CONCORDIA UNIVERSITY
Dept. of Computer Science and Software Engineering
COMP 335 – Introduction to Theoretical Computer Science
Fall 2017

$\mathcal{A}$ssignment 5
Submission through Moodle due on Monday December 4 at 23:55

1. (a) Suppose $L = \{a^i b^i c^{2i} : i \geq 0\}$ is context-free. Let m be the constant of the context-free pumping lemma for cfls. Choose $w = a^m b^m c^{2m}$. Clearly $w \in L$ and $|w| = 4m > m$. Let $w = uvxyz$ with $|vxy| \leq m$ and $|vy| \geq 1$. Then one of the following two cases holds:

   vy has no c's: Then if we pump down to get the string uxz, the number of c's in uxz has not reduced, that is, $n_c(uxz) = 2m$ but either $n_a(uxz) < m$ or $n_b(ux.) < m$ or both. So $uxz \notin L$, since either $n_c(uxz) \neq 2n_a(uxz)$ or $n_c(uxz) \neq 2n_b(uxz)$ or both.

   vy contains c's. In this case, since $|vxy| \leq m$, vy cannot contain any a's. Therefore if we pump down to get the string uxz, we have $n_c(uxz) < 2m$ but $n_a(uxz) = m$. So $uxz \notin L$ since $n_c(uxz) \neq 2n_a(uxz)$.

   We have shown in both cases that $uxz \notin L$, a contradiction to the pumping lemma. Therefore L is not context-free.

   (b) Suppose $L = \{w_1 c w_2 : w_1, w_2 \in \{a,b\}^*, w_1 \text{ is a substring of } w_2\}$ is context-free. Let m be the constant of the context-free pumping lemma for cfls. Choose $w = a^m b^m c a^m b^m$. Clearly $w \in L$ and $|w| = 4m + 1 > m$. Let $w = uvxyz$ with $|vxy| \leq m$ and $|vy| \geq 1$. The following possibilities are exhaustive:

   vy contains c: In this case, by pumping up once, we get the string uv^2xy^z which contains more than one c. Thus $uv^2xy^2z \notin L$.

   vy is in the part of the string before c. In this case, we pump up once. We get the string $uv^2xy^2z = w_1' c w_2$, but since $|w_1'| > |w_2|$, it is impossible that w_1' is a substring of w_2. Thus $uv^2xy^2z \notin L$.

   vy is in the part of the string after c. In this case, we pump down. We get the string $uxz = w_1 c w_2'$, but since $|w_1| > |w_2'|$, it is impossible that w_1 is a substring of w_2'. Thus $uxz \notin L$.

   vy contains symbols both before and after c: Note that since $|vxy| \leq m$, the string vy can only contain b's before the c and only contains a's after the c. Therefore $uxz = w_1' c w_2'$ but $n_a(w_2') < m$, while $n_a(w_1') = m$. Therefore it is impossible that w_1' is a substring of w_2', and $uxz \notin L$.

   In every case, we showed a contradiction to the pumping lemma. So L is not context-free.

   (c) Suppose $L = \{a^{n!} : n > 0\}$ is context-free. Then let $m = k + 1$ where k is the constant of the pumping lemma for cfls. Then $m \geq 2$. Choose $w = a^{m!}$. Clearly $w \in L$ and $|w| = m! \geq m$. Let $w = uvxyz$ with $|vxy| \leq m$ and $|vy| \geq 1$. Then $vy = a^j$ with $1 \leq j \leq m$. Since

$$m! + j \leq m! + m < m! + m(m!) = (m+1)m! = (m+1)!$$

   that is, $m! + j < (m+1)!$, we conclude that $uv^2xy^2z = a^{m!+j} \notin L$, a contradiction to the pumping lemma. Therefore L is not context-free.

2. $L \triangle F$ is not necessarily a regular language. As a counter-example, let L be a cfl that is not regular, such as $\{a^n b^n \mid n \geq 0\}$, and let $F = \emptyset$. Then $L - F = L$ and $F - L = \emptyset$. Therefore $L \triangle F = L$ which is not regular.

