UNIT-1 Automata: why study automata theory? Introduction to formal proof, Inductive Proofs, The central concept of automata theory.
Finite Automata: Deterministic Finite automata, Nondeterministic finite automata, An Application: Text Search, Finite automata with Epsilon – Transitions.

UNIT-2 Regular Expressions and Languages: Regular expressions, Finite automata and regular expressions, Applications of regular expressions, Algebraic Laws for Regular Expressions
Properties of Regular Languages: Proving Languages not to be regular, Closure properties of regular Languages, Decision properties of Regular Languages, Equivalence and minimization of Automata.

UNIT-3 Context Free Grammars and Languages: Context Free Grammars, parse Trees, Application of Context Free Grammars, Ambiguity in Grammars and languages.
Push Down Automata: Definition of the Pushdown Automaton, The Languages of a PDA, Equivalence of PDA's and CFG's, Deterministic Pushdown Automaton.

UNIT-4 Properties of Context Free Languages: Normal Forms for Context Free Grammars, The pumping Lemma for context Free Languages, Closure Properties of Context Free Languages, and decision properties of CFL's.

UNIT-5 Introduction to Turing Machine: Problems that computer cannot solve, The Turing Machine, Programming Techniques for Turing machines, Extensions to the basic Turing Machines, Turing machines and Computers, Undecidable Problems about Turing machines. An Introduction to intractable problems.

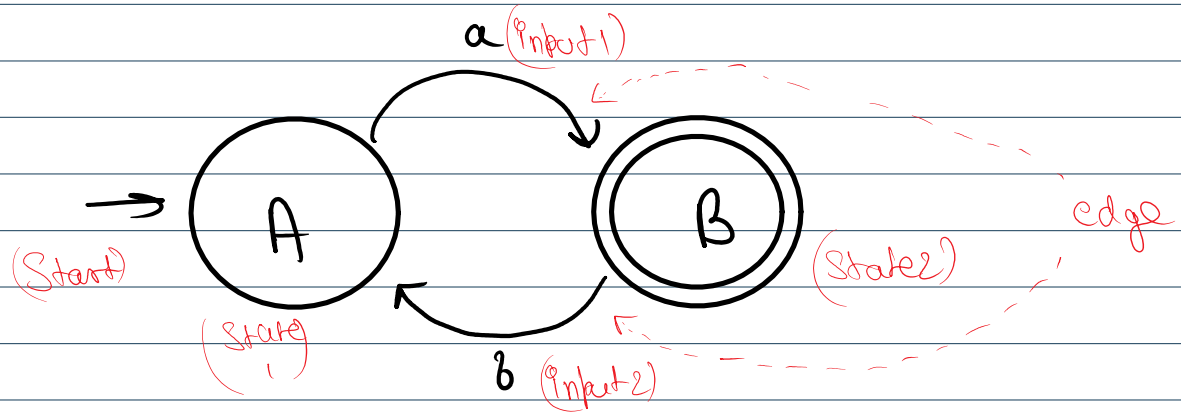
6

Automata : Study of abstract computing device or machines

Automata are used to design both hardware & software

Applications of Automata

- Software for designing & checking of digital circuits
- lexical analyzers → used in compilers
- Web crawlers → Scan large webpags does/data
- Communication protocols



State 2 is final State & State 1 is initial State

Structured representation of automata { Grammer + Regular Expression }

Grammer ; (Ex. parser in Compiler) Defines rules for arithmatic, conditional & nested etc. expressions

Regular Expression ; denote the structure of data (especially strings)

Length of Strings : denoted by || "mod"

$|10110| = 5$

Alphabets ; finite set of non empty set of symbols

$\Sigma = \{0, 1\}$; binary alphabets

Empty String ; String with 0 occurence of symbol

denoted " ϵ "

Power of Alphabet ; $* \Sigma = \{0,1\}$

$\Sigma^1 \Rightarrow 0, 1$

$\Sigma^2 \Rightarrow 00, 11, 01, 10$

$\Sigma^3 \Rightarrow 000, 111, 001, \ldots\ldots$

Σ^+ is union of $\Sigma^1 \ldots\ldots \Sigma^n$

$\Sigma^+ = \Sigma^1 + \Sigma^2 + \Sigma^3 \ldots\ldots$

Σ^* also includes Empty string

$\Sigma^* = \epsilon + \Sigma^1 + \Sigma^2 \ldots\ldots$

$= \Sigma^0 + \Sigma^+$

$\Sigma^* = \Sigma^0 + \Sigma^+$

Language : Set of Strings chosen from Σ^* that are accepted by a particular automata

denoted by "L"

If an automata A takes all strings starting from 0 then,

$$L(A) \Rightarrow \{0, 01, 00 \text{----} 0111 \text{---}\}$$

Language that doesn't contain any string is called a null language.

