Assignment One: Automata and Pattern Matching

John Pennell
ID: 834035412
University of Auckland

May 12, 2017

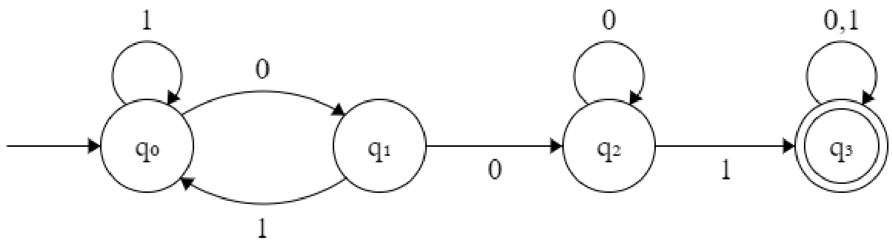
Question One

a) Let $A = \{0, 1\}, B = \{001\}, C = \{0, 1\}$, then
$L = \{\ xyz \mid x \in A^*, y \in B, z \in C^*\ \}$

b) Let L be the language stated in a) such that $x \in \{0, 1\}^m$ and $z \in \{0, 1\}^n$, where $m, n \geq 0$. Then the language L can have infinitely many strings containing the substring 001, where x and z are all possible strings of {0,1} that are greater than or equal to zero, i.e. $1001, 10011, 100111, ..., 100111^{n+1}$, where x loops once and z loops $n + 1$ times from the language. If m or n is zero, then the string x or z are considered the empty string ϵ. We know this is true as the concatenation of regular languages x, 001 and z are also regular, which we have proven in class. Thus, we have proven that there are infinitely many strings in L.

Suppose that we are looking for infinitely many strings that are not in L, then we would simply consider taking the complement of L as L', such that $L' = \{\ xyz \mid x \notin\ \ \{0, 1\}^*, y \notin \{001\}, z \notin \{0, 1\}^*\}$. This gives us all possible strings that are not in the language L and are outside of the set L. Thus, we have proven that there are infinitely many strings that are not in L.

c)



Proof: We shall show that the above DFA L = L(M).

Take some string w that is in L, so that $w = \sigma_1...\sigma_n$ and $run(w) = r_0...r_n$. We know that there are σ_{i-1}, σ_i and σ_{i+1} such that $\sigma_{i-1}\sigma_i\sigma_{i+1} = 001$. We note that r_{i-1} can potentially be any one of the states. If it is q_0, then by δ, $r_{i-1} = q_1$, $r_i = q_2$ and $r_{i+1} = q_3$. Another possibility is that r_{i-1} is q_3. We then note that if some $r_j = q_3$, then all following states will be q_3 , as for any input x, we have it that $\delta(q_3, x) = q_3$. Hence $r_n = q_3$. Thus, w is in L(M).

We shall now take some w in L(M), such that $w = \sigma_1...\sigma_n$ and $run(w) = r_0....r_n$. We know that $r_n = q_3$. Since that is not the starting state, we know that there is an $r_i \neq q_3$ such that $r_{i+1} = q_3$. The only such state to satisfy this proposition is q_2, and by δ we note that this must mean that $q_{i+1} = 0$. Since q_2 is not a starting state, then $i \neq 1$. This argument is repeated to deduce that $q_i = 1$. Thus the w indeed contains '001' , therefore $w \in L$.

Thus, from these two arguments we get that L = L(M).

d) $L = (0 + 1)^*001(0 + 1)^*$

Proof: As we have already proven that L is a regular language in part c), where L = L(M), as well as that the regular expression of L can be proven through the closure properties of regular languages, which we have learned in class. We can deduce that the correctness of the construction of the regular expression is equal to its regular language, where $L = (0+1)^*001(0+1)^* = \{\ u001v \mid u, v \in \{0, 1\}^*\}$, i.e. $10010, 100100, 1001000, ..., 1001000^n \in L$, but $1, 11, 111, ..., 1111^m \notin L$ as there is no substring 001. Thus, we have proven the correctness of the construction of the regular expression describing L.

Question Two

a) $L' = \{\oslash\}$. There is no connection between L and L' as ab^nc does not exist in the language. This is because there is no string which contains a, therefore cannot contain anything from L and cannot be accepted.

b) $L' = \{c^m b \mid m \geq 1\}$. There is a connection between L and L', as L' is a subset of L, where $a^nc \in L$, but $a^nc \notin L'$.

c) Suppose L is regular, then L' must also be regular, as we have proven this in class as part of the closure properties of regular languages, where the union of two regular languages is also regular. In this case it is the union of L and L', where $L_1 = L \cup L'$. We can also express this language through closure under intersection, where $L_2 = L \cap L' = \overline{\overline{L} \cup \overline{L'}}$. Thus, if L is regular, then L' is also regular.

Question Three

We have proven in class that it is algorithmically decidable whether a DFA M, accepts a string w. In this case w can be our string that contains an odd number of 1's. To prove this we would have to construct the trace of the computation of w on M and check whether its last state is final/accepted. Another way can be done by constructing a regular expression which describes all possible strings with an odd number of 1's and then proving it through constructing a DFA. Finally, we could also prove this using the Aho-Corasick algorithm to decide whether M recognizes exactly the binary strings that contain an odd number of 1's over the alphabet $\{0, 1\}$. Thus, these are some of the results proven in class which we have used.