CS 39: Theory of Computation **Winter '23**

PSET 1 — 2023-01-16

Prof. Chakrabarti ***Student: Amittai Siavava***

Credit Statement

I worked on these problems alone, with reference to class notes and the following books:

(a) **Introduction to the Theory of Computation** by **Michael Sipser**.

(b) **A Mathematical Introduction to Logic** by **Herbert Enderton**.

Problem 1.

For a language L over alphabet Σ, define

$$\textsc{Cycle}(L) = \{yx : x, y \in \Sigma^* \text{ and } xy \in L\},$$

$$\textsc{Half}(L) = \{x \in \Sigma^* : \exists y \in \Sigma^* (|x| = |y| \text{ and } xy \in L)\},$$

$$\textsc{HalfPalindrome}(L) = \left\{x \in \Sigma^* : xx^R \in L\right\}.$$

(a) Let $A = \{(01)^n : n \geq 0\}$ and $B = \{0^n 1^n : n > 0\}$. What is $\textsc{Cycle}$(A) and $\textsc{Cycle}$(B)?

$$\textsc{Cycle}(A) = A \cup \{(10)^n : n \geq 0\}$$

$$\textsc{Cycle}(B) = \left\{0^{\max(0,\, n-a)}\ 1^{\min(n,\, 2n-a)}\ 0^{\min(n,\, a)}\ 1^{\max(0,\, a-n)} \text{ where } a \geq 0\right\}$$

(b) Let $C = \{0^p 1^q 0^r : p, q, r \geq 0 \text{ and } q = p + r\}$. What is $\textsc{Half}$(C) and $\textsc{HalfPalindrome}$(C)?

$$\textsc{Half}(C) = \{0^m 1^n : m \geq 0, n \geq 0\}$$

$$\textsc{HalfPalindrome}(C) = \{0^m 1^n : m \geq 0, n = m\}$$

Problem 2.

A string u is a *proper prefix* of a string v if u is a prefix of v and $u \neq v$. For a language L over alphabet Σ, define

$$\textsc{Min}(L) = \{x \in \Sigma^* : x \in L \text{ and no proper prefix of } x \in L\},$$

$$\textsc{Max}(L) = \{x \in \Sigma^* : x \in L \text{ and } x \text{ is not a proper prefix of any string in } L\}.$$

(a) Let $A = \{0^m 1^n : m, n \geq 0\}$. What is $\textsc{Min}(A)$ and $\textsc{Max}(A)$?

Since $\varepsilon \in A$ and ε is a proper prefix of every string except itself, $\textsc{Min}(A) = \{\varepsilon\}$.

On the other hand, every string in A is a proper prefix of some string in A (just take the string and append a 1, for instance). Therefore, $\textsc{Max}(A) = \varnothing$.

(b) Let $B = \{0^m 1^n, m, n > 0\}$. Read carefully; $B \neq A$. What is $\textsc{Min}(B)$ and $\textsc{Max}(B)$?

$$\textsc{Min}(B) = \{0^m 1 : m > 0\}$$

$$\textsc{Max}(B) = \varnothing$$

(c) Specify an infinite language L such that $\textsc{Min}(L) = \textsc{Max}(L)$.

$$L = \{0^n 1^n : n > 0\}$$

$$\textsc{Min}(L) = \textsc{Max}(L) = L$$

Problem 3.

For each of the languages below over the alphabet $\Sigma = \{0, 1\}$, specify a simple DFA by drawing a state diagram. Simplicity is important here: don't design unnecessarily complicated DFAs.

(a) The language that accepts only the empty string.

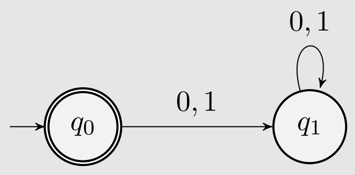

FIGURE 3.1. Only accept the empty string.

(b) The set of all strings that in Σ^* except 100 and 101.

