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 $\overline{U}$ be the bit not equal to $U$. However here for each case we loop over $x$ and $\overline{U}$ 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 $\overline{U}$ and $x$.

$$
\begin{align*}
&1, x \\
&x \\
&q_0 \quad \rightarrow \quad q_{0x} \\
&0 \\
&0, 1 \\
&x \\
&\vdots
\end{align*}
$$

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 $L(N) = even(A)$. Since $N$ can be transformed into an equivalent DFA, this will prove that $even(A)$ 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 $\epsilon$ transition. We can think of this as skipping the odd symbols when reading a string, yet transitioning anyway.

Formally, we define $N = (Q \times \{E, O\}, \Sigma, \delta', (q_0, E), F \times \{E, 0\})$ and

$$
\delta' : (Q \times \{E, O\}) \times \Sigma \rightarrow P(Q \times \{E, 0\})
$$

$$
\delta'(q, \sigma) = \begin{cases} 
\{\delta(q, x), 0\} & | x \in \Sigma \text{ if } p = E \text{ and } \sigma = \epsilon \\
\{\delta(q, \sigma), E\} & | p = 0 \text{ and } \sigma \neq \epsilon \\
\emptyset & \text{otherwise}
\end{cases}
$$

For any string $s = s_1 \ldots s_k \in A$, $even(s) = s_2s_4s_6 \ldots s_{k-1}/2 = \epsilon s_2s_4s_6 \ldots s_{k-1}/2 \epsilon \in even(A)$. The NFA begins in state $(q_0, E)$. States of the form $(x, E)$ have only $\epsilon$ transitions leaving them going to states of the form $(y, 0)$. States of the form $(y, 0)$ do not have any $\epsilon$ transitions, but do have transitions for every symbol in $\Sigma$ to states of the form $(x, E)$. Thus, every run of this NFA first takes an $\epsilon$ transition, then a non-$\epsilon$ transition, then an $\epsilon$ transition, and so forth. The $\epsilon$ transitions are those that the original DFA would have taken on odd positioned symbols (which get left out by $even(\cdot)$). The non-$\epsilon$ 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 \( \Sigma \) 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 \([\] \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 \([\] \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
  \[
  R = \{(s, q, at), (sa, q', t)) \mid s, t \in \Sigma^*, q, q' \in Q, a \in \Sigma, \delta(q, a) = (q', R)\}
  \cup \{(sh, q, at), (s, q', bat)) \mid s, t \in \Sigma^*, q, q' \in Q, a, b \in \Sigma, \delta(q, a) = (q', L)\}
  \cup \{(epsilon, q, [t]), (epsilon, q', [t])) \mid t \in \Sigma^*, q, q' \in Q, \delta(q, [\] = (q', L)\}
  \cup \{(s, q, []), (s, q', [)) \mid s \in \Sigma^*, q, q' \in Q, \delta(q, [\] = (q', R)\}

- 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 \( c_0, c_1, \ldots, c_n \) such that \( c_0 = ([, q_0, w] \) is the initial configuration, \( (c_i, c_{i+1}) \in R \) for all \( i = 0, \ldots, n - 1 \), and \( c_n = (s, q_a, t) \) for some \( s, t \in \Sigma^*\).