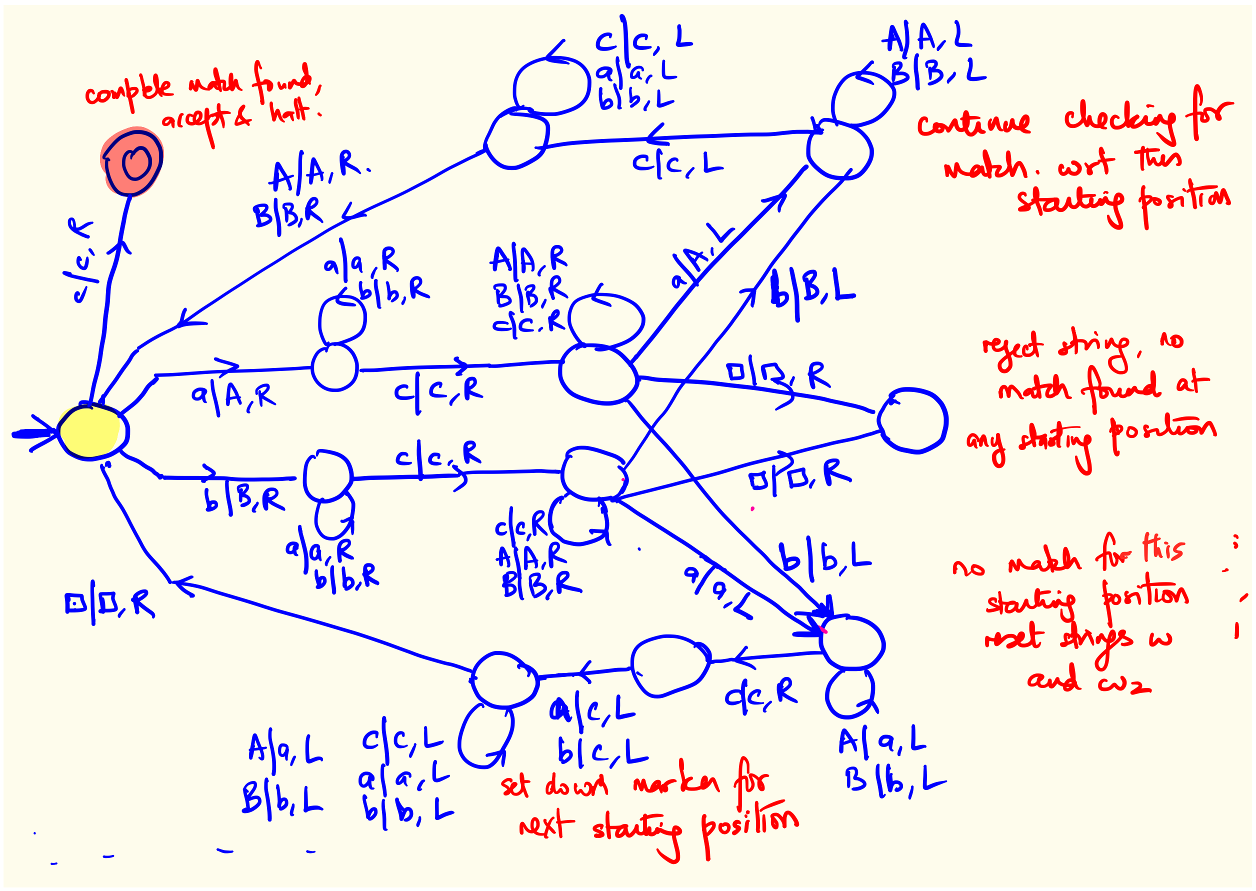
However, $L \triangle F$ must be a context-free language. First, notice that F is finite, and thefore regular, so $\overline{F}$ must be regular. It follows that $L - F = L \cap \overline{F}$ is context-free since it is the intersection of a cfl with a regular language. Secondly, $F - L$ must be finite, since $F - L \subseteq F$ and F is finite; therefore $F - L$ is context-free. We now see that $L \triangle F$ is the union of two context-free languages, and is therefore context free.

