Assignment 6

Unmarked Questions

Remember to comment out this section when submitting your assignment.

Here are a few simple warm-up problems. Make sure you are able to do them before proceeding to the marked questions.

Determine regularity of the following languages

For each of the following languages, state whether or not it is regular and prove your answer. If the language is regular, draw a finite automaton (DFA or NFA) which accepts the language using *no more than 10 states* and proof that it accepts the language (during an exam, you *will not* have to do this). Otherwise, prove that the language *is not* regular.

a. $L_1 = \{w \in \{0,1\}^* : w \text{ contains both 01 and 00 as substrings}\}$. For example, the strings $001, 010100, 000111 \in L_1$, while the strings $0, 1111, 010101 \notin L_1$.

   Solution. L_1 is regular. We will draw a DFA which accepts L_1. It is shown in Figure 1.

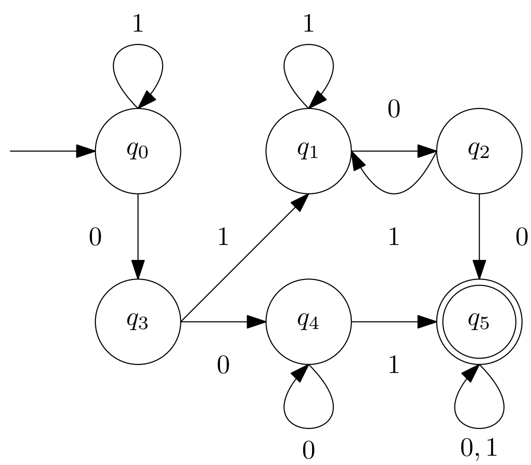


Figure 1: DFA M which accepts L_1.

Proof. To prove that M indeed accepts L, we will come up with and prove a state invariant for each of the six states. The predicate on a string $w \in \{0,1\}^*$ will be

$$P(w) := \delta^*(q_0, w) = \begin{cases} q_0 & w = 1^k \text{ for } k \geq 0 \\ q_1 & w = 1^k01(1^\ell + (01)^m)^n \text{ for } k, \ell, m, n \geq 0 \\ q_2 & w = 1^k01(1^\ell + (01)^m)^n0 \text{ for } k, \ell, m, n \geq 0 \\ q_3 & w = 1^k0 \text{ for } k \geq 0 \\ q_4 & w = 1^k00(0)^\ell \text{ for } k, \ell \geq 0 \\ q_5 & w \text{ constains both the string 01 and 00 as substrings} \end{cases} \tag{1}$$

We note that these six states are disjoint, i.e., no string can belong to two different states, and exhaustive, i.e., every string will end up at some state.

We prove $P(w)$ by structural induction with our standard recursive definition of Σ^*. In the base case $\delta^*(q_0, \epsilon) = q_0$ and indeed ϵ can be written as 1^k with $k = 0$.

Suppose for some string $w \in \Sigma^*$, $P(w)$ is true. We show that $P(w0)$ and $P(w1)$ are also true. For $w0$, we need to consider each of six possible cases for $\delta^*(q_0, w0)$.

$\delta^*(q_0, w) = q_0$ We know that $w = 1^k$ for $k \geq 0$ (IH). On a zero, δ, will take the transition to q_3 so $\delta^*(q_0, w0) = q_3$. Note that the invariant for q_3 is strings of the form 1^k0 for $k \geq 0$.

$\delta^*(q_0, w) = q_1$ We know that $w = 1^k01\left(1^\ell + (01)^m\right)^n$ for $k, \ell, m, n \geq 0$ (IH). On a zero, δ, will take the transition to q_2 so $\delta^*(q_0, w0) = q_2$. Note that the invariant for q_2 is exactly strings of the form $1^k01(1^\ell + (01)^m)^n0$ for $k, \ell, m, n \geq 0$.

$\delta^*(q_0, w) = q_2$ We know that $w = 1^k01\left(1^\ell + (01)^m\right)^n 0$ for $k, \ell, m, n \geq 0$ (IH). On a zero, δ, will take the transition to q_5 so $\delta^*(q_0, w0) = q_5$. Note that the invariant for q_5 is exactly strings containing both 01 and 00 as substrings and by adding a zero we guarantee that $w0$ now contains 00 as a substring (it already had a 01 substring).

