CSE 273 / L-01 / 19.01.2025 /

Quize ⇒ 15%. ⇒ Lowest one will be dropped

Attendance ⇒ 5%.

Assignment ⇒ 15%.

Midterm ⇒ 30%.

Final ⇒ 35%.

Theory of Computation

- study of abstract machine.
  → model human thought process, in a machine.

Alan Turing

- Turing Machine
  - simple but powerful model.

Alonzo Church

- Lambda Calculus
  - father of all functional programming language.
  - Lisp, Haskell, Scala
  - helps to explore the theoretical limits of what can be computed.
  - λ ⇒ representation of a function

⇒ Turing Thesis

- Hypothesizes that any function effectively calculable by an algorithm can be computed by a Turing Machine or expressed in Lambda Calculus.

- Unifier various models of computation
  - recursive function
  - Lambda Calculus
  - Turing Machines
- widely accepted despite not being formally provable.

Halting Problem:

- Undecidable Problem
  - when a function/program freeze, can't provide output or show the error within finite time.
    - stuck in a infinite loop.

Finite Automata:

- 1943
- Warren McCulloch and Walter Pitts

Stephen Kleene:

- regular expressions (kleene closure)
  - important for ~~fine~~ finite automata theory
  - groundwork for text processing and compiler construction.

L-02/21.01.2025

* applications of automata theory:
  - compilers & interpreters
  - text editor and processors
  - search engines
  - system verification components

* important terminologies:

* Alphabets:
  ⇒ every languages has some fixed alphabets.
  English = 26
  Bangla = 50
  ⇒ Σ, representation
  ⇒ in programming languages, we define the alphabets according to our needs.

* Strings:
  - collections or sequence of alphabets. (finite)
    ⇒ $\Sigma = \{0,1\}$; 0011 is a string
    ; 0012 not a valid string.

* Languages:
  - some rule, like we use grammar for our regular 'English' or 'Bangla' languages.
  - collection of valid string.

~~$\Sigma = \{0,1\}$; 0011~~

Basics of strings:

- Empty string : ϵ
- Length of a string :

$|0010| = 4$
$|aa| = 2$
} $|\ldots| =$ length

- substring :
  - maintain the sequence.
  - ⇒ 00111 ; 011 is a substring
    10 is not a substring
- subsequence :
  - no need to maintain the sequence.
- Prefix : aaabc
  - ~~Suffix~~
  - prefix(bc) = aaa
  - prefix(aaabc) = ϵ
  - prefix(ϵ) = aaabc
- Suffix : aaabc
  - suffix(aaa) = bc
  - suffix(aaabc) = ϵ
  - ~~Suffix(ϵ) = aaabc~~
  - suffix(ϵ) = aaabc

aaabc
$\equiv \epsilon$ aaabc ϵ

⇒ Proper prefix and proper suffix of a string is not equal to the string itself and non-empty.

Concatenation:

$w = aba$
$p = 01001$

$pw = 01001aba$
$wp = aba01001$

maintain sequence

Exponentiation:

$w = 010$

$w^0 = \epsilon$

$w^2 = w = 010$

$w^2 = w \cdot w = 010\,010$

Reversal:

$w = abc$

$w^R = cba$

Σ^k = all possible combination and permutation of length k from the given symbol defined by Σ.

$\Sigma = \{0, 1, 2\}$

$\Sigma^2 = \{01, 02, 12, 10, 20, 21, 00, 11, 22\}$

Problems:

- in automata theory, a problem is to decide whether a given string is a member of some particular language.

$\Rightarrow$ $\Sigma = \{0, 1\}$ $\Rightarrow$ allowed alphabets

$L = \{0^n 1^n \mid n \geq 1\}$ $\Rightarrow$ regular expression of a language.

$S_1 = 0011 \in L$ | $S_2 = 00011 \notin L$

L-03 / 26.01.2025

Machine:

- Check if a string is valid or not, depends on given language.

