Mathematically, a function $f \quad x \to f(x)$

Alphabet: a finite set Σ e.g. $\Sigma = \{0,1\}$, $\Sigma = \{a,b,c\}$

Inputs and outputs are finite strings with symbols from the alphabet
e.g. $\Sigma = \{0,1\}$ $x=0$, $x=1$, $x=001$ ...

- ε denotes the empty string
- $\emptyset$ denotes the empty set
- Σ^n denotes all the strings of length n using the alphabet Σ.
- Σ^* denotes all the finite string using alphabet Σ.
  e.g. $\Sigma = \{0,1\}$ $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11 \cdots\}$

In theory of computation, we will focus on the so called **decision problem**.

Remark: Every problem can be converted to a decision problem

e.g. $(x, y) \longrightarrow \begin{cases} \text{accept if } y = \text{correct output} \\ \text{reject if } y \neq \text{correct output} \end{cases}$

We will interpret decision problems as **sets**, which in the context of theory of computation we call **languages**. $L = \{x \mid x \text{ is a Yes input}\}$

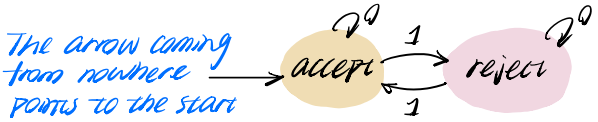
e.g. Alphabet $\Sigma = \{0,1\}$ Accept x if x contains an even number of 1's.
$L = \{\varepsilon, 0, 00 \cdots 11, 110, 101, 011, 1100 \cdots\}$

e.g. Alphabet $\Sigma = \{a,b\}$ Accept x if it has same number of occurances of a's and b's
$L = \{\varepsilon, ab, ba, aabb, abab, abba, baba, bbaa \cdots\}$

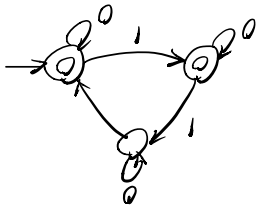
Deterministic Finite Automata (DFA)

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

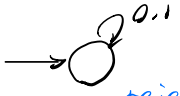
① $L = \{x \mid x \text{ has an even number of 1's}\} = \{\varepsilon, 0, 00, 11 \cdots\}$



② $L = \{x \mid$ number of 1's in x is $0 \pmod 2$ or $1 \pmod 2\}$

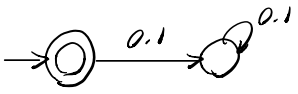


double circles are accept
single circles are reject.

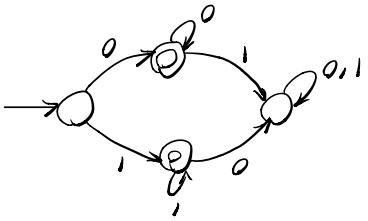
③ 

$L = \{\} = \phi$

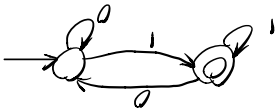
reject everything

④ 

$L = \{\varepsilon\}$

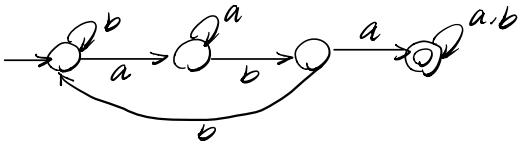
⑤ 

$L = \{x \mid x$ is all 1's or all 0's without $\varepsilon\}$

⑥ 

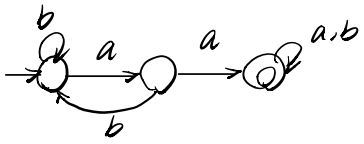
$L = \{x \mid x$ ends in $1\}$

⑦ Consider $\Sigma = \{a, b\}$



$L = \{x \mid x$ contains "aba"$\}$

⑧ Construct a DFA for $\Sigma = \{a, b\}$, $L = \{x \mid x$ contains "aa"$\}$

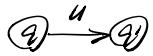


⑨ $L = \{x \mid x$ has the same number of a's and b's$\}$
Is there a DFA for L? NO

DEF A deterministic finite automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

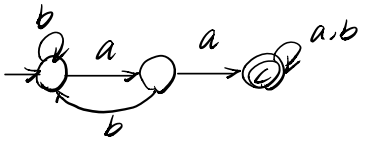
states Q is a finite set of state

alphabet Σ is a finite set

transition $\delta : Q \times \Sigma \to Q$ e.g. $\delta(q, u) = q'$ 

start $q_0 \in Q$ is the start state

accepts $F \subseteq Q$ is the set of accept states.