$\delta^*(q_0, w) = q_3$ We know that $w = 1^k0$ for $k \geq 0$ (IH). On a zero, δ, will take the transition to q_4 so $\delta^*(q_0, w0) = q_4$. Note that the invariant for q_4 is exactly strings of the form $1^k00(0)^\ell$ for $k, \ell \geq 0$. Here, we could take $\ell = 0$.

$\delta^*(q_0, w) = q_4$ We know that $w = 1^k00(0)^\ell$ for $k, \ell \geq 0$ (IH). On a zero, δ, will take the transition to q_4 so $\delta^*(q_0, w0) = q_4$. Note that the invariant for q_4 is still satisfied as we now have $1^k00(0)^{\ell+1}$ for $k, \ell \geq 0$.

$\delta^*(q_0, w) = q_5$ We know that w is a string with both 01 and 00 as substrings (IH). It follows that $w0$ also satisfies this state invariant as well.

We will also need do the same for $w1$, but it is a bit tedious so we won't write it out explicitly. By Structural induction, $P(w)$ is true for all $w \in \Sigma^*$ and our DFA *indeed* accepts L_1. $\square$

b. $L_2 = \{0^n : n \text{ is a perfect square}\}$. For example, the strings $\epsilon, 0, 0000$ are in L_2, but the strings $00, 000, 00000$ are not.

Solution. L_2 is not regular, suppose for a contradiction that L_2 is regular and let M be a DFA which accepts L_2. Suppose that the M has pumping length p. Consider string $w = 0^{(p+1)^2}$. Since $w \in L_2$, by the Pumping Lemma, we know that $w = xyz$ where $|xy| \leq p$, $|y| \geq 1$, and $xy^2z \in L_2$. We claim that this is a contradiction since $xy^2z \notin L_2$. Since w consists of only zeros, $y = 0^a$ for some constant $a \leq p$. Note that $xy^2z = x0^{2a}z$. We know that $|x0^{2a}z| > (p+1)^2$ since the right hand side is the length of w. Note however $|x0^{2a}z| = (p+1)^2 + a < (p+1)^2 + 2(p+1) + 1 = (p+2)^2$. Since there are no perfect squares greater than $(p+1)^2$ and less than $(p+2)^2$, $x0^{2a}z \notin L_2$.

Marked Questions

Q1. [10 Points] Construct DFA *(maximum 3 pages)*

Consider the marble rolling toy shown in Figure 2. A marble is dropped at A or B. There are flippers at positions x_1, x_2, and x_3 (indicated by the thick black lines) which fall either to the left or to the right. Whenever a marble encounters a flipper, it causes the flipper to reverse; the next marble will take the opposite branch.

