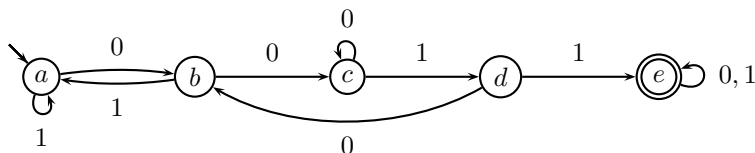
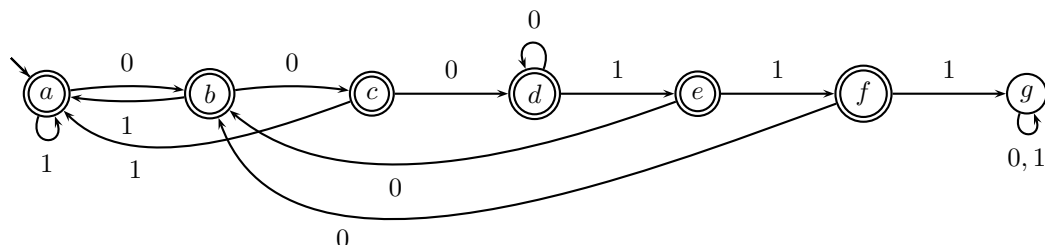


1 Deterministic Finite Automata

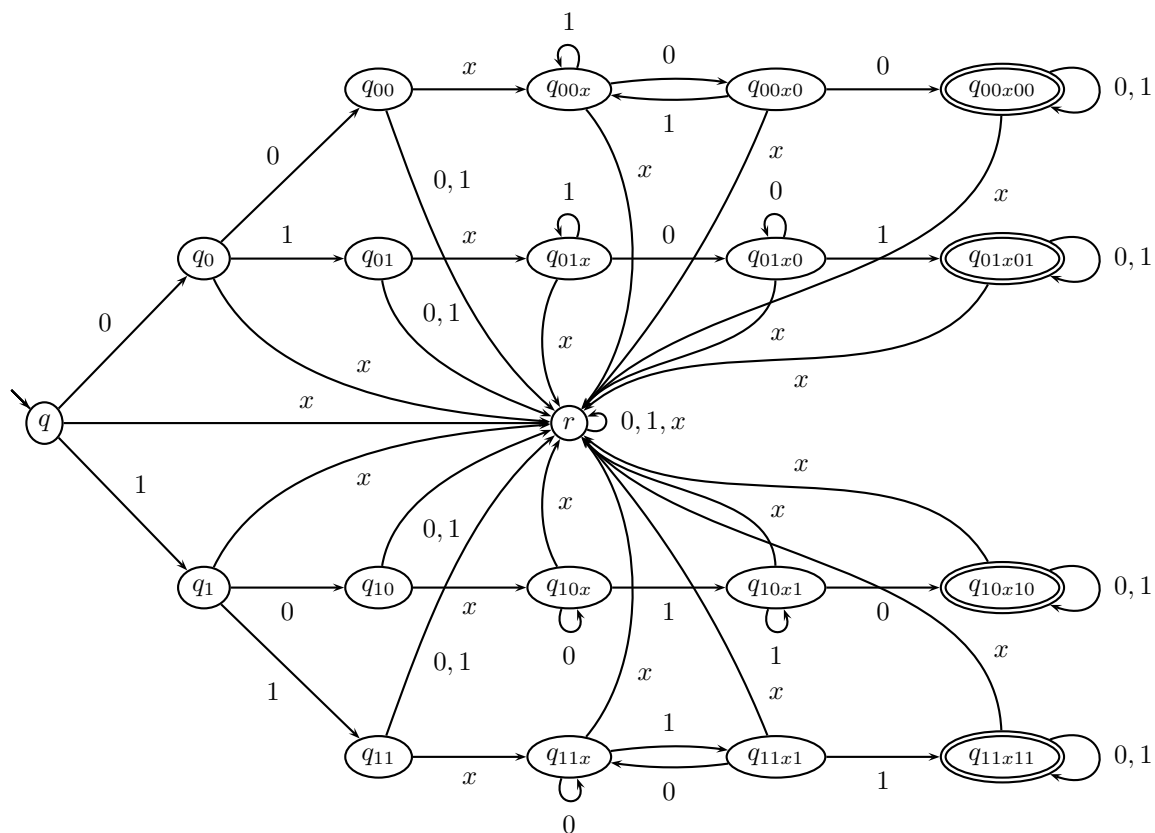
- (a) States a through e correspond to having seen 0 through 4 consecutive symbols of the string 0011. Once we reach state e , we stay there and accept.



