Deterministic Finite Automata

For each of the following languages, design a corresponding Deterministic Finite State Automaton. For each language, your solution should include a brief English description of the main ideas behind your construction, and a jflap file implementing the automaton. You should design your solution directly as a deterministic automaton, i.e., you are not allowed to use extended automata features or regular expressions provided by JFLAP, and then let JFLAP convert it into a deterministic automaton.

(a) The set of all binary strings \( w \in \{0, 1\}^* \) that contain the pattern 0011. For example, the strings 0011, 101010011101 are in the language, while \( \epsilon, 101011 \) are not in the language.

(b) The set of all binary strings \( w \in \{0, 1\}^* \) that do not contain the pattern 000111. For example, \( \epsilon, 0011, 111000 \) are in the language, while 100011101, 000111 are not.

(c) The set of all strings over the alphabet \( \{0, 1, x\} \) of the form \( w = UxV \) where \( U \in \{0, 1\}^2 \) is a binary string of length 2, and \( V \in \{0, 1\}^* \) is a binary string that contains the pattern \( U \). For example 01x0010, 11x11 are in the language, while 01010, 111x111, 00x11011, 00x00x are not.

(d) The set of all strings over the alphabet \( \{0, 1, x\} \) of the form \( UxV_1xV_2x \ldots xV_n \) such that \( U \in \{0, 1\} \) is a single bit, \( V_i \in \{0, 1\}^* \) are binary strings, and there exists exactly one \( V_i \) which contains the symbol \( U \). For example, 1x000x010x, 0x0x1x1x11 are in the language, while 00x00, 0x0x0, 1x00x00 are not.

Closure of Regular Languages

For any string \( w \in \Sigma^* \) of length \( n = |w| \), define \( \text{even}(w) = w_2w_4w_6 \ldots w_{\lfloor n/2 \rfloor} \) to be the string obtained by taking only the symbols of \( w \) that occur at even positions. For example, \( \text{even}(120390j1) = 209j \). This operation is extended to languages by applying it to each word in the language, i.e., \( \text{even}(L) = \{ \text{even}(w) : w \in L \} \). Prove that if \( A \) is a regular language, then \( \text{even}(A) \) is also a regular language. Your solution will be evaluated both for correctness and clarity. Your solution should contain all of the following:

1. An informal English description of the main ideas behind your solution
2. A mathematical description of all components of automaton \( M' \) (See hint below.)
3. A proof that \( L(M') = \text{even}(L(M)) \)
Hint: Your solution may start as follows: Let $L$ be an arbitrary regular language. By definition of regular language, there exists a deterministic finite state automaton $M = (Q, \Sigma, \delta, s, F)$ such that $L(M) = A$. We use the components of $M$ to build a new non-deterministic finite state automaton $M' = (Q', \Sigma', \delta', s', F')$ with $\epsilon$-transitions such that $L(M') = \text{even}(A)$ . . . . . . . Since $M'$ can be transformed into an equivalent deterministic finite automaton using the method studied in class, the language $\text{even}(A)$ is regular.

3 Modeling computation

In this problem you are asked to formally define a mathematical model of computation (which we may call “Read Only Automata”) along the lines followed in class when defining the deterministic finite state automaton. The new automaton takes as input a string $w \in \Sigma^*$, and can read the symbols of $w$ one at a time, but the symbols don’t get consumed as they are read. The automaton can move freely over the input, reading the same characters back and forth as many times as it wishes. At each step, the automaton can move either one symbol forward or one symbol backward on the input. The automaton accepts the input by entering a special state $q_a$ and halting. The input string $w \in \Sigma^*$ is given to the automaton enclosed within two special symbols $[ \text{ and } ]$ (not in $\Sigma$), so that the automaton can tell when it reaches the input boundaries. The automaton should be deterministic, i.e., there should be a unique computation that can be carried out by the automaton for each possible input.

The informal description given above may leave some details open to interpretation. You can resolve ambiguities in the informal description any way you like. Your solution should include:

1. A definition of Read Only Automata as mathematical objects, which specifies the type of all components of the automaton.
2. A definition of the set of configurations necessary to model the computation performed by the new automata, and the initial configuration of the automaton corresponding to a given input string $w \in \Sigma$.
3. A definition of the “next-configuration” relation $R$ that specifies how the system evolves over time.
4. A definition of the sets of strings accepted and rejected by the automaton.$^1$

As a reminder (and guideline), here is how we gave analogous definitions in class for the deterministic finite state automaton.

- A deterministic finite state automaton is a 5-tuple $M = (Q, \Sigma, \delta, s, F)$ where $Q, \Sigma$ are finite sets (called the set of states, and input alphabet), $s \in Q$ is the start state, $F \subseteq Q$ is a set of accepting states, and $\delta : Q \times \Sigma \to Q$ is a transition function.
- The set of configurations is $C = Q \times \Sigma^*$. Each configuration $(q, w) \in C$ consists of the current internal state of the automaton $q \in Q$, and the part of the input string $w$ which has not been read yet.
- Initial configuration on input $w$ is $(s, w)$, where $s$ is the start state of the automaton. The system configuration changes at each step as specified by the relation

  $$R = \{(q, aw), (\delta(q, a), w) : q \in Q, a \in \Sigma, w \in \Sigma^*\}$$

- The set of strings 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 $c_0, \ldots, c_n$ such that $c_0 = (s, w)$ is the initial configuration, $(c_i, c_{i+1}) \in R$ for all $i = 0, \ldots, n - 1$, and $c_n = (q, \epsilon)$ for some $q \in F$.

$^1$Notice that on certain inputs the automaton may loop, so there are strings that are neither accepted or rejected.