DFA & NFA

DFA & NFA

- In automata theory, a finite-state machine is called a deterministic finite automaton (DFA), if
- each of its transitions is *uniquely* determined by its source state and input symbol, and
- reading an input symbol is required for each state transition.
- A **nondeterministic finite automaton** (**NFA**), or **nondeterministic finite-state machine**, does not need to obey these restrictions.

DFA

Formal Example of DFA

Example 2.

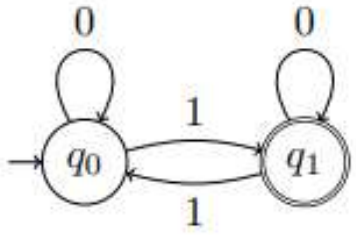


Figure 4: Transition Diagram of DFA

Formally the automaton is $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$ where

$$\delta(q_0, 0) = q_0 \qquad \delta(q_0, 1) = q_1$$
$$\delta(q_1, 0) = q_1 \qquad \delta(q_1, 1) = q_0$$

Ex

- Automaton accepts all strings of 0s and 1s
- Automaton accepts strings ending in 1
- Automaton accepts strings having an odd number of 1s
- Automaton accepts strings having an odd number of 1s and odd number of 0s

Solution

- https://courses.engr.illinois.edu/cs373/sp2013/Lectures/lec02.pdf

DFA Applications

- grep
- Thermostats
- Coke Machines
- Elevators
- Train Track Switches
- Security Properties
- Lexical Analyzers for Parsers

NFA

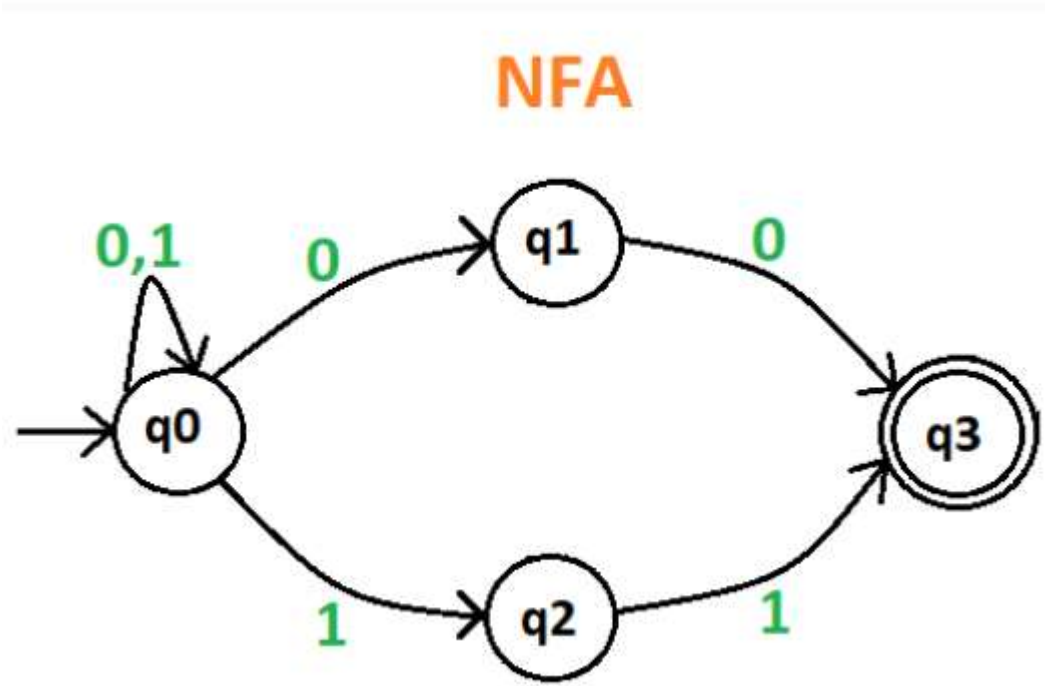
- NFA stands for **non-deterministic finite automata**.
- It is easy to construct an NFA than DFA for a given regular language.
- The finite automata are called NFA when there exist many paths for specific input from the current state to the next state.
- In particular, every DFA is also an NFA.
- Every NFA is not DFA, but each NFA can be translated into DFA.