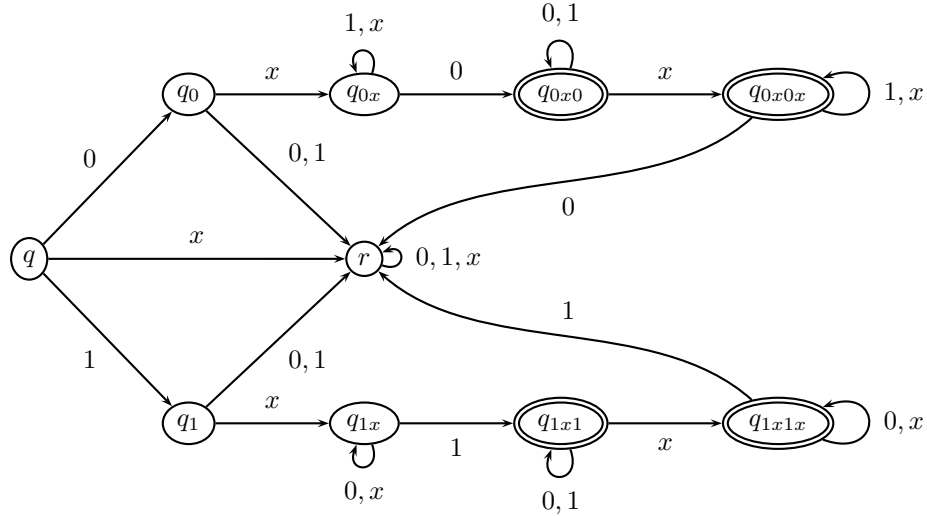
- (b) The idea for this DFA is exactly the same as the idea for (a), except we swap which states are accepting.



- (c) The four cases of $U = 00, 01, 10$, and 11 are considered independently. In each case, the DFA looks for U and then an x and then a string containing the U that was matched. The 4 “lines” of the DFA (other than the middle line) correspond to the four cases for U .



- (d) Like in part (c), we split on the possible cases for U . Let $\bar{U}$ be the bit not equal to U . However here for each case we loop over x and $\bar{U}$ until we find the V_i containing U . In the V_i containing U , we allow any combination of 0 and 1. After that, we allow any combination of $\bar{U}$ and x .



2 Closure of Regular Languages

For any regular language A , there exists a DFA $D = (Q, \Sigma, \delta, q_0, F)$ such that $L(D) = A$. Given D , we will construct a NFA N such that $L(N) = \text{even}(A)$. Since N can be transformed into an equivalent DFA, this will prove that $\text{even}(A)$ is regular.

Intuitively, we will construct N in two phases. First, we will transform D into an equivalent DFA D' with the property that we can partition the states of D' into two groups, even and odd, where we can only be in a state in the even group if we've read an even number of symbols and we can only be in a state in the odd group if we have read an odd number of symbols. Now we will replace every transition that goes from a state in the even group to a state in the odd group with an ϵ transition. We can think of this as skipping the odd symbols when reading a string, yet transitioning anyway.

