Deterministic Finite Automata

12 January 2021 23:16

A <u>language</u> over Σ is a set of strings over Σ. That is, a language over Σ is a subset $A \subseteq \Sigma^*$.

<u>Example</u>: Set of all binary strings with odd number of 1's is a language over $\{0,1\}$.

Languages can be specified in several ways.

1. Brute Force listing. $\{a, ab, abb, abbb, \ldots\}$
2. language operations. ab^*.
3. Other set theoretic descriptions.

<u>Finite Automata</u>

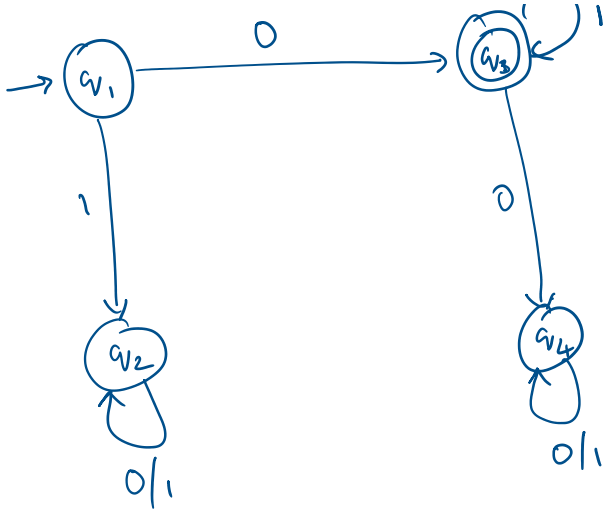
Computers with limited amount of memory.

State based devices.

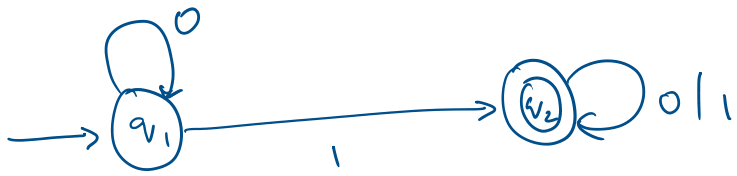
<u>Examples</u>: Timers, door open/close controller, Thermostat.

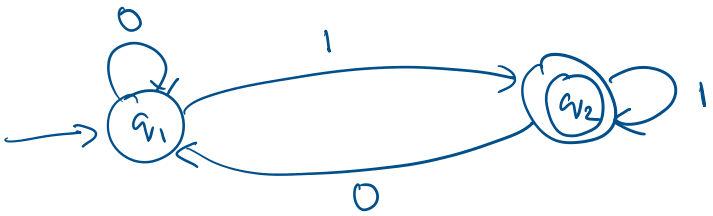
Why? These are abstract models that help us gain understanding.

<u>Deterministic Finite Automata</u>

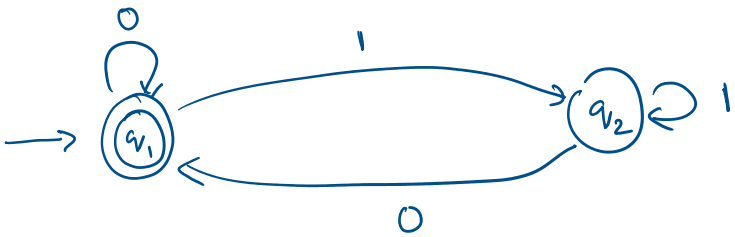


$A = \{0, 01, 011, 0111, 01111, \ldots\}$





$\Sigma^* 1$



$\Sigma^* 0 \cup \{\varepsilon\}$

DFA.

Def 1.5: A deterministic finite automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where.

