# TOC

© Difference b/w DFA and NFA.

1. DFA: DFA refers to Deterministic Finite Automation.
A Finite Automata (FA) is said to be deterministic if. Corresponding to an input symbol, there is a single resultant state i.e. there is only one transition. A deterministic finite automata is set of five tuples represented as,

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

- A non-empty finite set → A non-empty finite set of input symbol.
- A non-empty finite set of. → It is a non-empty ~~finite~~ set of final states $\in Q$
- It is starting state, $q_0 \in Q$.
- It is a transition function that takes 2 arguments, a state, and a input symbol, it return a single state.

2. NFA: NFA refers to Nondeterministic Finite Automation.
A Finite Automata (FA) is said to be non-deterministic if. there is more than one possible transition from one state on the same input symbol.
A non-deterministic finite automata is also a set of five tuples and represented as,

$M = \{Q, \Sigma, \delta, q_0, F\}$ ..... same as above.

## Extra :

① String: Finite set of symbols from alphabet or collection of alphabet or sequence of alphabet.

eg: {a, b, ab, ba, bb, ..}

length = 2.

{aa, ab, ba, bb}

② Sub-string: A substring is a contiguous part of a string i.e., a string inside another string.

③ Length of string: Length of a string is denoted as |w| and is defined as the number of positions for the symbol in the string.

[collection of string]

④ Working model of Finite Automata

IMP

| DFA | NFA |
| --- | --- |
| • DFA stands for Deterministic Finite Automata. | • NFA stands for Nondeterministic Finite Automata. |
| • For each symbolic representation of the alphabet, there is only state transition in DFA | • No need to specify how does the NFA reach according to some symbol. |
| • DFA cannot use Empty string transition. | • NFA can use Empty string transition |
| • DFA can be understood as one machine. | • NFA can be understood as multiple little machines computing at the same time. |
| • In DFA, the next possible state is distinctly set. | • In NFA, each pair of state and input symbol can have many possible next states. |
| • DFA is more difficult to construct | • NFA is easier to construct |
| • DFA rejects the string in case it terminates in a state that is different from the accepting state. | • NFA rejects the string in the events of all branches dying or refusing the string. |
| • Time needed for executing an input string is less | • Time needed for executing an input string is more. |
| • All DFA are NFA | • Not all NFA are DFA. |
| • DFA requires more space | • NFA requires less space then DFA. |

Q Grammer and language & difference between.

⇒ In theory of computation (TOC), a language is a set of sentences made from a set of basic symbols. A grammer is a set of rules that determine if these sentences are part of the language.

Language:

- A set of sentences formed from a set of basic symbols.
- A possibly infinite set of valid sequences of terminal symbols.
- A structured system of communication that consists of grammer and vocabulary.

Grammar:

- A set of rules that govern how we determine if these sentences are part of the language or not.
- A specification of a set of rules that define a language.
- A finite set of formal rules that are generating syntactically correct sentences.
- A set of production rules which are used to generate strings of language.

| Language. | Grammer |
|---|---|
| $L_1 = \{aa, bb, ab, ba\}$ <br> OR <br> $L(G) = \{s \mid s \in T^* \text{ and } S \to ts\}$ | $S \to a$ , $S \to b$. <br> $S \to aSb$ |

Q. Construct DFA equivalent to NFA($\{p, q, r, s\}, \{0, 1\}, p, \{s\}, \delta$) where $\delta$ is given by.

| → | S | 0 | 1 |
|---|---|---|---|
| → | p | p, q | p |
| | q | r | r |
| | r | s | - |
| | s | s | s |

⇒ 0, 1 → input
p → initial state
S → final state.

The states Q is a subset of $\{p, q, r, s\}$ is $\{\{\emptyset\}, \{p\}, \{r\}, \{s\}, \{p, q\}\}$.

