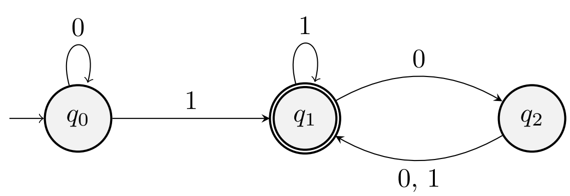
COMS 3261 Handout 2A: Deterministic Finite Automata Practice

Angel Cui and Owen Keith

Fall 2024

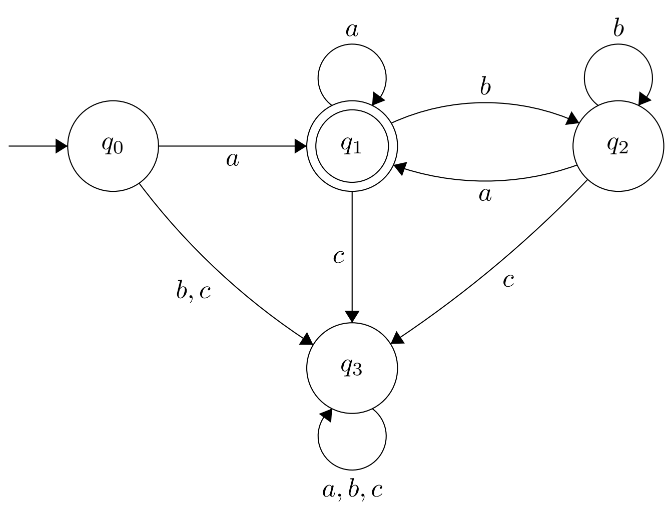
1

Determine which of ε, 11, 010, 10, 0101 is accepted by this DFA.



2

The DFA state diagram below is defined on the alphabet $\Sigma = \{a, b, c\}$. Write out its formal definition (as a 5-tuple). When specifying the transition function δ, draw a table.



3

Draw DFAs that recognize the following languages:

(a) $L = \{w \in \{a, b\}^* \mid w \text{ does not contain exactly two } a's\}$

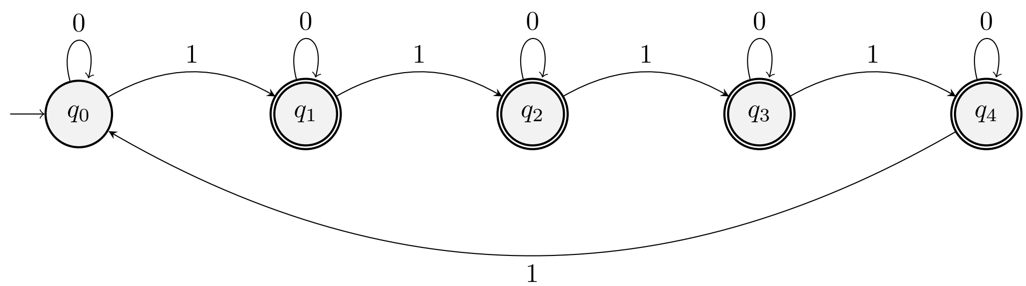
(b) $L = \{w \in \{a, b\}^* \mid w$ has even length and an odd number of a's$\}$

(c) $L = \{11, 101, 010, 0110\}$

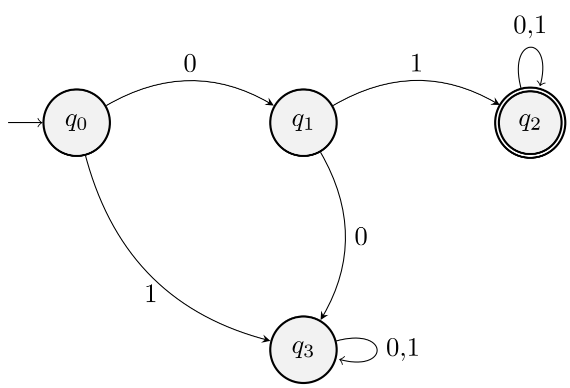
4

Given the DFA, specify the language it recognizes:

(a) DFA M:



(b) DFA N:



5 Non-Required Challenge Problem

NOTE: The following problem is just for fun and is **NOT** required material.

The Cantor set $\mathcal{C}$ is created by starting with the unit interval $[0, 1]$ and iteratively repeating the following process. First, the middle open third $\left(\frac{1}{3}, \frac{2}{3}\right)$ is removed, leaving two segments $\left[0, \frac{1}{3}\right] \cup \left[\frac{2}{3}, 1\right]$. Again, the open middle third of each remaining segment is removed leaving $\left[0, \frac{1}{9}\right] \cup \left[\frac{2}{9}, \frac{1}{3}\right] \cup \left[\frac{2}{3}, \frac{7}{9}\right] \cup \left[\frac{8}{9}, 1\right]$. At each step the open middle third of all remaining segments is removed, and the Cantor set $\mathcal{C}$ is the set of all points in $[0, 1]$ that are not deleted during any step of this infinite process.

Let $\Sigma = \{0, 1, 2\}$. Construct a DFA for the following language:

$$L = \{x \in \Sigma^* \mid \text{The base-3 number } 0.x \text{ is in the Cantor set}\}$$

6 Review of Definitions and Theorems

Alphabet

An ***alphabet*** Σ is a finite, non-empty, set of of symbols.

String

A ***string*** over an alphabet Σ is a finite sequence of characters from Σ.

The ***length*** of a string s, denoted $|s|$, is the number of characters in s.

The ***empty string***, denotes ε, is the string of length 0, i.e., $|\varepsilon| = 0$.

The ***concatenation*** of strings s and t, denoted $s \circ t$ (also denoted $s||t$), is the string obtained by appending t to the end of s.

$\mathbf{\Sigma^k}$ denotes the set of all strings of length k over Σ.

$\mathbf{\Sigma}^*$ denotes the set of all strings over Σ, i.e.,

$$\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k = \varepsilon \cup \Sigma^1 \cup \Sigma^2 \cup ...$$

Language

An ***language*** over an alphabet Σ is a set of strings $L \subseteq \Sigma^*$.

DFA

A ***deterministic finite automaton*** (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta : Q \times \Sigma \to Q$ is the ***transition function***,[a]
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.[b]

[a]A transition function maps each state and input symbol to a next state.
[b]Accept states are also called final states.

Definition of accepting computation

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$ and a string $w = w_1 ... w_n \in \Sigma^*$, **M accepts w** if and only if $\exists r_0, r_1, ..., r_n \in Q$ such that:

1. $r_0 = q_0$
2. $\forall i \in \{1, ..., n\}, \delta(r_{i-1}, w_i) = r_i$
3. $r_n \in F$

Definition of the language accepted by a DFA

For a DFA M, define

$$\mathbf{L(M)} = \{w \in \Sigma^* \mid M \text{ accepts } w\}$$

called "**the language recognized by M.**"

Regular Language

A language L is **regular** if and only if $\exists$ DFA M such that $L(M) = L$.

Theorem

Every finite language is regular.

Theorem

The class of regular languages is closed under **complement, union, intersection.** That is, if L is regular, then $\overline{L}$ is regular, if L_1 and L_2 are both regular, then $L_1 \cup L_2$ and $L_1 \cap L_2$ are both regular. Where

$$\overline{L} = \Sigma^* \backslash L = \{w \in \Sigma^* \mid w \notin L\}$$

$$L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ or } w \in L_2\}$$

$$L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ and } L_2\}$$