Assignment 6 (SOLUTIONS)

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

Consider the marble rolling toy shown in Figure 1. 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.

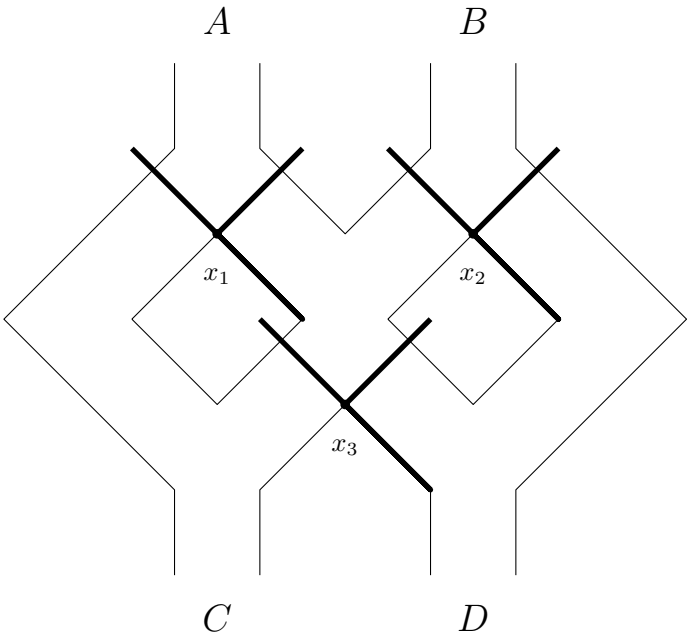


Figure 1: Default configuration of the marble rolling toy. Marbles are dropped in one of A or B and come out at one of C or D.

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.

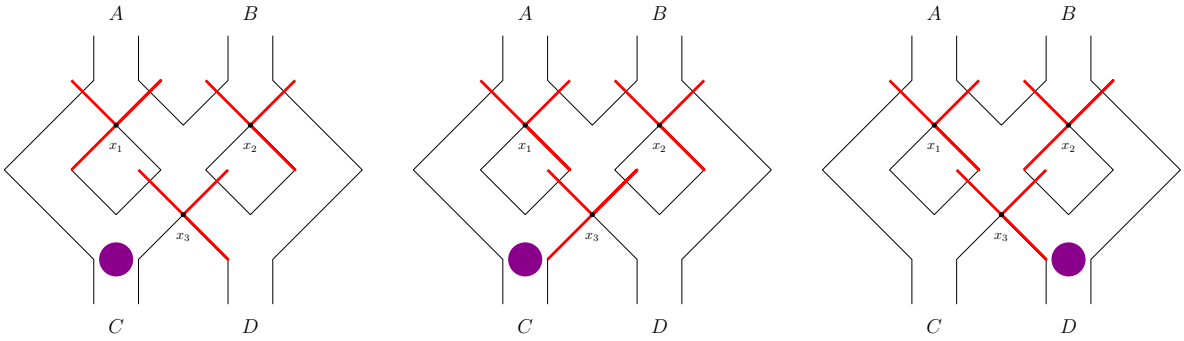
An example of the state of the machine after dropping marbles consecutively at A, A, and then B is shown in Figure 2.

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

Solution. Figure 3 shows the DFA which accepts the marble rolling toy language. Note that each state is labelled by $q_{x_1x_2x_3}$ which represents the direction the ball will fall if it hits each of the three flippers. Sometimes it is possible that a sequence of balls will result in the same state but fall through different exits. In these cases the states are labelled by $q_{x_1x_2x_3s}$ where $s \in \{c, d\}$ denotes the different exit that the final ball takes. The start state is q_{LLL}.

[6 Marks] Deduct 0.5 marks for each missing or incorrect state up to a total of 6 marks.

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.*



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

Figure 2: 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.