Finite Automata (FA):

(i) Deterministic (DFA)

(ii) Non-deterministic (NFA)

(iii) Non-deterministic with ϵ-transitions. (ϵ-NFA)

DFA:

- in the finite automata graph:
  - node ⇒ state
  - directed edge ⇒ weighted edge
  - moving one state to another state is known as transition.

DFA is a 5 tuple:

$$\langle Q, \Sigma, \delta, q_0, F \rangle$$

- Q ⇒ Finite set of states
- Σ ⇒ Finite set of input alphabet.
  - allowed character
- δ ⇒ transition function.
  - mapping $Q \times \Sigma$ to Q
  - transition table

- $q_0 \Rightarrow q_0 \in Q$, initial state
  - only one state
- $F \Rightarrow F \in Q$, final state
  - maybe multiple, zero or more

* In DFA, there is only one transition for each input symbol from each state.
* In the transition table, final state represent with a star sign and initial state with a arrow.
*

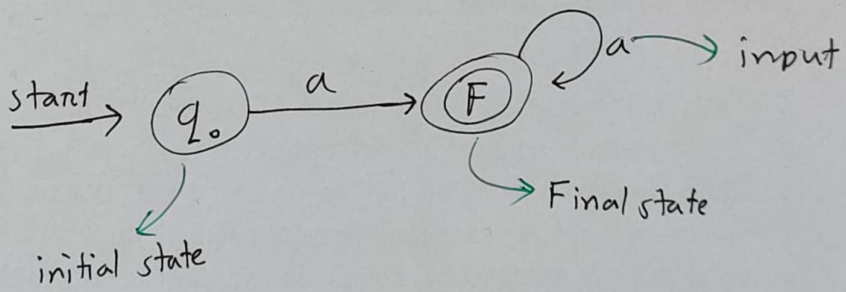
$\Sigma = \{a\}$

$L = \{a^n \mid n = 1, 2, \ldots\}$ (at least one 'a')

Example string:

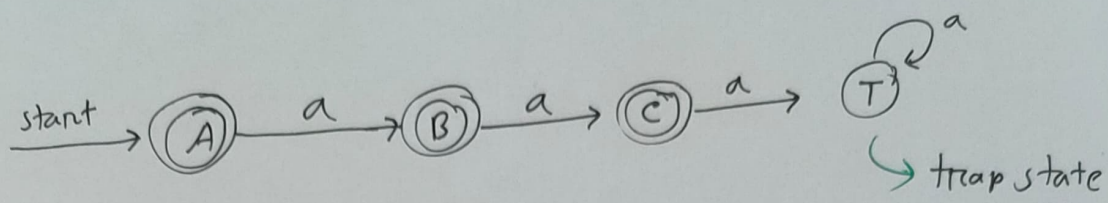
a
aa
aaa
aaaa
aaaaa
:

DFA:



* Read empty string wing DFA will be stucked on a trap state.

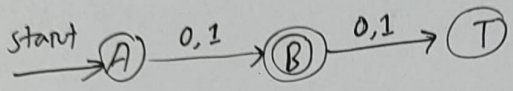
* at most 2's 'a':



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

L : length is 1

DFA:

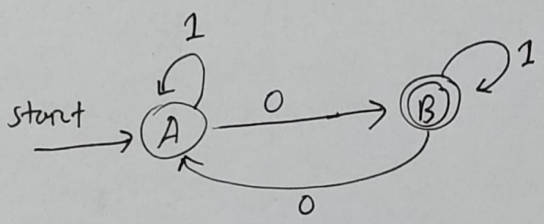


* From slide - 12:

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

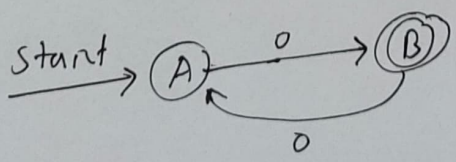
L : odd numbers of 0, any number of 1

DFA:



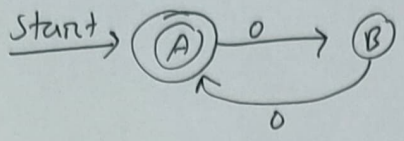
L-04 / 28.01.2025

* Odd length of zero:



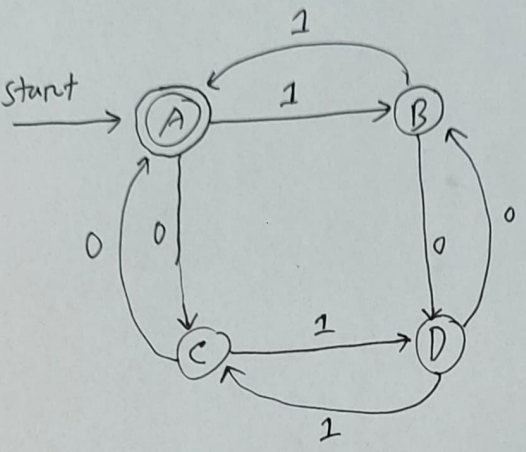
Even numbers of zeros:

DFA:



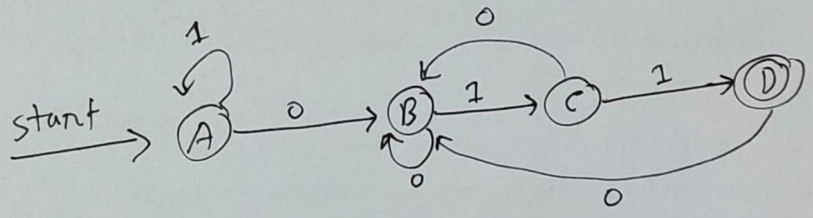
even numbers of zeros and even number of 1.

DFA:



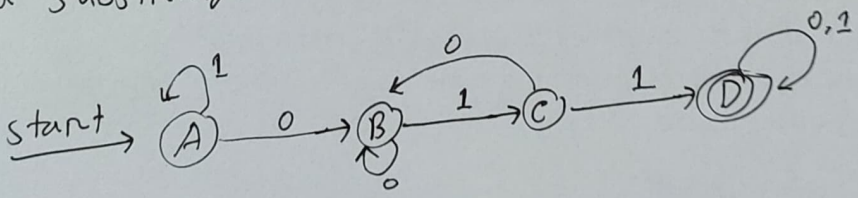
End with 011.

DFA:



011 as a substring:

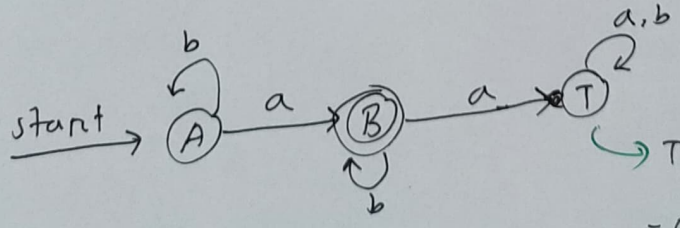
DFA:



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

L: only one a

DFA:

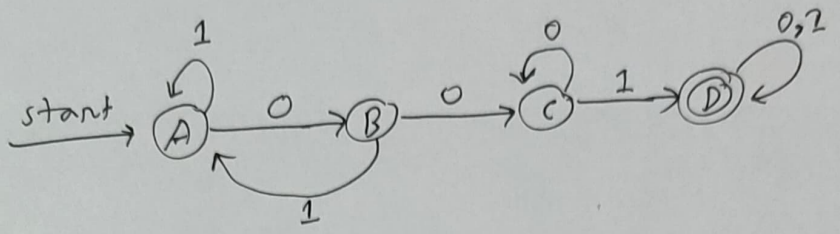


- only self loop allowed
- no outgoing transition

✳ $\Sigma = \{0, 1\}$

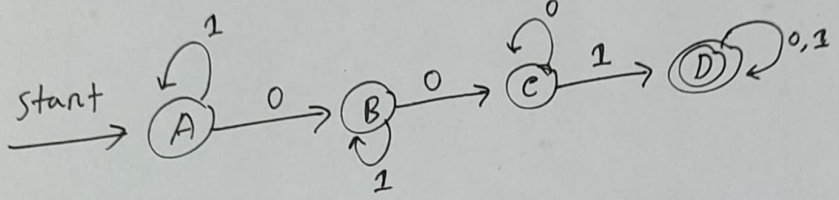
L : 001 as a substring

DFA:



✳ 001 as a subsequence:

DFA:



✳ Practice DFA construction from slide.
Make some unique and little bit challeges DFA and solve it.
Submit as a assignment on canvas. At least 5x.

