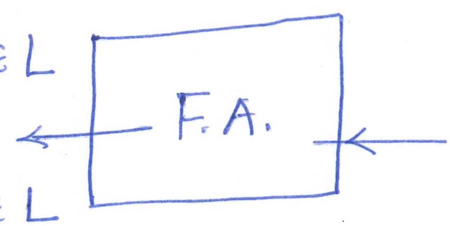
Finite Automata & Regular Languages

Regular Languages: Class of languages recognized by finite automata.

Abstractly, for a language recognized by a F.A.

YES/Accept if $w \in L$

NO/Reject if $w \notin L$



Input $w \in \Sigma^*$, $|w| = n$

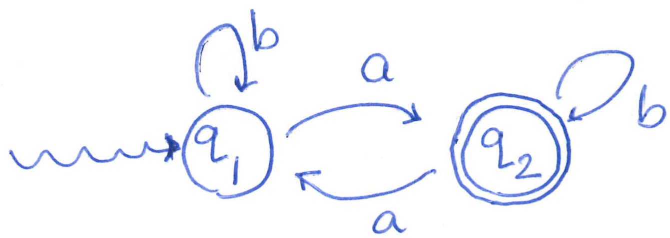
$w_1 w_2 \ldots w_n$

Input is read only once.

<u>Motivation</u> - Controllers for Sliding door, elevator, washing m/c etc.

- Constantly many "states".
- Change state according to input via simple rules.

<u>Examples of F.A.</u>



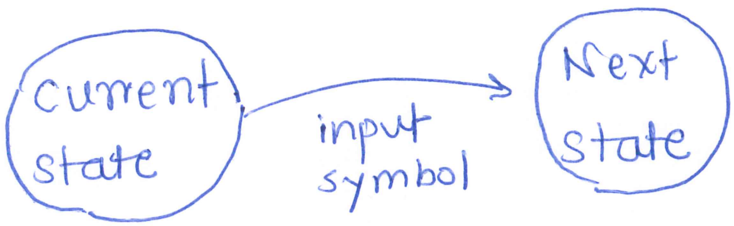
① $\Sigma = \{a, b\}$

q_1 : start state, indicated by squiggly in-arrow.

q_2 : accept state, " double circle.

There could be many (or no) accept states.

Given input say baabba, the FA starts in state q_1, scans input left to right, one symbol at a time and makes transitions accordingly. Generically



E.g. on input baabba

$$q_1 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_1 \xrightarrow{b} q_1 \xrightarrow{b} q_1 \xrightarrow{a} q_2.$$

Since the final state q_2 is an accept state, the F.A. is said to accept input baabba.

E.g. Inputs accepted: a, abbaa, babbaba, ...

Inputs rejected: ε, abba, aaaa, ...

If M denotes this F.A., the language recognized by it is denoted/defined as

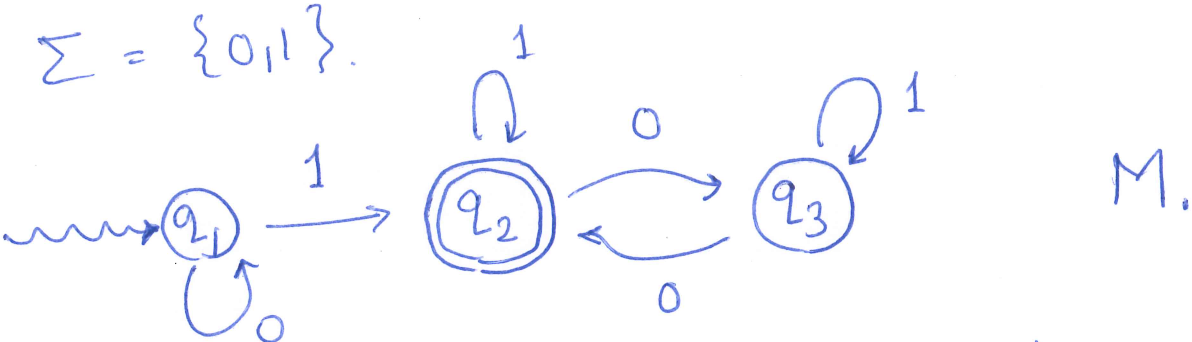
$$L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$$

I.e. running M on input w starting in state q_1 makes M end up in an accept state.