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

$$P(w) \coloneqq \delta^*(q_0, w) = q_{abcs}$$

for all $a, b, c \in \{L, R\}$ and $s \in \{c, d\}$ if it exists where the final gate state is $x_1 = a$, $x_2 = b$, $x_3 = c$ and the final ball fall through s if present. We note that these eight states are disjoint and exhaustive as every setting of the flippers are considered and putting a ball in either A or B will result in the flippers ending up in one of the stated configurations and the ball can only fall through one of two exists.

We prove $P(w)$ by structural induction with our standard recursive definition of Σ^*. In the base case $\delta^*(q_{LLL}, \epsilon) = q_{LLL}$ and indeed in the initial configuration all balls will fall to the left of a flipper. Suppose for some string $w \in \Sigma^*$, $P(w)$ is true. We show that $P(wA)$ is also true (the process for showing $P(wB)$ is true is similar). For wA, we need to consider each of twelve possible cases for $\delta^*(q_{LLL}, wA)$ (though only eight are unique).

(a) $\delta^*(q_{LLL}, w) = q_{LLL}$. By IH, we have that the flipper states are $x_1 = L$, $x_2 = L$, and $x_3 = L$. Note that if we drop another ball into A, then the flippers will be in the state $x_1 = R$, $x_2 = L$, $x_3 = L$, and the ball will exist through C which is exactly q_{RLLc}.

(b) $\delta^*(q_{LLL}, w) = q_{RLLc}$ and $\delta^*(q_{LLL}, wA) = q_{RLLd}$. By IH, we have that the flipper states are $x_1 = R$, $x_2 = L$, and $x_3 = L$ regardless of where the last ball exited. Note that if we drop another ball into A, then the flippers will be in the state $x_1 = L$, $x_2 = L$, $x_3 = R$, and the ball will exist through C which is exactly q_{LLRc}.

(c) $\delta^*(q_{LLL}, w) = q_{LRL}$. By IH, we have that the flipper states are $x_1 = L$, $x_2 = R$, and $x_3 = L$. Note that if we drop another ball into A, then the flippers will be in the state $x_1 = R$, $x_2 = R$, $x_3 = L$, and the ball will exist through C which is exactly q_{RRLc}.

(d) $\delta^*(q_{LLL}, w) = q_{LLRc}$ and $\delta^*(q_{LLL}, wA) = q_{LLRd}$. By IH, we have that the flipper states are $x_1 = L$, $x_2 = L$, and $x_3 = R$ regardless of where the last ball exited. Note that if we drop another ball into A, then the flippers will be in the state $x_1 = R$, $x_2 = L$, $x_3 = R$, and the ball will exist through C which is exactly q_{RLRc}.

(e) $\delta^*(q_{LLL}, w) = q_{LRR}$. By IH, we have that the flipper states are $x_1 = L$, $x_2 = R$, and $x_3 = R$. Note that if we drop another ball into A, then the flippers will be in the state $x_1 = R$, $x_2 = R$, $x_3 = R$, and the ball will exist through C which is exactly q_{RRR}.

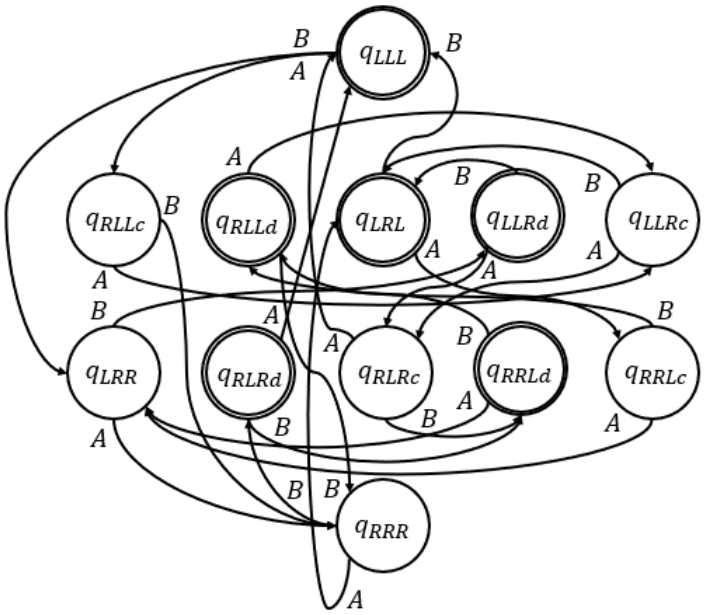


Figure 3: DFA which accepts strings of As and Bs where the last ball dropped falls through D.