EX

- Draw a deterministic and non-deterministic finite automaton which accept 00 or 11 at the end of a string containing 0, 1 in it

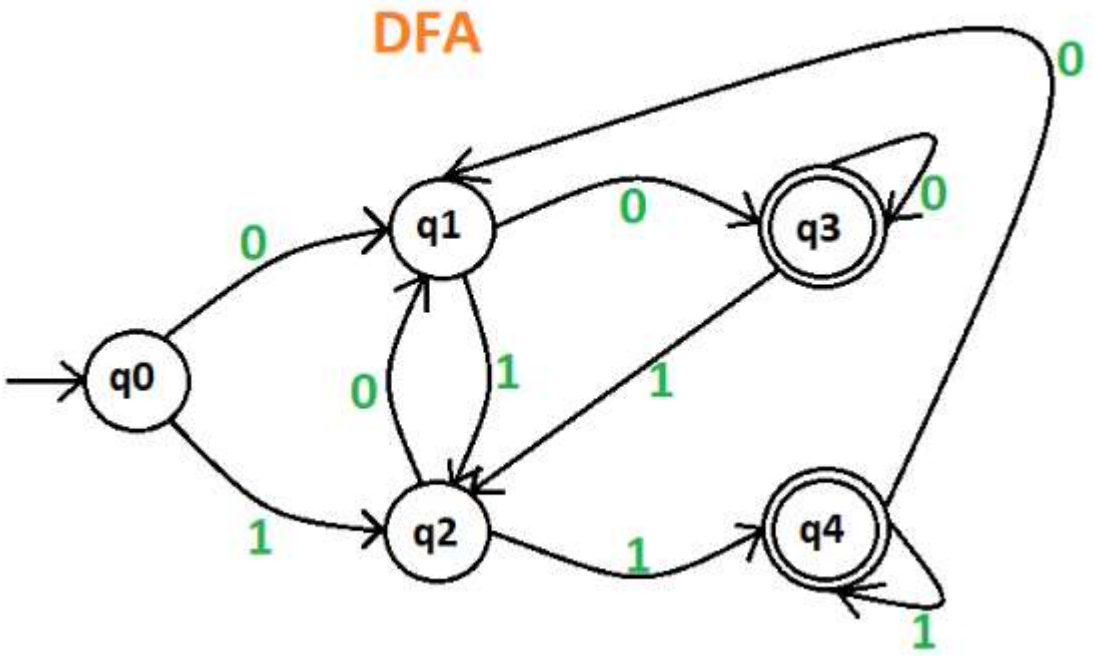
EX

- Draw a deterministic and non-deterministic finite automate which accept 00 and 11 at the end of a string containing 0, 1 in it



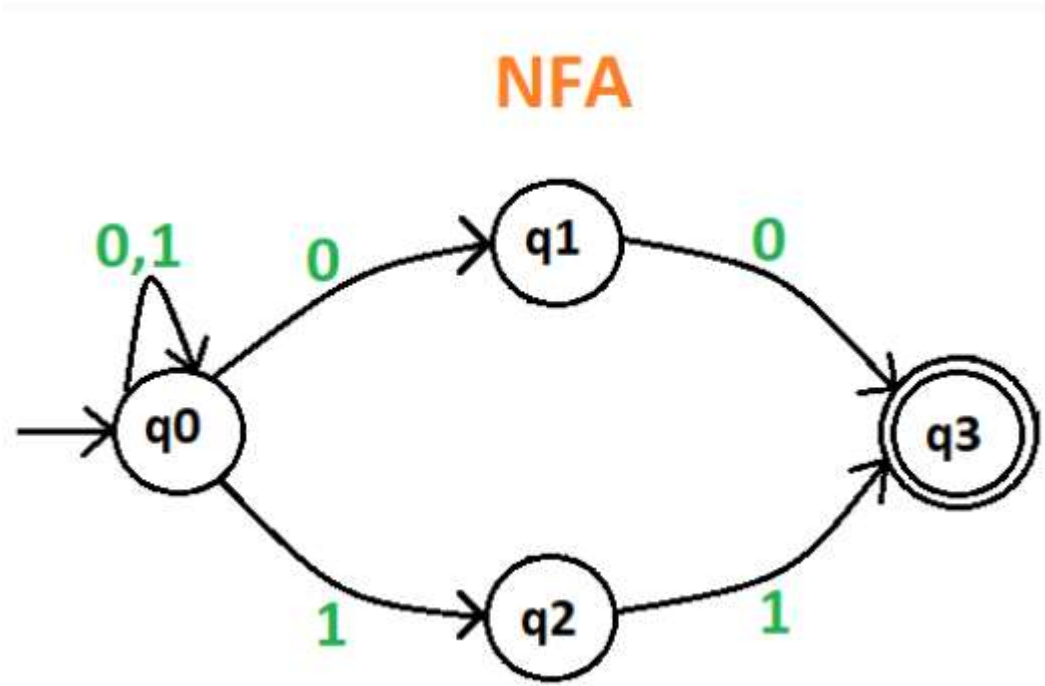
EX

- Draw a deterministic and non-deterministic finite automate which accept 00 and 11 at the end of a string containing 0, 1 in it



EX

- Draw a deterministic and non-deterministic finite automate which accept 00 and 11 at the end of a string containing 0, 1 in it



NFA Ex

- https://www.javatpoint.com/examples-of-non-deterministic-finite-automata

Adv of NFA

- It is important because NFAs can be used **to reduce the complexity of the mathematical work required to establish many important properties in the theory of computation**.
- For example, it is much easier to prove closure properties of regular languages using NFAs than DFAs.

