Regular Operations and Regular Languages

18 January 2021 00:02

<u>Theorem</u>: Regular languages are closed under complement.

<u>Def 1.23</u>: (Regular Operations): Let A and B be languages. The regular operations are union, concatenation and star, defined as follows:

(1) Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

(2) Concatenation: $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$

(3) Star: $A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0 \text{ and } x_i \in A \text{ for each } i\}$

We will see that regular languages are closed under regular operations.

<u>Example 1.24</u>: Let $\Sigma = \{a, b, c, \ldots x, y, z\}$.

$A = \{good, bad\}$ and $B = \{boy, girl\}$.

$A \cup B = \{good, bad, boy, girl\}$

$A \circ B = \{goodboy, goodgirl, badboy,$

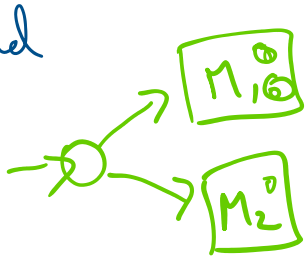
badgirl }

$A^* = \{ \varepsilon$, good, bad, goodgood, goodbad,
badbad, badgood, goodgoodgood,
goodgoodbad, ... $\}$.

<u>Theorem 1.25</u>: The class of regular languages is closed under the union operation.

We have A_1 and A_2. Since both are regular, this means there exist DFAs M_1, M_2 such that $L(M_1) = A_1$ and $L(M_2) = A_2$. Can we build a machine such that it recognizes $A_1 \cup A_2$?.

<u>Idea 1</u>: Simulate the processing of M_1 followed by the processing of M_2. Doesn't work because DFA cannot go back and re-read the input.



<u>Moral</u>: We need to keep track of the workings of M_1 and M_2 at the same time. How to accomplish this?

time. How to accomplish this?

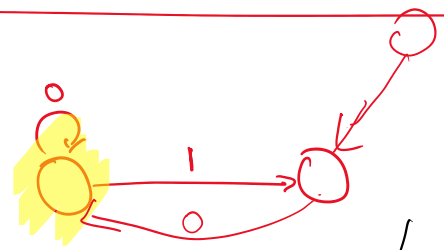
→ Cartesian Product.

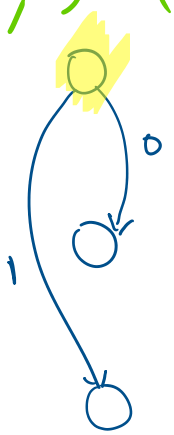
Machine whose states are (r, s)

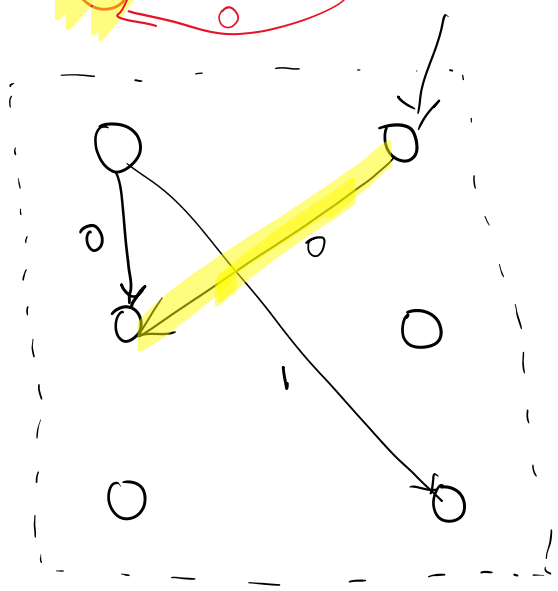
$\delta_1(r_1, a) = s_1$

$\delta_2(r_2, a) = s_2$

$\delta((r_1, r_2), a) = (s_1, s_2)$







F_1 F_2

$F_1 \times F_2$:

$F_1 \cap F_2$:

$F_1 \cup F_2$

<u>Proof</u>: We have A_1, A_2. Let M_1 recognize A_1 and let M_2 recognize A_2. Let $A_1, A_2 \subseteq \Sigma^*$.

Else $A_1 \cup A_2$ does not make sense.

$$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$$

$$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$$

<u>Goal</u>: Need to build $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = A_1 \cup A_2$.

(1) $Q = \{ (r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2 \}$

$= Q_1 \times Q_2$.

