Date 16 1 24.
M T W T F S S

AUTOMATA THEORY. CSC220.

1. Introduction to automata theory.
2. Deterministic finite automata. (Machine ☰ specific characteristics) DFA
3. Non-deterministic finite automata NDFA.
4. NDFA → DFA Conversion.
5. Moore and Mealy machines.
6. Classification of Grammars.
7. Regular grammar. (Discrete)
8. Context Free Grammar (CFG)
9. Push down automata PDA.
10. Turing Machine TM.
11. Decidability

Introduction.

- The term automata is derived from Greek word ☐ means self acting.
- An automaton (automata) is an abstract self propeld computing device ☐ follows a predetermined sequence of operations automatically.
- An automaton ☐ finite number of steps / positions / is called finite automata. FA. / finite state machine FSM.

* An automaton can be represented by a 5-tuple

1. Q = This rep finite set of states.
2. Σ = This rep finite set of symbols, called alphabet.
3. δ = This rep transition function.
4. q_0 = This rep first / initial state from where any input is processed.
5. ($q_0 \in Q$)
5. F = This rep set of final state / states
   ($F \in Q$)

finite symbols

1 Alphabet.

* finite set of symbols eg

$\Sigma = \{a, b, c, d\}$ = Alphabetical symbols are a, b, c, d.

* A word can be created here: bad, dad, baba

finite sequence of symbols

2 String

This is finite set of ~~sym~~ sequence formed from alphabetical symbols: bad, dad, baba

3 Length of string.

Number of symbols present in a string

If s = dad

$|s|$ = length of is is 3

$|s| = 3$.

* If length of $|s| = 0$ its called empty string

If $|s| = 0$ // empty string ~~den~~.

* empty smng is denoted by λ or ε.

4 Kleene Star (*) = Denoted by Σ^* works on alphabets

- This is unary operator on set of symbols or smngs.
- Kleene star gives infinite set of all posible smng of all posible lengths over alphabet including empty string ε

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

$\Sigma^* \{\varepsilon\} = 0$

$\Sigma^* \{a, b, c, d\} - 1$

$\Sigma^* \{aa, bb, cc, dd\} - 2$

$\Sigma^* \{-----\}$ = when 10.

5 Kleene Closure - (Kleene Plus) - Σ^* (+)

Works on alphabets

It exclude empty smng

- It is infinite set of all posible smngs of all lengths over alphabets excluding empty smng ε.

$\Sigma^+ = a, b, c, d, aa, bb, cc, dd.$

$\Sigma^+ = \Sigma^n$ where $n \geq 1$

$\Sigma^* = \Sigma^n$ where $n \geq 0$

6 Language.

- Is subset of Σ^* for some alphabet
- It can be finite or infinite.
- eg = If Language takes all posible strings of length 2 over alphabet $\Sigma = \{a, b\}$ then Language = L =

  $L = \{aa, bb, ab, ba\}$

Deterministic finite Automaton (DFA).

Finite automaton can be classified into two ?:

1. Deterministic Finite automaton DFA
2. Non-deterministic finite automaton NI

- In DFA for each input symbol, one can determine the state which machine will move hence called deterministic automato.
- And coz it has finite no! of states, machine is called deterministic finite machine or deterministic finite automaton.

Define DFA. Formally.

- defined by 5 tuples:

1. Q - finite set of states.
2. Σ - finite set of symbols called alphabets. /Input symbols)
3. δ - transition function where $\delta = Q \times \Sigma \rightarrow Q$
4. q_0 - initial state. ($q_0 \in Q$) from where any input is processed
5. F - Set of final state / states where ($F \in Q$)

Graphical representation of DFA

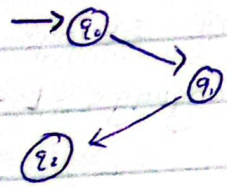
DFA is repd by di-graphs (direction graphs) called

* ~~DFA~~ State diagram where,

a - The vertices represent the states.