Acceptance/Recognition

- The language accepted or recognized by a DFA M over alphabet Σ is $L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$.
- A language L is said to be accepted/recognized by M if $L = L(M)$.

Acceptance/Recognition

$L(M)$, *language accepted* by a finite automaton M

consists of all strings u over its alphabet of input symbols satisfying $q_0 \xrightarrow{u}{}^{*} q$ with q_0 the start state and q some accepting state. Here

$$q_0 \xrightarrow{u}{}^{*} q$$

means, if $u = a_1 a_2 \ldots a_n$ say, that for some states $q_1, q_2, \ldots, q_n = q$ (not necessarily all distinct) there are transitions of the form

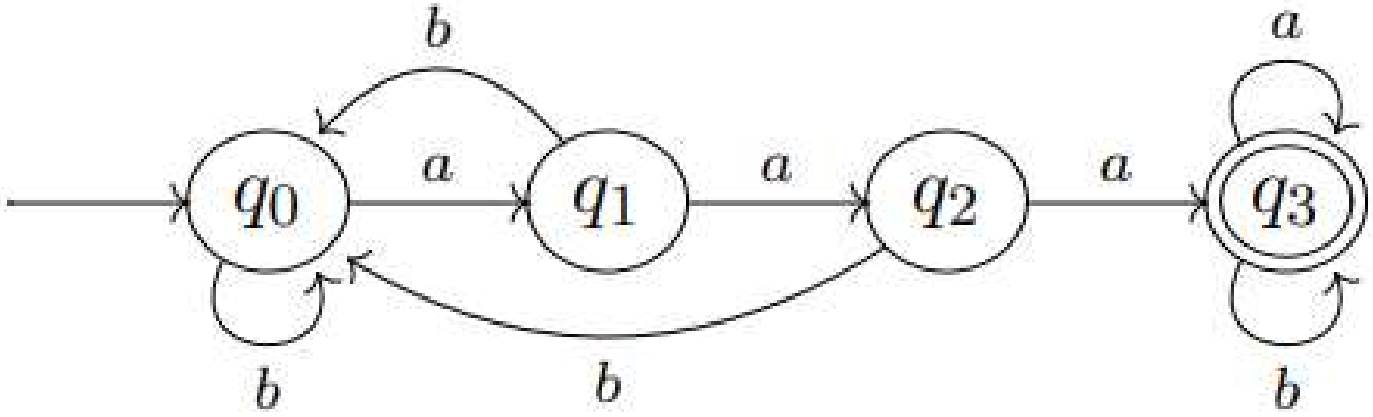
$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots \xrightarrow{a_n} q_n = q.$$

- A (formal) language L over an alphabet Σ is just a set of strings in Σ^*.
- Thus any subset $L \subseteq \Sigma^*$ determines a language over Σ.

Theorem (Kleene)

- The class of regular languages is exactly the same as the class of languages accepted by DFAs

Identify language?

