# COL352 Lecture 4

## Contents

**1 Recap** **1**

## 1 Recap

Recall that regular languages are closed under complementation, $\cap$ and $\cup$.
We also defined $L_1L_2 = \{x_1x_2 \mid x_1 \in L_1, x_2 \in L_2\}$.

**Claim 1.** *The class of regular languages is closed under concatenation.*

**Notation 1.** *Let $rev(x)$ denote the reverse of a string $x \in \Sigma^*$.*

**Notation 2.** *Let $rev(L)$ denote $\{rev(x) \mid x\imath L\}$.*

**Claim 2.** *The class of regular languages is closed under reversal.*

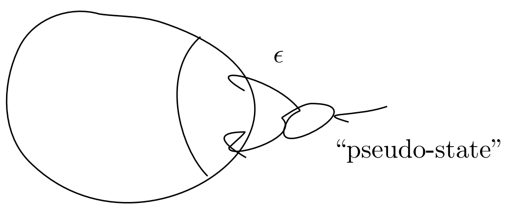


Figure 1: An example of thing that doesn't work

Consideration: $\epsilon$ transition? Gives NFA, not DFA.

So let's consider the following language:

$\Sigma = \{0, 1\}$, $L_1 = \{x \in \Sigma^* \mid x \text{ begins with } 01\}$.

$rev(\Sigma) = \{x \in \Sigma^* | x \text{ ends with } 10\}$.

Strategy: reverse all transitions of the DFA, swap accepting and starting states. Easy to make DFA for both, but using the reversal strategy, $\delta$ is not a function, and hence not a DFA.

Consider

$\Sigma = \{0, 1\}$, $L_1 = \Sigma^*$, $L_2 = \{1w \mid |w| = 3\}$. Easy to make DFA of both.

Using $\epsilon$ transitions, we can create an NFA for the concatenation of these languages.

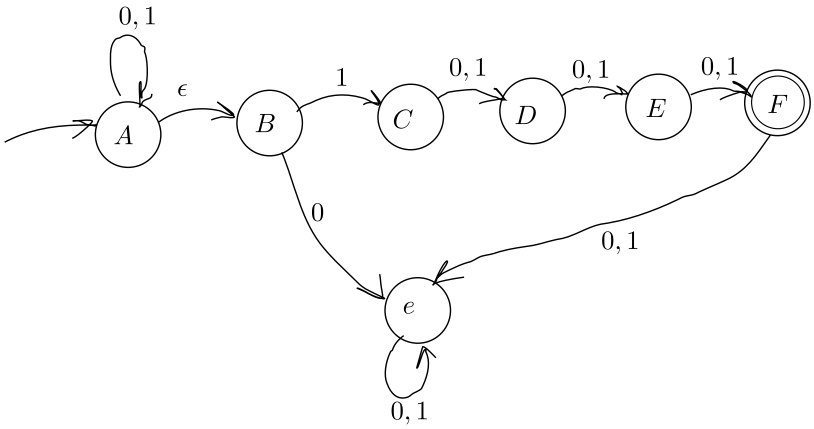


Figure 2: NFA example for $L_1 L_2$

**Definition 1.** *A non-deterministic finite automaton (NFA) is a 5-tuple* $(Q, \Sigma, \Delta, Q_0, A)$*, where*

1. $Q$ *is a finite non-empty set of states.*
2. $\Sigma$ *is the input alphabet (finite set).*
3. $\Delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$ *is the set of transitions.*
4. $Q_0 \subseteq Q$ *(possibly empty).*
5. $A \subseteq Q$ *is a set of accepting states.*

Use the first reversal example here.

**Note 1.** *NFA allows you to take/not take a transition.*

**Note 2.** *Note that point 3 is equivalent to* $\Delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$*; this equivalence arises by considering the set of* $z$ *such that* $(x, y, z) \in \Delta$*, and mapping* $(x, y)$ *to this set, which is a subset of* $Q$*.*

**Note 3.** *There might be (possibly 0) many runs of an NFA on an input.*

**Note 4.** *It might be possible that one run ends in an accepting state and another ends in a rejecting state.*

The corresponding NFA for the reversal problem is as follows:
$Q = \{q_0, q_1, q_2, q_e\}$.
$\Sigma = \{0, 1\}$.
$Q_0 = \{q_2\}, A = \{q_0\}$.
$\Delta = \{(q_2, 0, q_2), (q_2, 1, q_2), (q_2, 1, q_1), (q_1, 0, q_0), (q_e, 0, q_e), (q_e, 1, q_e), (q_e, 0, q_1), (q_e, 1, q_e)\}$.

Now let's see what happens on a run of this NFA with the input 1010.

Current state $(q_2, 1010)$.