$Q = \{A, B, C\}$
$\Sigma = \{a, b\}$
$q_0 = A$
$F = \{C\}$

$\delta =$

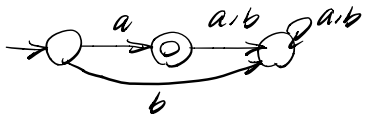
	a	b
A	B	A
B	C	A
C	C	C

Accept a string x if we finish at an accept state, otherwise, reject x.

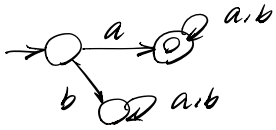
DEF The set of all strings accepted by a DFA M is called the **language** of M and is denoted by $L(M)$.

We say that DFA M **recognizes** L if $L = \{x \mid M \text{ accepts } x\}$ i.e. $L(M) = L$.

① Construct a DFA M with $L(M) = \{a\}$ $\Sigma = \{a, b\}$

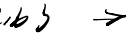


② Construct a DFA M with $L(M) = \{x \mid x \text{ starts with "a"}\}$, $\Sigma = \{a, b\}$



DEF A language over a finite alphabet Σ is called a **regular** language if can be recognized by a DFA

e.g. $\{x \mid x \text{ has same number of a and b}\}$ NOT regular

$\{a, b\}^* = \{\text{all finite strings with a,b}\}$ → 

regular

$L = \{x \mid x \text{ starts and ends with same letter}\}$ $\Sigma = \{0, 1\}$ regular

DEF The complement of a language L over an alphabet Σ is

$$\bar{L} = \Sigma^* \setminus L$$

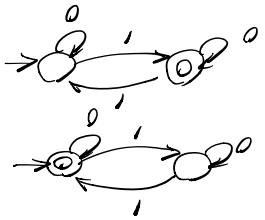
THM *If L is a regular language then $\bar{L}$ is a regular language*

Since L is a regular language, there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes L. That is $L(M) = L$. Then $\bar{M} = (Q, \Sigma, \delta, q_0, Q \setminus F)$ recognizes $\bar{L}$ and thus $\bar{L}$ is also regular. Here $\bar{M}$ is identical to M except that the roles of accept and reject states have switched.

Ex. $\Sigma = \{a, 1\}$

$L = \{x \mid x \text{ has an odd number of 1's}\}$

$\bar{L} = \{x \mid x \text{ has an even number of 1's}\}$



THM *If A and B are regular languages over an alphabet Σ, then $A \cup B$ is a regular language.*

A is recognized by $(Q, \Sigma, \delta, q_0, F)$ and B is recognized by $(Q', \Sigma, \delta', q_0', F')$
Then the following DFA recognizes $A \cup B$

- states: $Q \times Q' = \{(q, q') : q \in Q, q' \in Q'\}$
- alphabet: Σ
- transition: $\tilde{\delta}((q, q'), w) = (\delta(q, w), \delta'(q', w))$
- starts: (q_0, q_0')
- accepts: $(F \times Q') \cup (Q \times F')$

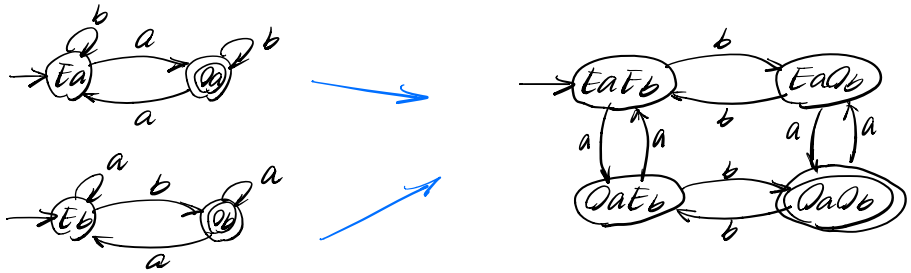
$(A \cap B \rightarrow F \times F')$

Remark $A \cap B = \overline{(\bar{A} \cup \bar{B})}$

Ex. $\Sigma = \{a, b\}$ $A = \{x \mid x \text{ has an odd number of a's}\}$

$B = \{x \mid x \text{ has an odd number of b's}\}$

$\Rightarrow$ Design $A \cup B = \{x \mid x \text{ has an odd number of a's and b's}\}$



