### Individual Homework

**Problem (1)**

For some $v \in \{0, 1\}^*$, define $v^c$ to be complement of $v$, meaning every 0 in $v$ is a 1 in $v^c$ and every 1 in $v$ is a 0 in $v^c$. For instance $\epsilon^c = \epsilon$, $(10)^c = 01$, and $(1011)^c = 0100$.

By definition of $L$ and $AccFut_L$, $uru \in AccFut_L(r)$ for any $u, r \in \{0, 1\}^*$. Furthermore, $r$ is the only string of length $|r|$ in $AccFut_L(r)$.

Let $s, t \in \{0, 1\}^*$ such that $t \neq s$. There are two cases:

**Case 1:** $s \notin AccFut_L(t)$

Since $s \in AccFut_L(s)$, then $AccFut_L(s) \neq AccFut_L(t)$.

**Case 2:** $s \in AccFut_L(t)$

Then $ts \in L$. If $|s| = |t|$, then $s = t$ which is a contradiction. Thus $s$ and $t$ have different lengths. Again there are two cases:

**Case 2.1:** $|s| > |t|$

Since $ts \in L$, then $s$ is of form $utu$ for some $u \in \{0, 1\}^+$. Consider $s' = u^c tu^c$, by definition $s' \in AccFut_L(t)$, but since $|s'| = |s|$ and $s' \neq s$, $s' \notin AccFut_L(s)$ as $s$ is the only string of length $|s|$ in $AccFut_L(s)$. Consequently, $AccFut_L(s) \neq AccFut_L(t)$.
Case 2.2: $|s| < |t|$

Similar to case 2.1. Since $ts \in L$, then $t$ is of form $usu$ for some $u \in \{0, 1\}^+$. Consider $t' = u^c su^c$, by definition $t' \in AccFut_L(s)$, but since $|t'| = |t|$ and $t' \neq t$, $t' \notin AccFut_L(t)$, as $t$ is the only string of length $|t|$ in $AccFut_L(t)$. Consequently, $AccFut_L(s) \neq AccFut_L(t)$.

Then for any $s \in \{0, 1\}^*$, $AccFut_L(s)$ is unique. There are infinitely many such $s$. By the Myhill-Nerode theorem any DFA representing $L$ must have infinitely many sates, so $L$ is nonregular.
Problem (2)

We give a state diagram of a Turing machine that recognizes

\[ L = \text{PALINDROME}_{0,1} = \{w \in \{0, 1\}^* : w = w^{rev}\} \]

First, we define the states \( Q \) and tape alphabet \( \Gamma \) of our Turing machine. \( Q = \{q_0, q_{\text{accept}}, q_{\text{reject}}, q_1, q_2, q_3, q_4, q_5\} \), where \( q_0, q_{\text{accept}}, q_{\text{reject}} \) are the respectively the initial, accepting, and rejecting states of the Turing machine. \( \Gamma = \Sigma \cup \{\sqcup\} = \{0, 1, \sqcup\} \). As for \( \delta \), we define \( \delta \) by the state diagram below and for any transitions that are not defined in the state diagram one can assume that they will map to the \( q_{\text{reject}} \).

\[ 0, 1 \to R \]
\[ 0 \to \sqcup, R \]
\[ \sqcup \to R \]
\[ \sqcup \to L \]
\[ \sqcup \to \sqcup, R \]
\[ 0 \to \sqcup, R \]
\[ 1 \to \sqcup, R \]
\[ 0 \to \sqcup, L \]
\[ 1 \to \sqcup, L \]
\[ 0, 1 \to L \]

The Turing machine works by first checking if we are in the base case, where the tape contains the empty string. If the tape contains the empty string then we move to the accepting state. Otherwise, if the tape is not the empty string then the Turing machine moves to either state \( q_2 \) or \( q_4 \) depending on whether the first element is a 0 or a 1, respectively, and writes on the tape \( \sqcup \),
a blank, in place of the first element and moving the head to the right along the tape. In the case of the first element being a 0, the Turing machine will remain in state $q_1$ and will continue moving right as long as it does not see a blank. This state corresponds to the Turing machine looking for the last non-blank cell of that tape. Once the Turing machine reaches the next blank cell it moves one step left to be at the last non-blank cell, and transitioning to state $q_2$.

