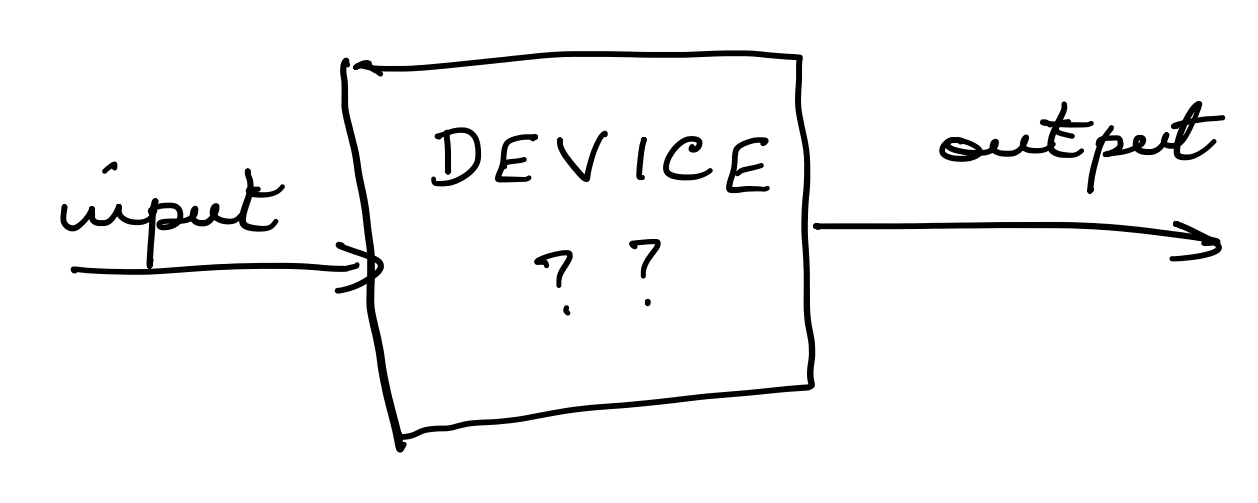
Lecture 2 DFA

Tuesday, January 12, 2021 9:46 AM



FIRST MODEL:

Finite memory machines

Finite state machines

input → sequence of symbols

collection of symbols : alphabet

each symbol : letter

Σ : alphabet

$\Sigma = \{a, b\}$ or $\Sigma = \{a\}$

$\Sigma = \{0, 1\}$ or $\Sigma = \{0, 1, 2, \dots 9\}$

words or strings are sequences of letters: $\Sigma = \{a, b\}$

a, abba, baba, aaaaab, ε (empty word)

Σ^* : set of all possible words

If $\Sigma \neq \emptyset$ then Σ^* is infinite as a set. We do not have infinitely long words.

$L \subseteq \Sigma^*$

↳ a language.

L can be finite or infinite.

Our DFA are going to read words one symbol at a time from left to right with no backtracking.

At the end the DFA is going to output "accept" or "reject".

The machine is telling you a set of words that it accepts i.e. is defining a language.

Q: Is it possible to design a DFA for every language?

