CS 2800 HW #6

KIRILL CHERNYSHOV

Problem 1

(a) Suppose $q_1 \sim q_2$. Then, $\hat{\delta}(q_1, \epsilon) \in A \iff \hat{\delta}(q_2, \epsilon) \in A$. But $\hat{\delta}(q_1, \epsilon) = q_1$ and $\hat{\delta}(q_2, \epsilon) = q_2$, by the definition of the extended transition function. Therefore, $q_1 \in A \iff q_2 \in A$, i.e. if one state in an equivalence class is an accepting state then all other states in that class are also accepting states. This means that any equivalence class can contain only accepting states or only non-accepting states.

(b) In this context, a function f that is well defined satisfies the following property: if $a \sim b$ then $f(a) = f(b)$. To prove this by contradiction, assume that there exist q_a and q_b such that $q_a \sim q_b$ but $\delta_{min}(q_a, k)$ is not the same equivalence class as $\delta_{min}(q_b, k)$ for some $k \in \Sigma$.

(c) Using the definition of δ_{min}: suppose $\delta_{min}(q, a)$ is accepting for some q and a. Then, we know that $\delta_M(q, a)$ is accepting, and vice versa if it's rejecting. By the theorem proven in part a, all states in an equivalence class are either accepting or rejecting. That means that if $\delta_{min}([q], a)$ is accepting then all states in $[\delta_M(q, a)]$ are accepting, and vice versa if it is rejecting. This means that M_{min} accepts an input iff M does, so their languages are the same.

Problem 2

(a) This proof would work for a DFA. It does not work for a NFA because x is said to be recognisable by NFA M if *one* of the states that x *could* end up in is an accepting state. If x can end up in either state q_a, an accepting state, or q_r, a rejecting state, inverting all the states doesn't stop x from being accepted, since q_r now becomes an accepting state, and x can end up in q_r. Since x is accepted by both the original NFA and the new inverted NFA, their languages cannot be complements: if they were, then any string accepted by one machine would be rejected by the other machine.

An example is simply as stated above: a machine M that has three states, q_a, q_r and the starting state q_s, that takes the alphabet $\{0, 1\}$, and upon 1 transitions to either q_a or q_r. 1 can end up in q_a, so it is in the language of M. If the states are inverted to form machine N, 1 is still accepted since it can end up in q_r, which is now accepting. Therefore 1 is also in the language of N, and so $L(M)$ and $L(N)$ cannot be complements.

(b) Kleene's theorem implies that any NFA-recognisable language can be expressed as a regex. This is because if L is NFA recognisable, there exists a NFA M such that L is the language of M. By Kleene's theorem, there exists a regex whose language is L. Again by Kleene's theorem, every regex can be expressed as a DFA that recognises the same language. From this DFA a "complement" DFA can be made by flipping all the states (accepted becomes rejected and vice versa); the flawed proof above proves that this can invert a DFA (not an NFA) and its language will be complementary to the previous DFA's language. However, a DFA is also a NFA. Since this NFA must exist, the complement of any NFA-recognisable language is also DFA-recognisable.

(c) This DFA accepts the language denoted by the regex $aa + ab*$, since the only way to end up in q_2 is $ab*$ and the only way to end up in q_4 is aa (the only two accepting states). We can construct

a DFA M with the same alphabet that accepts this regex. The states of M are the starting state q_s, the hell state q_h and the three accepting states q_3, q_4 and q_5. A string starts in the starting state (duh); upon b it goes to hell and upon a it proceeds to q_3. At q_3, upon b the string proceeds to q_4 and upon a to q_5. In q_4, upon b the string remains in q_4 and upon a goes to hell; in q_5 upon either a or b it goes to hell. In hell, upon either a or b the string remains in hell.

If we take this DFA and flip all of its states (the accepting states become rejecting and vice versa), it now accepts the complement of the above regex. Since it is also a NFA, it satisfies the properties required.

(d) Once again we can create a DFA that recognises the regex and flip it to make its complement. This DFA has the same alphabet as above. It has starting state q_s, hell state q_h, and three accepting states, q_3, q_4 and q_5. q_s is also accepting, as the regex accepts ϵ. From q_s, a transitions to q_3, b goes to hell. From q_3, a goes to q_4, b goes to q_5. From q_4, a remains in q_4 and b transitions to hell, and from q_5 either character causes a transition to hell. In hell, any character will keep the string in hell.

To invert this DFA, make q_s, q_3, q_4 and q_5 rejecting, and q_h accepting. Analysing this new DFA, it can be seen that it accepts the regex $b + aba + abb + aaa * b$. Since this DFA is complementary to the previous one, this regex must be the complement of $ab + a*$.