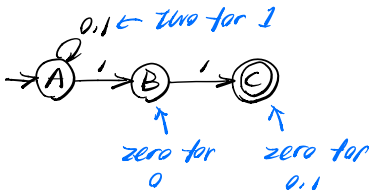
Nondeterministic Finite Automaton (NFA)

determinism: Every step of the computation follows in a unique way from the preceding one.

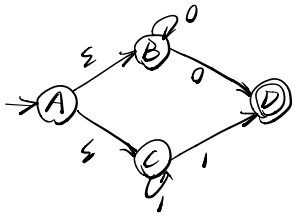
nondeterminism: At every step, we might face several choices. We'll accept a string if it's possible to make choices so that we will eventually end up in accept.

informal

NFA: For every state $q \in Q$ and symbol $s \in \Sigma$, there might be zero, one (DFA) or several arrows with label s leaving q. Moreover, we might have arrows with label ε.

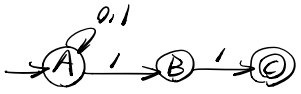


$\Sigma = \{0,1\}$



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

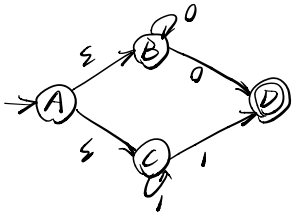
DEF (informal) An NFA accepts a string w if there exists a way of traversing the arrows (using w) to end up in an accept state.
In other words, there is a path that starts at q_0 and ends at an accept state s.t. reading the label on the arrow as we traverse the path will result in the word w.

① 

$w = 1011$ accepted

$A \xrightarrow{1} A \xrightarrow{0} A \xrightarrow{1} B \xrightarrow{1} C$

$L = \{x \mid x \text{ ends in "11"}\} = \{11, 011, 111, \dots\}$

② 

$L = \{x \mid x \text{ is all 0's or 1's}\}$
(NOT including ε)

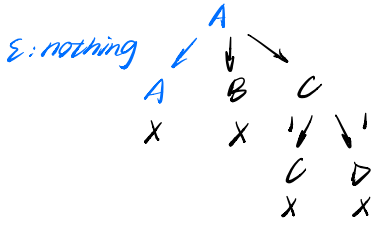
(i) $w = 00$ accepted

$A \xrightarrow{\varepsilon} B \xrightarrow{0} B \xrightarrow{0} D$

(ii) $w = 111$ accepted

$A \xrightarrow{\varepsilon} C \xrightarrow{1} C \xrightarrow{1} C \xrightarrow{1} D$

(iii) $w = 10$ reject



Notation $P(Q)$ = The set of all subsets of Q

e.g. $P(\{A,B\}) = \{\emptyset, \{A\}, \{B\}, \{AB\}\}$

DEF An NFA is a 5-tuple $N = (Q, \Sigma, \delta, q_0, F)$

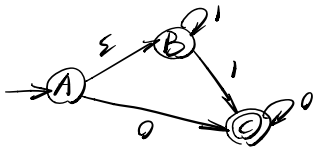
states Q is a finite set of state

alphabet Σ is a finite set

transition $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to P(Q)$

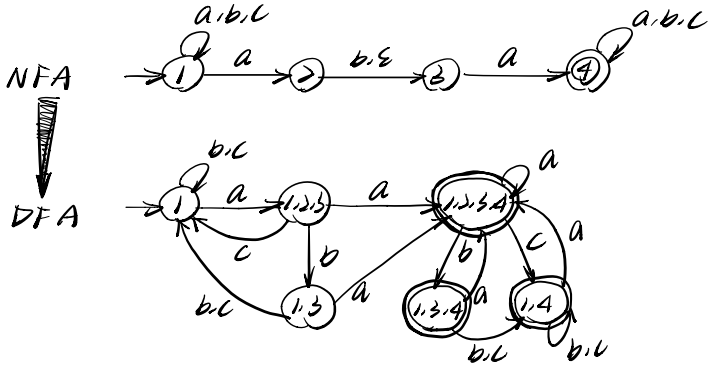
start $q_0 \in Q$ is the start state

accepts $F \subseteq Q$ is the set of accept states.



