Finite Automaton for Recognizing Strings Containing 'ab'

Jayashre
3rd Year B.Tech CSE - Cybersecurity - 22011103020
CS5694: Advanced Compiler Design

February 23, 2025

1 Introduction

This document presents a finite automaton (FA) that recognizes strings over the alphabet $\Sigma = \{a, b\}$ where the substring `"ab"` appears at least once. The automaton is implemented in C and follows a deterministic finite automaton (DFA) approach.

2 Regular Language Description

The language L consists of all strings that contain the substring "ab" at least once. Formally, we define it as:

$$L = \{w \in \{a, b\}^* \mid \text{"ab" appears in } w\}$$

Example Strings:

- Accepted: `"ab"`, `"aab"`, `"abb"`, `"bab"`, `"babab"`
- Rejected: `"a"`, `"b"`, `"aaa"`, `"bbb"`

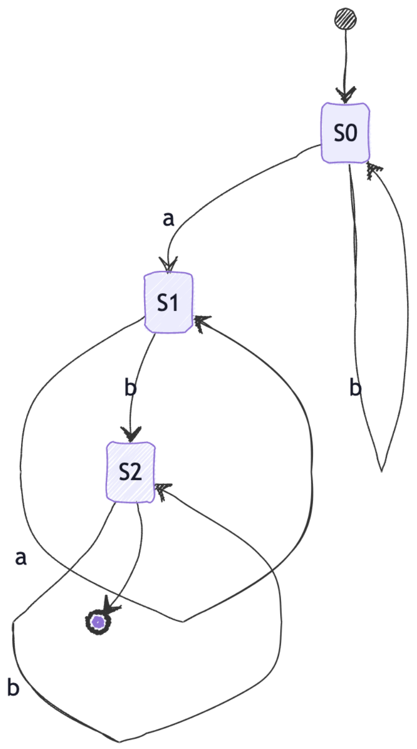
3 Finite Automaton Diagram

The DFA consists of three states:

- S_0 (Start State) - Moves to S_1 if 'a' is encountered.

- S_1 - Moves to S_2 upon receiving ’b’.
- S_2 (Accepting State) - Remains in S_2 upon receiving ’a’ or ’b’.

Below is the state transition diagram for our DFA:



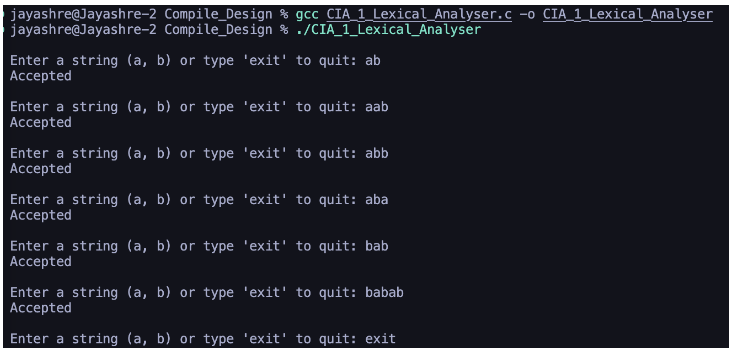
4 Screenshots of Program Execution

Below are screenshots demonstrating both acceptance and rejection of inputs.

5 C Code Implementation

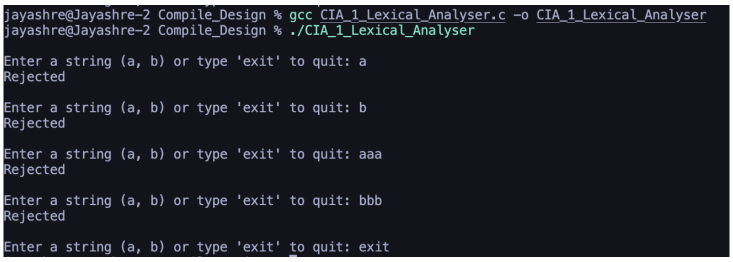
The following C program implements the DFA:

```c
#include <stdio.h>
#include <string.h>

int simulateDFA(const char *input) {
    int state = 0;
    int length = strlen(input);
```



```
jayashre@Jayashre-2 Compile_Design % gcc CIA_1_Lexical_Analyser.c -o CIA_1_Lexical_Analyser
jayashre@Jayashre-2 Compile_Design % ./CIA_1_Lexical_Analyser

Enter a string (a, b) or type 'exit' to quit: ab
Accepted

Enter a string (a, b) or type 'exit' to quit: aab
Accepted

Enter a string (a, b) or type 'exit' to quit: abb
Accepted

Enter a string (a, b) or type 'exit' to quit: aba
Accepted

Enter a string (a, b) or type 'exit' to quit: bab
Accepted

Enter a string (a, b) or type 'exit' to quit: babab
Accepted

Enter a string (a, b) or type 'exit' to quit: exit
```

Figure 1: Accepted input example (”ab”)



```
jayashre@Jayashre-2 Compile_Design % gcc CIA_1_Lexical_Analyser.c -o CIA_1_Lexical_Analyser
jayashre@Jayashre-2 Compile_Design % ./CIA_1_Lexical_Analyser

Enter a string (a, b) or type 'exit' to quit: a
Rejected

Enter a string (a, b) or type 'exit' to quit: b
Rejected

Enter a string (a, b) or type 'exit' to quit: aaa
Rejected

Enter a string (a, b) or type 'exit' to quit: bbb
Rejected

Enter a string (a, b) or type 'exit' to quit: exit
```

Figure 2: Rejected input example (”aaa”)

```
for (int i = 0; i < length; i++) {
    char c = input[i];

    switch (state) {
        case 0:
            if (c == 'a') state = 1;
            else if (c == 'b') state = 0;
            else return 0; // Reject on invalid
               character
            break;

        case 1:
            if (c == 'a') state = 1;
            else if (c == 'b') state = 2;
            else return 0;
            break;

        case 2:
            if (c == 'a' || c == 'b') state = 2;
```

```
                else return 0;
                break;

            default:
                return 0;
        }
    }

    return (state == 2);
}

int main() {
    char input[100];

    while (1) {
        printf("\nEnter␣a␣string␣(a,␣b)␣or␣type␣'exit'␣
           to␣quit:␣");
        scanf("%s", input);

        if (strcmp(input, "exit") == 0) break;

        if (simulateDFA(input)) {
            printf("Accepted\n");
        } else {
            printf("Rejected\n");
        }
    }

    return 0;
}
```

6 Conclusion

This document provides a detailed description of a DFA that recognizes strings containing the substring ”ab”. The implementation in C demonstrates correct behavior, as verified through testing. The screenshots confirm the working of the automaton.