01. Define Automata Differentiate between DFA and NFA?

Automata are abstract, self-operating Mathematical models that process input sequences to perform computations by moving through a finite number of states according to a set of rules accepting or rejecting input string based on whether a final state is reached.

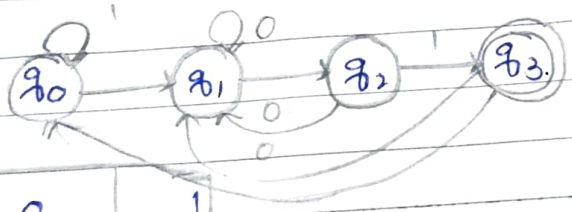
| DFA | NFA |
|---|---|
| 01. DFA stands for Deterministic Finite Automata. | 01. NFA stands for Non-Deterministic Finite Automata. |
| 02. For each symbolic representation of the alphabet, there is only one state transition in DFA. | 02. No need to specify how does the NFA react according to some symbol. |
| 03. DFA cannot use Empty string transition | 03. NFA can use empty string transition. |

| | |
|---|---|
| 04. DFA can be understood as one machine. | 04. NFA can be understand as Multeple lettle Machines computing at same teme. |
| 05. In DFA, the next possible state is destenct set | 05. In NFA, each pair of state and input can Can have many possible states. |
| 06. DFA is more difficult to Construct. | 06. NFA is easier to construct. |
| 07. Teme needed for executing an input string is less. | 07. Teme needed for executing an input string is more. |
| 08. All DFA are NFA | 08. Not all NFA are DFA. |
| 09. It requires more space. | 09. NFA requires less space than DFA. |
| 10. Epsilon move is not allowed in DFA. | 10. Epsilon more is allowed in NFA. |

02. Construct a DFA for the language Contaening Set of strings ending with 011.

A) Input alphabet $\Sigma = \{0, 1\}$
Given to construct a DFA ending with '011'.

DFA:




| $\Sigma$ | 0 | 1 |
|---|---|---|
| $q_0$ | $q_1$ | $q_0$ |
| $q_1$ | $q_1$ | $q_2$ |
| $q_2$ | $q_1$ | $q_3$ |
| $q_3$ | $q_1$ | $q_0$ |

03. Define ambiguous grammar, check whether the grammar is ambiguous or not for the string id + id + id ?

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow id$

A) Ambiguous Grammar:-

A ambiguous grammar is a grammar that can generate the same input string in more than one ways resulting in multiple distinct parse trees or left most derivations for that string

Given Grammar.

$E \rightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow id$

A grammar is ambiguous grammar if it satisfies left most derivation and right most derivation

Left most derivation:-

$E*E$

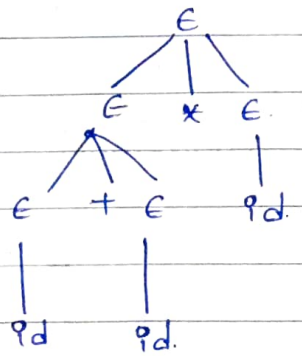
$E+E*E$

$id+E*E$

$id+id*E$

$id+id*id$

parse tree




Right most derivate

$E+E$

$E+E*E$

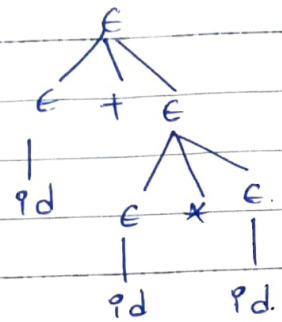
$E+E*id$

$E + E * id$

$E + id * id$

$id + id * id$

parse tree




04. Define the left factoring and left recursion. Elemenate the recursion for following grammar.

$E \rightarrow E+T/T$

$T \rightarrow T*F/F$

$F \rightarrow (E)/id$

Left Recursion

A production is left recursive if a non-terminal can derive a sentential form where leftmost symbol is the same non-terminal.

Ex:- $E \rightarrow E+T/T$

Left Factoring:-

Left factoring is a grammar transformation to remove common prefixes from alternatives. so a top-down parser can decide which alternative

to take by looking at a fenete look ahead

Ex:- $A \rightarrow ab|ac$

given grammar

$E \rightarrow E+T|T$

$T \rightarrow T*E|F$

$E \rightarrow (E)|id$

Both E and T have immediate left recursion

Eleminate left Recursion of E:

$E \rightarrow TE'$.

$E' \rightarrow +TE'|\epsilon$.

Eleminate left recursion of T:

$T \rightarrow FT'$.

$T' \rightarrow *FT'|\epsilon$

Resultant Grammar (No left Recursion)

$E \rightarrow TE$

$E' \rightarrow +TE'|\epsilon$

$T \rightarrow ET'$.