δ	ε	0	1
A	$\{B\}$	$\{C\}$	$\emptyset$
B	$\emptyset$	$\emptyset$	B
C	$\emptyset$	$\{C\}$	$\emptyset$

Remark Every DFA is an NFA i.e. DFA $\subseteq$ NFA



THM Every NFA N has an equivalent DFA M $L(N) = L(M)$

$\Rightarrow$ NFA's can only recognize regular languages.

Let $N = (Q, \Sigma, \delta, q_0, F)$.

For every $S \subseteq Q$, let $E(S) = \{q \in Q \mid$ there is a path of ε's from S to $q\}$

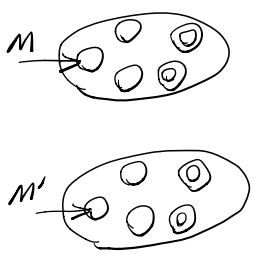
We'll construct $M = (Q', \Sigma', \delta', q_0', F')$.

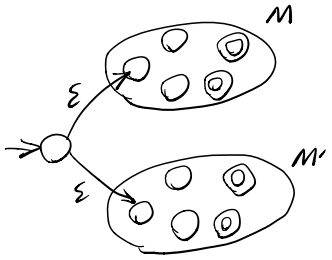
- $Q' = P(Q)$; $\Sigma' = \Sigma$; $q_0' = E(\{q_0\})$
- For $S \in Q'$ (meaning $S \in Q$), define $\delta'(S, a) = E\left(\bigcup_{A \in S} \delta(A, a)\right)$
- $F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$

The DFA is designed so that if we can reach to a state $S \subseteq Q$ for NFA it mean that in the NFA, we can reach to any single state in S that we wish using the same word.

Ex. Use NFA's to show that if A, B are regular languages over Σ, then so is $A \cup B$.

Let M be a DFA for A, M' a DFA for B.





DEF For languages A and B over Σ, let $AB = \{xy \mid x \in A, y \in B\}$

concatenation

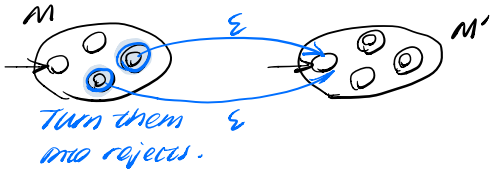
e.g. $A = \{\varepsilon, 00\}$ $B = \{1, 11\}$

$\Rightarrow AB = \{1, 11, 001, 0011\}$

THM If A and B are regular languages, then so is AB.

Let M be a DFA for A and M' be a DFA for B

It suffices to construct an NFA for AB



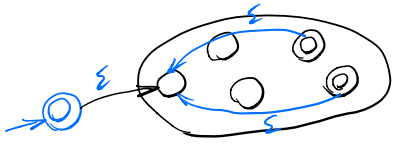
DEF For a language A over Σ let $A^n = \underbrace{AA \cdots A}_{n \text{ times}}$ $A^0 = \{\varepsilon\}$

$A^* = \bigcup_{n=0,1,\ldots} A^n$

e.g. $A = \{0, 11\}$ $A^* = \{\varepsilon, 0, 11, 00, 011, 110, 1111, 000 \ldots\}$

THM If A is a regular language, so is A^*.

Construct a DFA for A



Summary If A, B are regular languages over Σ, then so are

1. $\bar{A} = \Sigma^* \setminus A$
2. $A \cup B$
3. $A \cap B$
4. AB
5. A^*

1. If A is regular language, is it true that any subset $B \subseteq A$ is also regular?

   NO! If this was true, then every language $B \subseteq \Sigma^*$ would be regular.