(f) $\delta^*(q_{LLL}, w) = q_{RLRc}$ and $\delta^*(q_{LLL}, wA) = q_{RLRd}$. By IH, we have that the flipper states are $x_1 = R$, $x_2 = L$, and $x_3 = R$ regardless of where the last ball exited. Note that if we drop another ball into A, then the flippers will be in the state $x_1 = L$, $x_2 = L$, $x_3 = L$, and the ball will exist through D which is exactly q_{LLL}.

(g) $\delta^*(q_{LLL}, w) = q_{RRLc}$ and $\delta^*(q_{LLL}, wA) = q_{RRLd}$. By IH, we have that the flipper states are $x_1 = R$, $x_2 = R$, and $x_3 = L$ regardless of where the last ball exited. Note that if we drop another ball into A, then the flippers will be in the state $x_1 = L$, $x_2 = r$, $x_3 = R$, and the ball will exist through C which is exactly q_{LRR}.

(h) $\delta^*(q_{LLL}, w) = q_{RRR}$. By IH, we have that the flipper states are $x_1 = R$, $x_2 = R$, and $x_3 = R$. Note that if we drop another ball into A, then the flippers will be in the state $x_1 = L$, $x_2 = R$, $x_3 = L$, and the ball will exist through D which is exactly q_{LRL}.

We could also do the same for wB, 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 the language of the flipper toy. □

[4 Marks] 0.5 marks for each type of case listed above.

Q2. [10 Points] Regex Implies NFA *(maximum 3 pages not counting the diagrams for Regex to NFA)*

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.

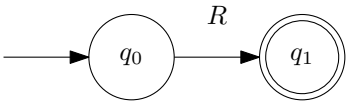


Figure 4: DFA M.

Solution. M is equivalent to R.

(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

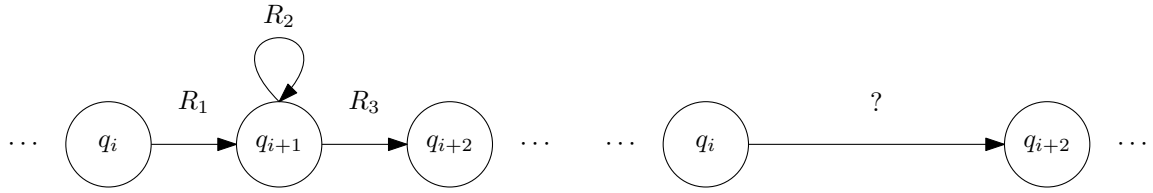


Figure 5: Part of DFA M. Find the regex which fits in ?.

that we can eliminate q_{i+1}.

Solution. The regular expression we should put on the transition $q_i \to q_{i+2}$ is $R_1(R_2)^*R_3$.

(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

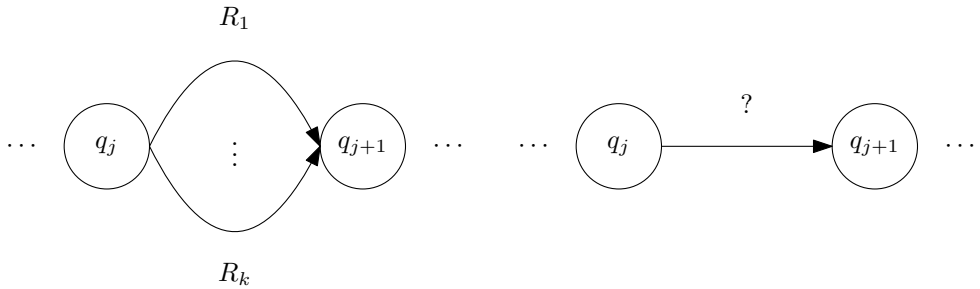


Figure 6: Part of DFA M. Find the regex which fits in ?.

