AUTOMATA AND COMPILER DESIGN

IMPLEMENTATION OF MACHINE

TOPIC : LEARNING AND DECISION – MAKING

Faculty in-charge : Dr. Vipasha Sharma

Section : H

TEAM MEMBERS:

Anshikka Mittal– 23FE10CSE00248

Anisha Sharma - 23FE10CSE00401

Learning and Decision Making using Finite Automata

Aim

To design and implement a Finite Automaton capable of simulating the process of learning a specific input pattern and making a deterministic acceptance or rejection decision based on the conformity of subsequent inputs to the learned pattern.

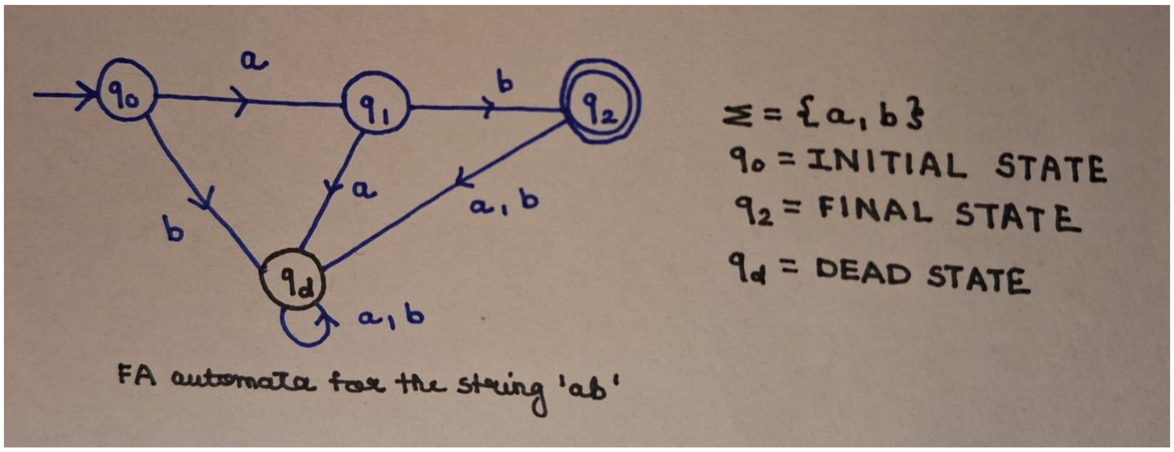
Introduction

Finite Automata (FA) are one of the most fundamental models in the theory of computation. They are widely used to recognize patterns and make decisions based on sequences of input symbols. While traditional finite automata are not designed to "learn" in the modern machine learning sense, they can be programmed to simulate a learning and decision-making behavior through state transitions and pattern recognition.

In the context of finite automata, learning refers to the process of constructing an automaton that can identify a specific sequence or pattern within an input. Once a pattern is identified or "learned," the automaton moves through a series of defined states when reading inputs that match this pattern.

Decision making in finite automata is realized through acceptance or rejection of input strings. After processing the entire input, the automaton checks whether it has ended in an accepting state. If yes, the automaton accepts the string — indicating the pattern was successfully matched. If not, the automaton rejects the string — indicating the pattern did not match.

Implementation Diagram of Finite Automata



Objective

This example simulates a Finite Automaton that "learns" a pattern (say "ab") and then decides whether an input string matches this pattern. The automaton accepts the string "ab" only. Any other input is rejected.

Python Code

```
def finite_automaton(input_string):
    state = 'q0'  # Start state

    for char in input_string:
        if state == 'q0':
            if char == 'a':
                state = 'q1'
            else:
                state = 'reject'
        elif state == 'q1':
            if char == 'b':
                state = 'q2'
            else:
                state = 'reject'
        else:
            state = 'reject'

    if state == 'q2':
        return 'Accepted: Pattern matched '
    else:
        return 'Rejected: Pattern not matched '
```

```
# Test examples
print(finite_automaton('ab'))      # Accepted
print(finite_automaton('a'))       # Rejected
print(finite_automaton('abc'))     # Rejected
print(finite_automaton('b'))       # Rejected
```

Explanation

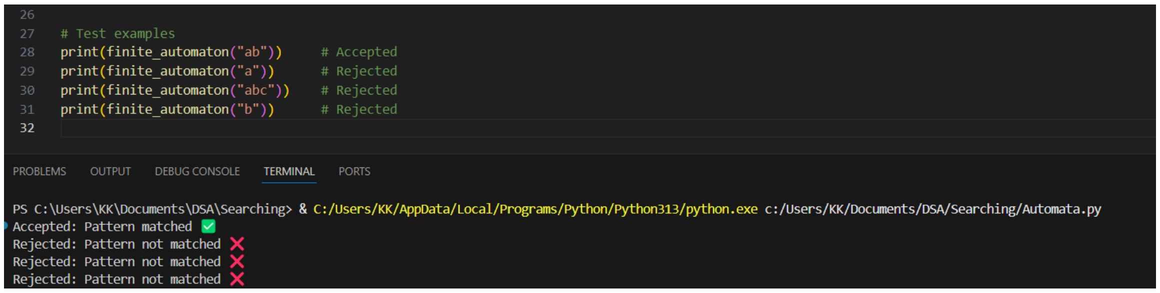
q0: Start state, expects 'a'

q1: After reading 'a', expects 'b'

q2: Accepting state (after reading 'ab')

Any deviation sends the automaton to a reject state.

Output



```
# Test examples
print(finite_automaton("ab"))      # Accepted
print(finite_automaton("a"))       # Rejected
print(finite_automaton("abc"))     # Rejected
print(finite_automaton("b"))       # Rejected
```

```
PS C:\Users\KK\Documents\DSA\Searching> & C:/Users/KK/AppData/Local/Programs/Python/Python313/python.exe c:/Users/KK/Documents/DSA/Searching/Automata.py
Accepted: Pattern matched ✅
Rejected: Pattern not matched ❌
Rejected: Pattern not matched ❌
Rejected: Pattern not matched ❌
```

```
Input: ab       -> Accepted: Pattern matched ☑
Input: a        -> Rejected: Pattern not matched ✗
Input: abc      -> Rejected: Pattern not matched ✗
Input: b        -> Rejected: Pattern not matched ✗
```