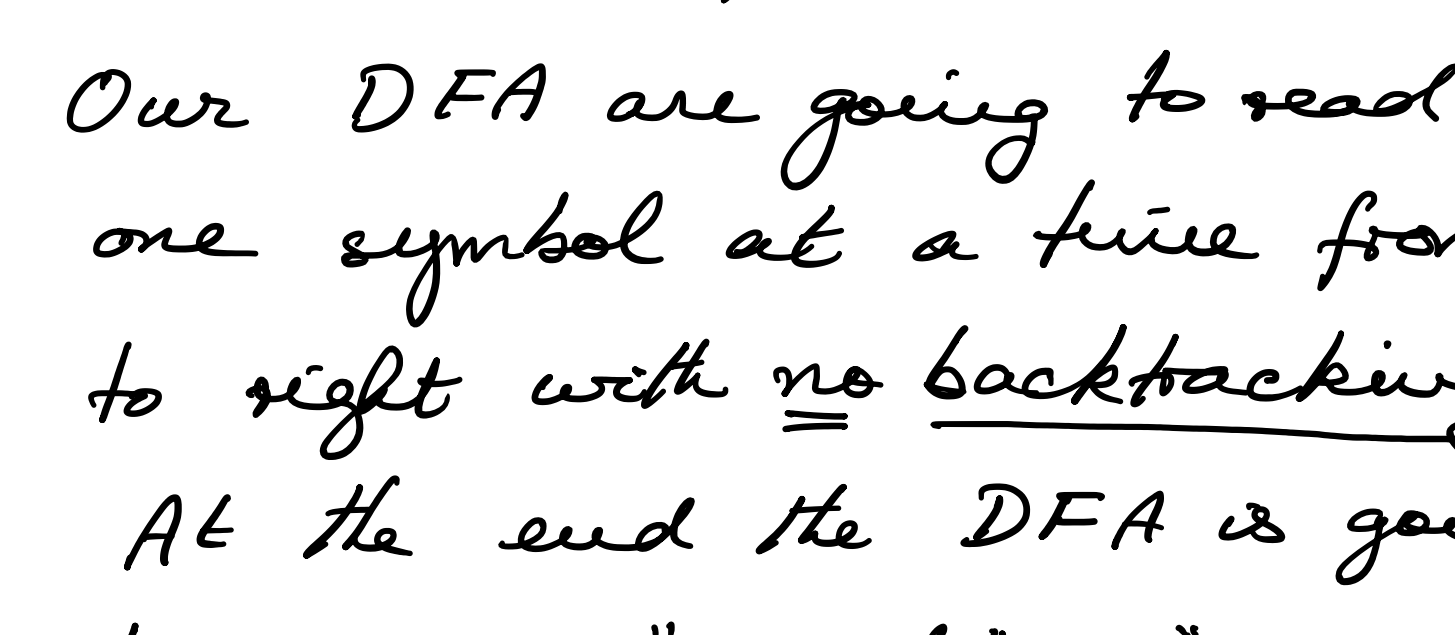
A: NO!

MACHINES :

states (finitely many)

transitions : jump between states

We will give a formal def. soon but let's draw pictures first:



aab ✗ $s_0 \xrightarrow{a} s_1 \xrightarrow{a} s_2 \xrightarrow{b} s_2$

aba ✓ $s_0 \xrightarrow{a} s_1 \xrightarrow{b} s_3 \xrightarrow{a} s_3$

set of states; s_0, s_1, s_2, s_3

start state : s_0

transitions are shown as labelled arrows:

for every letter from every state there is 1 arrow.

$s_2 \xrightarrow{a} s_2$ $\quad s_2 \xrightarrow{b} s_2$

Only one arrow for each letter the machine is deterministic.

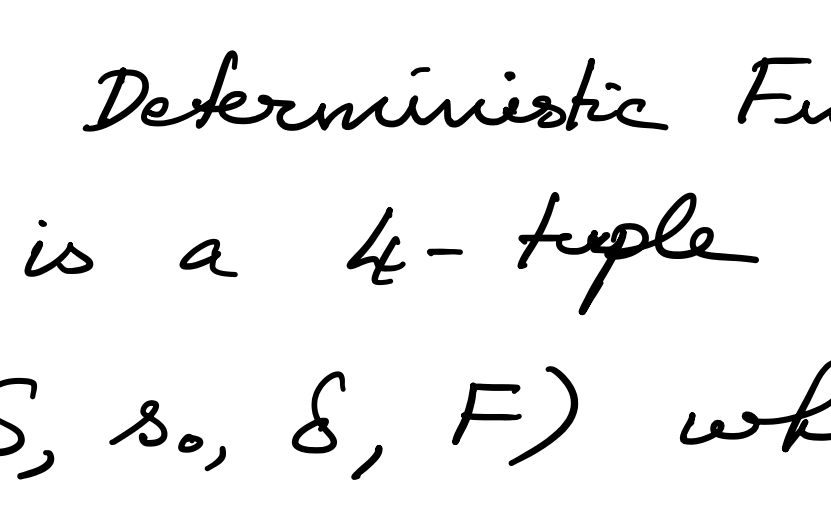
The state with the double circle is an accept state.

If the m/c ends up in an accept state at the end of the string, the string is accepted.