3. (a) Strategy: The main issue is to identify the possible starting position of the substring w_1 in the string w_2. We simply have to try all possibilities. So we

   i. Try to match w_1 with the substring immediately following the rightmost c (initially there is only 1 c but later we will write c's and will therefore have more). To do this, we convert the leftmost unmatched symbol in w_1 to the corresponding upper-case letter, move right, and try to find and match the leftmost unmatched symbol after the c. We keep doing this until the symbols keep matching. If a complete match is found, we go to a final state and halt.

   ii. If at some point while trying to match w_1, we find a mismatched symbol, we move left, converting uppercase symbols back to lower-case symbols. When we reach the rightmost c, we convert the next symbol to a c, thereby marking the position of the next starting point, before moving left once more to the leftmost symbol of the input string, all the while converting upper case As and B's to their lower case counterparts.

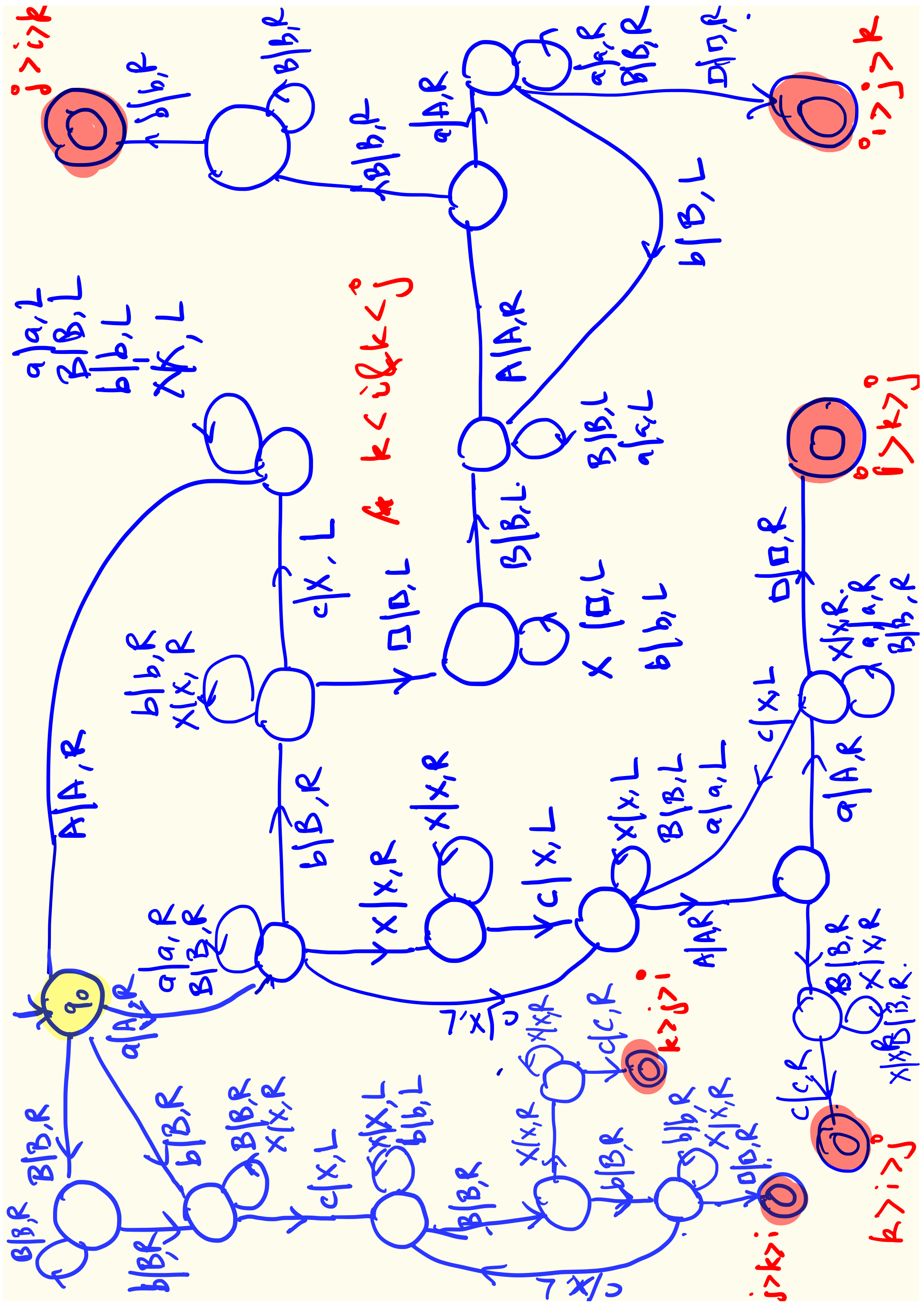
   iii. If while trying to match the next symbol of w_1, we fall off the end of the string, this indicates that there is no possible match, we simply halt in a non-final state.