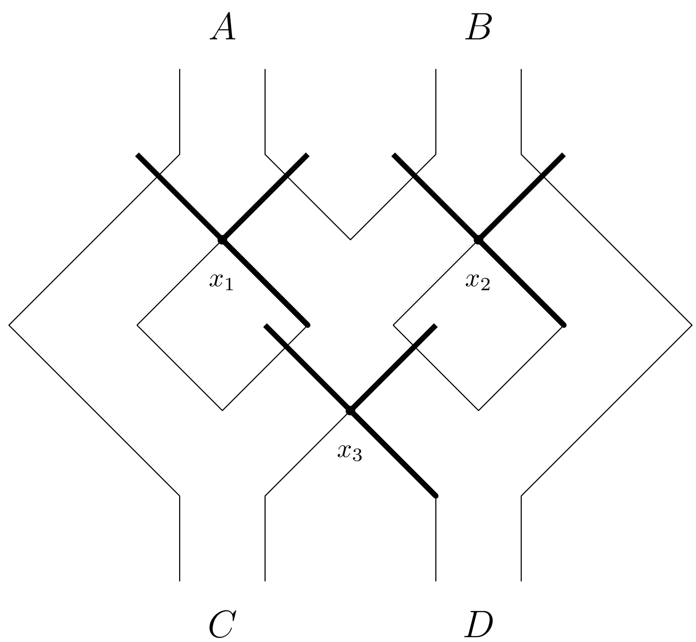


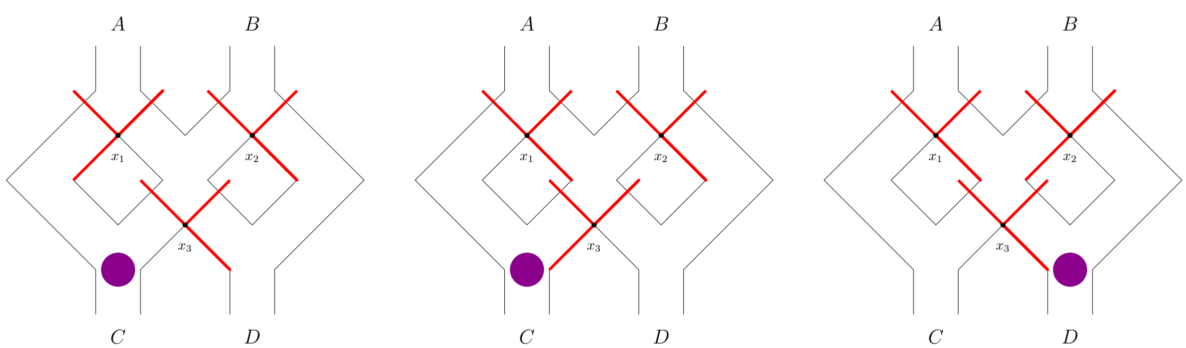
Figure 2: Default configuration of the marble rolling toy. Marbles are dropped in one of A or B and come out at one of C or D.

Model the toy as a DFA $M = (Q, \Sigma, \delta, s, F)$ with alphabet $\Sigma = \{A, B\}$. Inputs to the DFA are sequences of As and Bs indicating where the marbles will be dropped. M should *accept* a string of As and Bs if the *last* marble dropped exits the toy through D. Accept the empty string.

An example of the state of the machine after dropping marbles consecutively at A, A, and then B is shown in Figure 3.

a. Draw your DFA below. *You should not need more than sixteen states.*

Solution.



(a) Drop first marble at A. (b) Drop second marble at A. (c) Drop third marble at B.

Figure 3: Result of dropping three marbles at A, then A, then B. This corresponds to the three strings A, AA, and AAB. Your DFA should reject A and AA but accept AAB.

b. Formally prove that your DFA is correct. *In your inductive case, you only need to consider the addition of A to the end of a string w such that $P(w)$ is true for your predicate P.*

Proof. □

Q2. [10 Points] Regex Implies NFA *(maximum 2 pages not counting the diagrams for Regex to NFA solutions)*

In the following question we will complete the proof of the equivalence of regular expressions (Regex), deterministic finite automaton (DFA), and non-deterministic finite automaton (NFA). In class, we showed that DFAs have the same expressive power as NFAs. In this problem, we will first turn a DFA M into a regex R such that the language accepted by M, denoted $\mathcal{L}(M)$, is equivalent to the language represented by R, denoted $\mathcal{L}(R)$, i.e., we want to find a regular expression R which is *equivalent* to M. Then we will show that a regex R can be transformed into a NFA N such that $\mathcal{L}(R) = \mathcal{L}(N)$, i.e., we want to find an NFA N which is *equivalent* to R.

DFA→Regex We will walk through the *state removal method*. The goal is to remove states until there are only two remaining, the start state and a single accepting state with a single transition between them. We do this by replacing single character transitions with regular expressions.

(a) Let R be a regex and suppose the DFA M is as shown in Figure 4. Find a regex which is equivalent to M.

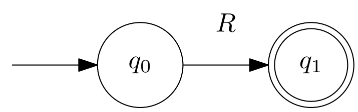


Figure 4: DFA M.

Solution.