This machine recognizes all words that start with ab

FORMAL DEF:

S: set of states (finite)



$\delta(s, a) = t$

$\delta : S \times \Sigma \longrightarrow S$

↳ δ is a (total) function

Def A Deterministic Finite Automaton (DFA) is a 4-tuple

(S, s_0, δ, F) where

S is a finite set of states

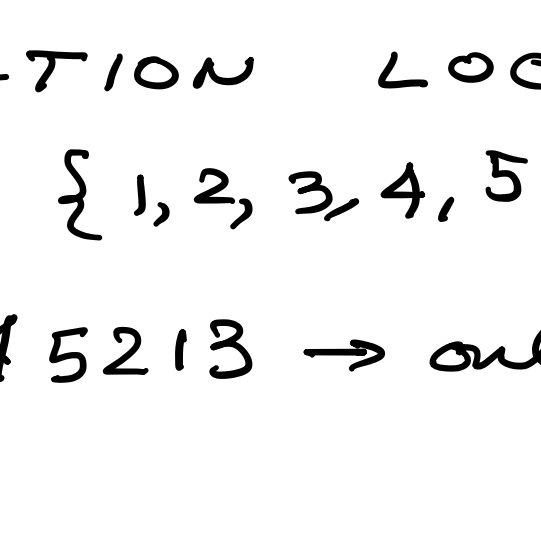
$s_0 \in S$ is the start state

$\delta : S \times \Sigma \longrightarrow S$ is the transition function

$F \subseteq S$ accept states (final states)

Def A language that can be recognized by a DFA is called a regular language.

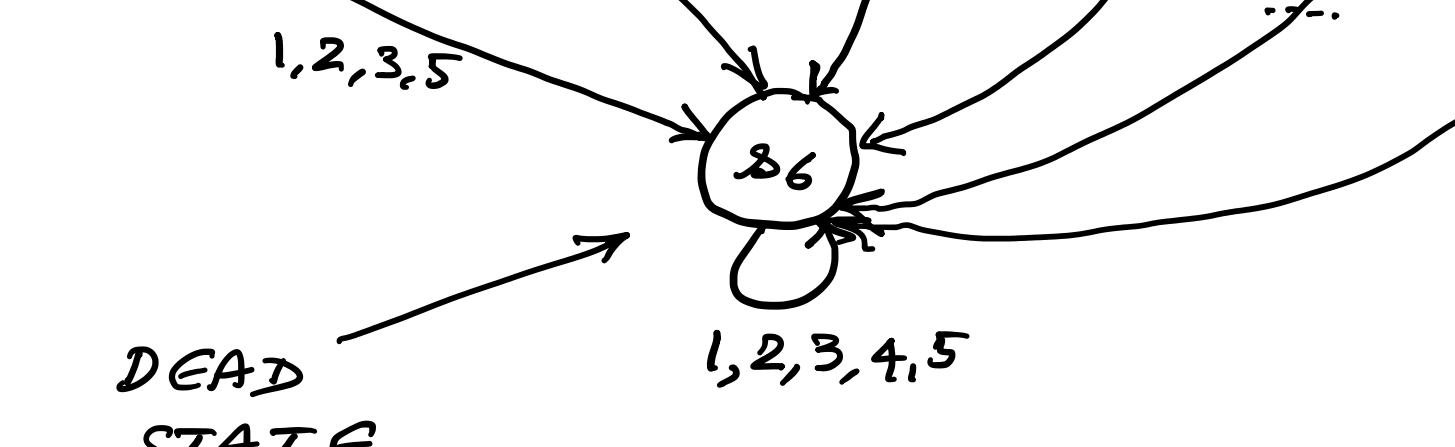
REMARK When I ask you to design a DFA for L it must accept every word in L and reject every word that is not in L.



It accepts all of Σ^*.

If the start state is an accept state then and only then ε is in the language.

