a, b\}^* \text{ for each } i \text{ and, for some } i, x_i \text{ is a palindrome}\}$.

☐ **Regular** ☐ **Context-free** ☐ **TM-Decidable**

Solution:

Context-free: L is the language of the following grammar:

$$\begin{aligned} S &\implies S\#A \mid A\#S \mid P \\ P &\implies aPa \mid bPb \mid \epsilon \\ A &\implies aA \mid bA \mid \#A \mid \epsilon \end{aligned}$$

L is not regular: By the way of contradiction assume that it is. Then $L' = L \cap L((a+b)^*)$ must be regular. L' is the set of all palindrome strings, which we know is not regular (For example using MNT: we can see that all these strings are distinguishable $\{1, 01, 001, \ldots, 0^i1, \ldots\}$; the suffix 0^i distinguishes 0^i1 and 0^j1 where $i \neq j$).

(e) $L = \left\{ \langle G, D \rangle \;\middle|\; G \text{ is a CFG}, D \text{ is a DFA}, \text{and } L(G) \subseteq L(D) \right\}$

☐ **TM-Decidable** ☐ **TM-Recognizable** ☐ **Not TM recognizable**

Solution:

TM-Decidable: $L(G) \subseteq L(D)$ iff $\overline{L(D)} \cap L(G) = \emptyset$. We compute a DFA for $\overline{L(D)}$ and intersect it with grammar G to obtain a new grammar G'. And then we check grammar G' for emptyness (we have seen algorithms for all these three steps).

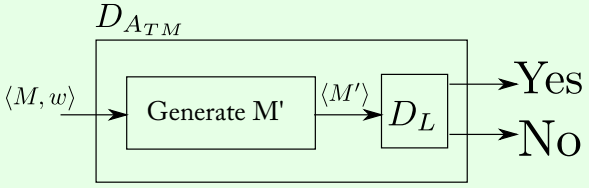
Problem 3: Reduction (10 points)

Prove that L is undecidable where:

$$L = \{\langle M \rangle \mid M \text{ is a TM and accepts some sting of odd length}\}$$

You are not allowed to use Rice's Theorem in this problem (although you can adapt the proof of Rice's Theorem to this problem).

Solution:



We reduce A_{TM}, which we know is undecidable, to L. That is we give the code for a decider of A_{TM}, $D_{A_{TM}}$ assuming that we have access to a subroutine (a TM) that decides L.

Now let's investigate the language of the following TM M'.

Algorithm $M'(x)$
1. $r \leftarrow$Simulate $M(w)$
2. **if** $r =$"yes"
3. **then return** "yes"
4. **else return** "no"

If $\langle M, w \rangle \in A_{TM}$, then the simulation in line 1 always returns "yes" and therefore always line 3 executes which means $L(M') = \Sigma^*$ and therefore $\langle M' \rangle \in L$. If $\langle M, w \rangle \notin A_{TM}$, then either $M(w)$ in line 1 returns "no" in that case $M'(x)$ returns "no" in line 4, or $M(w)$ never halts, in that case $M'(x)$ never halts; so in both these two cases $L(M') = \emptyset$ and hence $\langle M' \rangle \notin L$.

So the idea for implementing $D_{A_{TM}}$ is to construct $\langle M' \rangle$ from M and w and pass it to D_L to answer the question $\langle M' \rangle \stackrel{?}{\in} L$ which as mentioned above will determine the answer to question $\langle M, w \rangle \stackrel{?}{\in} A_{TM}$. In pseudocode:

Algorithm $D_{A_{TM}}(\langle M, w \rangle)$
1. Write down $\langle M' \rangle$ using $\langle M, w \rangle$
2. **return** $D_L(\langle M' \rangle)$

Problem 4: Decidability (10 points)

Is L decidable? Prove your claim.

$$L = \{\langle M \rangle \mid M \text{ is a TM and if we start } M \text{ with a blank input tape, then it will finally write some non-blank symbol on its tape.}\}$$

Solution:

L is decidable:
We know that at each step of running a TM, the next step is being determined using the current internal state of the TM and the character that the head scans. If the tape is initially blank and TM never writes down any non-blank character, then the tape always remains blank and the head always scans a blank character. Therefore if it the TM never prints a non-blank symbol, after at least $|Q| + 1$ steps (Q is the set of states of the TM), either the TM halts or a state would repeat. If a state repeats, definitely the TM starts looping (because the head always scans a blank and the same state transitions will happen once we visit a state twice).

So to decide L, we need to simulate M with a blank tape for at most $|Q|+1$ steps, if it never writes down a non-blank symbol during this period, it will never write one at all.

Problem 5: Palindrome (10 points)

Let $L = \left\{ \langle M \rangle \;\middle|\; M \text{ is a Turing machine and } M \text{ accepts at least one palindrome} \right\}$.