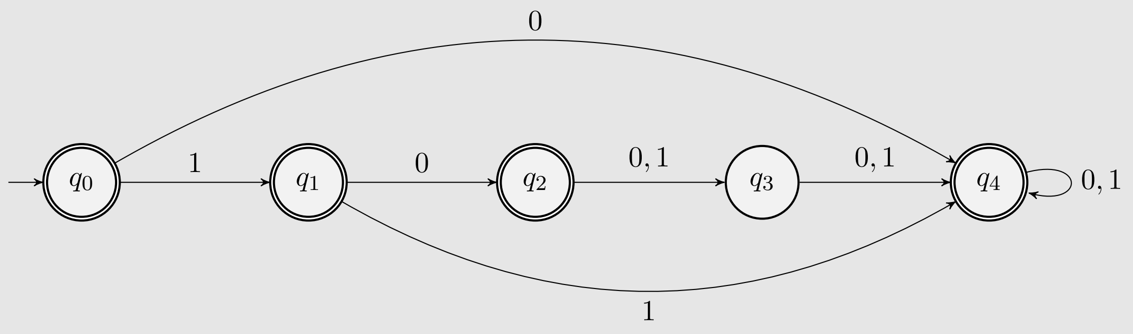

FIGURE 3.2. Accept all strings in $\{0, 1\}^*$ except 100 and 101.

(c) $\{x \in \Sigma^* : x \text{ does not contain the substring } 110\}$.

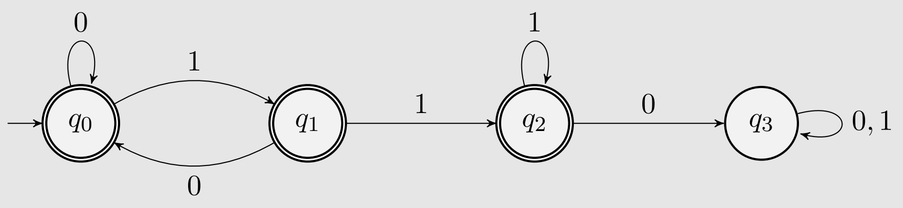

FIGURE 3.3. Accept all strings in $\{0, 1\}^*$ except those containing 110.

Problem 4.

Draw a state diagram of a DFA that recognizes the language consisting of all strings in $\{0,1\}^*$ such that each string is of length at least three and every block of three consecutive symbols has at least one 0.

(Thus, for example, 0011001 *is in the language, but* 0011100 *is not.)*

Explain, in one or two sentences, the idea behind your DFA construction

To make this problem easier to tackle, I split it into three incremental steps:

(i) Construct a DFA that accepts all strings of length at least 3 (figure 4.1).

(ii) Construct a DFA that accepts all strings with at least one 0 in each block of 3 letters (4.2, 4.3).

(iii) Combine the two DFAs (figure 4.4).

Part 1: DFA to accept all strings of length at least 3.

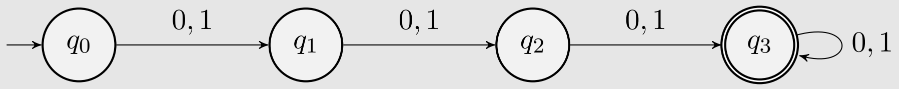

FIGURE 4.1. Accept all strings of length at least 3.

Part 2

DFA to accept all strings with at least one 0 in each block of 3 letters.

The simplest idea I had was to trap and reject once we encounter 111 since that is the only non-accepting 3-sequence block.

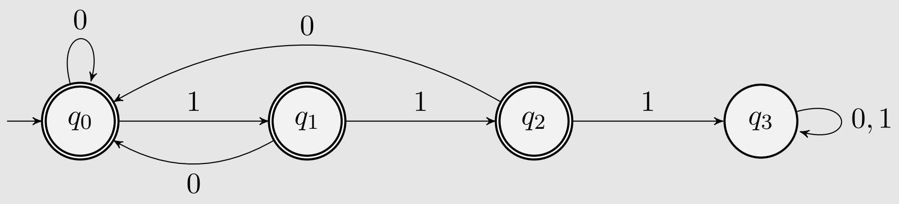

FIGURE 4.2. Accept all strings with at least one 0 in each block of 3.