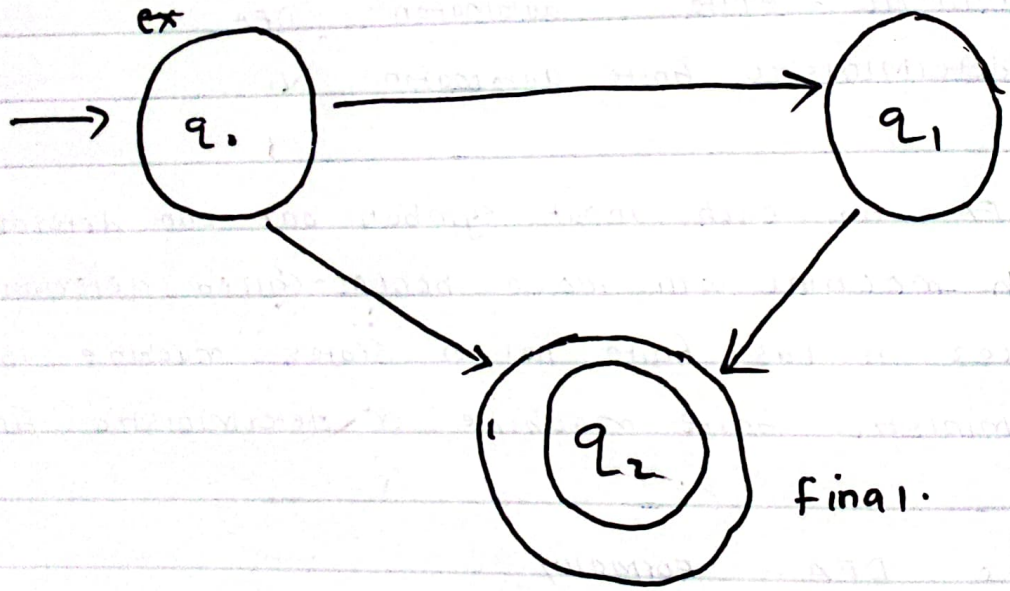
b - The edges lebled ∑ input alphabet show transition

c - The initial state is denoted by empty single incoming arrow.



d - The final state is indicated by double circles.

ex



ex

Let DFA be:

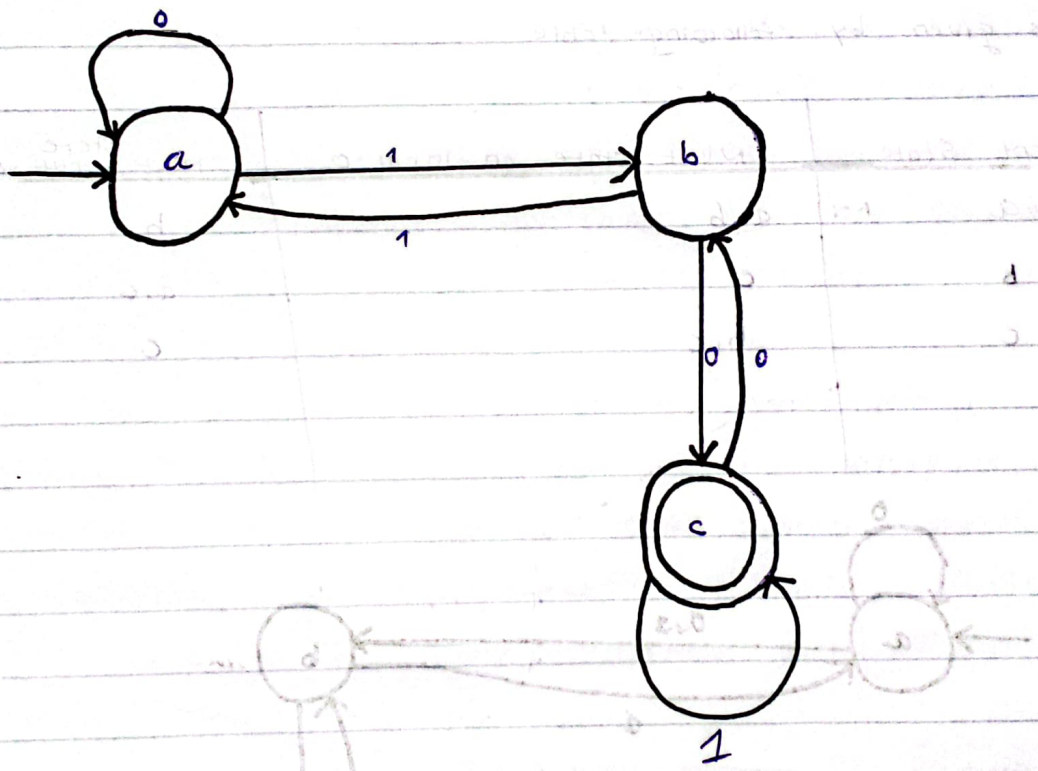
$$Q = \{q, b, c\}$$
$$\Sigma = \{0, 1\}$$
$$q_0 = \{a\}$$
$$F = \{c\}$$