Formally, we define $N = (Q \times \{\mathbf{E}, \mathbf{O}\}, \Sigma, \delta', (q_0, \mathbf{E}), F \times \{\mathbf{E}, \mathbf{O}\})$ and

$$\delta' : (Q \times \{\mathbf{E}, \mathbf{O}\}) \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q \times \{\mathbf{E}, \mathbf{O}\})$$

$$\delta'((q, p), \sigma) = \begin{cases} \{(\delta(q, x), \mathbf{O}) \mid x \in \Sigma\} & \text{if } p = \mathbf{E} \text{ and } \sigma = \epsilon \\ \{(\delta(q, \sigma), \mathbf{E})\} & \text{if } p = \mathbf{O} \text{ and } \sigma \neq \epsilon \\ \emptyset & \text{otherwise} \end{cases}$$

For any string $s = s_1 \dots s_k \in A$, $\text{even}(s) = s_2 s_4 s_6 \dots s_{\lfloor k/2 \rfloor} = \epsilon s_2 \epsilon s_4 \epsilon s_6 \dots s_{\lfloor k/2 \rfloor} \epsilon \in \text{even}(A)$. The NFA begins in state $(q_0, \mathbf{E})$. States of the form $(x, \mathbf{E})$ have only ϵ transitions leaving them going to states of the form $(y, \mathbf{O})$. States of the form $(y, \mathbf{O})$ do not have any ϵ transitions, but do have transitions for every symbol in Σ to states of the form $(x, \mathbf{E})$. Thus, every run of this NFA first takes an ϵ transition, then a non- ϵ transition, then an ϵ transition, and so forth. The ϵ transitions are those that the original DFA would have taken on odd positioned symbols (which get left out by $\text{even}(\cdot)$). The non- ϵ transitions are those that the original DFA would have taken on even positioned symbols. Thus, the NFA reads all the even positioned symbols and skips the odd positioned symbols.

3 Modeling Computation

- A Read Only Automata (ROA) is a 5-tuple $M = (Q, \Sigma, \delta, q_0, q_a)$ where Q and Σ are finite sets called the set of states and the input alphabet respectively, $q_0, q_a \in Q$ are called the start state and accepting state respectively, and $\delta : Q \times \Sigma_{[]} \rightarrow Q \times \{L, R\}$ is called the transition function. Here, $\Sigma_{[]}$ denotes $\Sigma \cup \{[,]\}$.
- The set of configurations is $C = \Sigma_{[]}^* \times Q \times \Sigma_{[]}^*$. Each configuration $(s, q, t) \in C$ consists of the part of the string before the current position (including the initial $[$) $s \in \Sigma_{[]}^*$, the current internal state $q \in Q$, and the part of the string from the current position to the end (including the final $]$) $t \in \Sigma_{[]}^*$.
- The initial configuration on input w is $([, q_0, w])$ where q_0 is the start state of the automata. The system configuration changes at each step are given by the relation

$$\begin{aligned}
 R = & \{((s, q, at), (sa, q', t)) \mid s, t \in \Sigma_{[]}^*, q, q' \in Q, a \in \Sigma, \delta(q, a) = (q', R)\} \\
 & \cup \{((sb, q, at), (s, q', bat)) \mid s, t \in \Sigma_{[]}^*, q, q' \in Q, a, b \in \Sigma, \delta(q, a) = (q', L)\} \\
 & \cup \{((\epsilon, q, [t), (\epsilon, q', [t)) \mid t \in \Sigma_{[]}^*, q, q' \in Q, \delta(q, [) = (q', L)\} \\
 & \cup \{((s, q,]), (s, q',])) \mid s \in \Sigma_{[]}^*, q, q' \in Q, \delta(q,] = (q', R)\}
 \end{aligned}$$

- The set of string accepted by the automaton is the set of all $w \in \Sigma^*$ such that there is an accepting computation of M on input W , i.e., a sequence of configurations $c_0, c_1, \dots, c_n$ such that $c_0 = ([, q_0, w])$ is the initial configuration, $(c_i, c_{i+1}) \in R$ for all $i = 0, \dots, n-1$, and $c_n = (s, q_a, t)$ for some $s, t \in \Sigma_{[]}^*$.