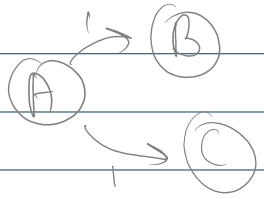
denoted by ϕ

Deterministic finite automata refers to the fact that on each input there exist one & only one state to which the automata can transition from its current state

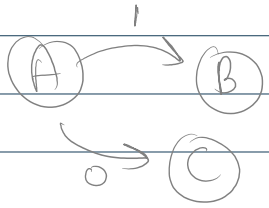
formal defination,

A DFA consists of the following tuples.

- Q ; set of States
- Σ ; set of input Symbols
- δ ; transition function / table diagram
- q_0 ; start state
- F ; set of final state



is not DFA



is DFA

$$A = (Q, \Sigma, \delta, q_0, f)$$

If w is a string of L(A) then $\hat{\delta}$ represents string processing

$$L(A) = \{ w \mid \hat{\delta}(q_0, w) \text{ is in } F \}$$

{ The language accepted by DFA (A) has string w, where $\hat{\delta}$ starting from q_0 to process w is in }

where $\hat{\delta}$ starting from q_0 to process w gives in final state

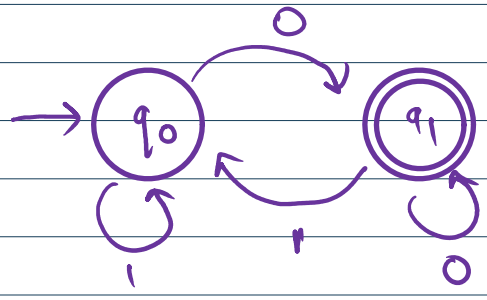
Example,

Design a DFA with input symbols $\Sigma = \{0, 1\}$ that accept all strings ending with 0

$F = \{0\}$

Ans:

A=



Checking for 0101

$\hat{\delta}(q_0, 0) = q_1$ ---> result of first transition

$\hat{\delta}(q_1, 1) = q_0$ ---> 2nd char

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

$\hat{\delta}(q_1, 1) = q_1 \Rightarrow$ not final state

had q_1 been q_0 then string would have been accepted

(String not accepted!)

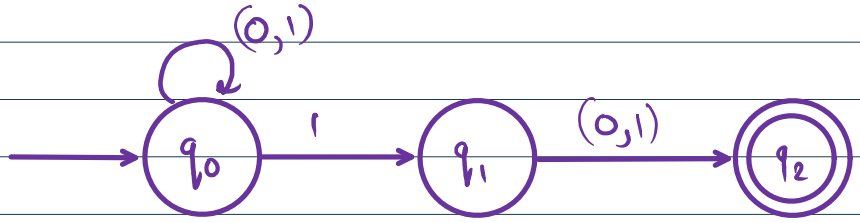
$A = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$

Unlike DFA the NFA can have any number of transition (inc. zero) to next states from given state

i.e it has power to be in several state at once

NFA $A = (Q, \Sigma, \delta, q_0, F)$

Example,



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

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

$\delta =$

	0	1
q_0	q_0	$\{q_0, q_1\}$ ← two different States
q_1	q_2	q_2
q_2	ϕ	ϕ

$q_0 = q_0$

$F = \{q_2\}$

Checking for 0101

$\delta(q_0, 0) = q_0$

$\delta(q_0, 1) = \{q_0, q_1\}$

$\delta(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$

$\Rightarrow \{q_0\} \cup \{q_2\}$

$\Rightarrow \{q_0, q_2\}$

$\delta(\{q_0, q_2\}, 1) = \delta(q_0, 1) \cup \delta(q_2, 1)$

$\Rightarrow \{q_0, q_1\} \cup \emptyset$

$\Rightarrow \{q_0, q_1\}$

$\delta(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$

$\Rightarrow \{q_0\} \cup \{q_2\}$

$\Rightarrow \{q_0, q_2\}$

As the final transition contains q_2 (which is final state) the string is accepted

