Construction of DFA...
Construction of DFA

Type-01: For strings ending with a particular substring

Steps Involved

Step-01: Decide the minimum number of states required in the DFA and draw them.

Rule: All strings ending with 'n' length substring will always require minimum \((n+1)\) states in its DFA.

Example: Consider the regular expression-

\[(a+b)^*z\]

over \(\Sigma = \{a, b\}\)

The strings of above language will always end on 'z'.

So, if \(|z| = n\), then the DFA requires minimum \((n+1)\) states.
Step-02: Decide the strings for which you will construct the DFA.

Step-03: Construct the DFA for the above decided strings.

Remember:
Always prefer to go with the existing path. Create a new path only when you can't go with.

Step-04: After drawing the DFA for the above decided strings, send the left possible combinations to the starting state not over the dead configuration.
Problem-01: Draw the DFA for the language accepting strings ending with '01' over input alphabet $\Sigma = \{0,1\}$

Solution:

Regular expression for the given language is-

$$(0+1)^*01$$

Minimum number of states in the DFA = 3

Required DFA is-

Strings we will check -

- 01
- 001
- 0101
Problem-02: Draw the DFA for the language accepting strings ending with 'abb' over input alphabet $\Sigma=\{a, b\}$

Solution: Regular expression for the given language is-

$$(a+b)^*abb$$

Minimum number of states in the DFA = 4

Required DFA is-

Strings, we will check-
- abbb
- aabbb
- ababbb
- abbabb
**Problem-03:** Draw the DFA for the language accepting strings ending with 'abba' over $\Sigma = \{a, b\}$.

**Solution:** Regular expression for the given language is -

$$(a+b)^*abba$$

Minimum number of states in the DFA = 5

The required DFA is -

Strings we will check -
- abba
- aabba
- ababba
- ababba
- abbaabba
Problem-04: Draw the DFA for language accepting strings ending with `0011` over $\Sigma = \{0,1\}$

Solution: Regular expression for the given language is-

$$(0+1)^*0011$$

Minimum number of states in the DFA = 5

Required DFA is-

**strings we will check** -
- 0011
- 00011
- 000011
- 0010011
- 00110011