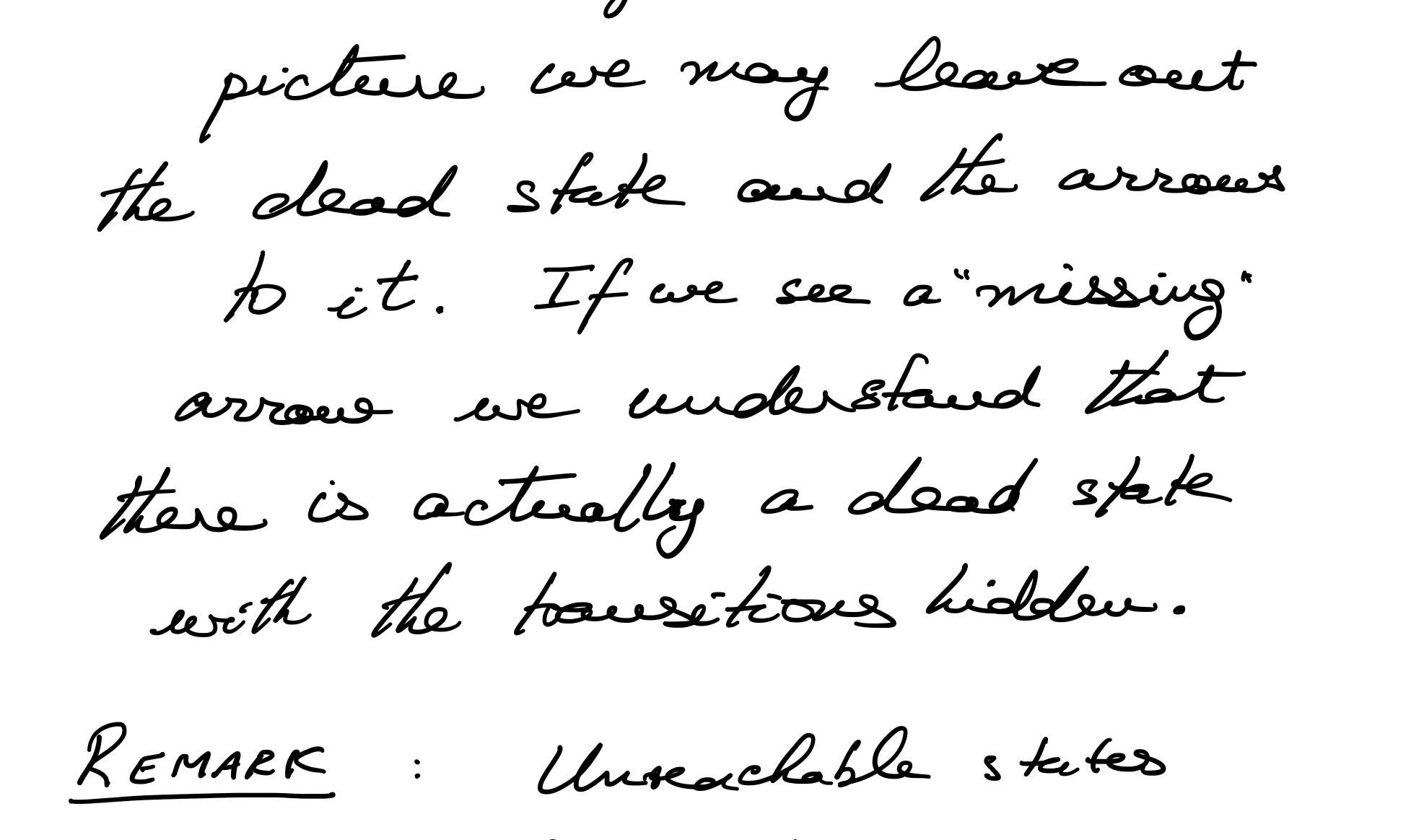
$\Sigma = \{a, b, c\}$



COMBINATION LOCK

$\Sigma = \{1, 2, 3, 4, 5\}$

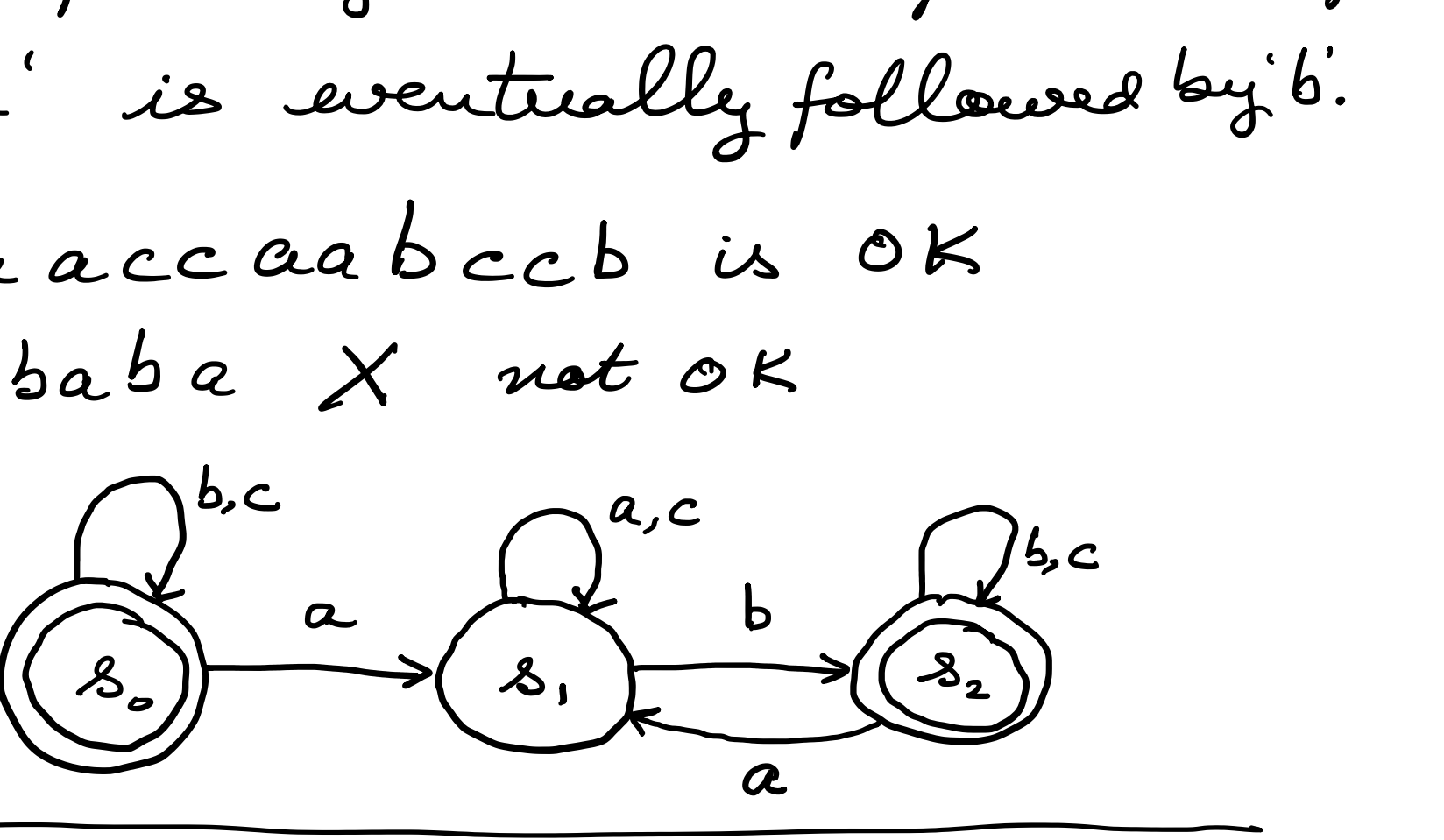
45213 → only valid comb.



In order to get a less cluttered picture we may leave out the dead state and the arrows