R_k so that we can eliminate all other transitions.

Solution. The regular expression on the transition $q_j \to q_{j+1}$ is $R_1 + \cdots + R_k$.

(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.

Proof. Let M be our DFA. We add a dummy start state s with an ϵ-transition to the start state of M. Further, we add a dummy state which will be the only accepting state and perform an ϵ-transition from every accepting state of M to this dummy accept state.

Let M' be the finite automaton after the transformation. Note that, for M', there are no transitions into the start state, no transitions out of the accepting state, and no loops on either the start or accepting states.

Now we will prove the following predicate for M' by induction on the number of states of M', n. Note that in the base case, there will be *two* states: the dummy start and accepting states. $P(n)$ states if M' has n states then, there exists a regular expression R which is equivalent to M'. In the base case, there are only two states with a transition between them. There is a regular expression on the transition function R. M' is equivalent to R.

Suppose in the inductive step, that for some fixed $k \geq 2$, we have that $P(2) \wedge \cdots \wedge P(k)$ is true. We want to show that $P(k+1)$ is true. Suppose M' is a finite automaton of the form previously described. M' has an internal node u, i.e., not a starting or accepting state. We will eliminate u and replace the transitions to and from u so that the language accepted by M' remains unchanged. First, let I be the set of states with a transition into u and O be the set of states we can reach by a transition from u. We will add the following transitions for every pair of $v \in I$ and $w \in O$: suppose $\delta^*(v, R_1) = u$, $\delta^*(u, R_2) = w$, and there u has a self-loop with the regular expression R_3 then we will add the transition $\delta^*(v, R_1(R_3^*)R_2) = w$. After this transformation, there may be pairs of nodes with multiple transition $T_1, ..., T_m$ from one to the other. We replace these with a single transition on the regular expression $T_1 + \cdots + T_m$. After removing u, we have a finite automaton with k states. By IH, it has an equivalent regular expression.

By the principle of mathematical induction, $P(n)$ is true for $\forall n$ with $n \geq 2$. □

[5 Marks] Two marks for part (d). One mark each for each of parts (a), (b), and (c).

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. See Figure 7.

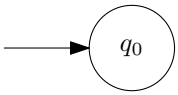


Figure 7: NFA for regex $\emptyset$.

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

Solution. See Figure 8.

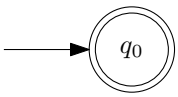


Figure 8: NFA for regex ϵ.

(c) For a fixed $a \in \Sigma$, draw an NFA which is equivalent to the regular expression a.
Solution. See Figure 9.

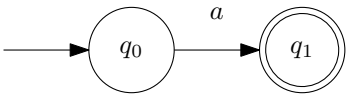


Figure 9: NFA for regex a for $a \in \Sigma$.

(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. Suppose M_1 has start state q_0 and accepting states $q_{f_1}, ..., q_{f_m}$ and M_2 has start state p_0 and accepting states $p_{g_1}, ..., p_{g_n}$. We create a dummy start state s with an ϵ-transition to q_0 and p_0. See Figure 10.

(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_1R_2.
Solution. With notation from the previous part, we add an ϵ-transition from each q_{f_i} to p_0 as shown in Figure 11.

(f) Suppose we have a regular expression R with corresponding DFA M. Use M to construct an NFA which is equivalent to R^*.
Solution. For every accepting state q_{f_i} we add an ϵ-transition back to the starting state. See Figure 12.

[5 Marks] Parts (a) and (b) should be grouped together for one mark. All the remaining parts should be one mark each.