If w is a string accepted by $L(B)$ where B is NFA then

$$L(B) = \{ w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset \}$$

{ In language of NFA "B", w is string for $\hat{\delta}$ starting from q_0 to process w has common state from F (i.e intersects) }

NFA → DFA

For,

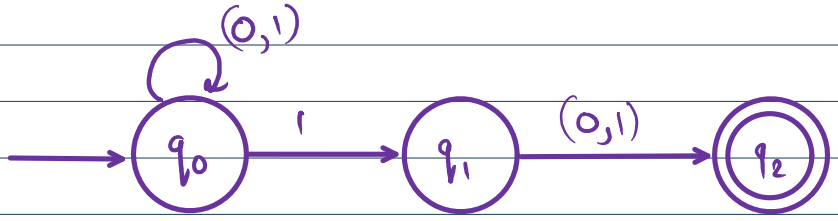
$M = (Q, \Sigma, \delta, q_0, F)$ ⇒ NFA which accepts lang. $L(M)$

$M' = (Q', \Sigma', \delta', q_0', F')$ ⇒ DFA $L(M')$

Such that $L(M') = L(M)$

1. Initially $Q' = \emptyset$
2. Add q_0 (NFA) to Q' (DFA)
   then find transition from this start stage
3. In Q' (DFA) find all possible sets of states for each input symbol
   If the set of state is not in Q' (DFA) then add it to Q'
4. In DFA, F' will have all states which were in F (NFA)

Example,



NFA $M = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_2\})$

$\delta \Rightarrow$

	0	1

$\delta \Rightarrow$

	0	1
q_0	q_0	$\{q_0, q_1\}$ ← - - - - - two different State
q_1	q_2	q_2
q_2	ϕ	ϕ

DFA $M' = \{ Q', \{0,1\}, \delta', q_0, F \}$

(known)

$\delta' \Rightarrow$

	0	1
known starting point → q_0	q_0	q_0q_1 ----> new state / This is a single Set (empty)
q_0q_1	q_0q_2	$q_0q_1q_2$ ----> new state
* q_0q_2	q_0	q_0q_1 (not a new state)
* $q_0q_1q_2$	q_0q_2 (not new state)	$q_0q_1q_2$ (not a new state)

Q'

Therefore,

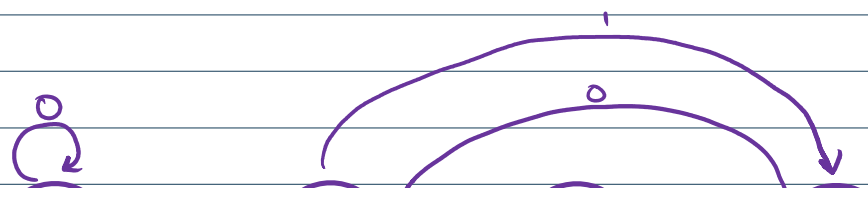
$Q' = \{ q_0, q_0q_1, q_0q_2, q_0q_1q_2 \}$

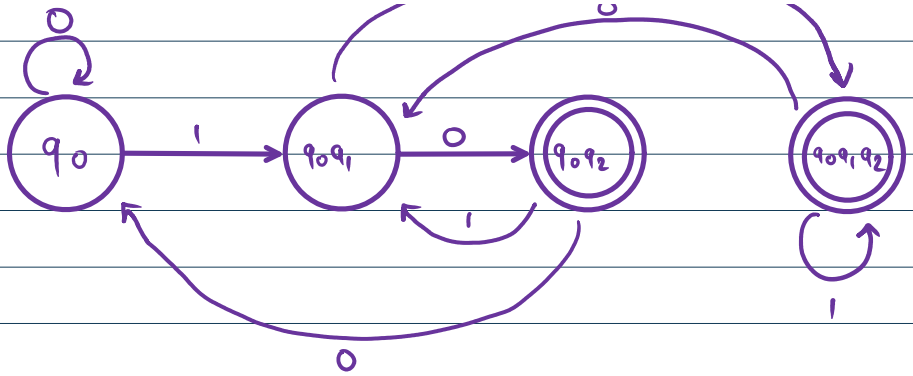
F' = any Q' containing FC (from NFA)

$\Rightarrow \{q_0q_2, q_0q_1q_2\}$

DFA $M = (\{q_0, q_0q_1, q_0q_2, q_0q_1q_2\}, \{0,1\}, \delta', q_0, \{q_0q_2, q_0q_1q_2\})$

Later q_0q_1, q_0q_2, $q_0q_1q_2$ can be renamed as A, B, C ...