$T' \rightarrow *FT'|\epsilon$

$F \rightarrow (E)|id$.

05. Consider the grammar

$S \rightarrow AA$

$S \rightarrow Aa/b$

perform LR(0) for the input "aabb"?

LR(0) parser

The first L represents read the input from left to right

The 'R' represents construct right most derivation in reverse order.

Steps to construct LR(0) parser:-

(i) Add augment production to grammar

(ii) Create canonical connection of LR(0) items

(iii) Draw data flow diagram (DFA).

(iv) Construct LR(0) parsing Table

(v) Check for input string.

Given string and Grammar

$S \rightarrow AA$

$A \rightarrow aA/b$.

String: aabb.

Step-01:-

Add augument production to grammar

$S' \rightarrow S.$

$S \rightarrow AA$

$A \rightarrow aA/b.$

Step-02:-

Create canonecal connections of LR(0) parser

$S' \rightarrow .S$

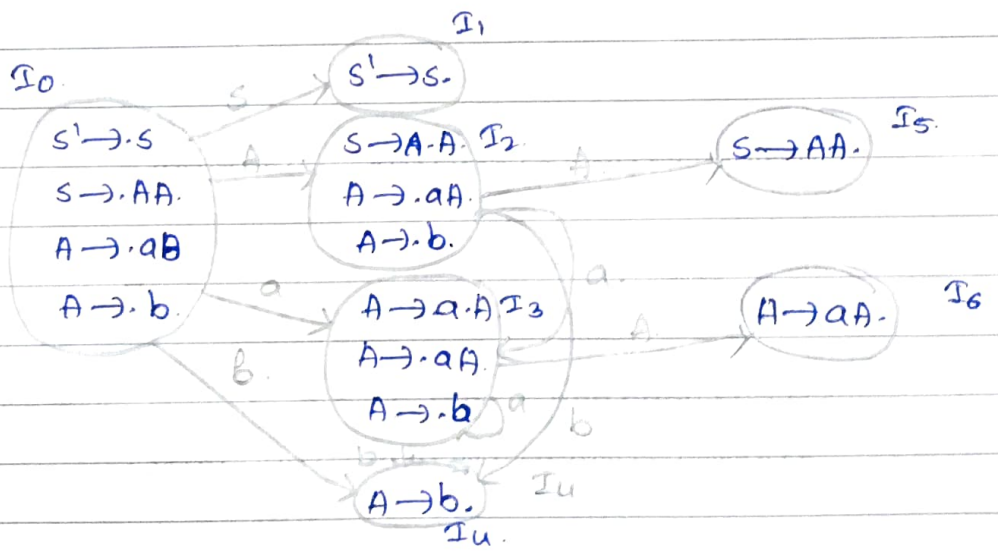
$S \rightarrow .AA \quad -r_1$

$A \rightarrow .aA \quad -r_2.$

$A \rightarrow .b \quad -r_3.$

Step-03:-

Draw DFA (Data Flow Diagram)



Step-4:- Table.

| | Actions | | | Goto. | |
|---|---|---|---|---|---|
| | a | b | $ | A | S |
| $I_0$ | $S_3$ | $S_4$ | | 2 | 1. |
| $I_1$ | | Accept | | | |
| $I_2$ | $S_3$ | $S_4$ | | 5. | |
| $I_3$ | $S_3$ | $S_4$ | | 6. | |
| $I_4$ | $r_3$ | $r_3$ | $r_3$. | | |
| $I_5$ | $r_1$ | $r_1$ | $r_1$ | | |
| $I_6$ | $r_2$ | $r_2$ | $r_2$. | | |

step - 5 :-

check for input string

| stack | Input | Output |
|---|---|---|
| $ | aabb$ | |
| $0 | aabb$ | shift 3 ($S_3$) |
| $0a3 | abb$ | shift 3 ($S_3$) |

| | | |
|---|---|---|
| $0a3a3 | bb$ | Sheft 4 (s4) |
| $0a3a3b4 | b$ | Reduce 3 ($r_3$) (A→b) |
| $0a3a3A | b$ | 6 |
| $0a3a3A6 | b$ | Reduce 2 ($r_2$) (A→aA) |
| $0a3A | b$ | 6 |
| $0a3A6 | b$ | Reduce 2 ($r_2$) (A→aA) |
| $0A | b$ | 2 |
| $0A2 | b$ | Sheft 4 (s4) |
| $0A2b4 | $ | Reduce 3 ($r_3$) (A→b) |
| $0A2A | $ | 5 |
| $0A2A5 | $ | Reduce ($r_1$) (S→AA) |
| $0S | $ | 1 |
| $0S1 | $ | Accept |

∴ The given string is acceptable