2. A is regular, B is finite $\Rightarrow$ $A \setminus B$ regular?

   $A \setminus B = A \cap \bar{B}$

   (A: regular, $\bar{B}$: regular)

Regular Expressions

DEF Consider an alphabet Σ. We say that R is a regular expression if

1. $R = a$ for $a \in \Sigma$ — $L(R) = \{a\}$
2. $R = \varepsilon$ — $L(R) = \{\varepsilon\}$
3. $R = \phi$ — $L(R) = \phi$

for reg expression R_1 and R_2

4. $R = (R_1 \cup R_2)$ — $L(R) = L(R_1) \cup L(R_2)$
5. $R = R_1 R_2$ — $L(R) = L(R_1) L(R_2)$
6. $R = R_1^*$ — $L(R) = L(R_1)^*$

(priorities ↑)

Ex. 1. $L(0^* 1 0^*) = \{x \mid x \text{ contains exactly one } 1\} = \{1, 01, 10, 010 \cdots\}$

2. $L((0^* 10)^*) = \{\varepsilon, 10, 010, 1010 \cdots\}$
3. $L(\Sigma^* 001 \Sigma^*) = \{x \mid x \text{ contains } 001\}$
4. $L((\Sigma\Sigma\Sigma)^*) = \{x \mid \text{the length of } x \text{ is a multiple of } 3\}$
5. $L((000)^* \cup (111)^*) = \{\varepsilon, 000, 111, 000000, 111111 \cdots\}$
6. $L(1^* \phi) = \phi$ $\quad L(1^* \varepsilon) = 1^*$
7. $L(\phi^*) = \{\varepsilon\}$

(k) $0\Sigma^* \cup \Sigma^* 0 = \{w \mid w \text{ starts or ends with } 0\}$

(l) $\{w \mid |w| \geq 2 \text{ and the second last letter is } 1\}$

$= \Sigma^* 1 \Sigma$ Note $\Sigma^* 1^* = \Sigma^*$

THM If R is a regular expression then $L(R)$ is a regular language

n=1 If the length is 1 then it's one of the three cases (a, ε, ϕ) and we have seen these are regular languages.

n→n+1 If R has length $< k$, then $L(R)$ is regular

If length of R is k, then $R = R_1 \cup R_2$, $R = R_1 R_2$, $R = R_1^*$

By the induction hypothesis, $L(R_1)$ and $L(R_2)$ are regular

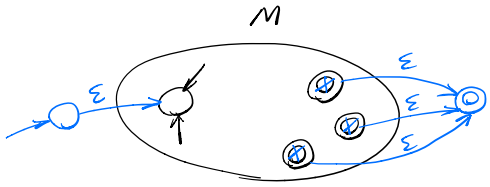
$\Rightarrow L(R_1) \cup L(R_2)$ is regular

The other two cases are similar.

THM Every regular language can be described by a regular expression.

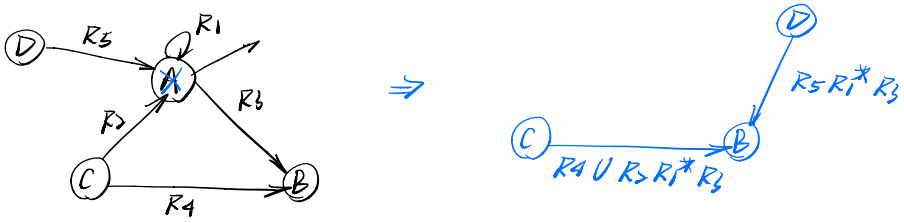
Let L be a regular language and let M be a DFA that recognizes L.

step 1 unique accept, no arrow out
unique start, no arrows in.

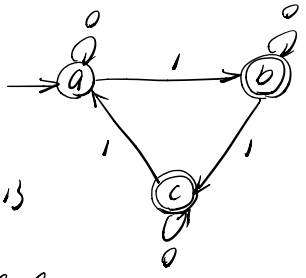


step 2 No multiple labels

step 3 Eliminating internal states one by one

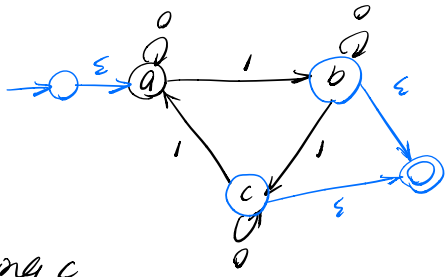


Ex.