where δ = given by:

Present State	Next state on input = 0	Next state on input = 1
→ a	a	b
b	c	a
c	b	c

It shld move to only 1 particular state.

Diagram.



NDFA.

For particular input symbol. the machine can move to any combination of state in machine

Meaning exact state to which machine moves cannot be determined hence NDFA.

- As it has finite no! of states, the machine is called, ~~No~~ NDFM or NDFA.

Formal definition of NDFA:

Replicate DFA definition except in transition fn $\delta = \delta: Q \times \Sigma = 2^Q$.

eg

Consider NDFA

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

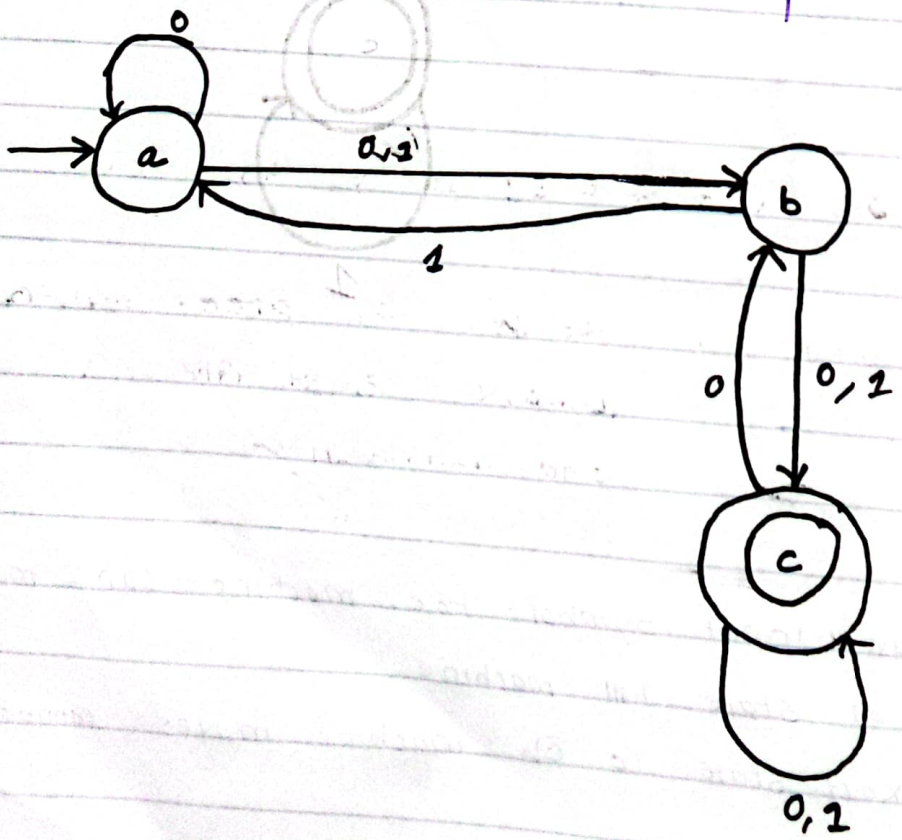
$\Sigma = \{0, 1\}$

$q_0 = \{a\}$

$F = \{c\}$

δ is given by following table.

