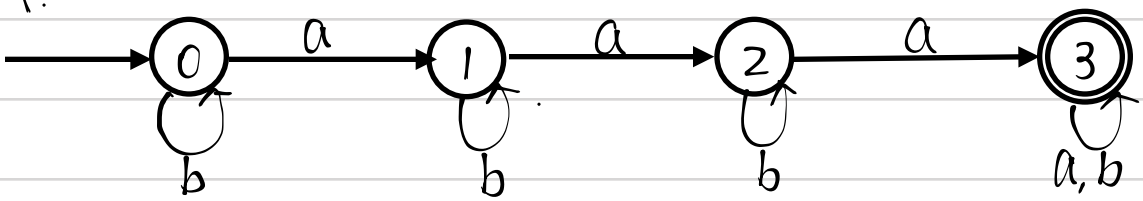
02-05 Lecture Introduction to DFAs

Deterministic Finite automaton:

Conceptually, a machine that takes in an input string $x \in \Sigma^*$, and consumes symbols in x one by one from left to right, and outputs a Yes (Accept) or No (Reject) answer.

M:



○: states a, b: symbols ⟶: transitions.

Formally, a DFA M is a structure consisting of 5 components.

$M = (Q, \Sigma, \delta, s, F)$

Q: a finite set of states $Q = \{0, 1, 2, 3\}$

Σ: the input alphabet $\Sigma = \{a, b\}$

(a finite set of single symbols)

s: the start state (always a single start state) $s = 0$

F: $F \subseteq Q$, a set of final/accept states $F = \{3\}$

may have multiple final states

δ: $Q \times \Sigma \to Q$ transition function

maps an ordered pair to a state.

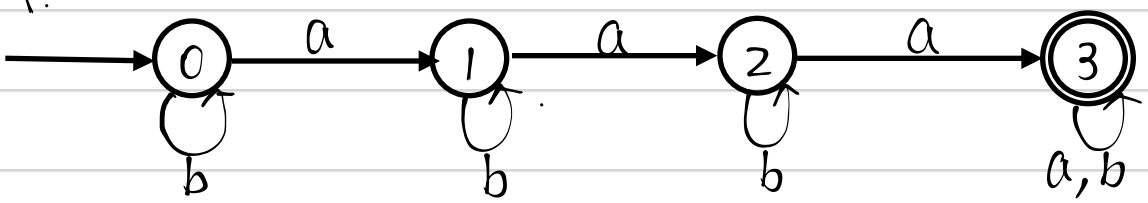
δ tells which state to move to in response to an input

$\delta(q, a) = r$

if M is currently in state q and the current input symbol to consume is a, then M consumes the symbol and moves to state r.

$|Q| \times |\Sigma|$ transitions: one transition out of each $q \in Q$ on each $a \in \Sigma$

M:



$\delta(0,a) = 1$, $\delta(1,a) = 2$, $\delta(2,a) = 3$, $\delta(3,a) = 3$

$\delta(0,b) = 0$, $\delta(1,b) = 1$, $\delta(2,b) = 2$, $\delta(3,b) = 3$

Specify a DFA

① list all the components of the DFA

② use a table

write. → to specify the start state

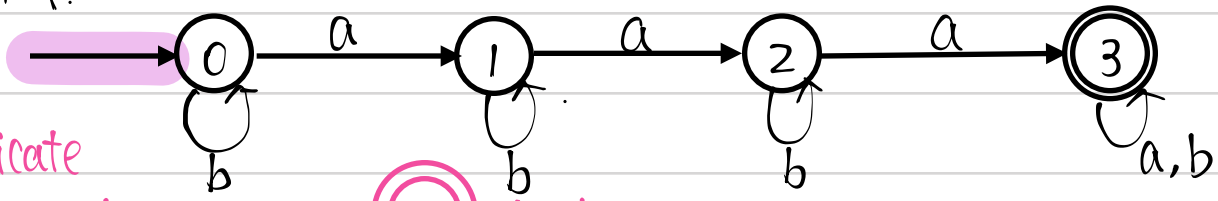
write F besides each final state.

δ	a	b
→ 0	1	0
1	2	1
2	3	2
F 3	3	3

$\delta(0,a) = 1$

③ Use a Diagram. (For homeworks)

M:



indicate start state

double circle indicate final states

Operation mechanism.

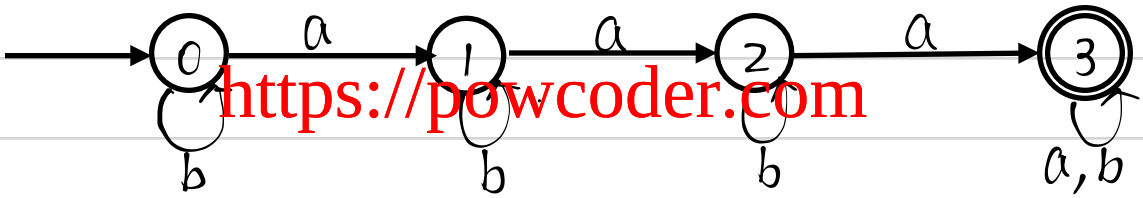
① An input can be any string $x \in \Sigma^*$

② Starting from the start state s, the DFA M consumes symbols in x one by one from left to right and moves to states according to the transition function δ