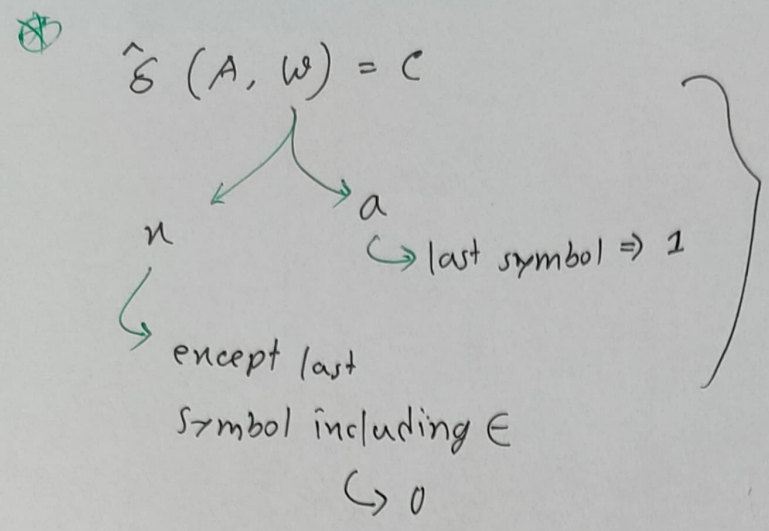
Slide - 21,22	H.W

L-05 / 04.02.2025

✳ Transition function support single input only. ⇒ δ

- We can extend for taking string as a input.
⇒ $\hat{\delta}$

⇒ $\delta(A, 0) = B$
$\delta(B, 1) = C$
} $\hat{\delta}(A, 01) = C$



Example - Slide - 26

$\hat{\delta}(q_0, 0101) = \delta(\hat{\delta}(q_0, 010), 1) = ? = \delta(q_3, 1) = q_0$

$\hat{\delta}(q_0, 010) = \delta(\hat{\delta}(q_0, 01), 0) = ? = \delta(q_2, 0) = q_3$

$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1) = ? = \delta(q_1, 0) = q_2$

$\hat{\delta}(q_0, 0) = \delta(\hat{\delta}(q_0, \epsilon), 0) = ? = \delta(q_0, 0) = q_1$

$\hat{\delta}(q_0, \epsilon) = ? = q_0$

$\therefore \hat{\delta}(q_0, 0101) = q_0 \in F$

Accepted.

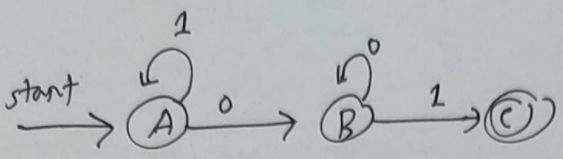
In terms of time, DFA is better than NFA.

- Computer algorithm use DFA only.

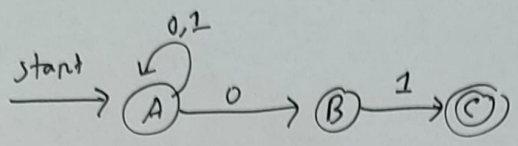
- NFA is easy to construct for human.
- There is an algorithm that can convert NFA to DFA easily.

Comparison: End with 01.