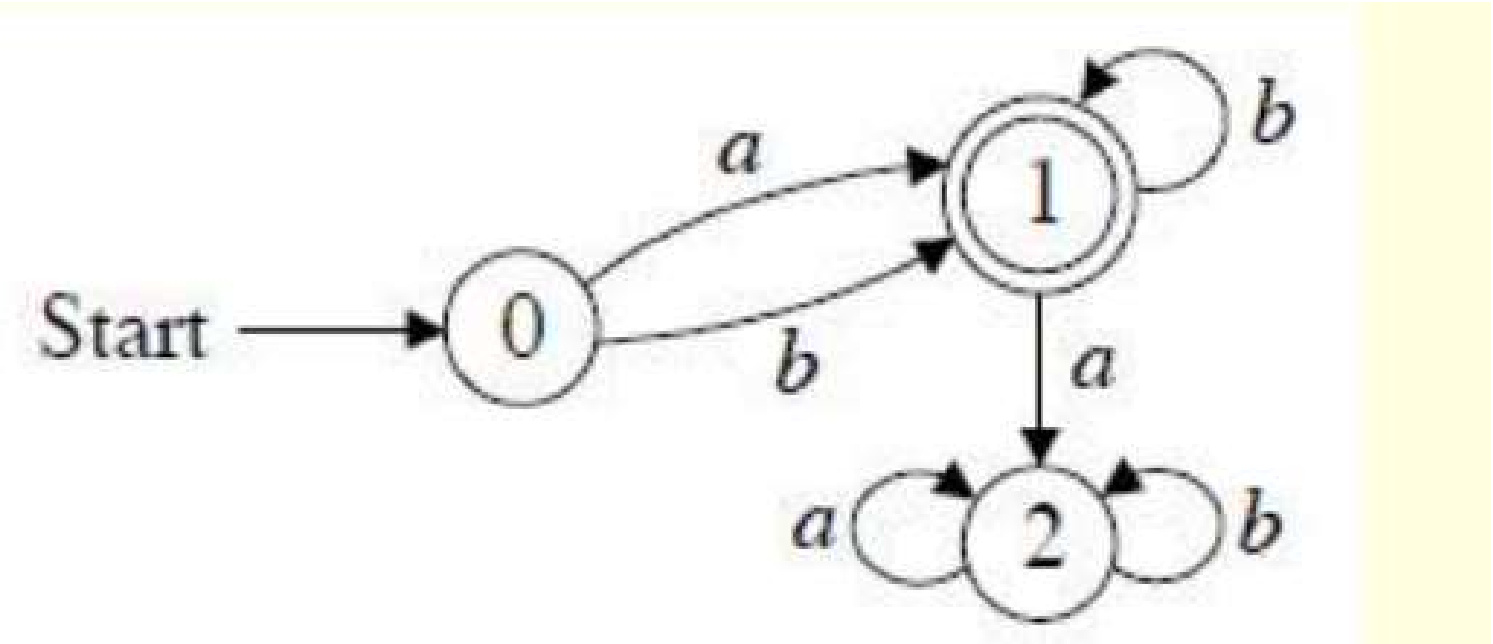


Identify language?

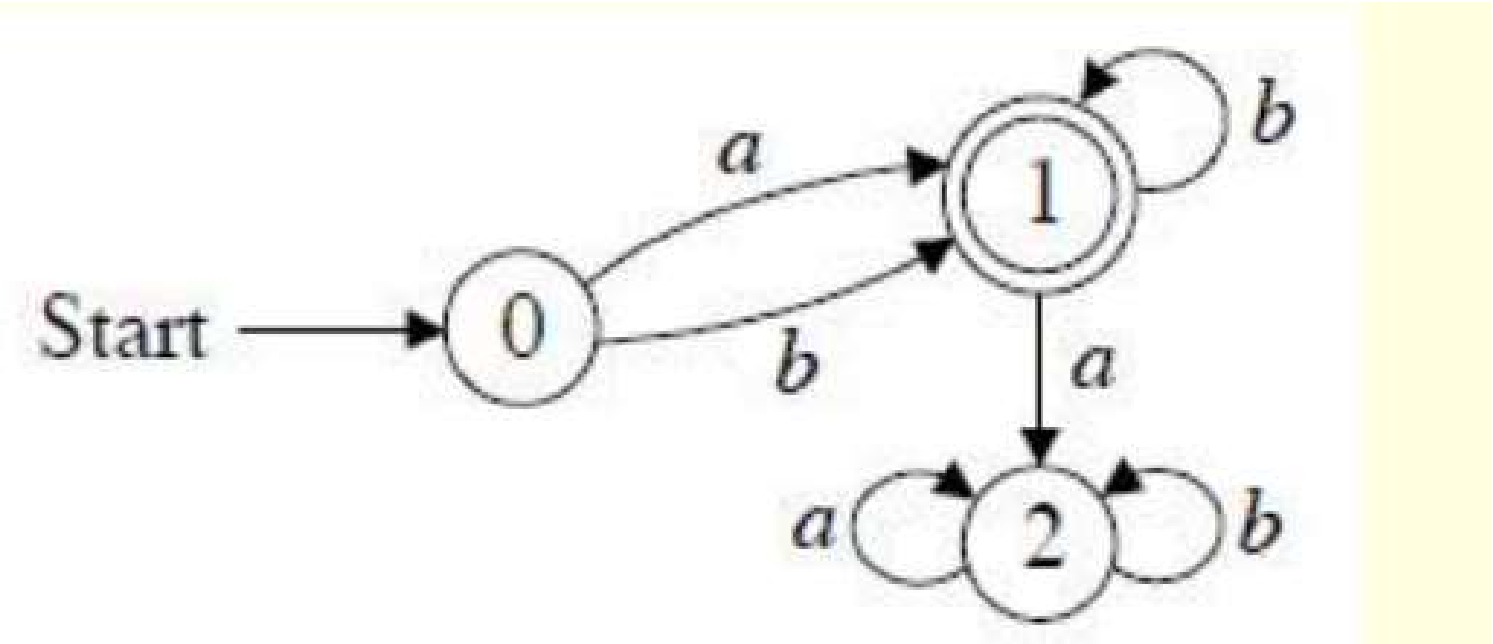
- L(M) = {u | u contains three consecutive a's}.