③ When the machine consumes the last symbol in x and lands on some state p,

- x is accepted (Yes) if $p \in F$
- x is rejected (No) if $p \notin F$

Example



input string baabb

baabb → aabb ⟶ abb ⟶ bb

0 — δ(0,b)=0 — 0 — δ(0,a)=1 — 1 — δ(1,a)=2 — 2

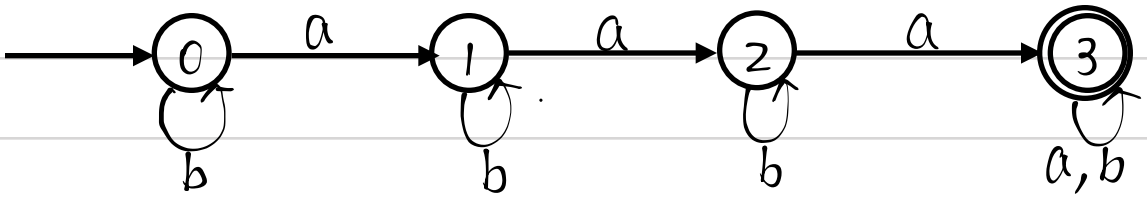
⟶ b ⟶ ↑

δ(2,b)=2 — 2 — δ(2,b)=2 — 2

end of the string lands on state 2

$2 \notin F$

baabb is rejected (No)



input string aaab

aaab $\xrightarrow{\hat{\delta}(0,a)=1}$ aab $\xrightarrow{\hat{\delta}(1,a)}$ ab $\xrightarrow{\hat{\delta}(2,a)=3}$ b $\xrightarrow{\hat{\delta}(3,b)=3}$

0 1 2 3 3

end of the string lands on state 3

$3 \in F$

aaab is accepted (Yes)

Formally, we define a function $\hat{\delta}: Q \times \Sigma^* \to Q$ to specify whether an input string is accepted or rejected by the DFA.

- $\hat{\delta}$ mas a state q and a string x to a new state $\hat{\delta}(q, x)$
- Intuitively $\hat{\delta}$ is the multistep version of δ

$$\hat{\delta}(q, \varepsilon) = q$$ ← stay put on input ε.

$$\hat{\delta}(q, ya) = \delta(\hat{\delta}(q, y), a)$$

the state M ends up in when started in state q and consuming all symbols in y according to the transition function δ

Def. A string x is accepted by the DFA M if $\hat{\delta}(s, x) \in F$
and is rejected - - - - - - if $\hat{\delta}(s, x) \notin F$

s is the start state of M, F is the set of final states of M

Def. The language accepted (recognized) by M is the set of all strings accepted by M, denoted as $L(M)$

$$L(M) = \{ x \in \Sigma^* \mid \hat{\delta}(s, x) \in F \}$$

Def. A language $A \subseteq \Sigma^*$ is a regular language if $A = L(M)$ for some DFA M.

Meaning there exists a DFA M that satisfies the two conditions.

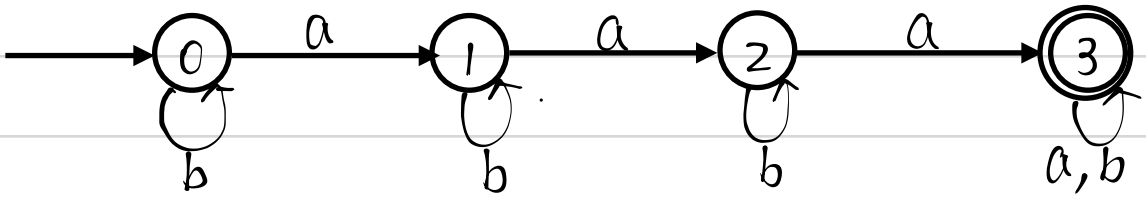
① $\forall x \in A$, M accepts x and

② $\forall x \notin A$, M rejects x

Example

$A = \{ x \in \{a, b\}^* \mid x \text{ contains at least three } a\text{'s} \}$

M:



A is regular because there exists a DFA M that accepts it.

$A = L(M)$

Example:

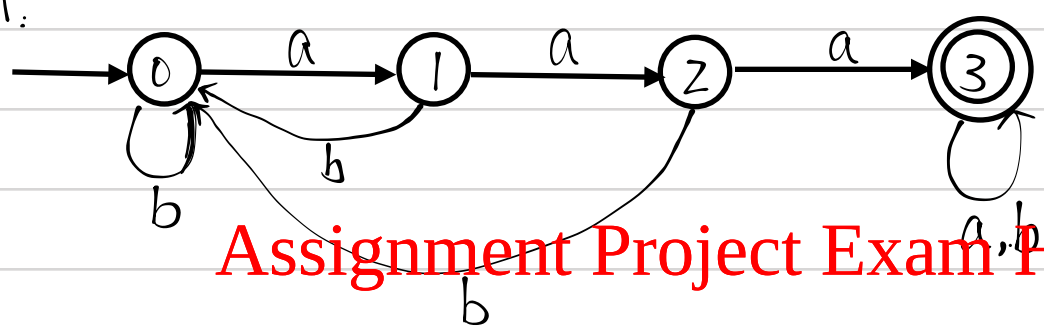
$A = \{xaaay \mid x, y \in \{a,b\}^*\}$

$= \{z \in \{a,b\}^* \mid z \text{ contains a substring of three consecutive a's}\}$

$baab \notin A$, $babaaaab \in A$

A is regular because there exists a DFA M that accepts it.

M:



the idea behind the design of M is

use the states to count the number of consecutive a's the machine have read (consumed) so far.

If M hasn't seen three a's in a row and sees a **b**, M goes back to the start state (recounting)

If M has seen three a's in a row, M stays in 3 no matter what symbol it sees (consumes) thereafter.