While working on Q_6, I realized that the multiple-zeros scenario necessitates tracking the full state since we will not necessarily have a single reject state. So I redesigned this part of the DFA to track the full state.

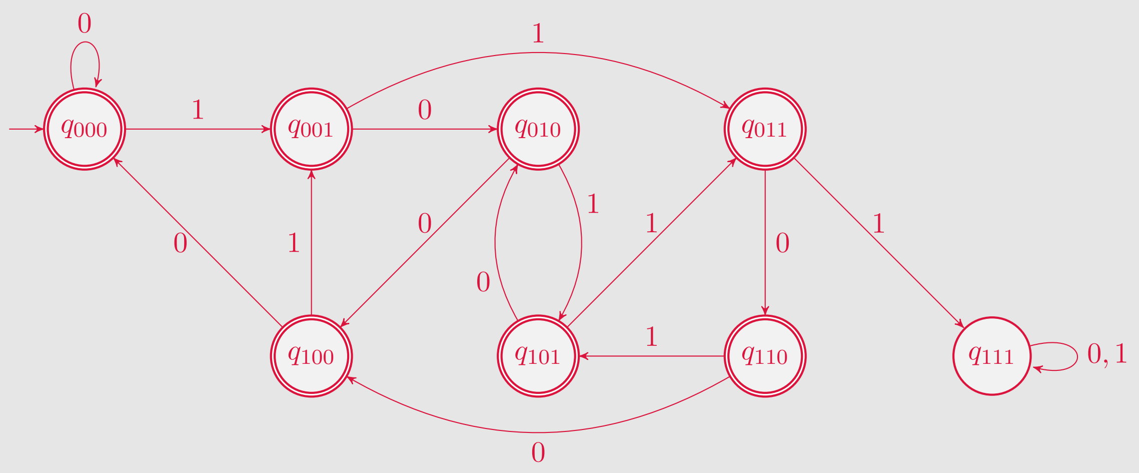

FIGURE 4.3. Accept all strings with at least one 0 in each block of 3.

Part 3: Combine DFA 4.1 and DFA 4.3.

I debated whether to use the simplified DFA (figure 4.2) or the full-state DFA (figure 4.3), eventually going with the full-state DFA as that would help when working on Q_6.

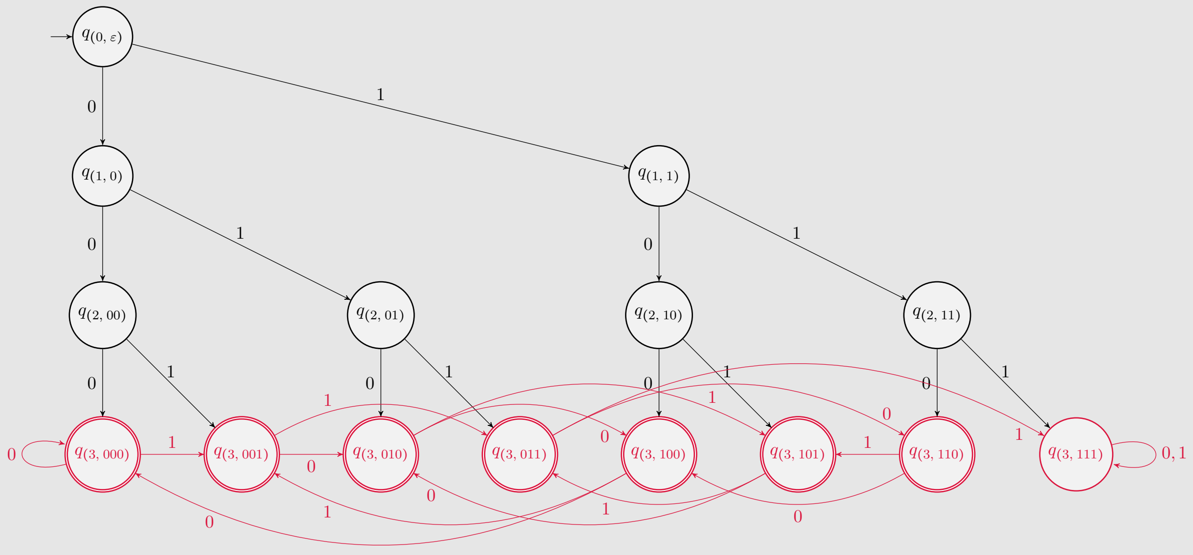

FIGURE 4.4. Accept all strings with length at least 3 and at least one 0 in each block of 3.