Transition table ⇒

| State | 0 | 1 |
|---|---|---|
| P | PQ | P |
| Q | R | R |

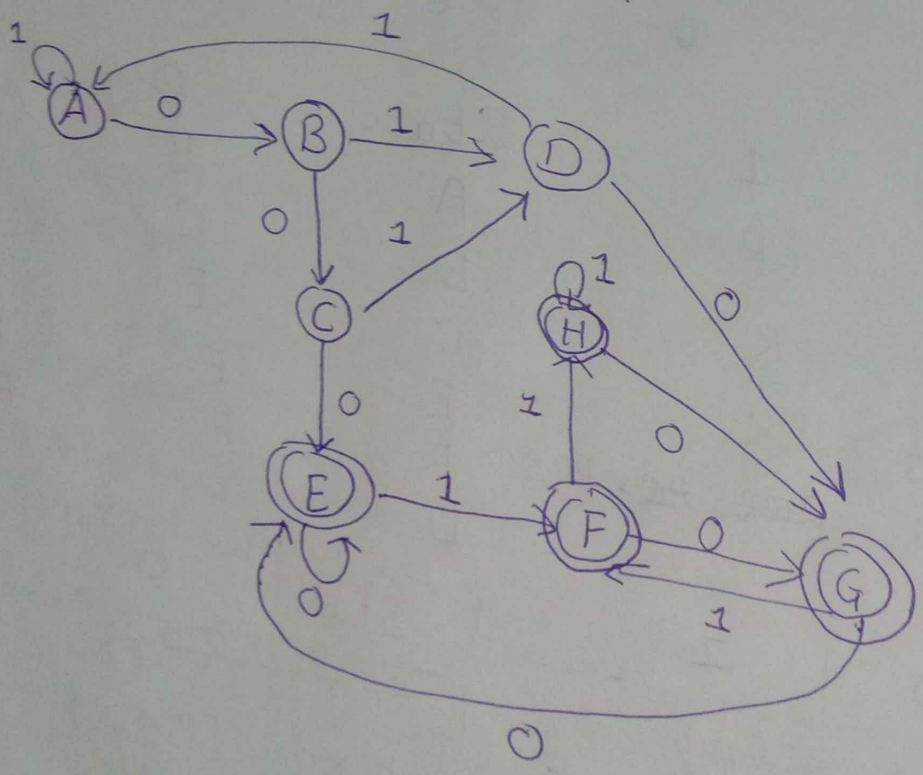
| State | 0 | 1 |
|---|---|---|
| P | PQ | P |
| PQ | PQR | R |
| PQR | PQRS | X |
| PQRS | PQRS | ~~PQRS~~ ~~PQX~~ PRS |

| State | 0 | 1 |
|---|---|---|
| A | B | A |
| B | C | D |
| C | E | F |
| E | E | E |
| F | F | F |
| D | | |

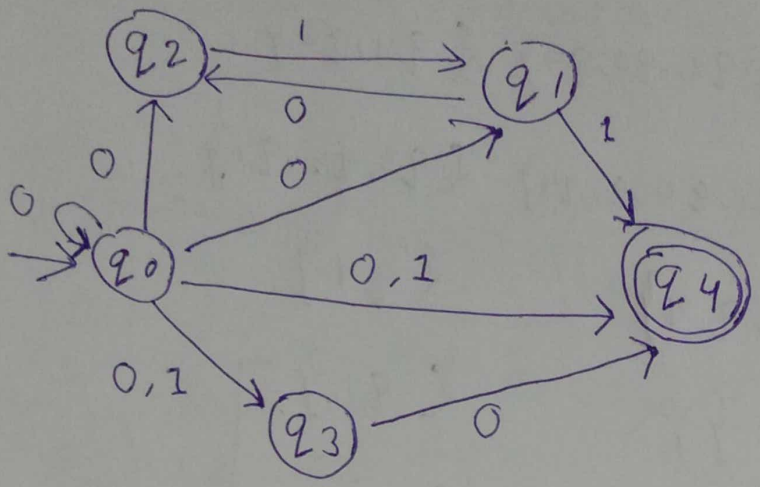
| State | 0 | 1 |
|---|---|---|
| {p} | {p, q} | {p} |
| {p, q} | {p, q, r} | {p, r} |
| {p, q, r} | {p, q, r, s} | {p, r} |
| {p, q, r, s} | {p, q, r, s} | {p, r, s} |
| {p, r, s} | {p, q, s} | {p, s} |
| {p, q, s} | {p, q, r, s} | {p, r, s} |
| {p, s} | {p, q, s} | {p, s} |

| State | 0 | 1 |
|---|---|---|
| ~~{p, r}~~ | ~~{p, r}~~ | |
| {p, r} | {p, q, s} | {p} |

| State | 0 | 1 |
|---|---|---|
| A | B | A |
| B | C | D |
| C | E | D |
| E | E | F |
| F | G | H |
| G | E | F |
| H | G | H |
| D | G | A |