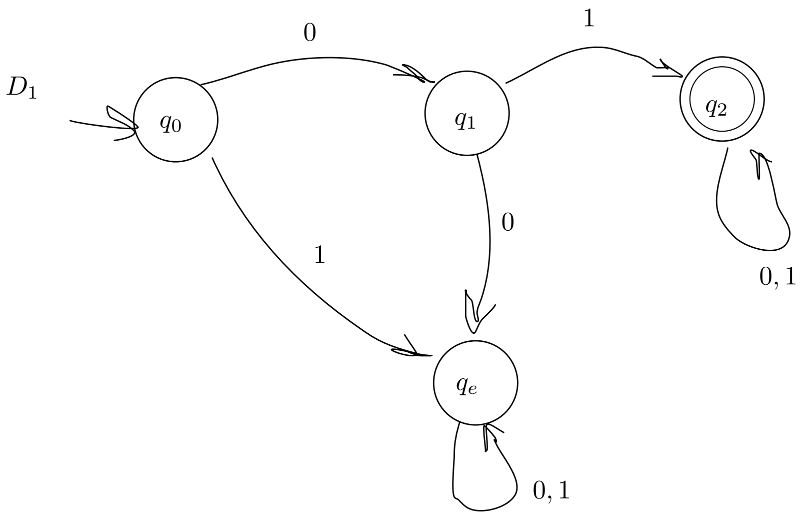


Figure 3: example beginning with 01

Can go to $(q_1, 010)$ or $(q_2, 010)$.

From $(q_1, 010)$, forced to go to $(q_0, 10)$, and from $q_2, 010$, forced to stay in $(q_2, 10)$.

The first run dies off, since no other possibilities.

From $(q_2, 10)$ we can go to $(q_1, 0)$ or $(q_2, 0)$, and from the first, we get to $(q_0, \epsilon)$, and from the second, we get to $(q_2, \epsilon)$.

Hence there are 3 possible runs, first rejecting, second accepting, and third rejecting.

For the second example, which has an $\epsilon$ transition, let's see how it works as well.

Consider the input $x = 00100$.

First step
$(A, 00100) \to (A, 0100), (B, 00100)$.

Second step
$(B, 00100) \to (error, 0100) \to \cdots$ (stays forever)
$(A, 0100) \to (A, 100), (B, 0100)$

Third step
$(A, 100) \to (B, 100) \to (C, 00) \to (D, 0) \to (E, \epsilon)$
$(A, 100) \to (A, 00) \to (B, 00) \to (error, 0) \to (error, \epsilon)$
$(A, 100) \to (A, 00) \to (A, 0)$

Fourth step
$(A, 0) \to (A, \epsilon) \to (B, \epsilon)$
$(A, 0) \to (B, 0) \to (error, \epsilon)$

So in this case, all runs of the NFA reject $x$.
We say that the NFA accepts a string $x$, if there is at least one accepting run of the NFA, and it rejects $x$ otherwise.

**Notation 3.** *Informally, we say NFA $N$ accepts string $x$ is $\exists$ an accepting run of $N$ on $x$; $N$ rejects $x$ if all*

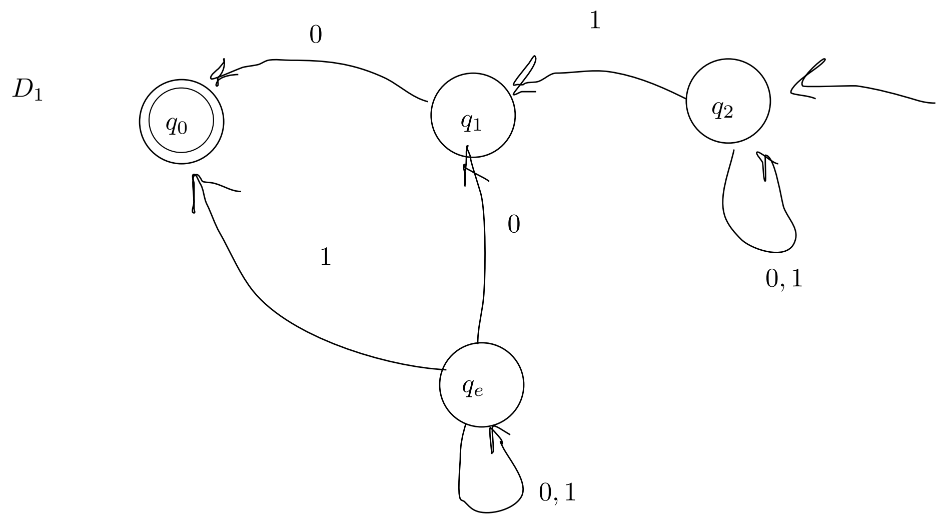


Figure 4: example ending with 10

*runs of $N$ on $x$ reject.*