Evidently, in this example,

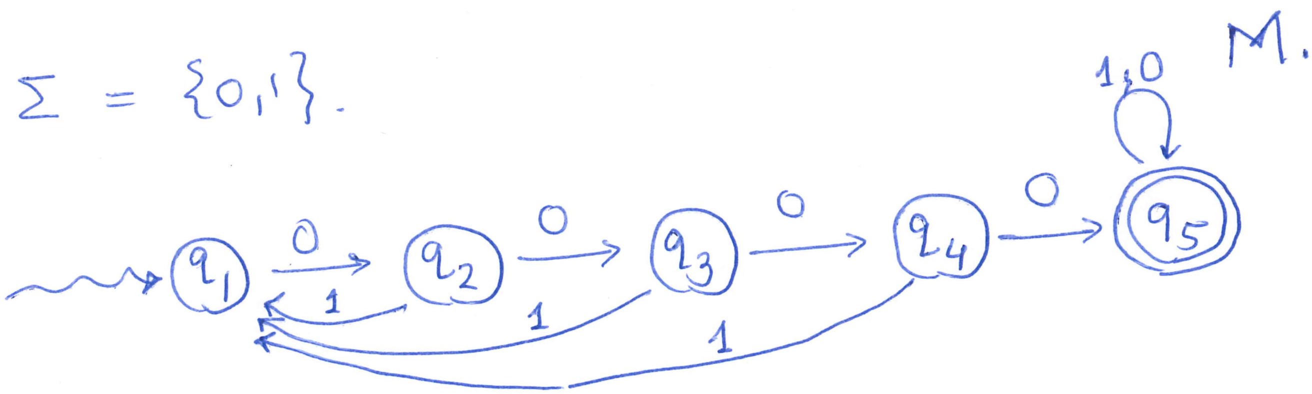
$L(M) = \{ w \in \{a,b\}^* \mid w$ has an odd number of $\underline{a}$'s. $\}$.

② $\Sigma = \{0,1\}$.



$L(M) = \{ w \in \{0,1\}^* \mid$ w has at least one $\underline{1}$ and after the first occurrence of $\underline{1}$ has an even number of $\underline{0}$'s. $\}$

③ $\Sigma = \{0,1\}$.



$L(M) = \{ w \in \{0,1\}^* \mid$ w has $\underline{0000}$ as a consecutive substring. $\}$

Formal Definition of F. A.

Def A finite automaton M is a 5-tuple

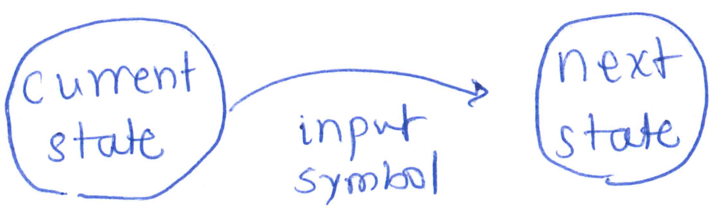
$M = (Q, \Sigma, \delta, q_1, F)$ where

- Q is a finite set of states.
- Σ is a finite alphabet.
- $q_1 \in Q$ is the start state
- $F \subseteq Q$ is the subset of accept states.
- $\delta : Q \times \Sigma \to Q$ is the transition function.

Note The transition function is interpreted as

δ(current state, input symbol read) = next state

i.e.



in the transition or state diagram.

The pictorial / graphical representation before is referred to as the state diagram.

Example 2, in this formal notation, is:

- $Q = \{q_1, q_2, q_3\}$.
- $\Sigma = \{0, 1\}$
- $F = \{q_2\}$.

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_3

Alternately as:

$\delta(q_1, 0) = q_1$ $\quad$ $\delta(q_2, 0) = q_3$ $\quad$ $\delta(q_3, 0) = q_2$

$\delta(q_1, 1) = q_2$ $\quad$ $\delta(q_2, 1) = q_2$ $\quad$ $\delta(q_3, 1) = q_3$.

Operation of F.A. (Informal) $(Q, \Sigma, \delta, q_1, F)$.