Q. Construct DFA which is equivalent to following NFA over $\Sigma = \{0, 1\}$.




State Table:

| State | 0 | 1 |
|---|---|---|
| → $q_0$ | $\{q_0, q_1, q_2, q_3, q_4\}$ | $\{q_4, q_3\}$ |
| $q_1$ | $\{q_2\}$ | $\{q_4\}$ |
| $q_2$ | $\{\phi\}$ | $\{q_1\}$ |
| $q_3$ | $\{q_4\}$ | $\{\phi\}$ |
| (q4) | $\{\phi\}$ | $\{\phi\}$ |

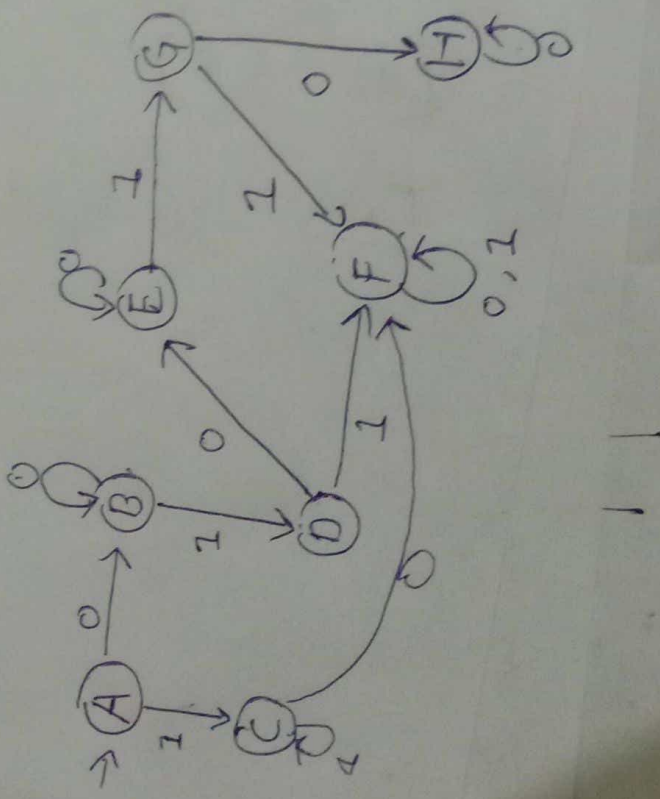
input → 0, 1.
initial state → $q_0$.
final state → $q_4$

Transition table:

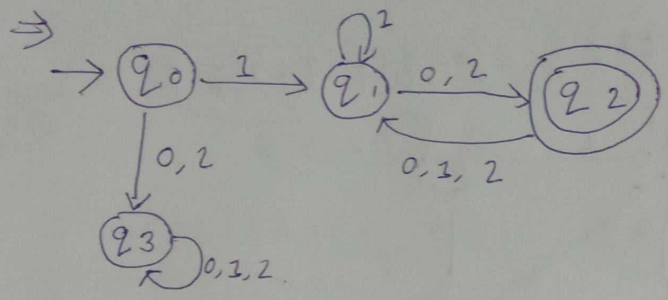
Transition Table:

| State | 0 | 1 |
|---|---|---|
| → {q0} | {q0,q1,q2,q3,q4} | {q4,q3} |
| {q0,q1,q2,q3,q3} | {q0,q1,q2,q3,q4} | {q3,q4,q1} |
| {q3,q4,q1} | {q2,q4} | {q4} |
| {q2,q4} | {ϕ} | {q1} |
| {q4} | {ϕ} | {ϕ} |
| {q4,q3} | {q4} | {ϕ} |
| {q1} | {q2} | {q4} |
| {q2} | {ϕ} | {q1} |