Present State	Next state on input 0	Next state on input 1
→a	a, b	b
b	c	a, c
c	b, c	c



Comparison btn DFA and NDFA.

1- The transition in DFA, for a state is to a single particular input for each hence deterministic

- In NDFA the transition can be to multiple next state in each input symbol. Thus Non-deterministic

2 IN DFA empty string transitions are not seen while in NDFA empty string transition is permited / Allowed.

Null transition

3 DFA allows backtracking while NDFA, backtracking is not posible

4 DFA requires more space while NDFA, requires less space.
(< in DFA, transition in any state must be stored coz you hv to backtrack).
(< NDFA no backtracking >.

5 A string is accepted by DFA if it transist to final state while a string is accepted by NDFA if atleast one of all posible transition, ends in final state.

* 01001 = Not accepted by DFA.

* 01001 = First case accepted. NDFA. Thus accepted since one route is going accordingly.

Acceptors, classifiers and Transducers.

* An acceptor/recognizer - is an automaton that computes a boolean function
  - All states of acceptor is either accepting (true) or rejecting (false) inputs given to it.

* Classifier → Has more than 2 final states and it gives ~~str~~ single outputs when it terminates.

* **Transducer** → is an automaton that produces output based on current input and ~~previ~~ or previous state
  → Transducers can be of 2 types:

① Mealy machine

- The output depends on current state and current input.
-

② Moore Machine

The output depends only on current state.

Acceptability of DFA and NDFA.

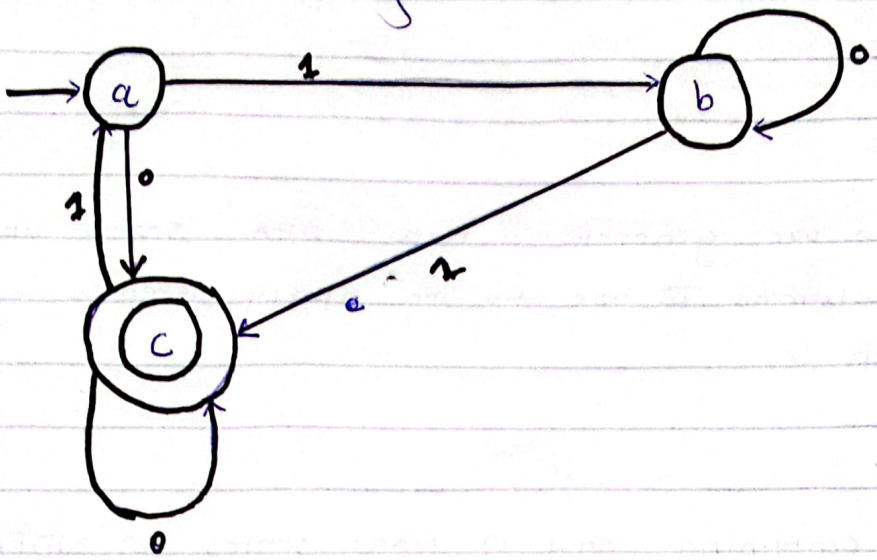
→ A string is accepted by DFA or NDFA ~~ff~~ iff the DFA or NDFA starting at initial state ends in an accepting state (any of final state) after reading the string wholly.

* ~~Automata~~ Shld begin from initial
* It should transit one state after the other →
* Must reach final state ~~c~~ all symbols in a string consumed.

} Acceptab condition

Ex

Consider the following



1. 0, 1010, 11, 110, 101 - All accepted.

2. 1, 011, 0110, - All rejected.

~~CONVERSION BTN DFA → NDFA.~~

NDFA to DFA Conversion.

Algorithm.

Input - NDFA.

output - Equivalent DFA.

Step 1:

1. Create state table from given, NDFA.

Step 2:

2. Create blank state table under posible input alphabet for equivalent DFA.

Step 3:

3. mark start state of DFA by q0 same as that of NDFA

Step 4:

- Find out combination of states for each posible input alphabet.

Step 5:

Each time we generate a new DFA state under input alphabet columns we hv to apply step 4 again otherwise go to step 6.