- Start in q_1.
- Scan input left to right, one symbol at a time, change state according to δ.
  transition
- When input is exhausted, accept iff the F.A. is in an accept state.

Note F.A. accepts the empty string ε iff $q_1 \in F$.

Operation of F.A. (Formal). $(Q, \Sigma, \delta, q_1, F) = M$

On input $w = w_1 w_2 \cdots w_n \in \Sigma^*$,

Let $r_1 = q_1$

Let $r_{i+1} = \delta(q_i, w_i)$ for $i = 1, 2, \ldots, n$.

Accept iff $r_{n+1} \in F$.

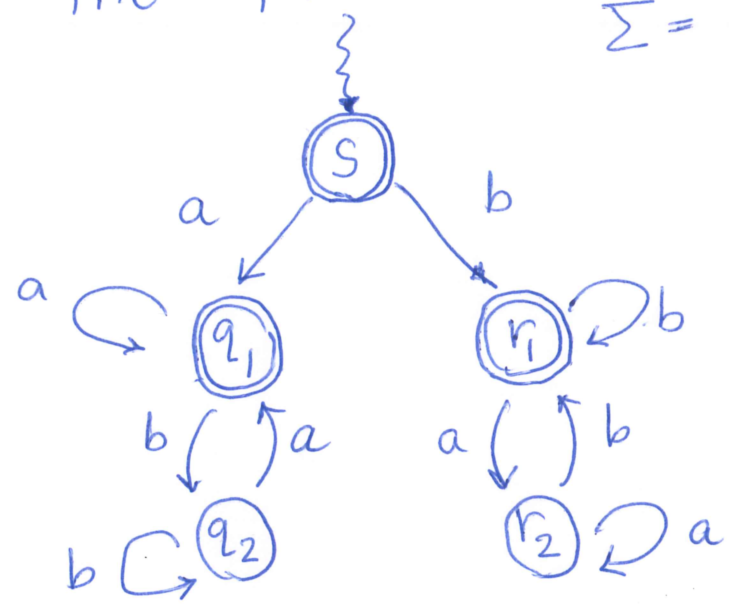
$L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w. \}$.

Def: The class of languages accepted by finite automata is called the class of regular languages.

Alternately, a language $L \subseteq \Sigma^*$ is called regular iff $L = L(M)$ for some finite automaton M with alphabet Σ.

Intuition: F.A. are a computational model with
- constant amount of memory and
- read once input.

Consider the FA $\Sigma = \{a, b\}$



It is not difficult to see that it recognizes

$$L = \left\{ w \in \{a,b\}^* \;\middle|\; \begin{array}{l}\text{the first and the last} \\ \text{symbol of } w \text{ is the same.}\end{array} \right\}$$

(with understanding that ε, $\underline{a}$, $\underline{b} \in L$).

Intuitively the F.A. "remembers" the first symbol and depending on it, "branches out", and the two branches separately "check" that the last symbol matches the "remembered" first symbol.

Similarly,

$$L = \left\{ w \;\middle|\; \begin{array}{l}\text{the first 5 symbols of } w \text{ match the} \\ \text{last 5 symbols in reverse}\end{array} \right\}$$

is regular. One "remembers" 5 symbols now.

However, as we'll show later, the following languages are not regular:

$L = \{ w \in \{a,b\}^* \mid w$ is a palindrome (reads same forward and backward) $\}$.

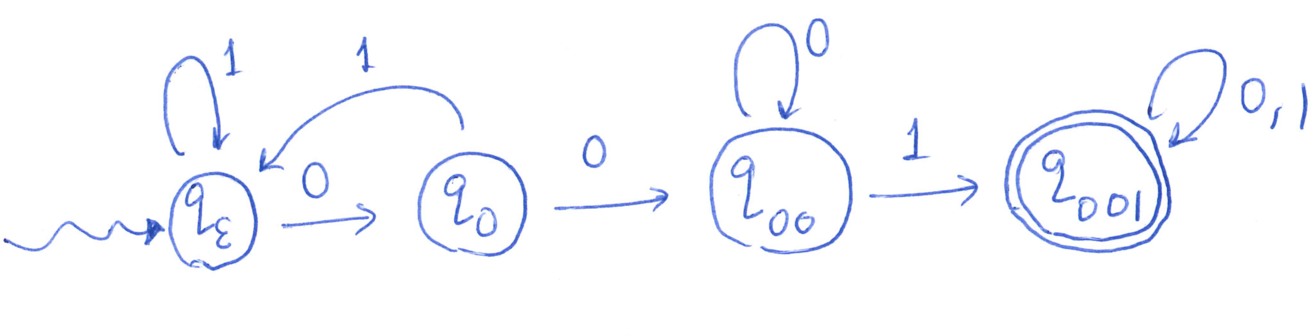
$L = \{ a^n b^n \mid n \geq 0 \}$.