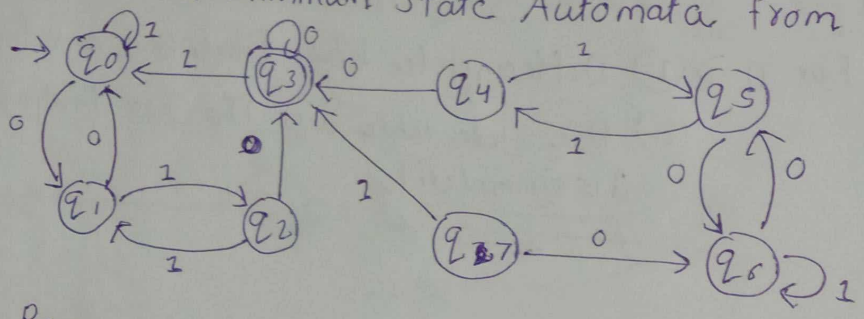
| State | 0 | 1 |
|---|---|---|
| → A | B | C |
| B | B | D |
| D | E | F |
| E | E | G |
| F | F | F |
| G | H | F |
| H | H | G |
| C | F | C |



Q Design DFA which accept string starting with '1' and does not end with '1' over $\Sigma = \{0, 1, 2\}$




Q Construct minimum state Automata from the following



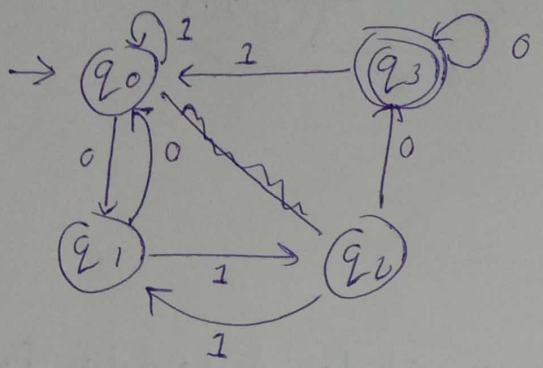

⇒ Remove $q_4, q_5, q_6, q_7$ as it is not reachable from final state.

| State | 0 | 1 |
|---|---|---|
| $q_0$ | $q_1$ | $q_0$ |
| $q_1$ | $q_0$ | $q_2$ |
| $q_2$ | $q_3$ | $q_1$ |
| $(q_3)$ | $q_3$ | $q_0$ |

$\pi_0 = \{q_0, q_1, q_2\}\{q_3\}$

$\pi_1 = \{q_0, q_1\}\{q_2\}\{q_3\}$

$\pi_2 = \{q_0\}\{q_1\}\{q_2\}\{q_3\}$

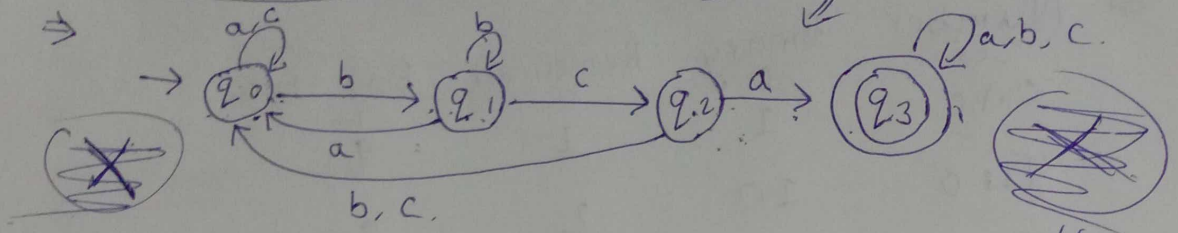



Note: For $\pi_1$ ⇒ (1) Differenciate b/w final and non-final.

(2) Use state table and $\pi_0$ for finding out dissimilarity

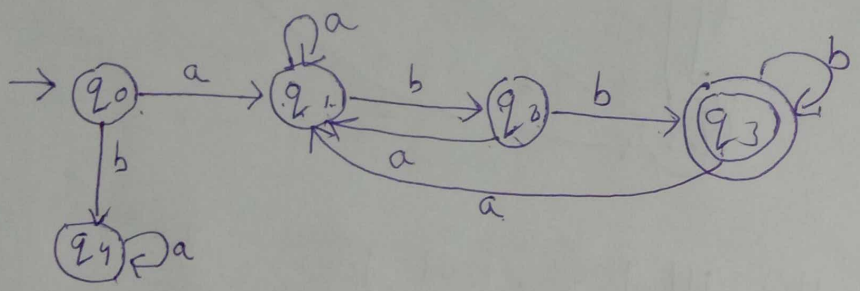
① Construct a DFA for the set of string over {a, b, c} having ~~bca~~ ~~abc~~ abc as a substring. but bca substring is right