- RE
- r = (a|b)$*$ aaa (a|b)$*$

Identify language and RE?

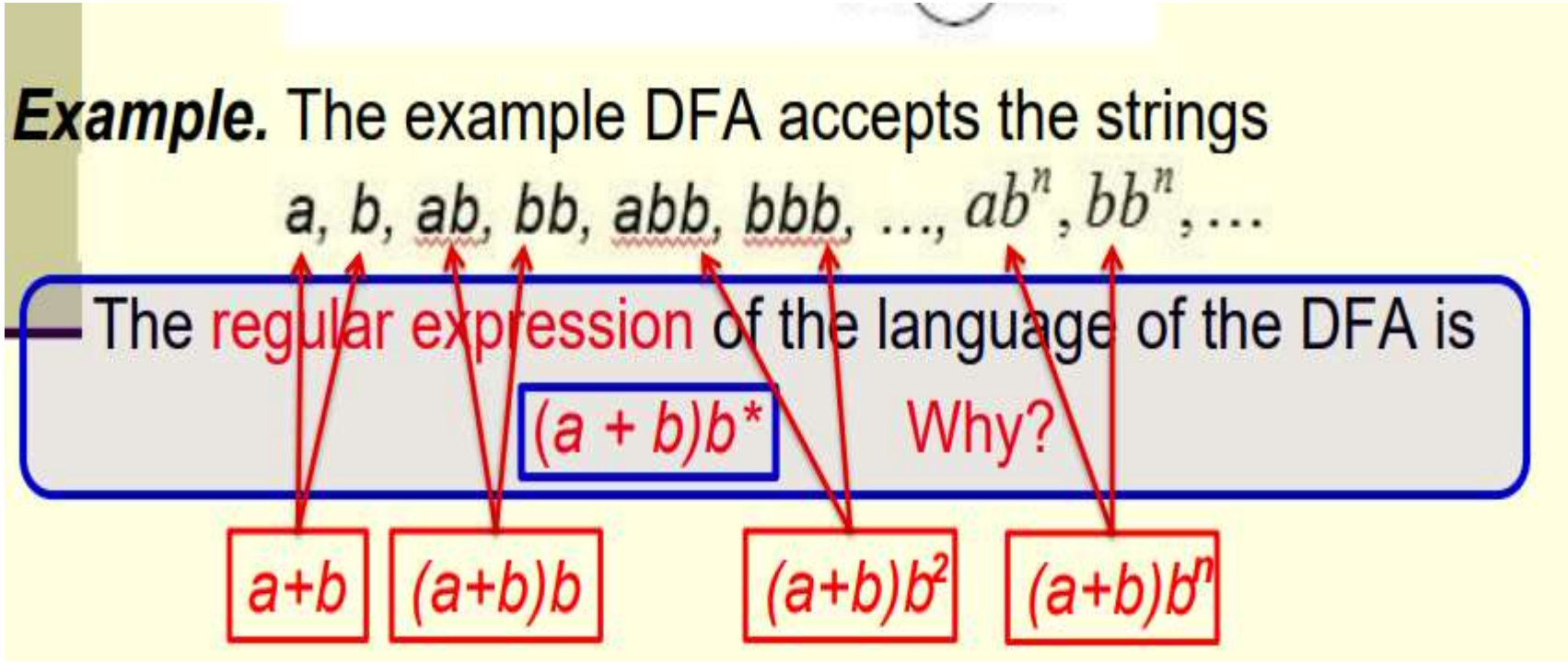


Identify language?