(2) Σ is the same alphabet. If A_1 and A_2 are not over the same alphabet, choose $\Sigma = \Sigma_1 \cup \Sigma_2$ and view A_1 and A_2 over Σ.

(3) δ needs to keep track of δ_1 and δ_2.

$$\delta : (Q \times \Sigma) \to Q.$$

$$\delta((r_1, r_2), a) = (\delta(r_1, a), \delta(r_2, a))$$

(4) Starting State : $q_0 = (q_1, q_2)$

(5) Accepting States: We accept w if it ends at (r_1, r_2) if either $r_1 \in F_1$ or $r_2 \in F_2$.

$$F = \{ (r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2 \}$$

$$= \underline{(F_1 \times Q_2) \cup (Q_1 \times F_2)}$$

Why not $F_1 \times F_2$?

Correctness of this proof is straightforward.

Correctness of this proof is straightforward.

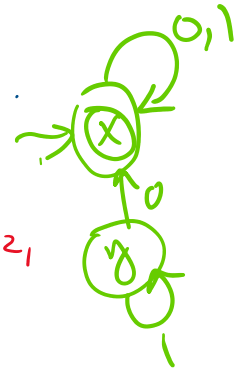
Let $w \in A_1 \cup A_2$. That $w \in A_1$ or $w \in A_2$. Suppose $w \in A_1$. Then there is a sequence of states $s_0, s_1, s_2 \ldots s_n$ such that.

(1) $s_0 = q_1$, (2) $\delta(s_{i-1}, w_i) = s_i$, and

(3) $s_n \in F_1$.

$(s_n \in F_1, r_j)$

When M reads w, then M ends in $F_1 \times Q_2$.



Q: Is it enough to show that for all $w \in A_1 \cup A_2$, w is accepted by M?.

Q: Regular languages are closed under intersection. How to show this?

Theorem 1.26: Regular languages are closed under the concatenation operation.

0110100111

If A_1 and A_2 are regular, then $A_1 \circ A_2$ is regular.

Can we try to simulate M_1 and M_2 on two pieces of the input? The issue is that the input could be split anywhere, and we cannot go back and try different splits. We will introduce non determinism

different splits. We will introduce non determinism to solve this.

Non deterministic Finite Automata (NFA)

Till now, we have seen Deterministic FA. Now we will see NFA. Let us see the major differences.

1) DFA has exactly one transition for each symbol and state.

   NFA can have 0, 1 or many.

2) DFA has arrows labelled with symbols in the alphabet.

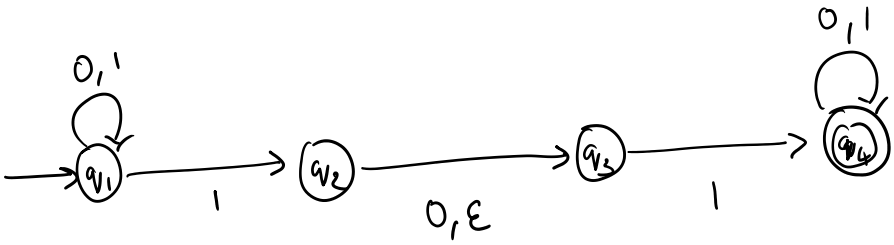
   NFA can have arrows marked with ε.

   → Can have 0, 1 or many arrows marked with ε.

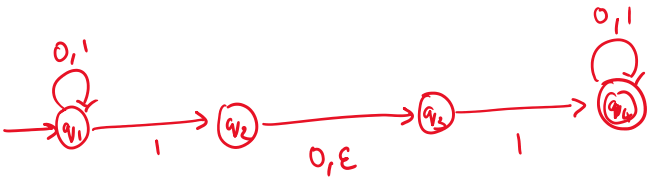
3) NFA can have multiple choices to be made at each state, possibly. It could have many possible computation paths.