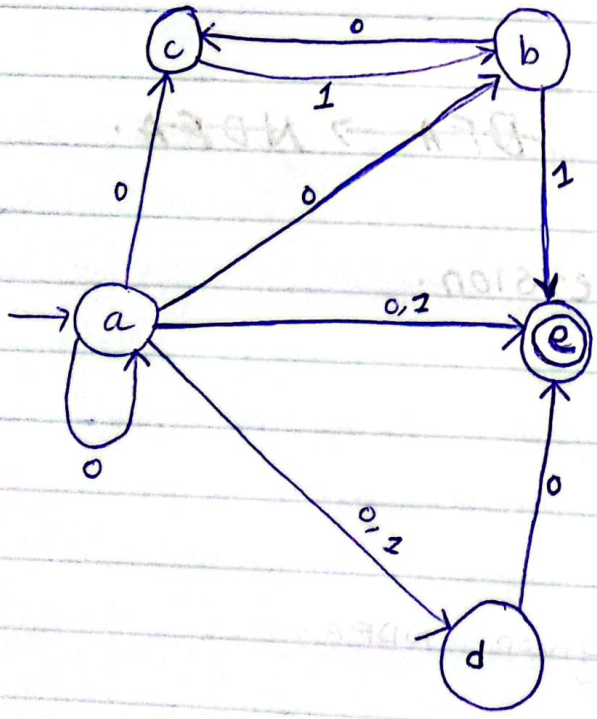
Step 6:

The state containing any of final states of NDFA are final states of equivalent DFA.

Ex.

Consider following NDFA.

⟹



Convert to DFA.

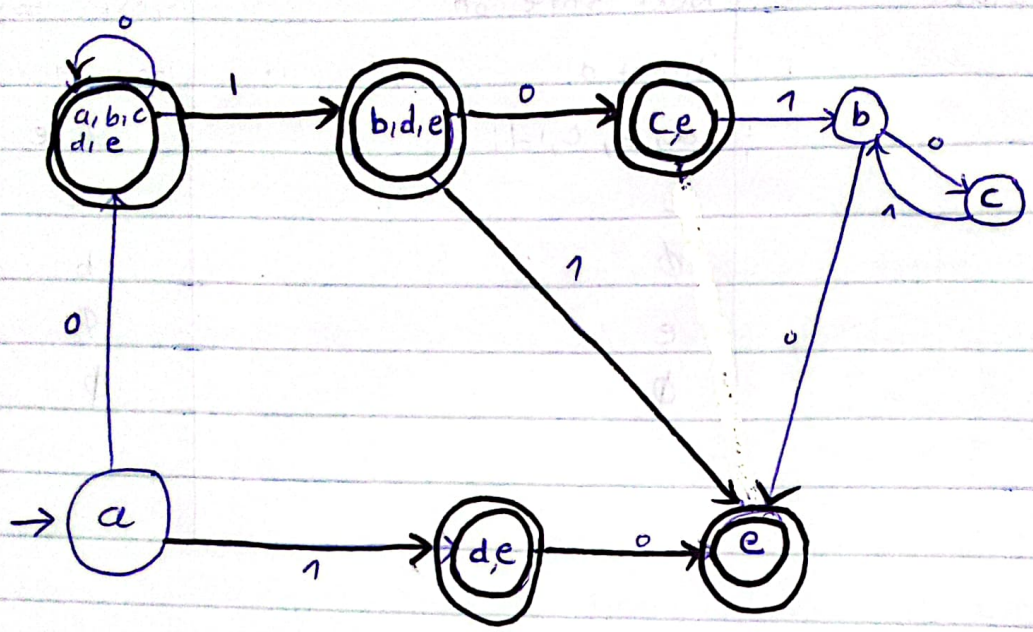
Step 1

Present state	Next state on Input 0	Next state on Input 1
→ a	a, b, c, d, e	d, e
b	c	e
c	ϕ	b
d	e	ϕ
e	ϕ	ϕ

Step 2:

Present State	Next state on Input 0	Next state on Input 1
→ a	a, b, c, d, e	d, e - COPY NOF
[a, b, c, d, e]	a, b, c, d, e	b, d, e
[d, e]	e,	ϕ
[b, d, e]	c, e,	e,
e	ϕ	ϕ
c, e	ϕ	b
b	c	e
c	ϕ	b

Diagram.