In both cases, the m/c would (intuitively) need super-constant amount of memory ($\frac{|w|}{2}$ symbols to "remember" the first half of w <u>and</u> in the second example, $\log_2 n$ bits to "count" to n).

<u>Designing F.A., More Illustrative Examples</u>

<u>Question</u> Design FA that recognizes

$L = \{ w \in \{0,1\}^* \mid w$ has $\underline{001}$ as a consecutive substring. $\}$.



- 4 states corresponding to prefixes of 001.

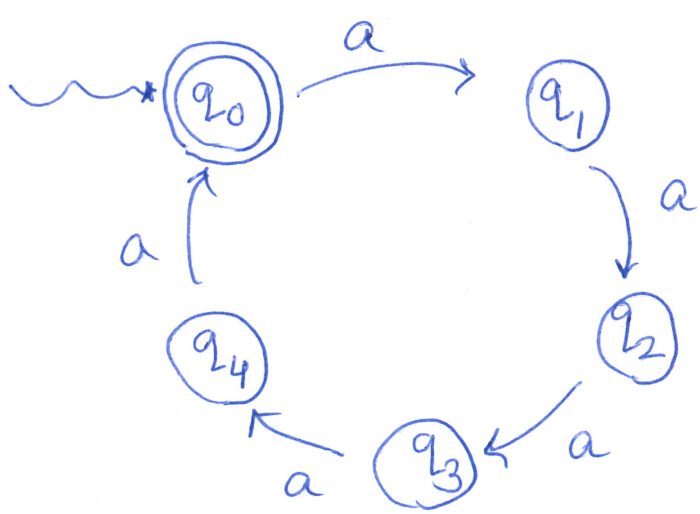
  ε, 0, 00, 001.

- Transitions capture "progress".

Exercise Construct FA that detects a consecutive substring 00101.

Question Design FA that recognizes

$$L = \{ w \in \{a\}^* \mid |w| \text{ is divisible by } 5 \}.$$



Note After reading prefix a^i, the F.A. is in state $q_{(i \bmod 5)}$.

How would one recognize

$$L = \{ w \in \{a\}^* \mid |w| \text{ is either } 2 \text{ or } 4 \bmod 5 \}$$

?

Question Design FA that recognizes $L = \{ w \in \{0,1\}^* \mid w$ in binary represents an integer divisible by 5. $\}$.

E.g. ε, 101, 1010, 00101, ... $\in L$.

1, 001, 1001, 1110, ... $\notin L$.

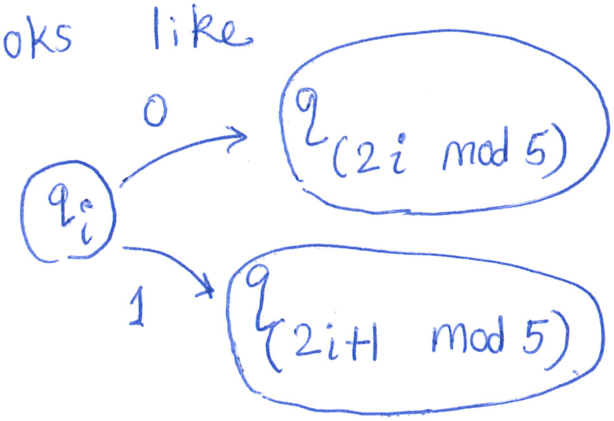
Observation If the string $w_1 w_2 \cdots w_k$ represents integer N in binary then