State $q_2$ corresponds to checking whether the last non-blank element on the tape is equivalent to the first non-blank element on the tape. In state $q_2$, if the element below the head of the Turing machine is a 1 then the input is not a palindrome and so the Turing machine moves to $q_{reject}$. However, if the element in the cell is a 0 then the input may be a palindrome and so the Turing machine writes a blank into that cell and then proceeds to move its head left and transitioning to state $q_3$. The third case in state $q_2$ is that if the Turing machine sees a blank in the cell then it must be that the first and last non-blank cells on the tape are the same cell, which means that the tape contains a palindrome and so the Turing machine transitions to the accepting state.

In state $q_3$ the Turing machine continues to move left until it sees a blank cell, in which case the Turing machine transitions to state $q_0$. States $q_4$ and $q_5$ work analogously for the case where 1 is the first non-blank element on the tape. Essentially, the Turing machine checks whether the first and last elements of the input are the same and in the process it overwrites those cells with blanks, creating a new string that is two elements shorter. Then the Turing machine repeats the procedure on the smaller string until it determines whether the string is a palindrome.
Group Homework

Problem (1)

(a)

$w$ is a single letter (some element in $\Sigma$), so we can make the following DFA to recognize it.

Since $L$ is a regular language, it can be decided by a DFA. Similar to the procedure described in the solution to question 2 part (a) of the individual section of homework 4, we can combine the DFA that decides $L$ with the DFA that decides $\{w\}$ to construct a new DFA that decides $L \cup \{w\}$. Therefore, $L \cup \{w\}$ is regular.
(b)

$w$ is a single letter (some element in $\Sigma$), so we can make the following DFA to recognize it. Call this DFA $M_w$.

![DFA Diagram]

Since $L$ is a regular language, it can be decided by a DFA $M_L$. Similar to the procedure described in the solution to question 2 part (b) of the individual section of homework 4, we can generate new a DFA $M_{L\{w\}}$ that decides $L\{w\}$. $M_{L\{w\}}$ is defined as such.

$$Q_{M_{L\{w\}}} = \{(q_i, q_j)|q_i \in Q_{M_L}, q_j \in Q_{M_{\{w\}}}\}.$$  

$$\delta_{M_{L\{w\}}}((q_i, q_j), a) = (q_i', q_j') \text{ iff } \delta_{M_L}(q_i, a) = q_i' \text{ and } \delta_{M_{\{w\}}}(q_j, a) = q_j' \text{ for } a \in \Sigma, q_i \in Q_{M_L}, q_j \in Q_{M_{\{w\}}}.$$  

$$(q_0, q_0) \text{ is the start state of } M_{L\{w\}}.$$  

$$F_{M_{L\{w\}}} = \{(q_i, q_j)|q_i \in F_{M_L}, q_j \notin F_{M_{\{w\}}}\}.$$  

Key idea here is that $M_{L\{w\}}$ only accepts if $M_L$ will accept the input string and $M_{\{w\}}$ will reject it. Therefore $M_{L\{w\}}$ only accepts strings in $L\{w\}$. Consequently, $L\{w\}$ is regular.
(c)

For the sake of contradiction suppose \( L \cup \{w\} \) is regular.
If \( w \in L \), then \( L \cup \{w\} = L \), so \( L \) is regular, which is a contradiction.
If \( w \notin L \), then \( (L \cup \{w\}) \setminus \{w\} = L \). From part (b) of this question we know
\( (L \cup \{w\}) \setminus \{w\} \) is regular (given our assumption that \( L \cup \{w\} \) is regular).
Then \( L \) is regular, which is a contradiction. Thus, \( L \cup \{w\} \) is nonregular.

(d)

For the sake of contradiction suppose \( L \setminus \{w\} \) is regular.
If \( w \notin L \), then \( L \setminus \{w\} = L \), so \( L \) is regular, which is a contradiction.
If \( w \in L \), then \( (L \setminus \{w\}) \cup \{w\} = L \). From part (a) of this question we know
\( (L \setminus \{w\}) \cup \{w\} \) is regular (given our assumption that \( L \setminus \{w\} \) is regular).
Then \( L \) is regular, which is a contradiction. Thus, \( L \setminus \{w\} \) is nonregular.
Problem (2)

(a)

True
The two languages are decided by some DFA each. We can combine the DFAs so that the new DFA decides the union of the two languages. Apply the same procedure as the one described in the solution to Question 2 part (b) of the individual section in homework 4.