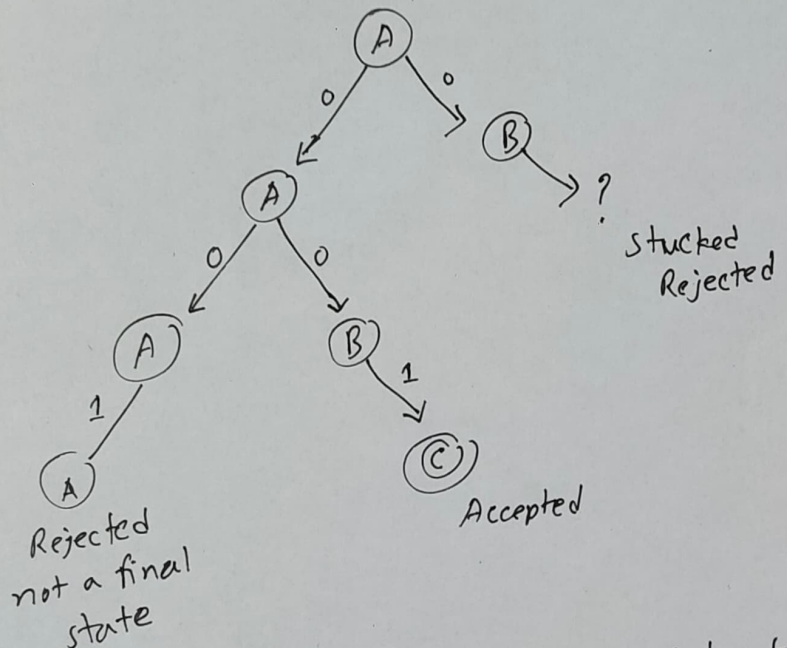
DFA:



NFA:



001



∴ For NFA, we need to traverse all possible way then we can accept or reject a string.

Extended transition function for NFA:

- ~~if~~ results are showed in a set.
- if any state that belongs to final state then accepted otherwise rejected.

Slide - 30

L-06 / 09.02.2025

∈-NFA:

Comparatively slower than DFA.

Subset Construction algorithm:

- can convert ∈-NFA or NFA to DFA

∈-NFA ⇒ NFA ⇒ DFA

- using the concept of kleene closure

kleen closure:

⇒ ∈-closure (T): all possible states reachable from any state in T ~~usin~~ without any cost + T

→ input of the function

- can be three type
  - (i) state
  - (ii) multiple state
  - (iii) transition function

From the diagram of Slide - 31

∈-Closure (C) : {C, D, A}

∈-Closure (A, B, E) = {A, B, F}

~~∈-Closure (T)~~

∈-Closure (δ(B, b)) = ∈-Closure (D, E) = {A, D, E}

∈-Closure (δ(B, a)) = ∈-Closure (ϕ) = ϕ = { }

Subset Construction algorithm:

- ∈-Closure (initial state) = ∈-Closure (1) = {1, 2}

Example: slide-33

	a	b
→ {1,2}	{3,5}	{4,5}
{3,5}	φ	{4}
* {4,5}	{5}	{5}
{4}	{5}	{5}
* {5}	φ	φ

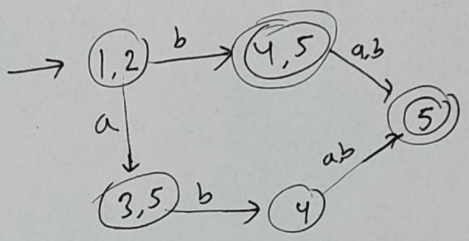
ϵ-Closure($\delta(1,a)$) = {3}

ϵ-Closure($\delta(1,b)$) = {4}

ϵ-Closure($\delta(2,a)$) = {5}

ϵ-Closure($\delta(2,b)$) = {4,5}

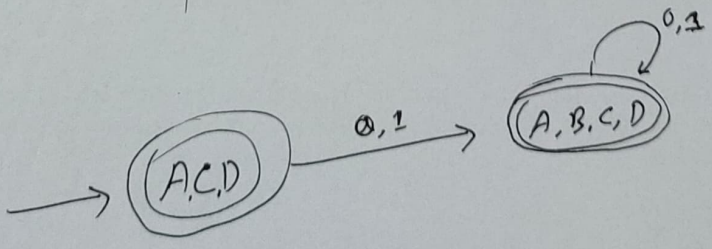
DFA:



Example from slide-35

	0	1
→ {A,C,D}	{A,B,C,D}	{A,B,C,D}
* {A,B,C,D}	{A,B,C,D}	{A,B,C,D}

DFA:



Worst case complexity of subset construction is

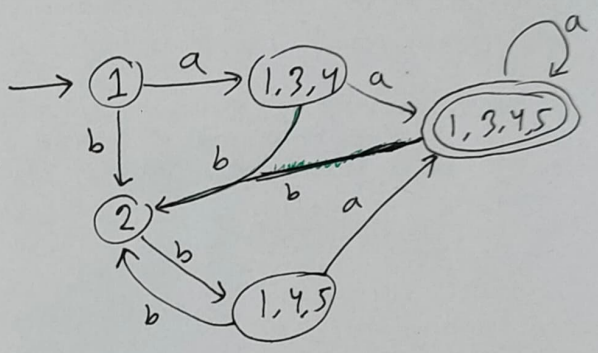
$2^n - 1$

Assignment-2:

- at least 2 NFA of different length which end up with worst case scenerio. ↪ 3, 4 Recommended.

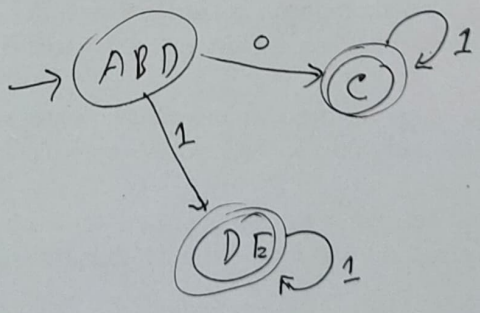
Example: from slide-37

	a	b
→ {1}	{1, 3, 4}	{2}
{2}	ϕ	{1, 4, 5}
{1, 3, 4}	{1, 3, 4, 5}	{2}
* {1, 4, 5}	{1, 3, 4, 5}	{2}
* {1, 3, 4, 5}	{1, 3, 4, 5}	{2}



Example: from slide-39

	0	1
→ {A, B, D}	{C}	{D, E}
* {C}	ϕ	{C}
* {D, E}	ϕ	{D, E}



DFA minimization algorithm:

- Table filling method
  - ↳ hopecroft algorithm
    - distinguishable table
    - partitioning algorithm

Example: from slide-41

⇒ partition based on characterization:

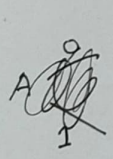
⇒ S_0:

A B C D E F G H

⇒ Assume all same

⇒ S_1:

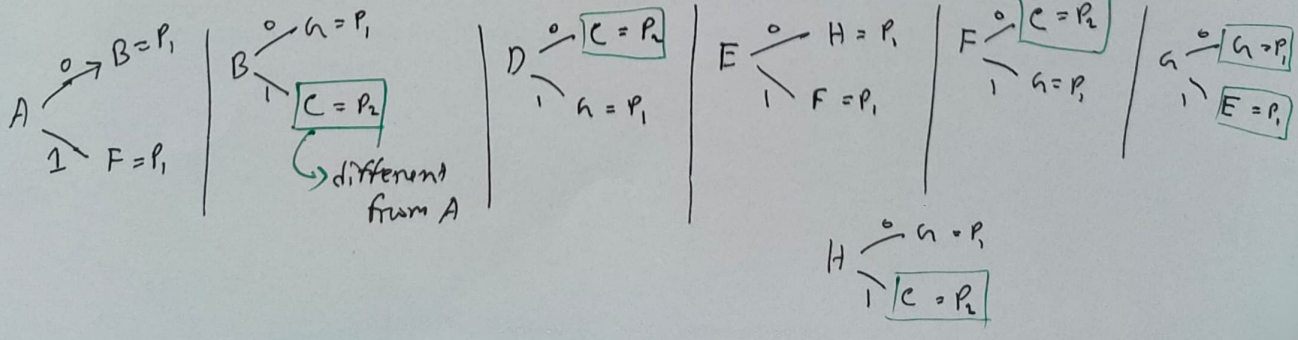
- choose a leader from an existing group and based on its characteristic make another group.
- first divide them based on initial and final state.