Since final state was e, any group ∋ e is final state.

DFA MINIMIZATION.

DFA minimization by Myphill - Nerode Theorem or Table filling algorithm.

Input: DFA

Output: Minimized DFA.

Step 1:

Draw table for all pairs of states not necessarily connected directly

Step 2:

Consider every state pair in ~~if~~ DFA where one state is a final state and other one is not final state and mark them.

Step 3:

~~Reapk~~ Repeat these step till we can't mark any more state.

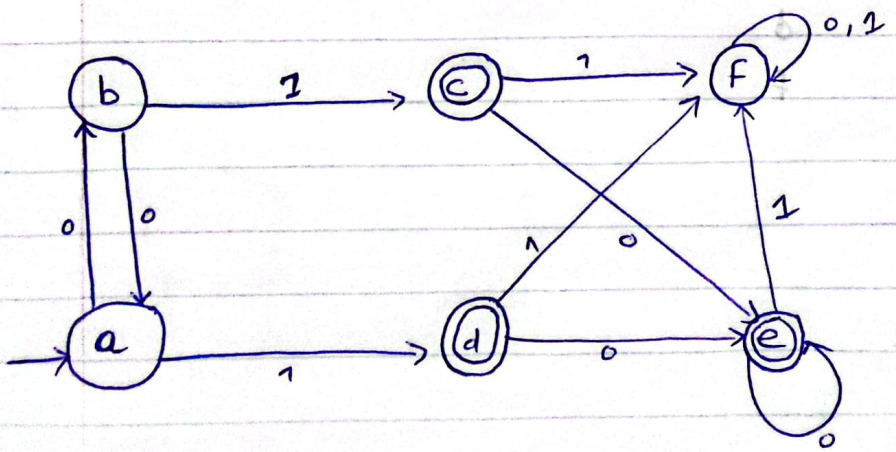
- If there's unmarked pair (q_i, q_j) mark it if the pair (Transition from q_i, A), Transition (q_j, A) is marked // $\delta(q_i, A)\ \delta(q_j, A)$ for some input alphabet A.

Step 4:

Combine all unmarked pairs and make them single state in reduced DFA

Example

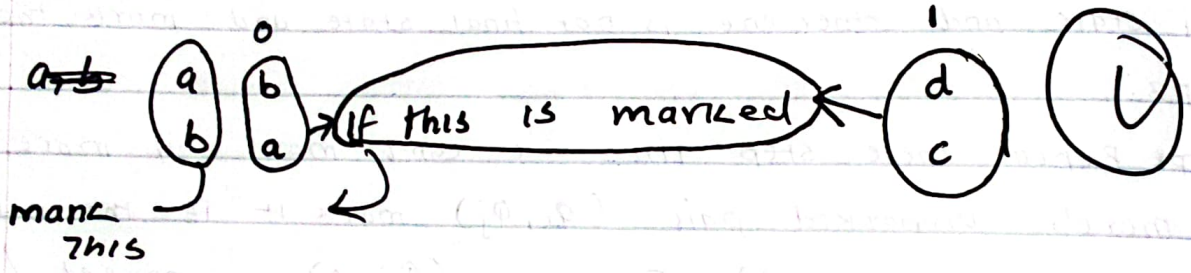
Consider following ~~DFA~~ DFA.



1 = final
2 = Not.

	a	b	c	d	e	f
a						
b						
c	✓	✓				
d	✓	✓				
e	✓	✓				
f	✓	✓	✓	✓	✓	

Step 3:



	0	1
d	e x	f x
c	e	f
e	e	f x

	0	1
a		d
f	f	f

	0	1
b	a	c
f	f	f

step 4:

combine unmarked pairs

(a,b), (c,d), (c,e), (d,e). } to avoid similarity.

(c,d) (c,e) (d,e) are similar

(c,d,e)

(a,b), (c,d,e)

Redraw.

Any group ⊆ a is initial state.