Main Idea:

We track states as a tuple of the length of the string processed so far (say, n) and the most recent n characters processed from the string. While n is less than 3, each transition appends the current character to the right of the string we are tracking and increments n. Once n reaches 3, we no longer need to keep incrementing it since the length requirement is satisfied. We also start watching for and trapping at invalid states. While transitioning in valid states, we drop the left-most character of the string we are tracking and append the new character to the right.

Finally, we accept if we have processed at least 3 characters (i.e. we are at a state $q_{(3,\, s)}$) and the latest 3 characters have at least one 0 (i.e. $\#_0(s) \geq 1$).

DFA Specification:

To simplify the expressions, we define the following functions:

$$\textsc{DropFirst}\,(s) = \begin{cases} \varepsilon & \text{if } s = \varepsilon \\ x & \text{otherwise, where } s \text{ is parsed as } \{0,1\}^1\, x \end{cases}$$

$$\beta(s) = n \text{ where } n \text{ is the integer obtained by interpreting } s \text{ as a binary number.}$$

$$\#(s) = n \text{ where } n \text{ is the length of } s.$$

$$\#_a(s) = n \text{ where } n \text{ is the number of occurrences of } a \text{ in } s.$$

We define the DFA as follows:

$$M = (Q,\ \Sigma,\ \delta,\ q_0,\ F) \text{ where:}$$

$$Q = \left\{ q_{(i,\, s)} : 0 \leq i \leq 3,\ \ s \in \{0,1\}^* \ \text{ and } \#(s) = i \right\}$$

$$\Sigma = \{0,1\}$$

$$\delta(q_{(i,\, s)}, x) = \begin{cases} q_{(i+1,\, sx)} & \text{if } i < 3 & \text{// Still too short.} \\ q_{(i,\, \textsc{DropFirst}(s)x)} & \text{if } i = 3 \text{ and } \#_0(s) \geq 1 & \text{// Valid length and condition.} \\ q_{(i,\, s)} & \text{otherwise.} & \text{// Invalid condition, trap in the state.} \end{cases}$$

$$q_0 = q_{(0,\, \varepsilon)}$$

$$F = \left\{ q_{(3,\, s)} : \#_0(s) \geq 1 \right\}$$

Problem 5.

For a string $x \in 0, 1^*$, let $\beta(x)$ denote the integer obtained by interpreting x as a binary number. Thus, for example, $\beta(11001) = 25$ and $\beta(0011) = 3$. We also define $\beta(\epsilon) = 0$ for convenience.

Design a DFA for the language $L = \left\{x \in \{0,1\}^* : \beta(x) \text{ is divisible by } 5\right\}$. *Draw the state diagram and also specify the DFA formally*. Hint: For integers m, n, p, we have $(mn + p) \mod 5 = ((m \mod 5) \cdot n + p) \mod 5$

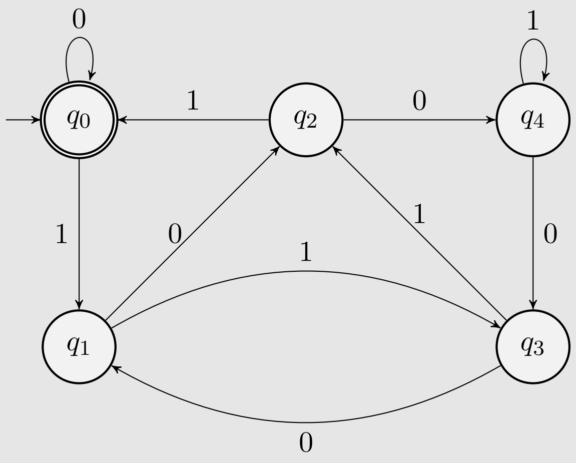

Figure 5.1. Accept all strings x where $\beta(x) \equiv 0 \pmod 5$.