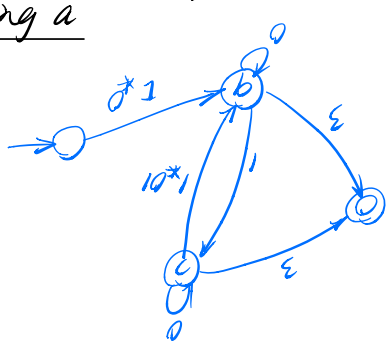


$\Sigma = \{0,1\}$

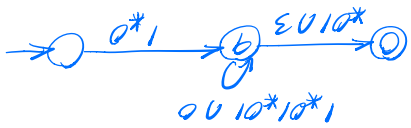
step 1



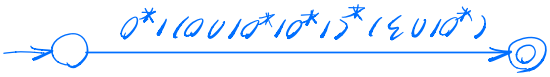
removing a



removing c



removing b



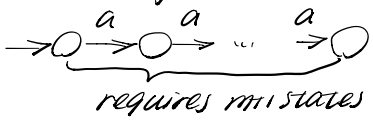
DFA ⟺ NFA ⟺ regular expression

Prove NOT Regular

Proposition The language $L = \{a^n b^n \mid n \in \mathbb{N}\}$ over $\Sigma = \{a,b\}$ is NOT regular.

Suppose otherwise. Let M be a DFA for L and m be the number of states in M. Consider $w = a^m b^m \in L$, and run the DFA M on this. In particular, focus on the first part (a^m).

Since we have only m states, we have to revisit a state when considering a^m.



That is, there is $0 \le j < j+k \le m$ s.t. a^j and a^{j+k} end up at the same state
$\Rightarrow$ Contradiction bc we need to accept $a^j b^j$ and reject $a^{j+k} b^j$.

Ex. Show that $L = \{w \mid w$ has the same number of 0's and 1's$\}$ NOT regular

For some $0 \le j < j+k \le \overset{\text{\# states}}{m}$, both 0^j and 0^{j+k} end up in the same state
$\Rightarrow$ Contradiction, $0^j 1^j$ and $0^{j+k} 1^j$ end up in the same state.

Ex. $L = \{a^{n^2} \mid n \geq 0\} = \{\varepsilon, a, aaaa, \cdots\}$ NOT regular

For some $0 \leq i < i+k \leq m$ (# states), both a^i and a^{i+k} end up in the same state

since $a^i a^{m^2-i} = a^{m^2} \in L$, $a^{i+k} a^{m^2-i} = a^{m^2+k} \in L$

Note that $m^2 < m^2+k \leq m^2+m < (m+1)^2$ $\Rightarrow$ m^2+k cannot be full square

$\Rightarrow$ Contradiction.

Ex. $L = \{a^n \mid n \text{ is a prime}\}$ $\Sigma = \{a\}$

For some $0 \leq i < i+k \leq m$ (# states), both a^i and a^{i+k} end up in the same state

Let p be a prime large than m

$a^i a^{p-i} = a^p \in L$

$\Rightarrow a^{i+k} a^{p-i} = a^{p+k} \in L$

$\Rightarrow a^{i+k+k} a^{p-i} = a^{p+2k} \in L$

$\cdots$ $\cdots$

$\Rightarrow a^{i+pk} a^{p-i} = a^{p+pk} \in L$

But $p+pk = p(1+k)$ not prime $\Rightarrow$ contradiction

Ex. $L = \{0^r 1^s \mid r > s\}$ NOT regular

For some $0 \leq i < i+k \leq m$ (# states), both 0^i and 0^{i+k} end up in the same state

$0^i 1^i \notin L$ $0^{i+k} 0^i \in L$ $\Rightarrow$ contradiction

Pumping Lemma

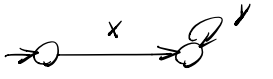
L is a regular language $\Rightarrow$ then there is a number m s.t. if $s \in L$ and $|s| \geq m$, then we can divide s into $s = xyz$ s.t.

① $|xy| \leq m$

② $|y| > 0$

③ $xy^i z \in L$ for every $i \geq 0$

proof Take m to be the number of states in a DFA for L



and whatever comes after xy, call it z.