The language of the DFA is
{ ab^n , bb^m | n ∈ W, m ∈ W }

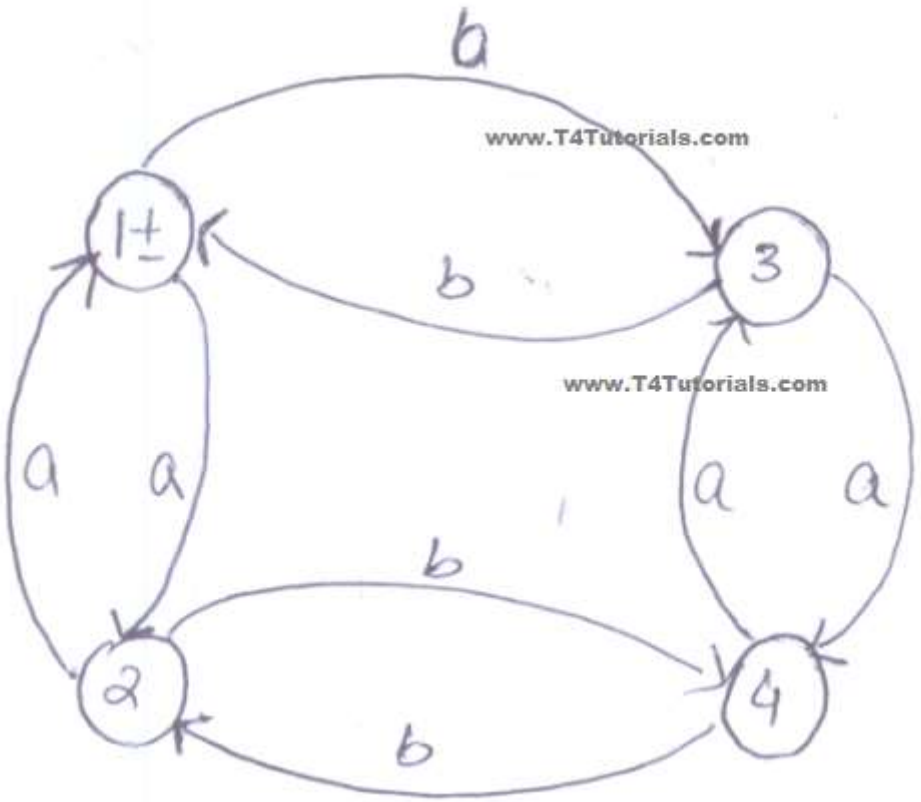
Identify language?



DFA Ex

- Design FA with ∑ = {0, 1} accepts even number of 0's and even number of 1's

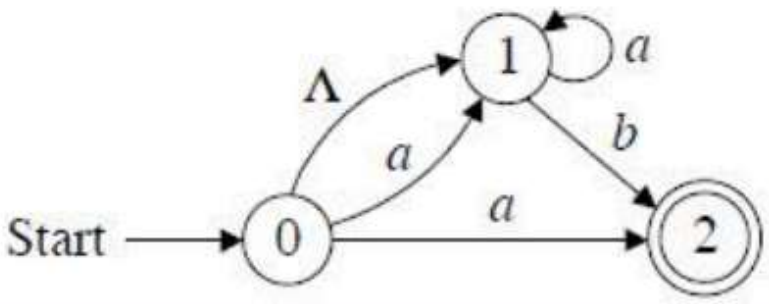
Identify lang



even number of a's and even number of b's

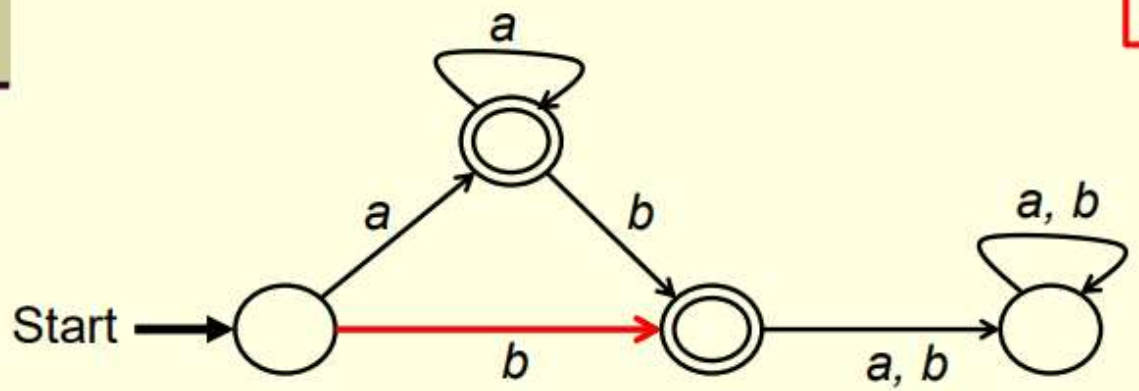
- State 1: even number of a's and even number of b's
  State 2: odd number of a's and even number of b's
  State 3: even number of a's and odd number of b's
  State 4: odd number of a's and odd number of b's

Example. NFA for the language of $a + aa^*b + a^*b$.