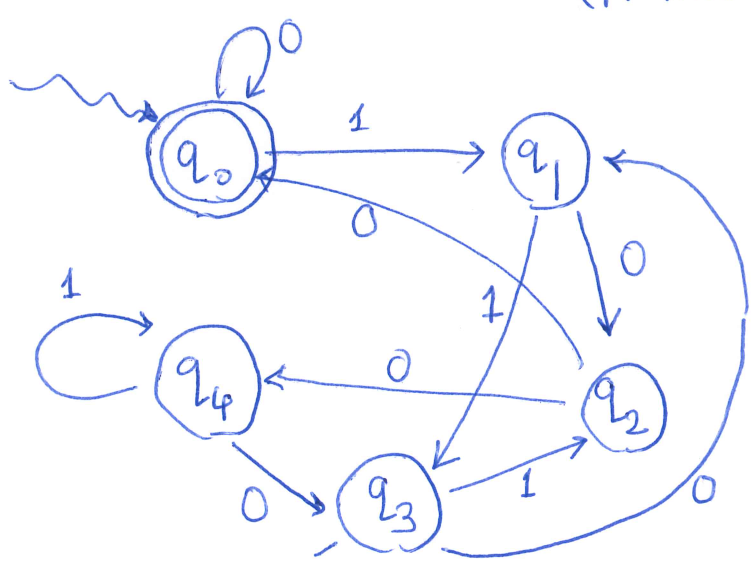
the string $w_1 w_2 \cdots w_k 0$ represents $2N$ in binary,

" $w_1 w_2 \cdots w_k 1$ " $2N+1$ " .

As before, after reading input $w_1 w_2 \cdots w_k$, the F.A., by design, will be in state $q_{(N \bmod 5)}$.

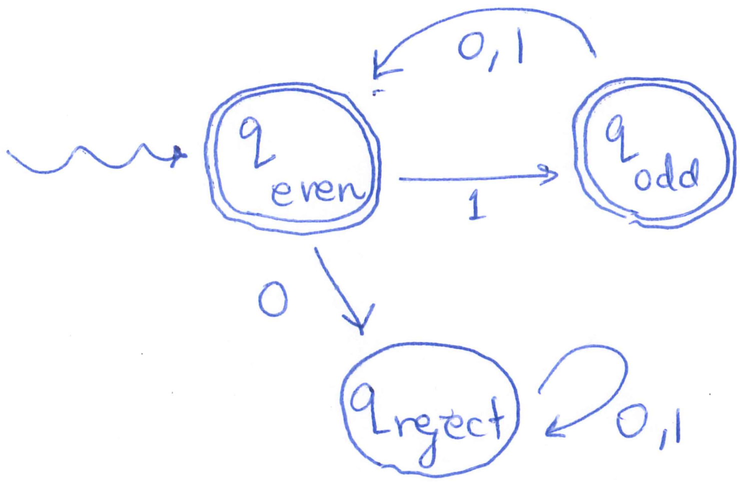
By above obeservation, a typical transition looks like





Question Design FA that recognizes

$L = \{ w \in \{0,1\}^* \mid w \text{ has } 1 \text{ in every odd position} \}$.



It is understood that $\varepsilon \in L$..

Regular Operators on Languages: $\cup, \circ, *$

Def Let A, B be languages (over some alphabet). The regular operators $\cup, \circ, *$ are:

