Date :- Aug 21, 2020

AFLL

FSM

Acceptor (Recognizer)

$w \in L(M)$ ⟷ yes / No

DFA = NFA = λ-NFA

$M = (Q, \Sigma, q_0, F, \delta)$

$\delta : Q \times \Sigma \to Q$

Transducers
Recognizing + o/p

Mealy

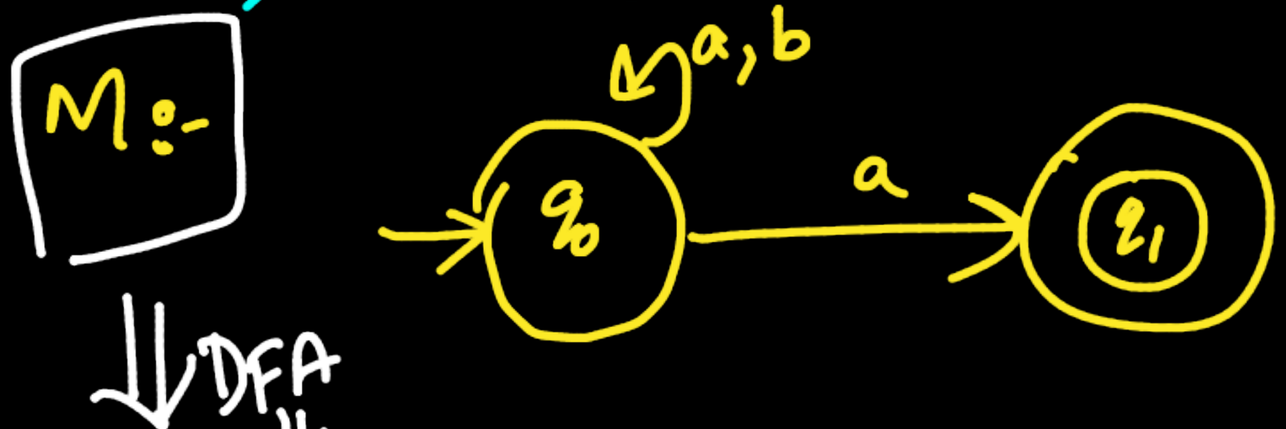
Moore

(Not in Syllabus)