0
1
0
0
1
1
0
q_0
q_0q_1
q_0q_2
$q_0q_1q_2$

E - NFA

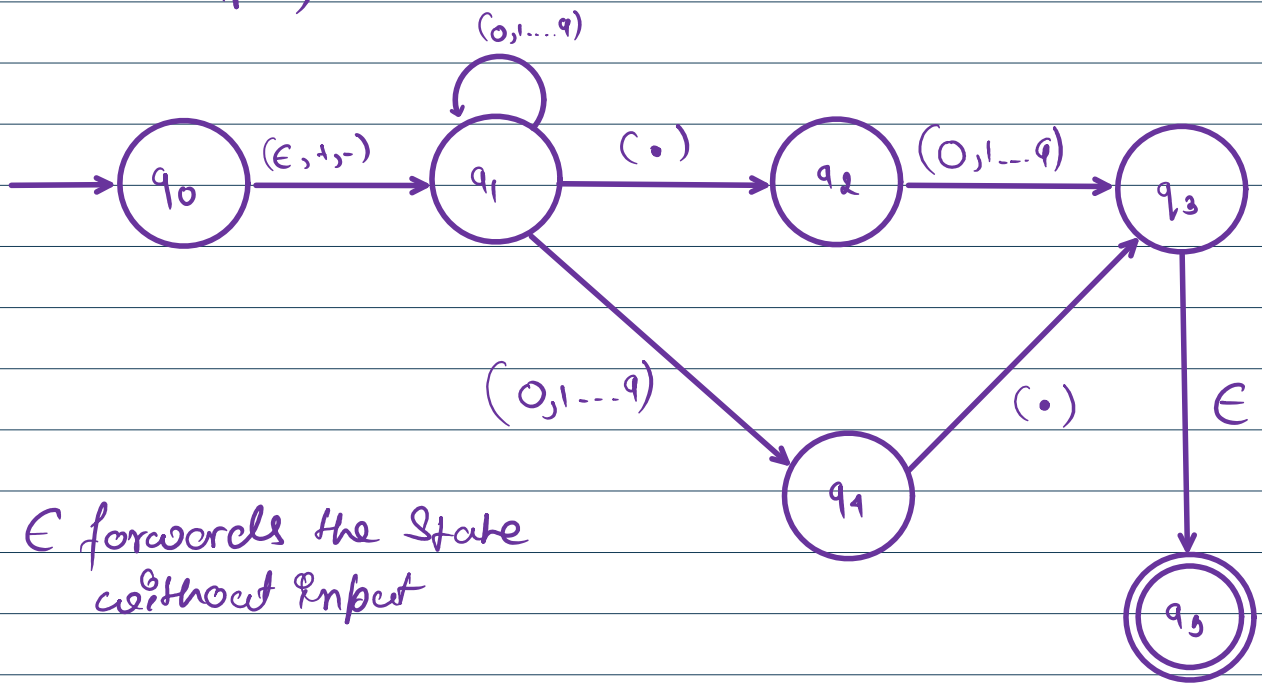
23 September 2022 05:02 PM

ϵ represents an input symbol which takes no string

ϵ-NFA $A = (Q, \Sigma, \delta, q_0, F)$

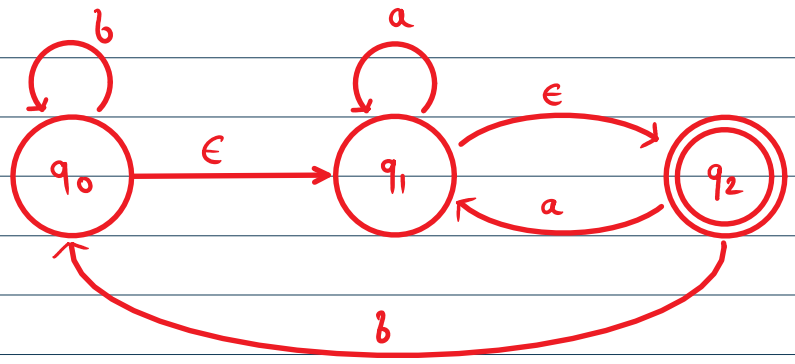
where Σ = Input Symbols $\cup \epsilon$

Example,



ϵ forwards the state without input

Example,



ε closure $(q_0) = \{q_0, q_1, q_2\}$ → (A)

