Documentation

Regular Expression:

RE = (ba)* b* a [a + ba(a+b)]*

Alphabet:

Σ = {a, b}

1. Explanation of the Regular Expression

Part 1: (ba)*

Means zero or more repetitions of the string ba.

Examples: ε, ba, baba, bababa.

Part 2: b*

Means zero or more b.

Examples: ε, b, bb, bbb.

Part 3: a

Mandatory single a.

Every valid string must contain this a.

Part 4: [a + ba(a+b)]*

Allows zero or more occurrences of:

a, baa, bab.

Overall Meaning:

A valid string starts with zero or more ba, may have extra b,

must contain at least one a, and can end with combinations of a, baa, or bab.

2. Thompson's Construction (ENFA)

Thompson's construction is used to build the ENFA.

Symbols are converted to basic NFAs.

Operators:

* for closure using epsilon loops,

+ for union using epsilon branching,

concatenation using epsilon transitions.

3. ENFA to DFA Conversion

Subset construction algorithm is used.

Each epsilon-closure becomes a DFA state.

States containing final ENFA state are accepting.

4. DFA Description

Start State: q0

Accepting State: q3

States:

q0 – Start, processing (ba)*

q1 – Seen b, expecting a

q2 – Processing b*

q3 – After mandatory a (Accepting)

q4 – After b in final part

q5 – After ba in final part

DEAD – Invalid transition

5. DFA Transition Table

q0: a→DEAD, b→q1

q1: a→q0, b→DEAD

q2: a→q3, b→q2

q3: a→q3, b→q4

q4: a→q5, b→DEAD

q5: a→q3, b→q3

6. DFA Implementation

The DFA is implemented in C++ using enumerated states and a transition function.

The input string is processed character by character.

The string is accepted if the final state is an accepting state.

7. Sample Test Cases

a → Accepted

bba → Accepted

bababaa → Accepted

bb → Rejected

ba → Rejected

8. Conclusion

The regular expression was analyzed.

Thompson’s ENFA was constructed and converted to DFA.

The DFA was implemented in C++ and correctly validates strings.