Show that L is TM-recognizable, i.e. explain how to construct a Turing machine that accepts $\langle M \rangle$ exactly when M accepts at least one palindrome. Of course, M might run forever on some input strings.

Solution:

Idea: We start by an enumeration of all the strings. For each string w that is a palindrome, we start a simulation $M(w)$. We perform all these simulations using dovetailing. Once a simulation $M(w)$ halts and accepts, we stop and accept since we just found a palindrome that M accepts. Since we are trying all strings and we perform all simulations for arbitrarily many steps, if there is a plaindome that M accepts, we will find it.

Algorithm $R_L(\langle M \rangle)$

1. Check $\langle M \rangle$ to be valid TM code, reject if not.
2. **for** $i = 1, \ldots, \infty$
3. **for** $j = 1, \ldots, i$
4. Compute w_j from our fixed enumeration of strings.
5. **if** w_j is a palindrome
6. **then** Simulate M on w_j for at most i steps.
7. Accept if the previous simulation accepted.

Problem 6: Grammar design (10 points)

Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Let

$$J = \left\{ w \;\middle|\; \#_{\mathtt{a}}(w) = \#_{\mathtt{b}}(w) \text{ or } \#_{\mathtt{b}}(w) = \#_{\mathtt{c}}(w) \right\},$$

where $\#_z(w)$ is the number of appearances of the character z in w. For example, the word $x = \mathtt{baccacbbcb} \in \mathsf{L}(J)$ since $\#_{\mathtt{a}}(x) = 2$, $\#_{\mathtt{b}}(x) = 4$, and $\#_{\mathtt{c}}(x) = 4$. Similarly, the word $y = \mathtt{abbccc} \notin \mathsf{L}(J)$ since $\#_{\mathtt{a}}(y) = 1$, $\#_{\mathtt{b}}(y) = 2$, and $\#_{\mathtt{c}}(y) = 3$.

Give a context-free grammar whose language is J. Be sure to indicate what its start symbol is. (Hint: First provide a CFG for the easier language $K = \left\{ w \in \{\mathtt{a}, \mathtt{b}\}^* \;\middle|\; \#_{\mathtt{a}}(w) = \#_{\mathtt{b}}(w) \right\}$ and modify it into the desired grammar.)

Solution:

Let's first concentrate on writing productions that generate strings with the same number of a's and b's. These strings either start with an a and end with a b ($S \implies aSb$), start with a b and end with an a ($S \implies bSa$), or start and end with the same symbol (in this case we can always break the string into two strings each with equal number of a's and b's, i.e. $S \implies SS$. To see this observe that when we scan the string awa from left to right, initially the number of a's is more than $b's$ and near the end when we reach at the substring aw the number of b's is more than a's. Therefore somewhere in the middle, where the transition happens, the number of a's and b's should be equal in both sides.). Corresponding to these cases we can write the following grammar:

$$S \implies aSb \mid bSa \mid SS \mid \epsilon$$

Now we allow symbol c's to appear every where:

$$\begin{aligned} S &\implies aSb \mid bSa \mid SS \mid CS \mid \epsilon \\ C &\implies cC \mid \epsilon \end{aligned}$$

Similarly we can write a grammar for those strings that have equal number of b's and c's.

$$\begin{aligned} R &\implies bRc \mid cRb \mid RR \mid AR \mid \epsilon \\ A &\implies aA \mid \epsilon \end{aligned}$$

Putting these together we have the following grammar for J. Variable S_0 is the start symbol.

$$\begin{aligned} S_0 &\implies S \mid R \\ S &\implies aSb \mid bSa \mid SS \mid CS \mid \epsilon \\ C &\implies cC \mid \epsilon \\ R &\implies bRc \mid cRb \mid RR \mid AR \mid \epsilon \\ A &\implies aA \mid \epsilon \end{aligned}$$

Problem 7: Non-CFLness (10 points)

1. Use pumping lemma for CFLs or the corollary to the pumping lemma for CFLs to prove that L is not regular:

$$L = \{w\#w \mid w \in \{0,1\}^*\}$$

Solution:

Assume L is a CFL. Then by the corollary to the pumping lemma, there exists a p such that for every $w \in L$, $|w| \geq p$, there is a way to split w, $w = xty$ such that $|t| \leq p$ and there is a strict substring t' of t such that $xt'y \in L$.

Let p be the number assured by the corollary to the pumping lemma. Consider the word $w = 0^p1^p\#0^p1^p$. Since $w \in L$ and $|w| \geq p$, there exists x, t, y such that $w = xty$, $|t| \leq p$, and a strict substring t' of t such that $xt'y \in L$. If t occurs completely to the left of $\#$, then clearly, for any strict substring t' of t, $xt'y$ cannot belong to L, since we will be contracting the word left of $\#$ only. Similarly, t cannot occur completely to the right of $\#$.