ε closure $(q_1) = \{q_1, q_2\}$ → (B)

ε closure $(q_2) = \{q_2\}$

{Closure is basically states it could go without any input from itself}

$$\delta'(A,a) = \epsilon\text{Closure}(\delta(A,a)) \quad - (1)$$

for a in A

gives B

$$\delta(A,a) = \delta(q_0,a) \cup \delta(q_1,a)\ \delta(q_2,a)$$
$$\Rightarrow \phi \cup q_1 \cup q_1$$
$$\delta(A,a) = \{q_1\}$$

$$\delta'(A,a) = \epsilon\text{Closure}(\delta(A,a)) = \varepsilon\text{Closure}(q_1)$$

$$\delta'(A,a) = \varepsilon\text{Closure}(q_1) = \{q_1, q_2\}\ (B)$$

not a new state

for b in A

for b in A

$$\delta'(A,b) = \varepsilon\text{Closure}(\delta(A,b))$$

gives A

$$\delta(A,b) = \delta(q_0,b) \cup \delta(q_1,b) \cup \delta(q_2,b)$$
$$= q_0 \cup \phi \cup q_0$$
$$= q_0$$

$$\delta'(A,b) = \varepsilon\text{Closure}(q_0)$$
$$= \{q_0 q_1 q_2\} \quad (A)$$

not a new state

for a in B

$$\delta'(B,a) = \varepsilon\text{Closure}(\delta(B,a))$$

gives B

$$\delta(B,a) = \delta(q_1,a) \cup \delta(q_2,a)$$
$$= q_1 \cup q_1$$
$$= q_1$$

$$\delta'(B,a) = \varepsilon\text{Closure}(q_1)$$
$$= \{q_1 q_2\}$$

not a new state

(B)

for b in B

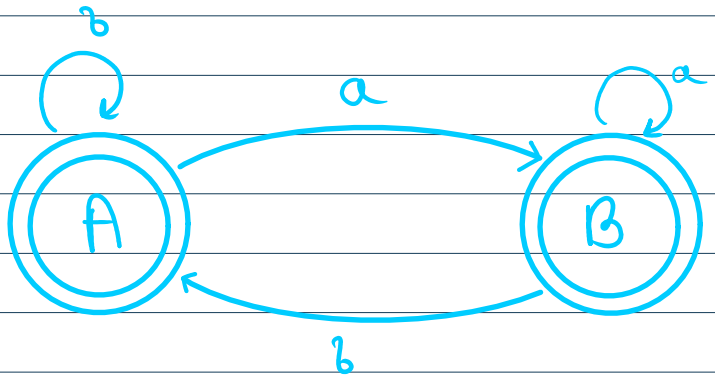
$$\delta'(B,b) = \varepsilon\text{Closure}(\delta(B,b))$$

$$\delta(B,b) = \delta(q_1,b) \cup \delta(q_2,b)$$
$$= \phi \cup q_0$$
$$= q_0$$

gives A

$$\delta'(B,b) = \varepsilon\text{Closure}(q_0)$$

$$\delta'(B,b) = \varepsilon\text{Closure}(q_0)$$
$$= \{q_0, q_1, q_2\} \quad (A)$$



A & B both contain q_2 so both final.

?E-NFA to DFA

23 September 2022 05:14 PM

??

31 October 2022 05:55 PM

I don't Know How I didn't Write anything for Regular Expression

Application of Regular Expression

1. Lexical Analyzer
2. Finding pattern in text

?Context Free Grammar

31 October 2022 06:03 PM

Content Free Grammer (formal definition)

CFG : { V, T, P, S }

V : Finite set of Variables (Non-terminal)

T ; Finite set of Terminals

P : Production Rules

S : Start Variables

Example

_Parse Trees

23 December 2022 04:20 AM

Ambiguous Grammar

22 December 2022 04:20 PM

A grammar is said to be ambiguous if there exists two or more derivation tree for a string

(* >> + in precedence)

Example,

$G = \{ \{S\}, \{a, b, +, *\}, P, S \}$

where P consists of

$S \longrightarrow S+S \mid S*S \mid a \mid b$

now, string $a+a*b$ will be generated as

$S \rightarrow S + S$ ($S \rightarrow a$)

$S \rightarrow a + S$ ($S \rightarrow S*S$)

$S \rightarrow a + S*S$ ($S \rightarrow a$)

