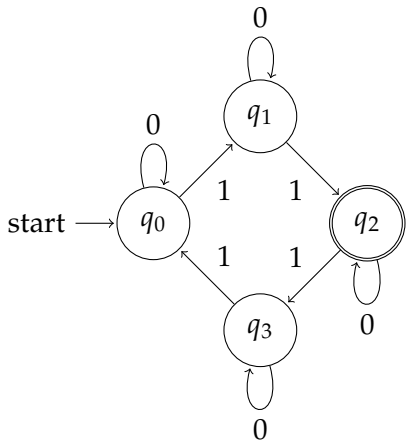
Problem Set 0

- This problem set is meant for students who have perhaps not seen automatas earlier.
- You are encouraged to discuss with your class mates but however such discussion should not break the spirit of doing these problems in the first place. *The goal should always be to improve your understanding of the subject and not answers (in fact, often mistakes are more helpful in this regard than solutions).*
- You do not have to submit this problem set, but you are highly encouraged to if you do not have prior exposure to automatas.

1 A brief crash course on automatas

In class, we saw examples of (deterministic) automata such as the following:



In words, we have a set of states Q, and we have a transition function $\delta : Q \times \Sigma \to Q$. Among the states, there is a specified start state (q_0 above) and some subset of them deemed as accepting states ($\{q_2\}$ above).

Question 1.1. *Let $\Sigma = \{0,1\}$. Let $L \subseteq \Sigma^*$ be the language consisting of strings such that the 5-th symbol in the string is 0. Build an automata for this language.*

How many states did you use? Do you think your answer is optimal?

Question 1.2. *Suppose M is an automata, and suppose q is one of the states of the automata and $x, y \in \Sigma^*$ such that M ends up in state q when it is run on x and also when it is run on y.*

Show that, for every string $z \in \Sigma^*$, we have that M accepts xz if and only if M accepts yz.

Question 1.3. *Using the above problem, show that the language*

$$L = \{0^n 1^n \ : \ n \in \mathbb{N}\} = \{01, 0011, 000111, 00001111, \ldots\}$$

cannot be accepted by any automata.

[Hint: Try running the automata on 0, 00, 000, etc. What happens when two of these strings reach the same state in the purported automata for L?]

1.1 Operation on languages

Complementation

Define the complement of a language L as $\overline{L} = \{x \in \Sigma^* \ : \ x \notin L\}$.

Question 1.4. *Given a finite automata M for a language L, describe how you will construct an automata for $\overline{L}$. How many states does $\overline{L}$ need in relation to the number of states in M?*

Union and intersection

Question 1.5. *Suppose M_1 is an automata for L_1, and M_2 is an automata for L_2.*
Construct an automata for $L_1 \cap L_2$.
Construct an automata for $L_1 \cup L_2$.
How many states did you need, in relation to the number of states in M_1 and the number of states in M_2?

[Hint: Set up your states so that you can "run both in parallel".]

Reversal

Another natural operation is the notion of "reversal". That is, $L^{\text{rev}} = \{\text{rev}(x) : x \in L\}$ (where $\text{rev}(x)$ is just x written in reverse).

Question 1.6. *Consider the reverse of the language in Question 1.1. That is, consider the set of all strings such that the* 5*-th symbol* from the right of the string *is* 0*.*
Construct an automata for this language. How many states do you need? Is it optimal?

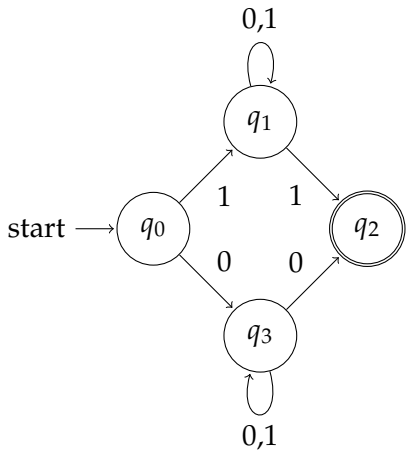
1.2 Introducing non-determinism

All automatas described above were "deterministic" in the sense that given a state q and a symbol $a \in \Sigma$, the transition function move the automata to a specific state q' (that is, $\delta(q, a) = q'$).

In non-deterministic finite automata (NFA), we relax this to say that the transition can move you to "more than one state". Formally, the transition function of an NFA is of the type

$$\delta : Q \times \Sigma \to \mathcal{P}(Q)$$

where $\mathcal{P}(Q)$ refers to the power set of Q (or the set of all subsets of Q). Pictorially, this can be interpretted diagrammatically such as the following:



Here, we have situations such as $\delta(q_1, 1) = \{q_1, q_2\}$ and $\delta(q_2, 0) = \delta(q_2, 1) = \emptyset$ etc. (such things would not be allowed in a usual automata (also called a deterministic finite automata or a DFA)).

The above now adds a confusion of what it means for an NFA to accept a string x. In the above example, what happens when we run the string on $x = 111$? Do we end up in state q_1 or in q_2 (since both seem to be legitimate runs).

We will say that a string x is accepted by an NFA if *some legitimate run* ends in an accepting state.

(The above example NFA in fact accepts all strings of length at least two whose first and last symbols are the same.)

Question 1.7. *Go back to all the earlier questions and answer them with NFAs in mind instead. (Specifically, try to build an NFA for language in Question 1.6. How many states did you need for that question now?)*