NFA ≅ DFA

Never accept invalid strings

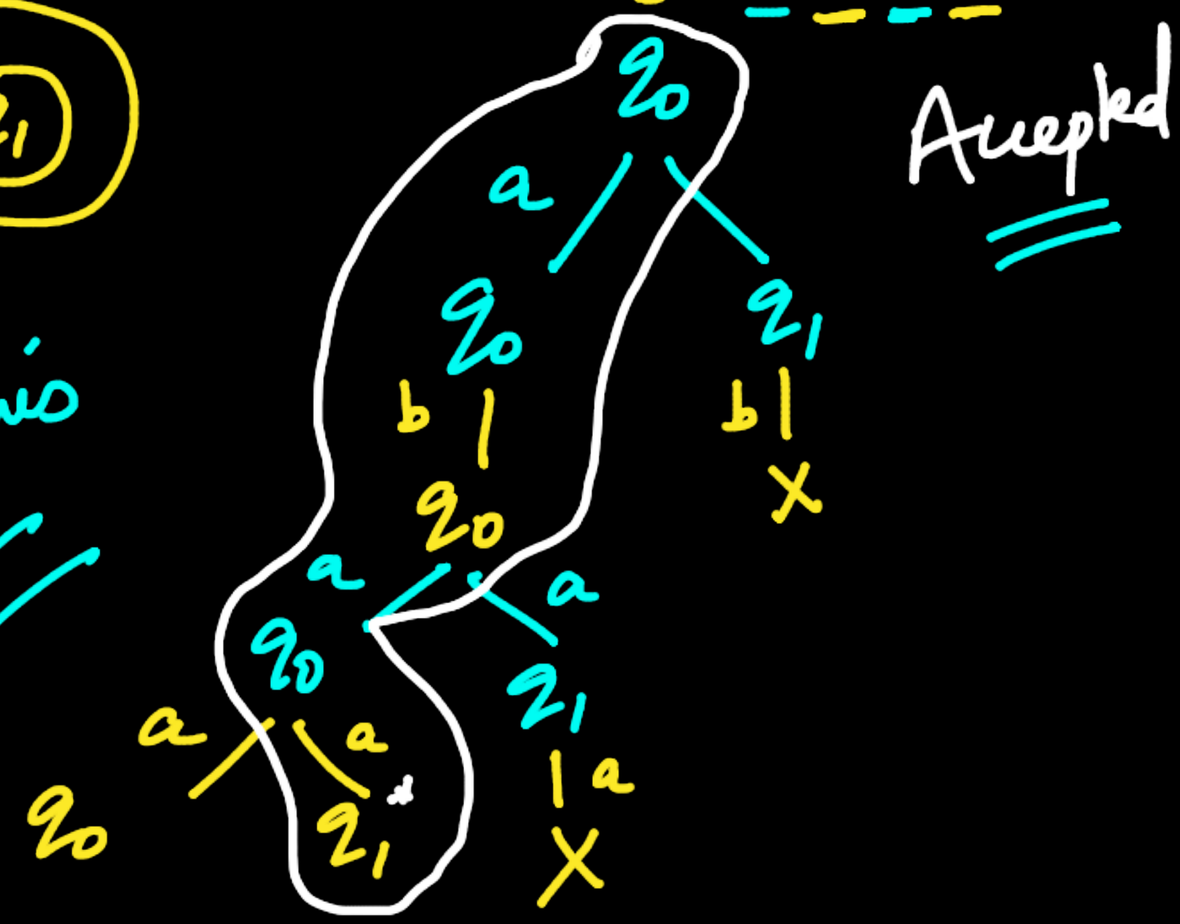
1) $L = \{ wa \mid w \in \{a,b\}^* \}$



Simulate this easily ✓

Tree

w = abaa



Accepted

Arm Expression Prog:- Ambiguous behavior

(3 + (4 * 5))

NFA → brain

↓

DFA (m/c)

((3 + 4) * 5)

wrong

Ambiguity is Another aspect to be dealt with Compilers

fib

0 or 1

return -

else

fib(n-1) + fib(n-2)

main

{

- -

- -

printf - -

Scanf

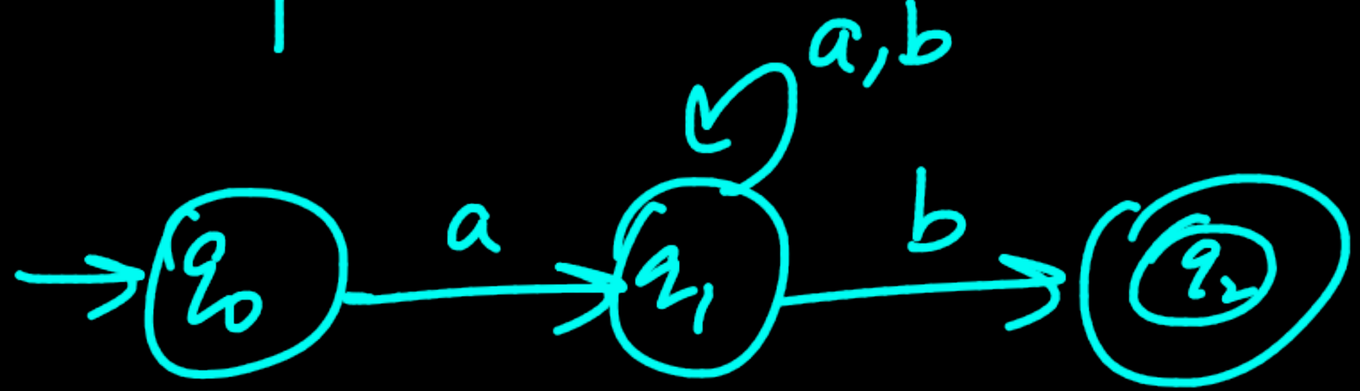
fib(n)

printf()