$S \rightarrow a + a*S$ ($S \rightarrow b$)

$S \rightarrow a + a*b$

additionally it can be done as

$S \rightarrow S*S$ ($S \rightarrow S+S$)

$S \rightarrow S+S*S$ ($S \rightarrow a$)

$S \rightarrow S+S*S$ $(S \rightarrow a)$

$S \rightarrow a+S*S$ $(S \rightarrow a)$

$S \rightarrow a+a*S$ $(S \rightarrow b)$

$S \rightarrow a+a*b$

Thus the Grammer is ambiguous

Another Example,

$E \rightarrow E+E \mid E*E \mid id$

↓

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$f \rightarrow id$

This isn't ambiguous

Push Down Automata

22 December 2022 06:37 PM

Basically a PDA is an NFA with a Stack

Formal definition:

$$P = \{ Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \}$$

Q : Set of States

Σ : Set of Input Symbols

Γ : Set of Stack Symbol

δ : Transition function

q_0 : Start State

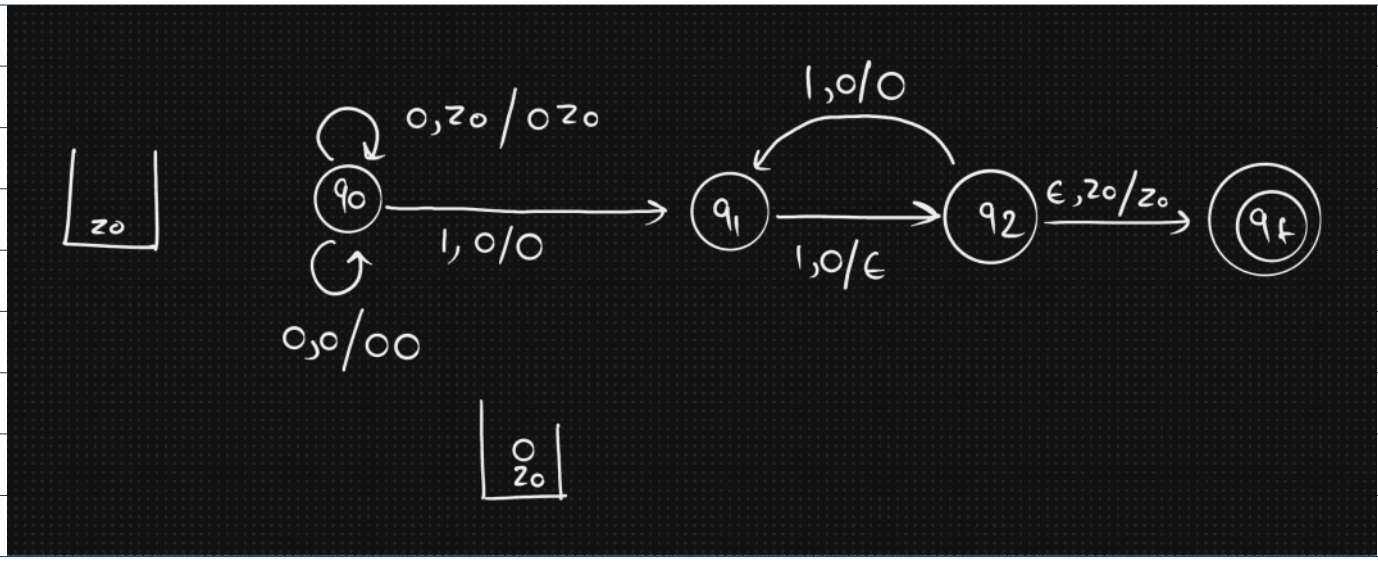
Z_0 : Top of Stack

F : Set of final states

Example,

Design a PDA for the CFG $0^n 1^{2n}$

Example String 00111 { for 1 zero 2 one }


Z0
q0
0,Z0/0Z0
1,0/0
0,0/00
q1
1,0/0
1,0/ϵ
q2
ϵ,Z0/Z0
qf
0
Z0

!Deterministic PDA

22 December 2022 06:37 PM

Applications of Context free Grammar

- Parsers
- YACC - Parser generator
- Markup language

Decision Properties of CFL's & it's Applications

23 December 2022 04:07 AM

Decision properties states that we can develop algorithm that answer the following questions

- is given string in the language (membership)
- is language empty (emptyness)
- is language finite (finiteness)

- Cannot directly feed the string as PDA is Non-deterministic so it can go in all possible path
- Pumping lemma can derive !? (2n-1) ways