P_1	P_2
A B D E F G H	C

⇒ S_2:

P_1	P_2	P_3
A E G	C	B D F H



⇒ S_3 :

P_1	P_2	P_3	P_4	P_5
AE	C	BH	G	DF

A: 0 → B = P_3, 1 → F = P_3

E: 0 → H = P_3, 1 → F = P_3

G: 0 → G = P_1, 1 → E = P_1

B: 0 → G = P_1, 1 → C = P_2

D: 0 → C = P_2, 1 → G = P_1

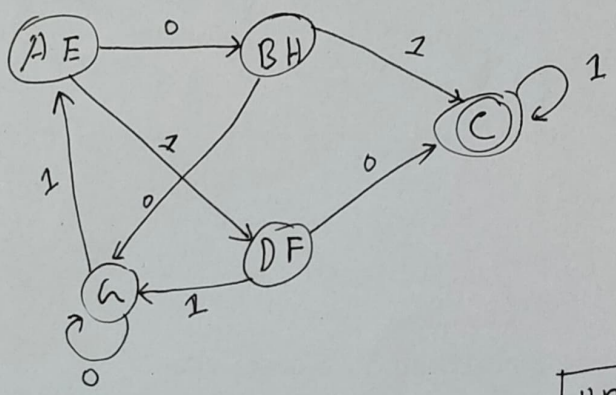
F: 0 → C = P_2, 1 → G = P_1

H: 0 → G = P_1, 1 → C = P_2

⇒ S_4 :

No change all same characteristic in their partition.

DFA:



upto slide-41

L-07 / 11.02.2025 /

✳ Table filling method :

- need to do some pre-process
  - dead state elimination
    - no incoming, no outgoing, non-final
  - non-reachable state elimination
    - no incoming, non-final

✳ For equivalence testing:

- treat them as a single combined DFA and start partitioning algorithm.

✳ Example: from slide-42

S_0 :

P_1
A B C D E

S_1 :

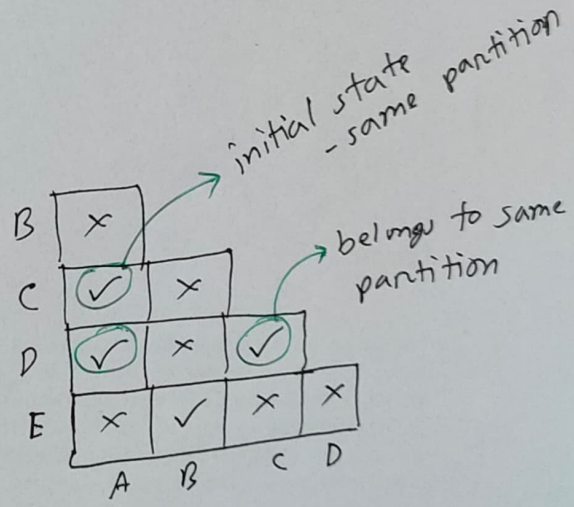
P_1	P_2
A C D	B E

S_2 :

P_1	P_2
A C D	B E

No change

table for equivalence test:



∴ They are equivalent.

⇒ if the starting state of both DFA belongs to same partition, then they are equivalent.

Finite automata with output:

⇒ without output was 5 tuple. but with output has 6 tuples.

output alphabet, may not exists on Σ

$$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

output function

For Moore machine,

$\lambda : Q \rightarrow \Delta$ — state for output, we can say that the final state when output will be provided. Doesn't matter how to reach there.

For Mealy machine,

$\lambda : \Sigma \times Q \rightarrow \Delta$

how to reach the state to provide output. every path can carry different weight, so it provides more variation on output.

Moore machine:

⇒ 'ϵ'-string provide the output of initial string.

Slide-44

total length of output string: $(n+1)$ — for ϵ

Mealy Machine:

⇒ 'ϵ'-string doesn't provide any output.

Slide-45

length of output string: $|n|$

Quiz-1 upto this 23.02.2025