to it. If we see a "missing" arrow we understand that there is actually a dead state with the transitions hidden.

REMARK : Unreachable states

They are allowed but useless so we usually remove them.



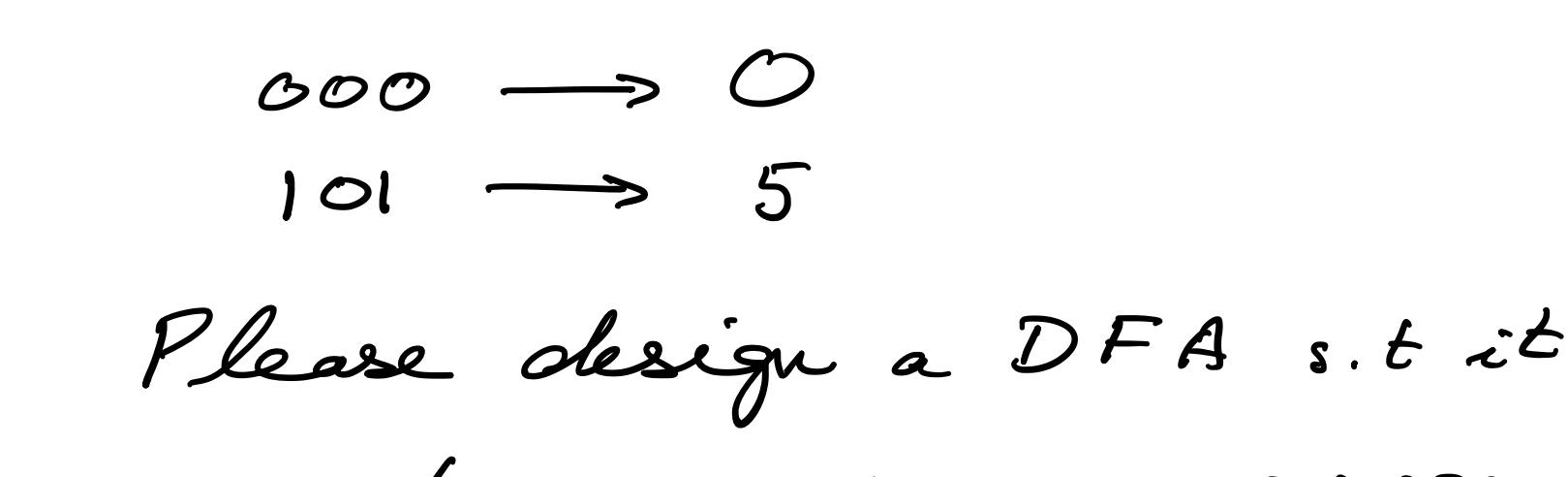
WE WILL REMOVE THEM.