⇒




L = {bca, bbbbca, aacccbca, bcabbcc, ~~abc~~ aabbcaacc, .... }

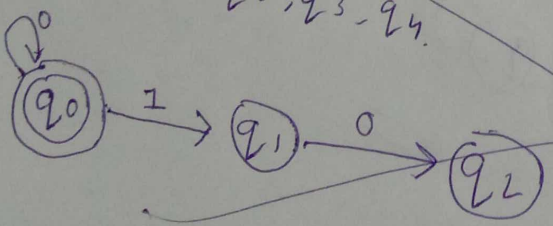
② Construct ~~DFA~~ DFA starting with a & ending with bb over.
Σ = {a, b}




L = {abb, aaababb, ababb, ...}

③ Design a DFA to check whether the given binary number is divisible by 5.

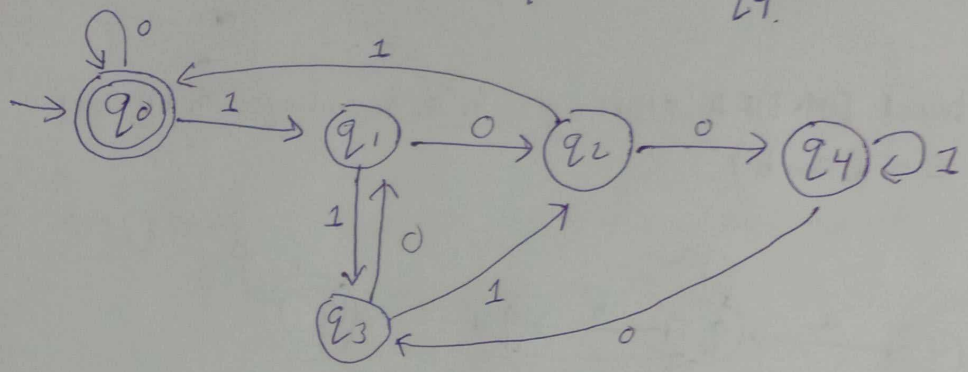
$q_0, q_1, q_2, q_3, q_4$.



Q] Design a DFA to check weather the given binary number is divisible by 5.

⇒

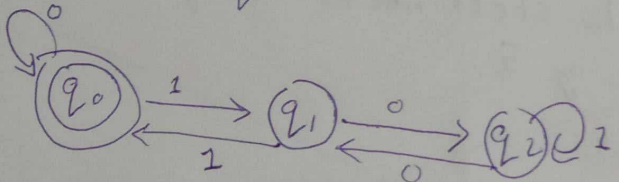
| Number | Binary | Remainder | End-state |
|---|---|---|---|
| Zero | 0 | 0 | $q_0$ |
| one | 1 | 1 | $q_1$ |
| Two | 10 | 2 | $q_2$ |
| Three | 11 | 3 | $q_3$ |
| Four | 100 | 4 | $q_4$ |



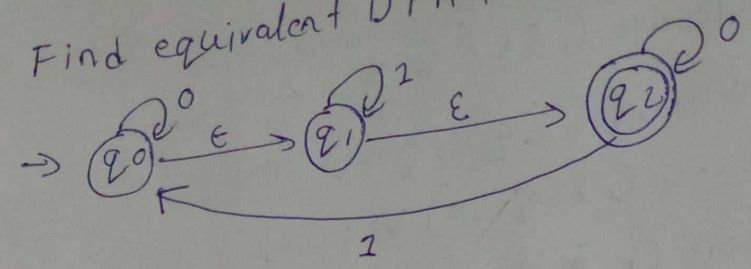

$L = \{101, 110, 111, 1000, 1001, 1010, \ldots\}$

Q] Design a DFA to check weather the given binary number is divisible by 3.

⇒ $q_0, q_1, q_2$.



Q Find equivalent DFA for following diagram.