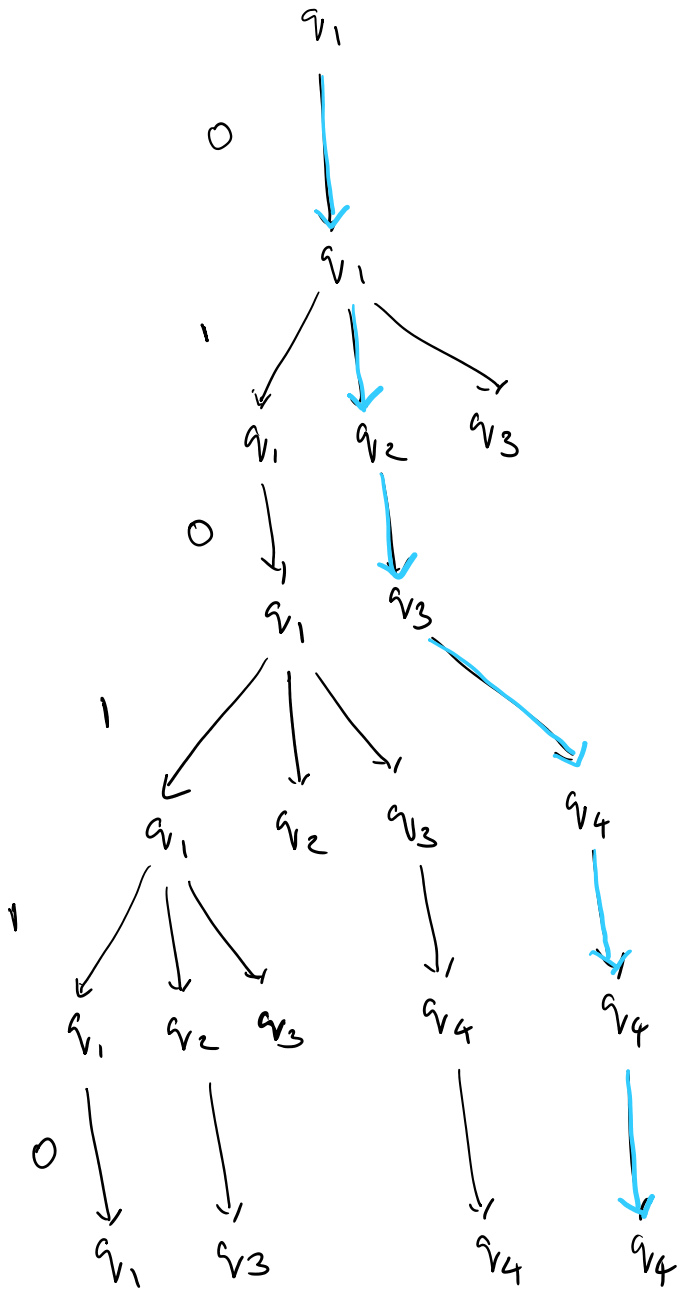
4) NFA accepts a string if there is at least

4) NFA accepts a string if there is at least one accepting computation path.



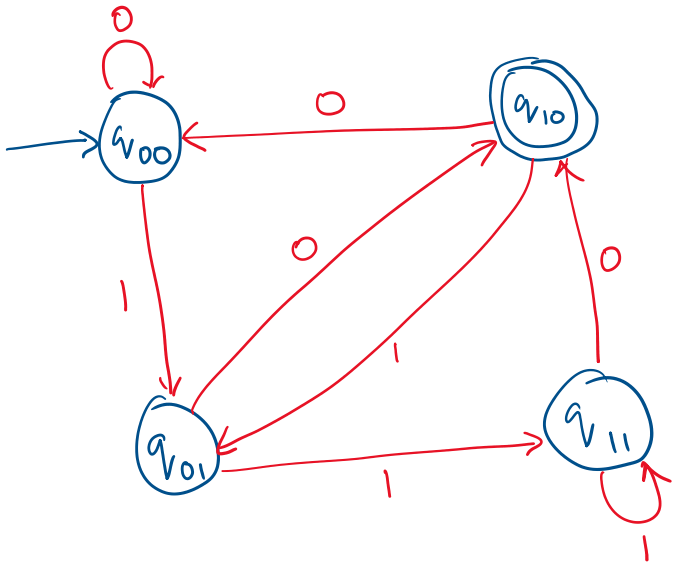
5) Nondeterminism can be viewed as guessing.