(b) Suppose instead that M has many more states and let a part of M be shown on the left of Figure 5. We want to eliminate state q_{i+1} so that M looks like the right of Figure 5 afterwards. Find a regex to put on the transition $q_i \to q_{i+2}$ using R_1, R_2, and R_3 so

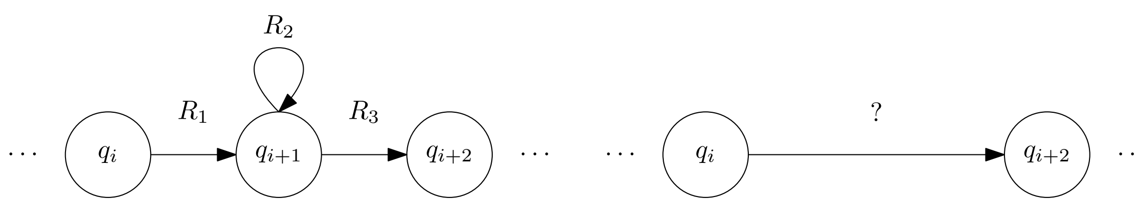


Figure 5: Part of DFA M. Find the regex which fits in ?.

that we can eliminate q_{i+1}.

Solution.

(c) Consider a part of M shown on left of Figure 6. We want to collate all transitions between states q_j and q_{j+1} so that M looks like the right of Figure 6 afterwards. Each R_i is a regex. Find a regex to put on the single transition $q_j \to q_{j+1}$ using R_1, ..., and R_k so that we can eliminate all other transitions.

Solution.

(d) Using the previous parts, prove the predicate $P(n) :=$ every DFA with n states has an equivalent regex. Inducting on the number of states. For n, fix some DFA M with n states and show that there exists a regex which is equivalent to M.

Hint: You may want to add a dummy start state and a dummy accept state with ϵ-transitions to the start state and from the accepting states respectively.

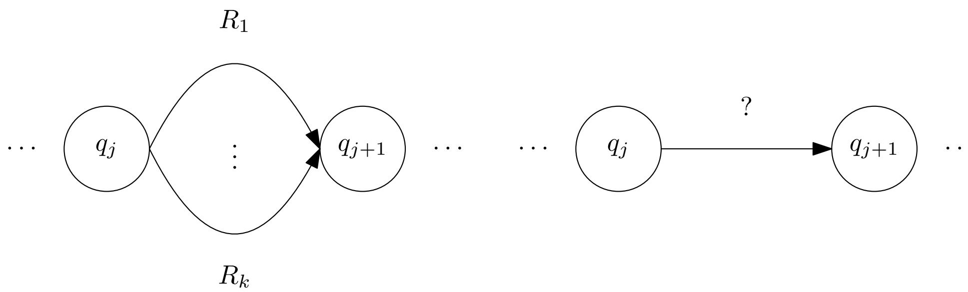


Figure 6: Part of DFA M. Find the regex which fits in ?.

Proof. □

Regex→NFA Recall that regexs are defined recursively on the set of characters Σ. For each part of the recursive definition of regexs, we construct an equivalent NFA. *A diagram is enough; you do not have to prove the correctness of the NFA.*

(a) Draw an NFA which is equivalent to the regular expression $\emptyset$.
Solution.

(b) Draw an NFA which is equivalent to the regular expression ϵ.
Solution.

(c) For a fixed $a \in \Sigma$, draw an NFA which is equivalent to the regular expression a.
Solution.

(d) Suppose we have two regular expressions R_1 and R_2 with corresponding DFAs M_1 and M_2. Use M_1 and M_2 to construct an NFA which is equivalent to $R_1 + R_2$.
Solution.

(e) Suppose we have two regular expressions R_1 and R_2 with corresponding DFAs M_1 and M_2. Use M_1 and M_2 to construct an NFA which is equivalent to $R_1 R_2$.
Solution.

(f) Suppose we have a regular expression R with corresponding DFA M. Use M to construct an NFA which is equivalent to R^*.
Solution.

Additional Questions

Remember to comment out this section when submitting your assignment.

If you would like more exercises consider trying the following problems from your primary and supplementary textbooks. *We will not be providing solutions to these questions* though you are free to find the solution online and discuss them with your peers.

1. David Liu's notes Chapter 5: Exercises 8, 9
2. Hopcroft, Motwani, Ullman Chapter 2.3: Exercises 1, 2, 3, Chapter 4.1: Exercises 2, 3, Chapter 4.2: Exercises 2, 3, 6