(b)

False
Let $\Sigma = \{0, 1\}$, define $L_0$ and $L_1$ as languages over $\Sigma$ as such:

$L_0 = \{0^k | k \in \mathbb{N}\}$
$L_1 = \{1^k | k \in \mathbb{N}\}$

From the lectures we know $L_0$ and $L_1$ are nonregular. Consider $L_0 \cup L_1$. Trivially $AccFut_{L_0 \cup L_1}(0^k) = AccFut_{L_0}(0^k)$ for all $k \in \mathbb{N}$, therefore by the Myhill-Nerode theorem any DFA representing $L_0 \cup L_1$ must have infinitely many states, so $L_0 \cup L_1$ is nonregular.

(c)

False
Let $\Sigma = \{0\}$, define $L_0$ and $L_1$ as languages over $\Sigma$ as such:

$L_0 = \{0^k | k \in \mathbb{N}\}$
$L_1 = \{0\}^* = \Sigma^*$

From the lectures we know $L_0$ is nonregular. Clearly $L_1$ is regular as it describes all the strings over $\Sigma$ (decided by a DFA with a single state which loops to itself). Consider $L_0 \cup L_1$. Since $L_0 \subset L_1$, we have $L_0 \cup L_1 = L_0$. As a result $L_0 \cup L_1$ is regular.
(d)

False
Let \( \Sigma = \{0\} \), define \( L_0 \) and \( L_1 \) as languages over \( \Sigma \) as such:

\[
L_0 = \{0^k \mid k \in \mathbb{N}\}
\]

\[
L_1 = \{0\}^* = \Sigma^*
\]

From the lectures we know \( L_0 \) is nonregular. Clearly \( L_1 \) is regular as it describes all the strings over \( \Sigma \) (decided by a DFA with a single state which loops to itself). Here \( L_0 \circ L_1 = 0(0)^* \) which is a regular language.
Problem (3)

We give a state diagram of a Turing machine that recognizes

\[ L = \{ w \in \{0, 1\}^* : \text{w has more 0's than 1'} \} \]

First, we define the states \( Q \) and tape alphabet \( \Gamma \) of our Turing machine. \( Q = \{ q_0, q_{\text{accept}}, q_{\text{reject}}, q_1, q_2, q_3, q_4 \} \), where \( q_0, q_{\text{accept}}, q_{\text{reject}} \) are the respectively the initial, accepting, and rejecting states of the Turing machine. \( \Gamma = \{0, 1, \sqcup, x\} \). As for \( \delta \), we define \( \delta \) by the state diagram below and for any transitions that are not defined in the state diagram one can assume that they will map to the \( q_{\text{reject}} \).

The Turing machine works by iteratively removing a 0 and a 1 from the tape. If the first non-blank cell contains a 0 but no subsequent 1’s can be found (in
order to be removed with the 0) then the Turing machine accepts the input as the input string contains at least one more 0 than it contains 1’s. On the other hand if the first non-blank cell contains a 1 but no subsequent 0’s can be found (in order to be removed with the 1) then the Turing machine rejects the input as the input string contains at least one more 1 than it contains 0’s. There is a simple case where the Turing machine rejects immediately if the tape is empty, in which case the number of 0’s equals the number of 1’s. However, in the case where the first cell contains a 0 and the Turing machine is then able to find a 1 as it sweeps its head right it must then somehow remove both of these values from the tape and continue the procedure from the beginning of the tape until only 0’s, 1’s, or no 0’s and no 1’s remain.

The way the Turing machine achieves this is by first overwriting the first cell with a ⊥ and then moving its head right and transitioning to either state $q_1$ or $q_3$. We consider without loss of generality the case where the first cell contains a 0 and the Turing machine transitions to state $q_1$ and moves its head right, after overwriting the current cell with ⊥. State $q_1$ then corresponds to continuously moving the head right until the first 1 is found or no 1’s are found in which case the string contains at least one more 0 than 1’s and is thus accepted. Similarly, $q_3$ serves an identical purpose, although searching for the first 0 and rejecting if no 0’s are found. Then once the first 1 is found the Turing machine overwrites this cell with an $x$, indicating that this element of this string is deleted, however, not confusing the Turing machine later as to whether this cell is the end of the string if it were to have used ⊥ instead of $x$, and then moves the tape head left and transitions to state $q_2$. Now the string contains one fewer 0’s and one fewer 1’s, and so the tape head must move to the leftmost non-blank cell of the tape, which corresponds to the beginning of this new substring, in order to repeat this procedure again. This procedure occurs in state $q_2$, and similarly in state $q_4$. Note that $x$’s are ignored whenever moving left and right along the tape as they represent elements of the string that are deleted, however, since cells on the tape cannot be deleted we must have some placeholder character for ignoring such cells.