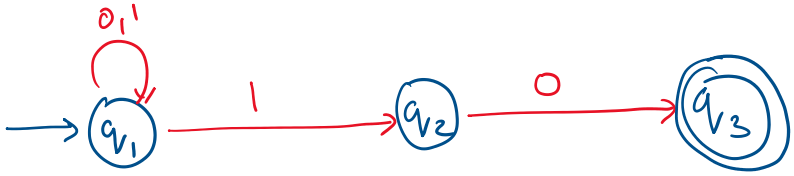


$A = \{ w \mid w \text{ ends in } 10 \}$

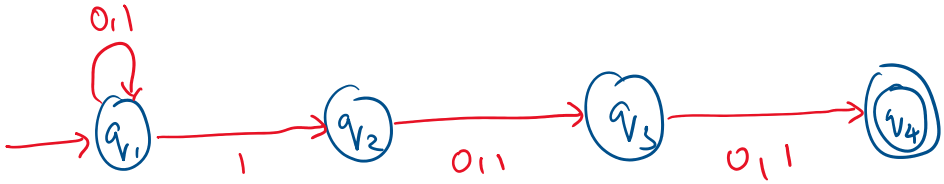
DFA for A



NFA for A:



<u>Example 1.30</u>: language of strings where the third symbol from the end is a 1.

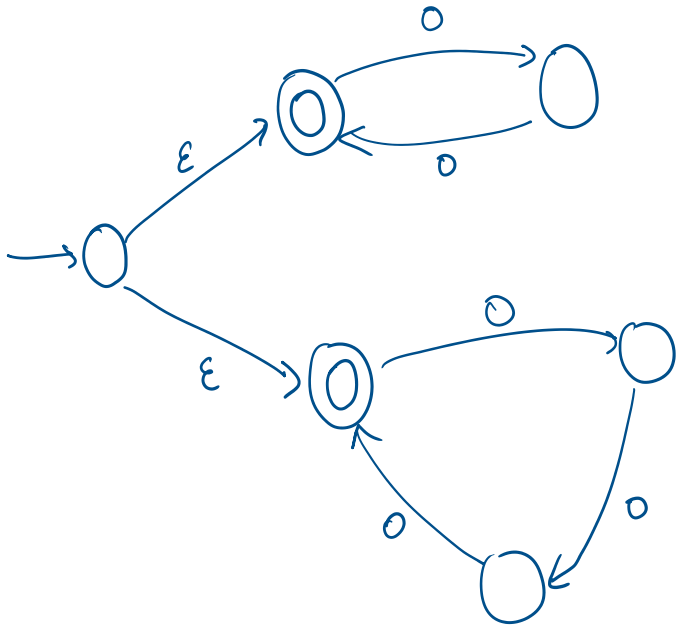


The DFA requires 8 states and is more involved.

Read the book and see the DFA.

Example 1.33 : Unary language.

$A = \{ 0^k \mid k \text{ is a multiple of } 2 \text{ or } 3 \}$



Def 1.37: A nondeterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

(1) Q is a finite set of states

(2) Σ is a finite alphabet.

(3) $\delta : Q \times \Sigma_\varepsilon \to \underline{\mathcal{P}(Q)}$ is the transition function.

Power set of Q

(4) $q_0 \in Q$ is the start state

(5) $F \subseteq Q$ is the set of accepting states.

$\mathcal{P}(Q)$ = Power set of Q = Set of all subsets of Q.

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.

NFA computation is similar to DFA computation.

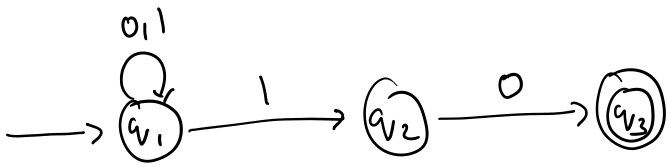
NFA N accepts w if we can write $w = y_1 y_2 \cdots y_m$ where $y_i \in \Sigma_\varepsilon$ and a sequence of states $r_0, r_1, \ldots r_m \in Q$ s.t.

(1) $r_0 = q_0$

(2) $\delta(r_i, y_{i+1}) \ni r_{i+1}$ for $i = 0, 1, \ldots m-1$

(3) $r_m \in F$.

<u>Example</u>:



$Q = \{q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

$F = \{q_3\}$

Start State $\Big\} = q_1$

$\delta(q_1, 0) = \{q_1\}$

$\delta(q_1, 1) = \{q_1, q_2\}$

$\delta(q_2, 0) = \{q_3\}$

$\delta(q_2, 1) = \phi$

$\delta(q_3, 0) = \delta(q_3, 1) = \phi$.

Read <u>Example 1.38</u>.