Applications

- Used in Compilers
- Used in finite state machines
- Used in parsers & markup languages

Let L be CFL, then there exists a constant n such that z is any string in L such that |z| is at least n

So Suppose z = uvwxy

@ break in five part & pump 2nd & 4th

Such that,

① $|vwx| \leq n$ i.e middle part won't too large

② $vx \neq \varepsilon$ @ one strings we pump should't be empty

③ $\forall i \geq 0$

$uv^i wx^i y$ is in L

?Simplification of CFG

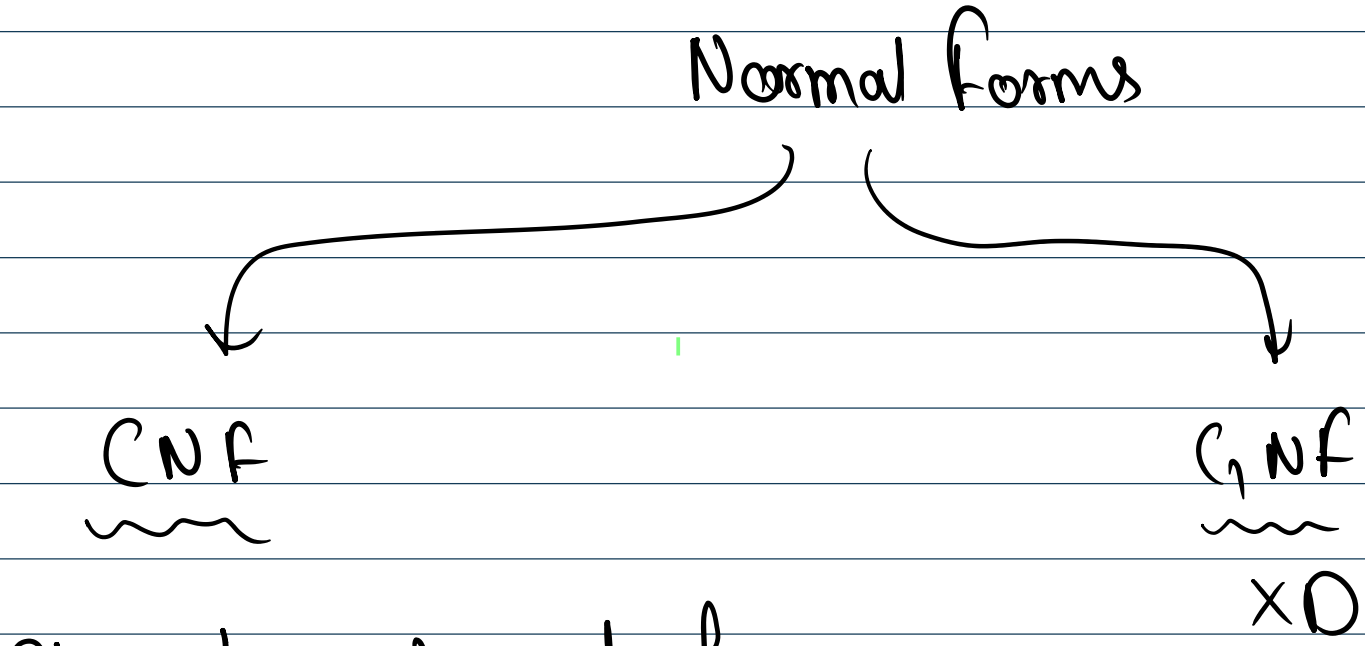
23 December 2022 04:03 AM

Simplification of CFG

1. Removal of CFG
2. Elimination of ε - productions
3. Removal of Unit productions

?Normal Forms of CFG

23 December 2022 04:22 AM



Chomskey Normal form

- $A \rightarrow BC$

$A \rightarrow a$

Turing Machine

$M = \{ Q, \Sigma, \Gamma, \delta, q_0, B, F \}$

Q : Finite Set of States

Σ : Set of Input Symbols

Γ : Complete set of tape symbols

δ : Transition function

q_0 : Initial State

B : Blank Symbol

F : Set of final States

$\delta(q, x) \rightarrow (p, y, D)$

- q : Current State
- x : tape Symbol
- p : next State
- y : New tape Symbol to be replaced with
- D : Direction (L, R)

Turing machine is a more capable machine model than

U CFL as it has the capacity to read as well as write on both direction on tape