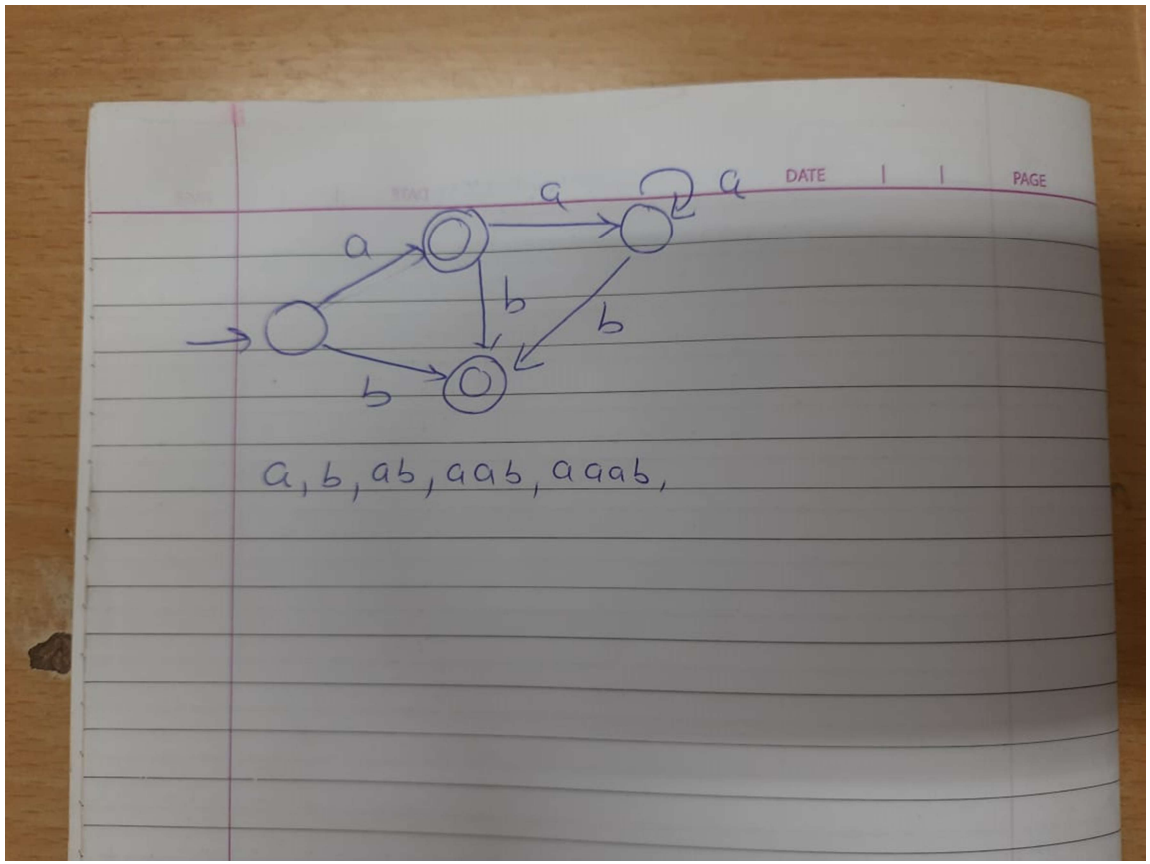
DFA for the language of $a + aa^*b + a^*b$.