$\Sigma = \{a, b, c\}$

EXAMPLE Design a m/c s.t it accept only those strings s.t every 'a' is eventually followed by 'b'.

aaaccaabccb is OK

ababa ✗ not OK



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

we will interpret the strings as binary numbers

$\varepsilon \longrightarrow 0$

$000 \longrightarrow 0$

$101 \longrightarrow 5$

Please design a DFA s.t it accepts a sequence corresponding to a binary number that is divisible by 3.

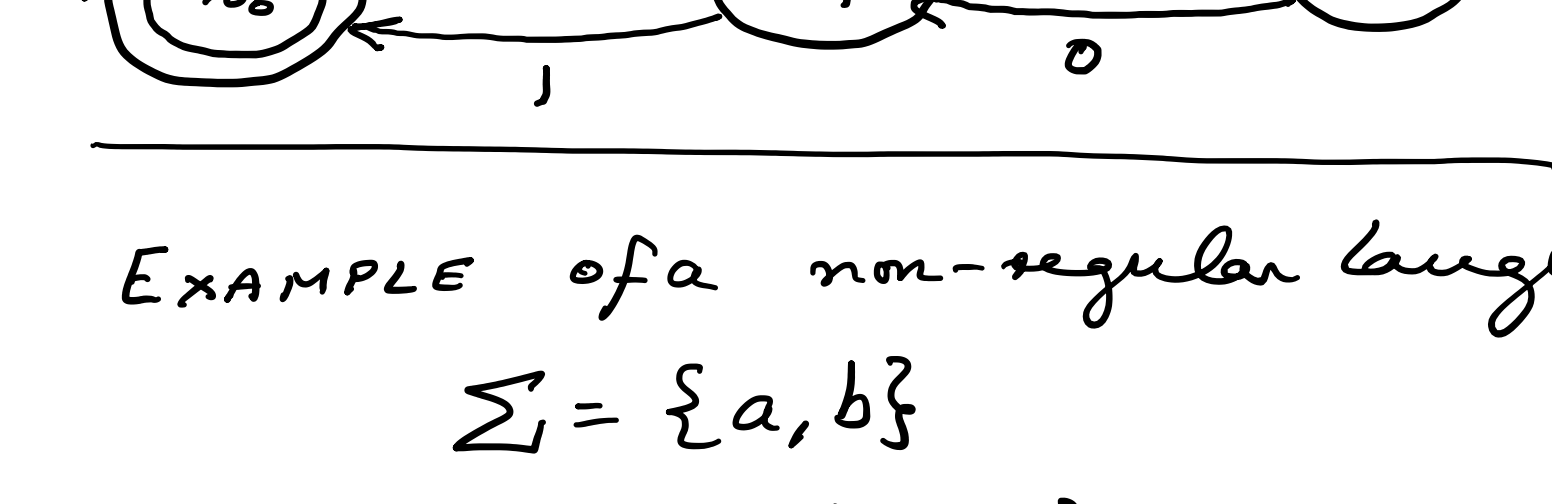
We are reading the string from left to right.

suppose the number you have seen so far is n

- then you read 0, what is the new number? $2n$

OR then you read 1, what is the new number? $2n+1$

The states of our DFA will encode the mod 3 value of the number seen so far



EXAMPLE of a non-regular language

$\Sigma = \{a, b\}$

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

a^n means 'a' repeated 'n' times

ε, ab, aabb, aaabbb, ...

FINITE AUTOMATA CANNOT COUNT!

They can only count mod some finite number.

If L is finite is it necessarily regular? YES!!

PROVE IT!