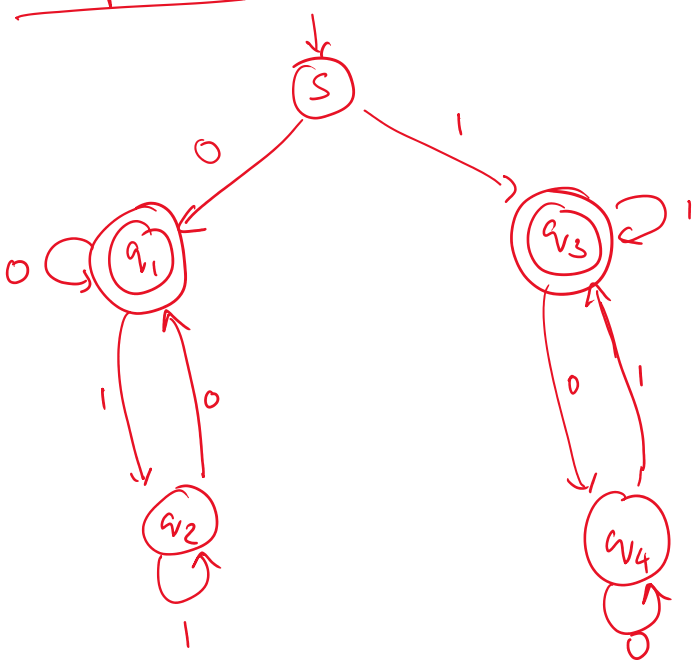
1. Q is a finite set of states

1. Q is a finite set of states
2. Σ is a finite alphabet.
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accepting states.

Read example 1.11, 1.13, 1.17, 1.21.

Read definition 1.16.

Example 1.11



$0\Sigma^*0$ does not contain the string 0.

$0\Sigma^*0 \cup 1\Sigma^*1 \cup \{0,1\}$

$Q = \{s, q_1, q_2, q_3, q_4\}$

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

$\delta(s,0) = q_1 \quad \delta(s,1) = q_3$

$B = \{ w \in \{0,1\}^* \mid$ first symbol of w ...

$\delta(s,0) = q_1 \quad \delta(s,1) = q_3$

$\delta(q_1,0) = q_1 \quad \delta(q_1,1) = q_2$

$\delta(q_2,0) = q_1 \quad \delta(q_2,1) = q_2$

$\delta(q_3,0) = q_4 \quad \delta(q_3,1) = q_3$

$\delta(q_4,0) = q_4 \quad \delta(q_4,1) = q_3$

$q_0 = s. \quad F = \{q_1, q_3\}.$

$B = \{ w \in \{0,1\}^* \mid$ first symbol of w = last symbol of $w \}$

$M = (Q, \Sigma, \delta, q_0, F)$ accepts a string $w = w_1 \ldots w_n$ if there is a sequence of states $r_0, r_1, \ldots r_n$ s.t.

START (1) $r_0 = q_0$.

MOVE (2) $\delta(r_i, w_{i+1}) = r_{i+1} \quad \forall\ 0 \le i \le n-1$

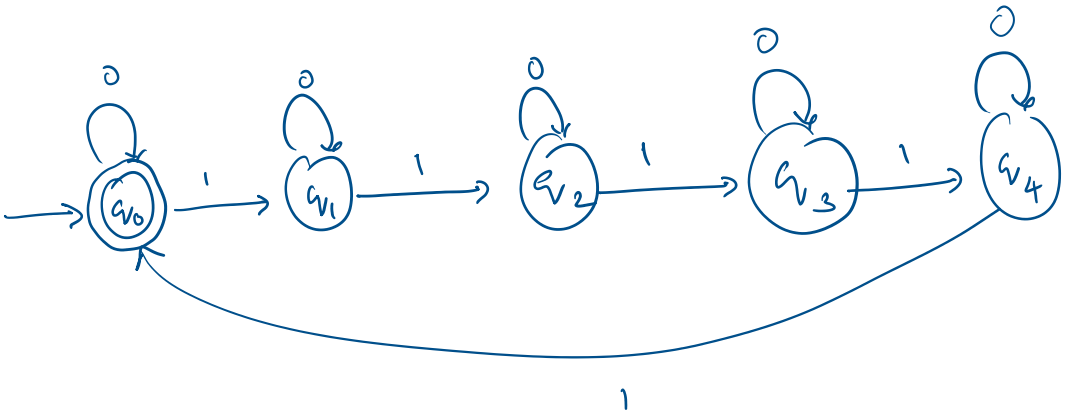
END (3) $r_n \in F$.

M recognizes the language A if

$$A = \{ w \mid M \text{ accepts } w \}.$$

How will you make a DFA that accepts all strings for which no. of 1's is divisible by 5?

- We need to track the number of 1's that we have already seen.
- We don't need full count. But only count mod 5.



$A = \{ w \in \{0,1\}^* \mid \text{no. of 1's in } w \equiv 2 \bmod 5 \}$

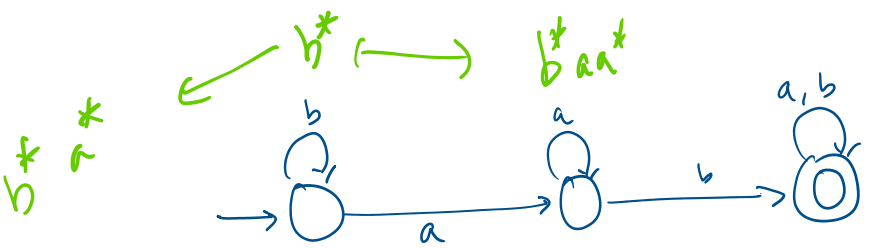
How do you make L such that $L(M) = A$?

