

Automata (Fall 2008) Qual Exam

Answer ALL questions

Question 1 (20 points + 20 points + 15 points)

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, we overload the function name δ to define a function that maps letters in Σ to functions from Q to Q . Specifically, we define $\delta : \Sigma \rightarrow (Q \rightarrow Q)$ such that $\delta(a)(q) = q'$ iff $\delta(q, a) = q'$ where $a \in \Sigma$ and $q, q' \in Q$.

We can further extend δ to define $\delta : \Sigma^* \rightarrow (Q \rightarrow Q)$ inductively such that $\delta(aw) = \delta(a) \circ \delta(w)$ where $a \in \Sigma$, $w \in \Sigma^*$ and \circ denotes function composition. That is, $\delta(aw)(q) = \delta(a)(\delta(w)(q))$ for $q \in Q$. For the base case of the inductive definition, $\delta(\epsilon)$ is defined to be the identity function such that $\delta(\epsilon)(q) = q$ for $q \in Q$.

With respect to a DFA $M = (Q, \Sigma, \delta, q_0, F)$, we define an equivalence relation \equiv_M on strings in Σ^* such that $w \equiv_M w'$ iff $\delta(w) = \delta(w')$.

With respect to a given language L , we define an equivalence relation \equiv_L on strings in Σ^* such that $w \equiv_L w'$ iff $\forall \alpha, \beta \in \Sigma^*$, $\alpha w \beta \in L \iff \alpha w' \beta \in L$.

In the following questions, M is assumed to be a DFA.

(a) Show that \equiv_M has finite index; that is, there are finitely many \equiv_M equivalence classes. You are required to give an upper bound on the number of equivalence classes in terms of n , which is the number of states in M .

(b) Show that $w \equiv_M w'$ implies $w \equiv_L w'$ where L is the language of M .

(c) Illustrate with an example to show that $w \equiv_L w'$ does not necessarily imply $w \equiv_M w'$ where L is the language of M .

Question 2 (15 points + 15 points + 15 points)

Let Σ be a finite alphabet. For any string $w = a_1a_2 \dots a_n$, where $a_i \in \Sigma$, the reverse of w , written w^R , is the string w in reverse order, $a_n \dots a_2a_1$. For any language L , $L^R = \{w^R \mid w \in L\}$.

- (a) If L is regular, is L^R always regular? Justify your answer.
- (b) If L is context-free, is L^R always context-free? Justify your answer.
- (c) If L is deterministic context-free, is L^R always deterministic context-free? Justify your answer. (Note that a language is deterministic context-free if the language can be recognized by a deterministic pushdown automaton.)

Answers:

1 (a) Each equivalence class is characterized by the δ value which is a function from Q to Q . As there are n^n different possible functions from Q to Q , there are finitely many equivalence classes.

1 (b)

$$\begin{aligned} & \alpha w \beta \in L \\ \iff & \delta(\alpha w \beta)(q_0) \in F \\ \iff & (\delta(\alpha)\delta(w)\delta(\beta))(q_0) \in F \\ \iff & (\delta(\alpha)\delta(w')\delta(\beta))(q_0) \in F && \text{since } w \equiv_M w' \text{ implies } \delta(w) = \delta(w') \\ \iff & \delta(\alpha w' \beta)(q_0) \in F \\ \iff & \alpha w' \beta \in L \end{aligned}$$

1 (c)

Consider the DFA M with two states q_0 and q_1 over the alphabet $\{0, 1\}$ such that $\delta(q_0, 0) = q_0$, $\delta(q_0, 1) = q_1$, $\delta(q_1, 0) = q_0$ and $\delta(q_1, 1) = q_1$ where q_0 is the starting state, and both q_0 and q_1 are final states. Let $L = L(M) = \{0, 1\}^*$. That is, $w \equiv_L w'$ for all $w, w' \in \{0, 1\}^*$. However, $\delta(0)$ differs from $\delta(1)$. Specifically, $\delta(0)(q_0) = q_0$, $\delta(0)(q_1) = q_0$, while $\delta(1)(q_0) = q_1$, $\delta(1)(q_1) = q_1$.

2 (a) Yes, L^R is always regular. Given a DFA for a regular language L , we can reverse all the arrows and reverse the roles of starting and accepting states to obtain an NFA for L^R .

2 (b) Yes. L^R is always context. Given a CFG for L , we can reverse the right hand side string of every production rule. The resulting context-free grammar generates L^R .

2 (c) No. L^R may not be deterministic context-free. Consider

$$L = \{x0y \mid |x| = |y|, x \in 1^*, y \in \{0, 1\}^*\}$$

The language L is deterministic context-free. A deterministic PDA makes use of the stack to check to see if the two sides have the same length with the first occurrence of a 0 considered as the middle symbol of the input. Reversing the language will present difficulty for a deterministic PDA. While a machine can easily identify the first occurrence of a 0, it cannot decide which occurrence of 0 is the last occurrence unless the machine has finished reading the whole input.