a, b, ab, aab, aaab, …

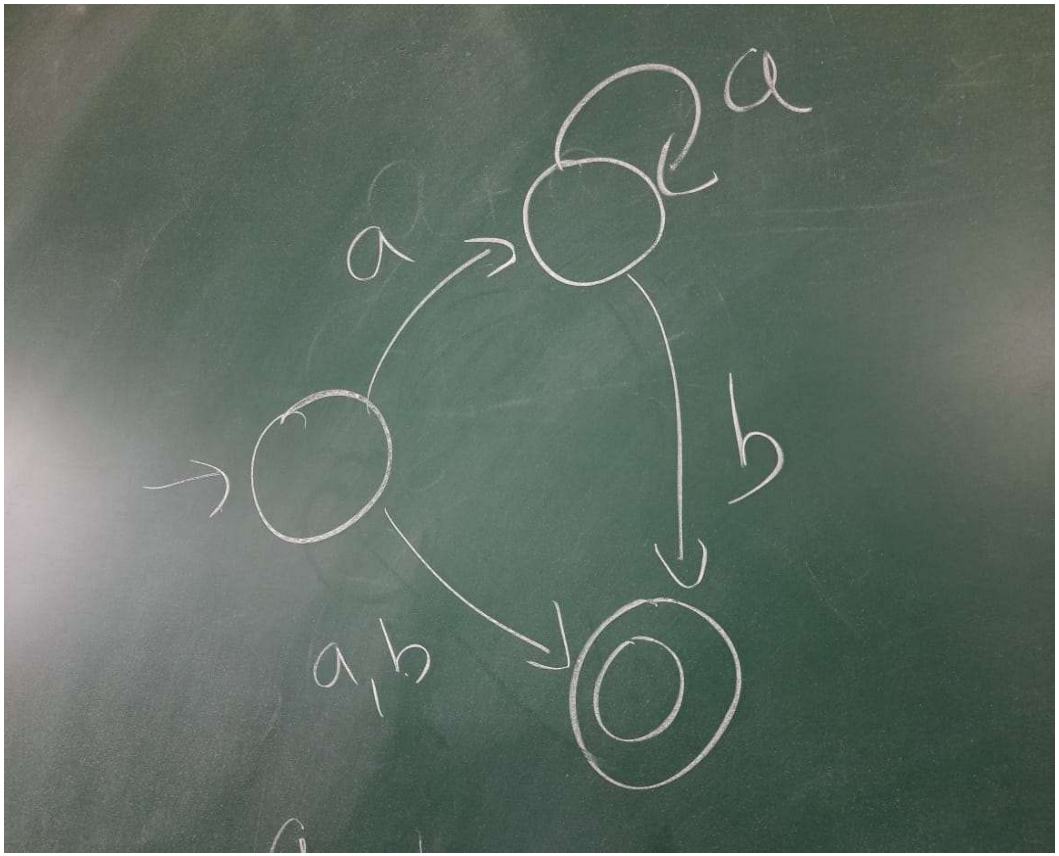


Whats wrong in this DFA? It accepts aa,aaa,…

Corrected DFA for a+aa*b+a*b



- A better DFA



Example. Find a DFA

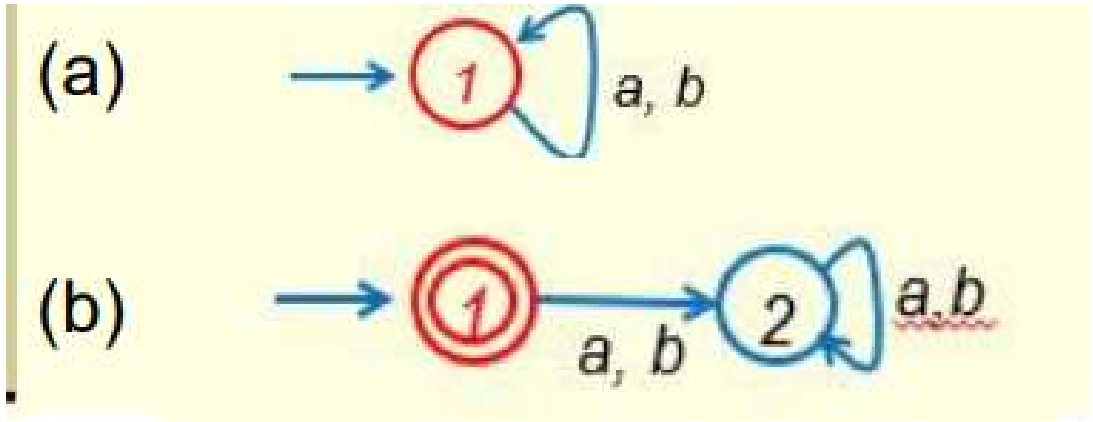
(a) $\emptyset$. (b) $\{\Lambda\}$.

a) Phi is an empty set, having no element. i.e. this dfa accepts nothing

b) This dfa accepts only null string, i.e. str having no characters

Example. Find a DFA

(a) $\emptyset$. (b) $\{\Lambda\}$.



DFA Vs NFA

S. No.	DFA	NFA
1	For Every symbol of the alphabet, there is only one state transition in DFA.	We do not need to specify how does the NFA react according to some symbol.
2	DFA cannot use Empty String transition.	NFA can use Empty String transition.
3	DFA can be understood as one machine.	NFA can be understood as multiple little machines computing at the same time.
4	DFA will reject the string if it end at other than accepting state.	If all of the branches of NFA dies or rejects the string, we can say that NFA reject the string.
5	Backtracking is allowed in DFA.	Backtracking is not always allowed in NFA.
6	DFA is more difficult to construct.	NFA is easier to construct.

DFA Vs NFA

S. No.	Title	NFA	DFA
1.	Power	Same	Same
2.	Supremacy	Not all NFA are DFA.	All DFA are NFA
3.	Transition Function	Maps $Q \rightarrow (\Sigma \cup \{\lambda\} \rightarrow 2^Q)$, the number of next states is zero or one or more.	$Q \times \Sigma \rightarrow Q$, the number of next states is exactly one
4.	Time complexity	The time needed for executing an input string is more as compare to DFA.	The time needed for executing an input string is less as compare to NFA.
5.	Space	Less space requires.	More space requires.