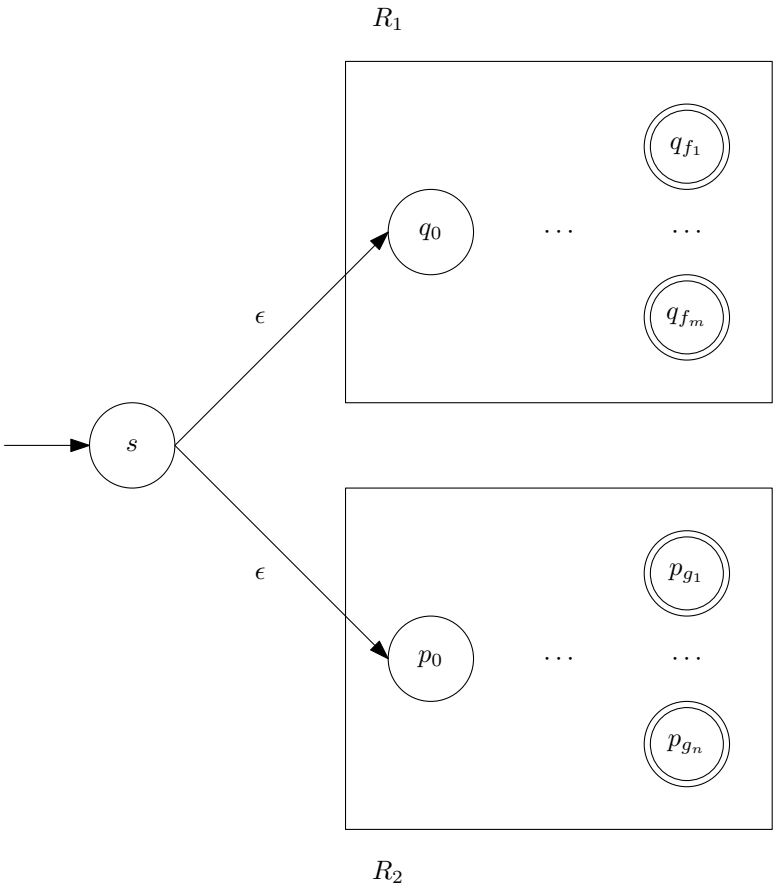


Figure 10: NFA for regex $R_1 + R_2$.

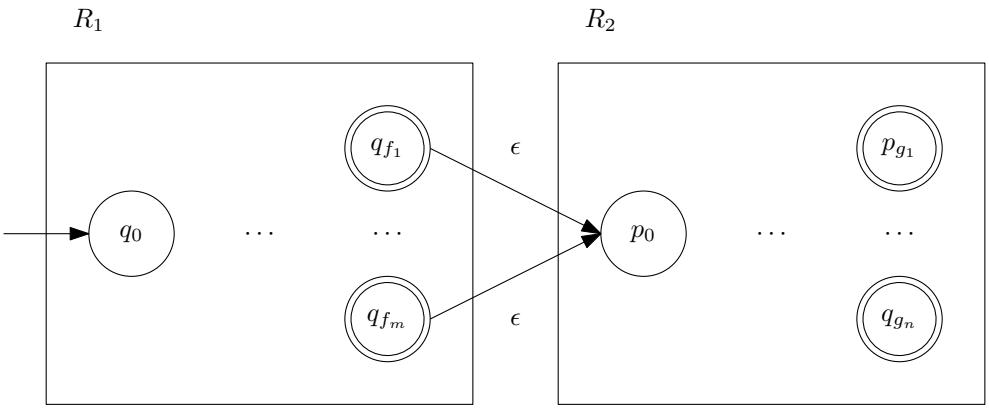


Figure 11: NFA for regex $R_1 R_2$.

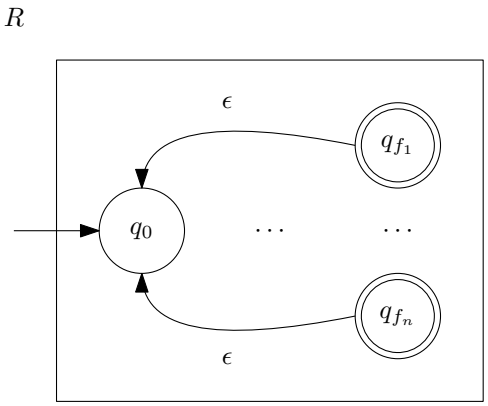


Figure 12: NFA for regex R^*.