⇒

State table:

| State | 0 | 1 | ε |
|---|---|---|---|
| → $q_0$ | $q_0$ | - | $q_1$ |
| $q_1$ | - | $q_1$ | $q_2$ |
| (*$q_2$) | $q_2$ | $q_0$ | - |

ε-closure of $q_0 = \{q_0, q_1, q_2\}$

—11— $q_1 = \{q_1, q_2\}$

—11— $q_2 = \{q_2\}$

$\delta(q_0, 0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, \epsilon), 0))$

$= \epsilon\text{-closure}(\delta(q_0, q_1, q_2), 0))$

$= \epsilon\text{-closure}(\delta((q_0, 0) \cup (q_1, 0) \cup (q_2, 0)))$

$= \epsilon\text{-closure}(q_0 \cup \phi \cup q_2)$

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

$\delta(q_0, 1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, \epsilon), 1))$

$= \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, q_1, q_2), 1))$

$= \epsilon\text{-closure}(\delta((q_0, 1), (q_1, 1), (q_2, 1)))$

$= \epsilon\text{-closure}(\phi \cup q_1 \cup q_0)$

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

$\delta(q_1, 0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_2, \epsilon), 0))$

$= \epsilon\text{-closure}(\delta(q_1, q_2), 0))$

$= \epsilon\text{-closure}(\delta(q_1, 0) \cup (q_2, 0))$

$= \epsilon\text{-closure}(\delta \phi \cup q_2)$

$= \epsilon\text{-closure}(q_2)$

$= q_2$

$\delta(q_1, 1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1, \epsilon), 1))$

$= \epsilon\text{-closure}(\delta(q_1, q_2), 1))$

$= \epsilon\text{-closure}(\delta(q_1, 1) \cup (q_2, 1))$

$= \epsilon\text{-closure}(q_1 \cup q_0)$

$= q_0, q_1, q_2$.

$\delta(q_2, 0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_2, \epsilon), 0))$

$= \epsilon\text{-closure}(\delta(q_2), 0))$

$= \epsilon\text{-closure}(\phi(q_2, 0))$

$= \epsilon\text{-closure}(q_2)$

$= q_2$

$\delta(q_2, 1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_2, \epsilon), 1))$

$= \epsilon\text{-closure}(\delta(q_2), 1))$

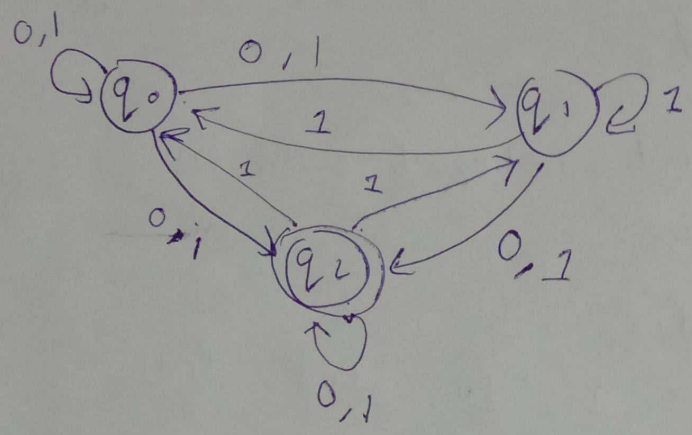
$= \epsilon\text{-closure}(\delta(q_2, 1))$

$= \epsilon\text{-closure}(q_0)$

$= q_0, q_1, q_2$.

Transition table

| State | 0 | 1 |
|---|---|---|
| $q_0$ | $\{q_0, q_1, q_2\}$ | $\{q_0, q_1, q_2\}$ |
| $q_1$ | $q_2$ | $\{q_0, q_1, q_2\}$ |
| $(q_2)$ | $q_2$ | $\{q_0, q_1, q_2\}$ |




---

Note! Mathamatical Induction

is Is Remaining

Define:

i) Prefix of a string: A substring with the sequence of beginning symbols of a given string is called "prefix"

ii) Suffix: Ending symbols of given string.

iii) Proper prefix:

A prefix that is not equal to the string itself and is non-empty.

iv) Proper Suffix:

A suffix that is not equal to the string itself and is non-empty.

Eg: String "India".

Proper prefix: "I", "In", "Ind", "Indi".

Proper suffix: "a", "ia", "dia", "ndia".

v) Star closure of a language:

The star closure of a language is the Kleene Closure, which is the concatenation of zero, one, two, or any countable number of strings

eg: $*\{\epsilon, 1, 11, 111, \ldots\}$

vi) Positive closure of a language:

The set of finite-length strings that can be generated by concatenating arbitrary elements of set of strings allowing the use of the same element multiple times.

$+\{1, 11, 111\}$