(b) Strategy:

i. We try to match a's, b's and c's by repeatedly checking off the leftmost a, moving to the right to check off a matching (and leftmost) b, and moving again to the right to check off a matching leftmost c.

ii. If we run out of one type of symbol, we make sure there is at least one more of each of the other two symbols, and then proceed to make sure they don't match exactly.



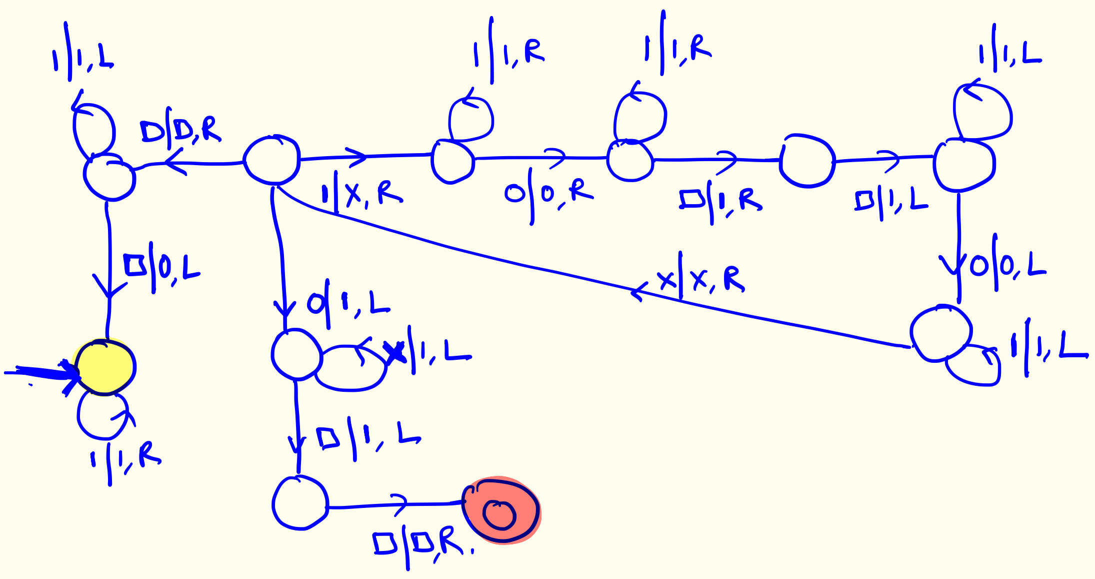
4. (10 marks) Give the transition diagram of a Turing machine that computes the following functions.

   (a) Strategy:

      i. Go to the end of the string, write a 0.

      ii. Repeatedly convert the leftmost 1 before the 0 to an X, move right, past the 0, and we write two more 1's at the very end of the string.

      iii. When we run out of 1's before the 0, we convert the 0 to a 1, we move to the left end of the string, converting X to 1 as we go, and we add one more 1 to the very left end of the string, making sure to leave the tape head at the left end of the output.



   (b) Strategy:

      i. First check that neither n nor m is 0 - if so, make sure the output is a blank tape.

      ii. If $n, m > 0$, go the end of the input string and write a \$ and then move to the left end of the string.

      iii. Now repeatedly erase the leftmost 1 before the 0, move past the 0, and write 1^m at the very end of the string - do this by