$$A \cup B = \{ x \mid x \in A \text{ or } x \in B \}.$$

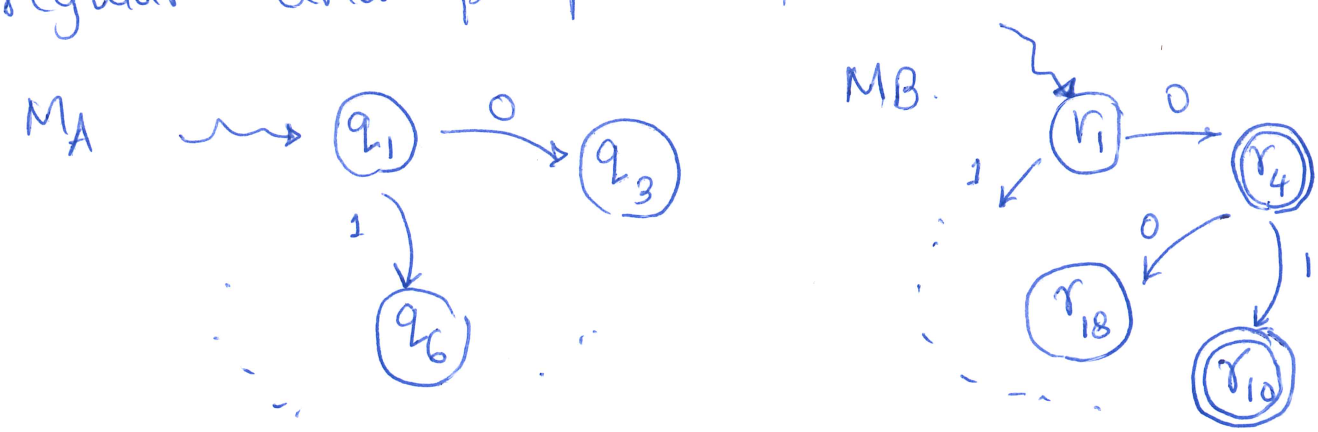
$$A \circ B = \{ x \cdot y \mid x \in A \text{ and } y \in B \}$$

$$A^* = \{ x_1 x_2 \cdots x_k \mid x_i \in A \ \forall\, 1 \le i \le k,\ k \ge 0 \}.$$

By definition $\varepsilon \in A^*$ (corresponds to $k=0$).

Theorem The class of regular languages is closed under the regular operators. I.e. if A, B are regular, so are $A \cup B$, $A \circ B$, A^*.

Proof (for $A \cup B$) We'll show that $A \cup B$ is regular and postpone proofs for $A \cdot B$, A^*.



Let M_A, M_B be F.A. that recognize A, B respectively.

We'll build a F.A. M that recognizes $A \cup B$.

For every input x (e.g. $x = 001001110$),

M accepts x $\Leftrightarrow$ Either M_A accepts x
OR M_B accepts x.

<u>First attempt</u> Try "running" first M_A on x
and then M_B on x
and accept if either accepts

However after running M_A on x, one runs out of the input.

<u>Second attempt</u> So we try to "run" both M_A and M_B on x <u>simultaneously</u> and accept if either accepts.

To run/simulate M_A, "remember" its state q_i.
" M_B, " r_j.

Hence, to run/simulate M_A, M_B simultaneously, "remember" (q_i, r_j).

Thus our F.A. M will have states as pairs (q_i, r_j) and it will simulate

M_A on the first co-ordinate

and M_B " second " of the pair and accept if either accepts.

Formal construction of M

Let $M_A = (Q_1, \Sigma, \delta_1, q_1, F_1)$

$M_B = (Q_2, \Sigma, \delta_2, r_1, F_2)$.

Then define $M = (Q_1 \times Q_2, \Sigma, \delta, (q_1, r_1), F)$

where

(1) $\delta : (Q_1 \times Q_2) \times \Sigma \to Q_1 \times Q_2$ is defined as

$$\delta((q,r), a) = (\delta_1(q,a), \delta_2(r,a))$$

$$\forall q \in Q_1, r \in Q_2, a \in \Sigma.$$

Let

(2) $F = F_1 \times Q_2 \cup Q_1 \times F_2$.

Exercise Show that M recognizes $A \cup B$. $\blacksquare$

Example $\Sigma = \{0,1\}$. We know that

$$A = \{ w \in \Sigma^* \mid |w| \text{ is even} \},$$

$$B = \{ w \in \Sigma^* \mid w \text{ has } 001 \text{ as consecutive substring} \}$$

are both regular, with the corresponding F.A. M_A, M_B with 2 and 4 states resp.

Hence $A \cup B = \{ w \mid |w| \text{ is even } \underline{\text{or}}\ w \text{ has } 001 \text{ as cons. substr.} \}$

is recognized by a FA with $2 \times 4 = 8$ states.

It turns out that showing that $A \cdot B$, A^* are regular (provided A, B are) is more difficult. We first define a new variant of F.A. called <u>Non-deterministic Finite Automata</u> (NFA), show that NFA are <u>equivalent</u> to F.A. (henceforth referred to as Deterministic FA (DFA). It is then easy to construct NFAs for $A \cdot B$ and A^*!