}

NFA

L = Strings Start with a & end in b



$Q = \{q_0, q_1, q_2\}$

$2^Q =$

$\delta(q_0, b) = \{\}$

$\delta : Q \times \Sigma \rightarrow 2^Q$ $\{\{\},$

$\{q_0\}, \{q_1\}, \{q_2\}$

$\{q_0, q_1\}$ $\{q_1, q_2\}$

$\{q_0 q_2\}$ $\{q_0 q_1 q_2\}$

$\delta(q_1, b) = \{q_1 q_2\}$

NFA

↳ Strings that start & end with same symbol

$\Rightarrow$ DFA ✓

Is every DFA an NFA? Yes

$\delta: Q \times \Sigma \to Q$

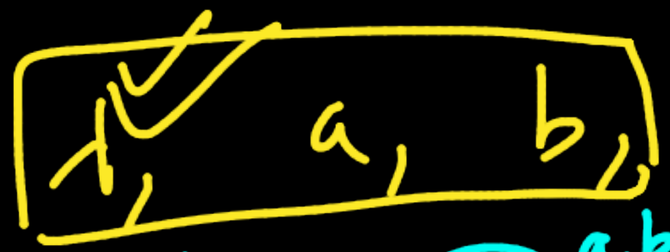
$\delta: Q \times \Sigma \to 2^Q$

$Q \subseteq 2^Q$

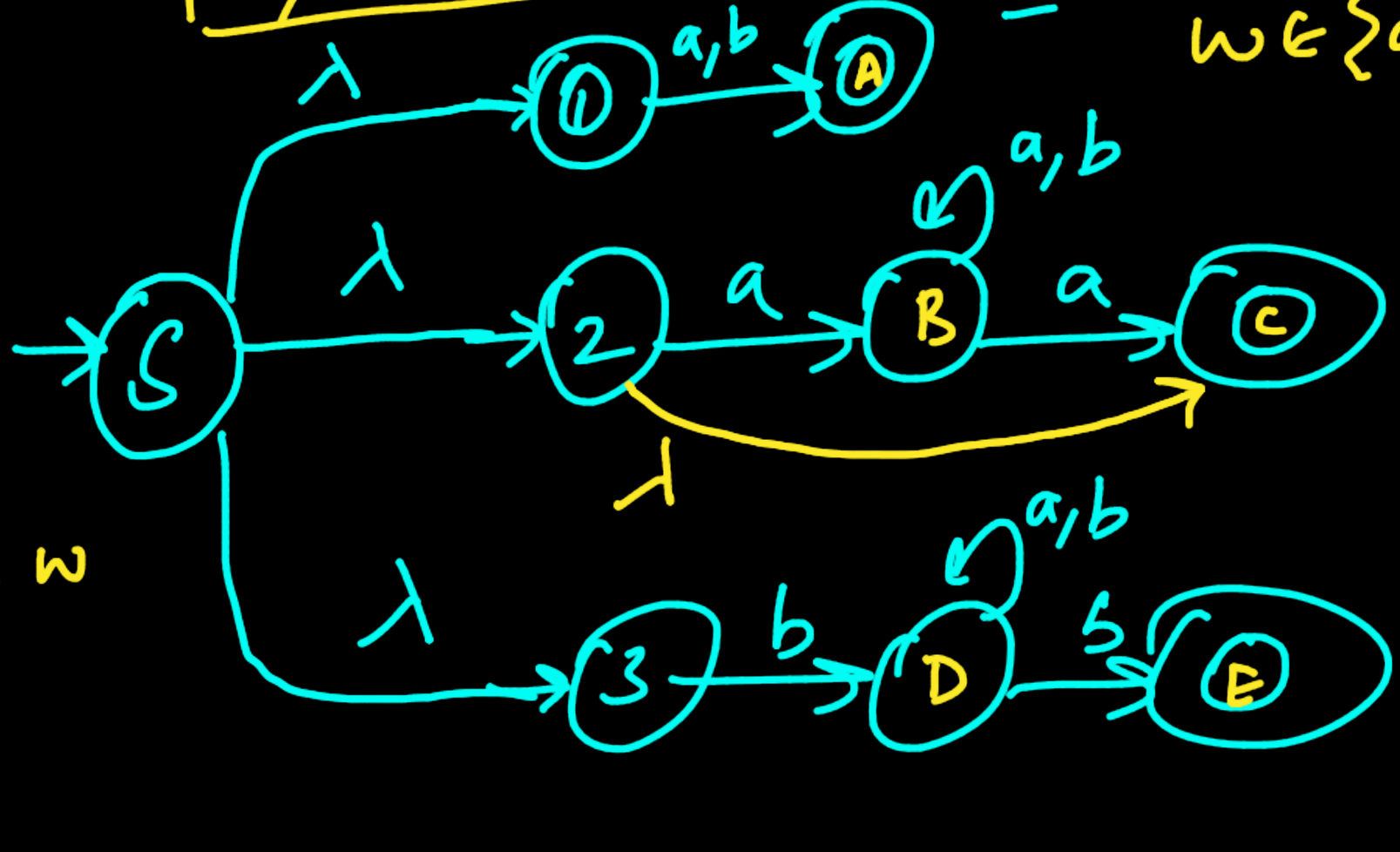
$L = \{\lambda, a, b, aa, bb, awa, bwb$

$w \in \{a,b\}^*$ $w \in \{a,b\}^*$ ✓

λ-NFA = Strings start & end with same symbol



$w \in \{a, b\}^*$



$\lambda . w = w$

S	- -
1	A -
2	B C*
3	- D

a a ✓

Transition fn. of λ-NFA

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

$$\delta : Q \times (\Sigma \cup \underline{\lambda}) \rightarrow 2^Q$$

Every L(DFA) = L(NFA) = L(λ-NFA)