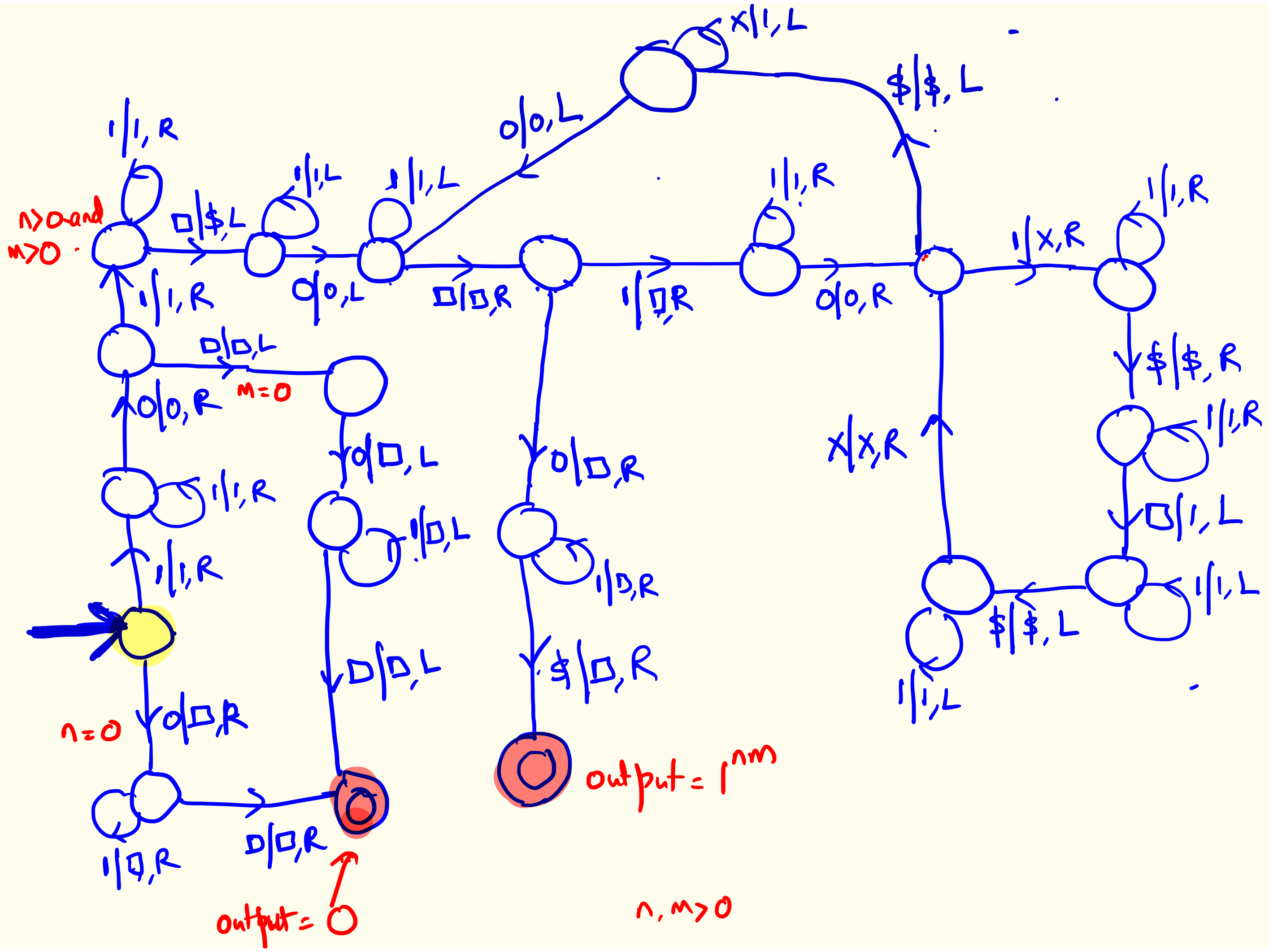
         A. repeatedly converting the leftmost 1 after the 0 to an X and

         B. moving to the very end of the string to write a 1, before

         C. moving left to find the leftmost 1 between the 0 and the \$. If there is no such 1, then we have written another block of m 1s.

      iv. Now move left to the left end of the string, on the way converting the X's in between 0 and \$ back to 1s.

      v. If there are no 1's on the left of 0, simply erase all symbols before and including the \$ and stop.



n>0 and m>0
1|1,R
□|$,L
1|1,L
1|1,L
0|0,L
1|1,R
0|0,L
x|1,L
$|$,L
1|1,R
1|X,R
1|1,R
□|□,R
1|□,R
0|0,R
1|1,R
0|□,L
m=0
0|0,R
1|1,R
1|1,R
$|$,R
1|1,R
0|□,L
1|□,L
0|□,R
X|X,R
□|1,L
1|□,R
1|1,L
$|$,L
1|1,L
□|□,L
$|□,R
n=0
0|□,R
1|□,R
□|□,R
output = 0
output = 1^nm
n,m>0