DFA Specification

For convenience, we define the function

$$\beta(x) = n \text{ where } n \text{ is the value of } x \text{ interpreted as a binary number.}$$

We specify the DFA as follows:

$M = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$ where:

$Q = \{q_i : 0 \leq i < 5\}$

$\Sigma = \{0, 1\}$

$$\delta(q_i, x) = q_j \text{ where } j = ((2i + \beta(x)) \mod 5) = \begin{cases} q_0 & \text{if } i = 0 \text{ and } x = 0 \\ q_1 & \text{if } i = 0 \text{ and } x = 1 \\ q_2 & \text{if } i = 1 \text{ and } x = 0 \\ q_3 & \text{if } i = 1 \text{ and } x = 1 \\ q_4 & \text{if } i = 2 \text{ and } x = 0 \\ q_0 & \text{if } i = 2 \text{ and } x = 1 \\ q_1 & \text{if } i = 3 \text{ and } x = 0 \\ q_2 & \text{if } i = 3 \text{ and } x = 1 \\ q_3 & \text{if } i = 4 \text{ and } x = 0 \\ q_4 & \text{if } i = 4 \text{ and } x = 1 \end{cases}$$

$q_0 = q_0$

$F = \{q_0\}$

Problem 6.

Formally specify a DFA for the set of all strings in $\Sigma = \{0,1\}^*$ such that each string is of length at least 2000 and every block of 2000 consecutive symbols has at least ten 0s. Explain the design in one or two sentences.

DFA Specification

To simplify the expressions, we define the following functions:

$$\textsc{DropFirst}(s) = \begin{cases} \varepsilon & \text{if } s = \varepsilon \\ x & \text{otherwise, where } s \text{ is parsed as } \{0,1\}^1 x \end{cases}$$

$$\beta(s) = n \text{ where } n \text{ is the integer obtained by interpreting } s \text{ as a binary number.}$$

$$\#(s) = n \text{ where } n \text{ is the length of } s.$$

$$\#_a(s) = n \text{ where } n \text{ is the number of occurrences of } a \text{ in } s.$$

We specify the DFA as:

$$M = (Q,\ \Sigma,\ \delta,\ q_0,\ F) \text{ where:}$$

$$Q = \left\{ q_{(i,\ s)} : 0 \le i \le 2000,\ \ s \in \{0,1\}^* \text{ and } \#(s) = i \right\}$$

$$\Sigma = \{0,1\}$$

$$\delta(q_{(i,\ s)}, x) = \begin{cases} q_{(i+1,\ sx)} & \text{if } i < 2000 & \text{// Still too short.} \\ q_{(i,\ \textsc{DropFirst}(s)x)} & \text{if } i = 2000 \text{ and } \#_0(s) \ge 10 & \text{// Valid length and condition.} \\ q_{(i,\ s)} & \text{otherwise.} & \text{// Invalid condition, trap in the state.} \end{cases}$$

$$q_0 = q_{(0,\ \varepsilon)}$$

$$F = \left\{ q_{(2000,\ s)} : \#_0(s) \ge 10 \right\}$$

Main Idea:

We track states as a tuple of the length of the string processed so far (say, n) and the most recent n characters processed from the string. While n is less than 2000, each transition appends the current character to the right of the string we are tracking and increments n. Once n reaches 2000, we no longer need to keep incrementing it since the length requirement is satisfied. We also start watching for and trapping at invalid states. While transitioning in valid states, we drop the left-most character of the string we are tracking and append the new character to the right.

Finally, we accept if we have processed at least 2000 characters (i.e. we are at a state $q_{(2000,\, s)}$) and the latest 2000 characters have at least ten 0 (i.e. $\#_0(s) \geq 10$).