DFA → NFA
DFA → λ-NFA

(No conversion required)

NFA → Translate → DFA
λ-NFA → DFA

Subset-Construction

But L(DFA) = L(NFA) = L(λ-NFA)

All equivalent in Power.

What may not work with FA?

DFA

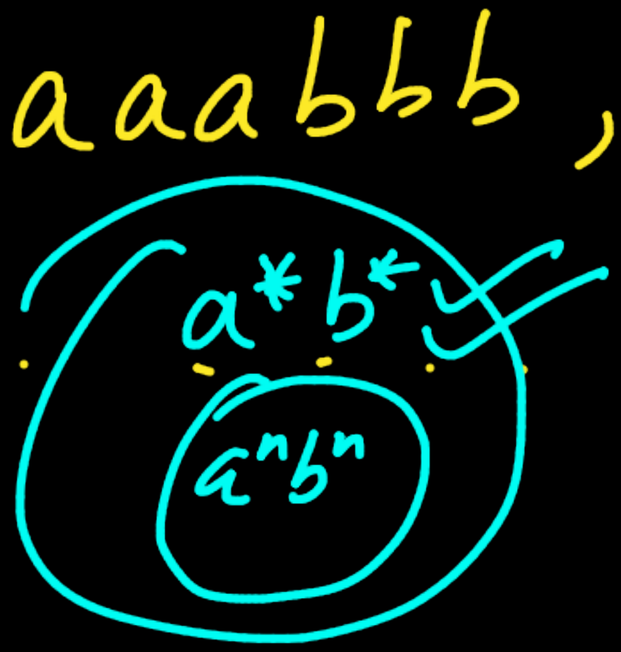
NFA

λ-NFA

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

$$L = \{ \lambda, ab, aabb, aaabbb, aaaabbbb \ldots \}$$

Infinite !



$$L = \{a^n \mid n \geq 1\}$$

$$L = \{a^{2n} \mid n \geq 1\}$$

Power = linear

$$L = \{a^{n^2} \mid n \geq 1\}$$

Not possible

$$L = \{a^i \mid i \text{ is prime}\}$$

FA :- Memory is limited

States

mod - counter

palindrome www^R

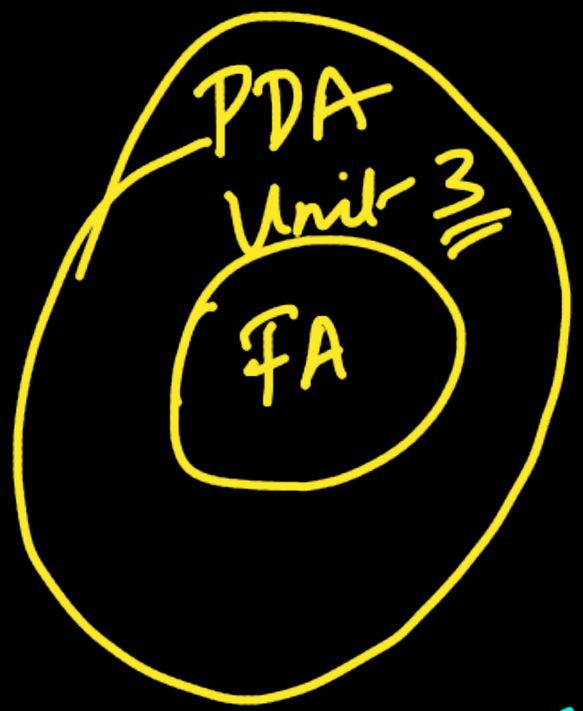
$a^n b^n$

abba

DFA to accept palindromes of length ≤ 4 ≤ 100 ≤ 1000

(wwᴿ) $w \in \{a,b\}^*$

FA has limits!!



formal way to prove that you cannot FA for L

$\Rightarrow$ Pumping lemma → Unit-2

Regular language

FA = M

(DFA, NFA, λ-NFA)

Power

E:

L(M) = regular language