Assume hence that $\#$ occurs in t. Since $|t| \leq p$, t must be of the form $1^i\#0^j$ for some $0 \leq i, j \leq p$. If t' has fewer than i 1's, then $xt'y$ will have fewer 1's to the left of $\#$ than to the right of it, and hence will not be in L. Similarly, if t' has fewer than j 0's, then $xt'y$ will have fewer 0's to the right of $\#$ than to the left of it, and hence will not be in L. Finally, t' cannot have the same number of 0's and 1's as t has, unless it has no $\#$, in which case too $xt'y$ is not in L. Hence there is no t' that is a strict substring of t such that $xt'y$ is in L. This contradicts the corollary to the pumping lemma. Hence our assumption that L is a CFL is wrong, and we conclude that L is not a CFL.

2. Prove that L is not context free using closure properties:

$$L = \{a^nb^nw \mid n \geq 0, w \in \{c,d\}^*, |w| = n\}.$$

Note: $\{a^nb^nc^n \mid n \geq 0\}$ is the only language that you may assume we already knew that is not context free.

Solution:

Assume L is a CFL. Since for any CFL L_1 and a regular language L_2, $L_1 \cap L_2$ is a CFL, $L \cap a^*b^*c^*$ is a CFL. But $L \cap a^*b^*c^* = \{a^nb^nc^n \mid n \geq 0\}$, which we know is not a CFL. This contradiction proves that our assumption that L is a CFL is wrong. Hence L is not a CFL.

Problem 8: Normal Form (16 points)

Write this grammar in Chomsky Normal Form. Then use CYK algorithm to determine whether *ababb* is in the language of this grammar. (S is the start symbol)

$$S \to ASA \mid aB$$
$$A \to B \mid S$$
$$B \to b \mid \epsilon$$

Solution:

- First, we add the new start variable S':

$$S' \to S$$
$$S \to ASA \mid aB$$
$$A \to B \mid S$$
$$B \to b \mid \epsilon$$

- Now we need to find all variables that can yield ϵ: A and B can go to ϵ, but not S. We need to eliminate all $\epsilon - productions$ and compensate them, by adding new productions.

$$S' \to S$$
$$S \to ASA \mid aB \mid SA \mid AS \mid a$$
$$A \to B \mid S$$
$$B \to b$$

- Next, we need to get rid of unit rules. We have these unit-rules: $S' \to S$, $A \to B$ and $A \to S$. Once again, we must compensate for the absense of these rules by adding new ones.

$$S' \to ASA \mid aB \mid SA \mid AS \mid a$$
$$S \to ASA \mid aB \mid SA \mid AS \mid a$$
$$A \to b \mid ASA \mid aB \mid SA \mid AS \mid a$$
$$B \to b$$

- Next step is to get rid of the rules of improper form. We achieve this by adding new variables.

$$S' \to T_{AS}A \mid T_aB \mid SA \mid AS \mid a$$
$$S \to T_{AS}A \mid T_aB \mid SA \mid AS \mid a$$
$$A \to b \mid T_{AS}A \mid T_aB \mid SA \mid AS \mid a$$
$$B \to b$$
$$T_{AS} \to AS$$
$$T_a \to a$$

Problem 9: Proof (10 points)

Prove fomally (preferably using induction) that all strings that the following grammar generates have even length. S is the start symbol.

$$S \to SB \mid aa$$
$$B \to bSBb \mid ab$$

Solution:

We need to show that for all w such that $S \Rightarrow^* w$, $|w|$ is even.

We will prove, by induction over n, the following claim for every n:

Claim: If S derives w in at most n steps or B derives w in at most n steps, then w is of even length.

Base case: $n = 1$
If S derives w in 1 step, then $w = aa$, which is of even length. If B derives w in 1 step, then $w = ab$, which is also of even length.

Induction step: $n + 1 > 1$
Assume the claim is true n. If S derives w in $n + 1$ steps, then the first step in the derivation must be $S \Rightarrow SB$ and where SB derives w in n steps. Hence $w = w_1 w_2$ where S derives w_1 in n steps or less, and B derives w_2 in n steps or less. Using the induction hypothesis, w_1 and w_2 are of even length, and hence w is of even length.

If B derives w in $n + 1$ steps, then the first step in the derivation must be $B \Rightarrow bSBb$ and where $bSBb$ derives w in n steps. Hence $w = bw_1w_2b$ where S derives w_1 in n steps or less, and B derives w_2 in n steps or less. Using the induction hypothesis, w_1 and w_2 are of even length, and hence w is of even length.

Hence all words derived from S (and B) are of even length, and hence every word in the language of the grammar is even.