Read Example 1.21.

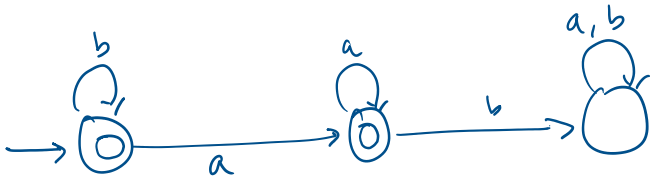
<u>Regular languages</u> (Def 1.16): A language is <u>regular</u> if some finite automaton recognizes it.

This is a machine model based definition. Later we will see a language | grammar based definition.

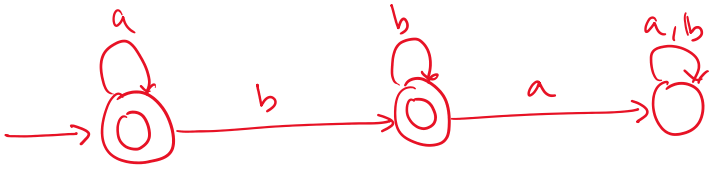
1.5 (a). $\{ w \in \{a,b\}^* \mid w \text{ contains } ab \text{ as a substring} \}$

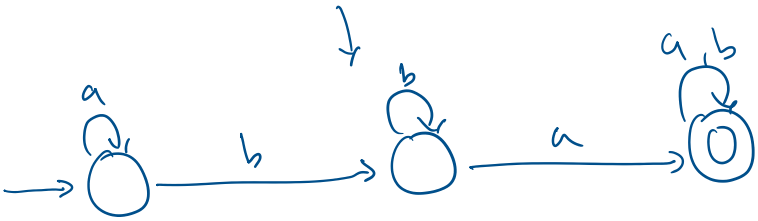


How do you design a DFA for the complement of the above language?



1.5 (d) $\{ w \in \{a,b\}^* \mid w$ is a string not in $a^* b^* \}$





1.6. (a) $\{ w \in \{0,1\}^* \mid w$ begins with 1 and ends in $0 \}$

<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 : $A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0 \text{ and } x_i \in A$ 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 $B = \{boy, girl\}$.

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

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

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

<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

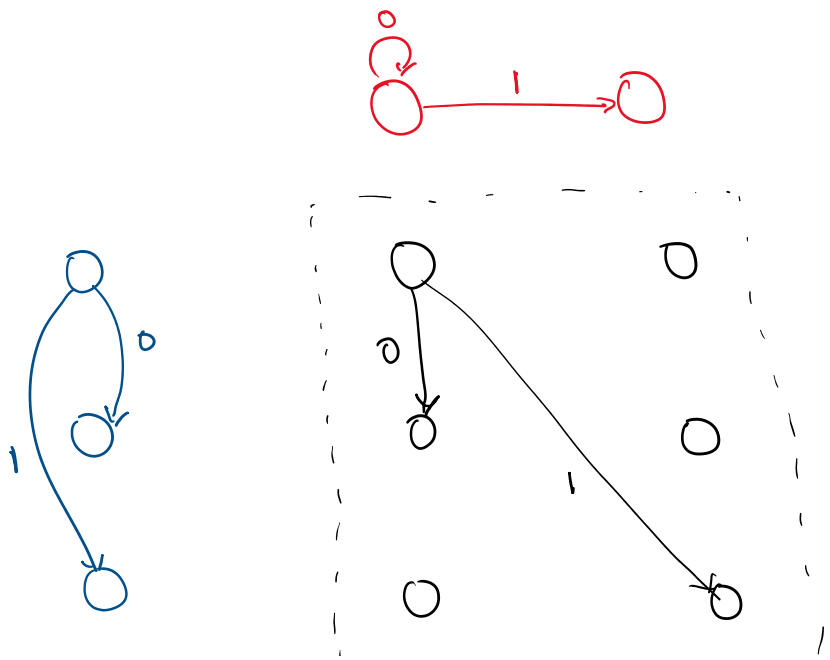
regular, this means there exist DFAs M_1, M_2 such that $L(M_1) = A_1$ and $L(M_2) = A_2$. Can we build a machine such that it recognizes $A_1 \cup A_2$?.

<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?

$\longrightarrow$ Cartesian Product.

Machine whose states are (r, s)



<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.

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) $s_n \in F_1$.

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?.