College Of Engineering Trivandrum

Compiler Design Lab

CS431

Submitted By:
Neethu S
S6 CSE Roll No:42

TVE18CS043

Department of Computer Science

October 8, 2021

Contents

1 ϵ - NFA to NFA conversion

1.1 Aim

Write a program to convert NFA with ϵ transition to NFA without ϵ transition.

1.2 Theory

Non-deterministic Finite Automata (NFA) is a finite automata having zero, one or more than one moves from a given state on a given input symbol. ϵ-NFA is the NFA which contains epsilon move(s)/Null move(s).

An ϵ-NFA is represented formally by a 5-tuple, (Q, Σ, Δ, q0, F), consisting of

- a finite set of states Q
- a finite set of input symbols Σ
- a transition function Δ : Q $\times$ (Σ U ϵ)$\rightarrow$ P(Q)
- an initial (or start) state q0 ϵ Q
- a set of states F distinguished as accepting (or final) states F $\subseteq$ Q

Here, P(Q) denotes power set of Q.

1.3 Algorithm

```
Step 1 - Find out all the epsilon-transitions from each state from Q. That will
be called as epsilon-closure(qi) where, qi belongs to Q.

Step 2 - Then, q1 transitions can be obtained. The q1 transitions means an
epsilon-closure on q moves.

Step 3 - Step 2 is repeated for each input symbol and for each state of given NFA.

Step 4 - By using the resultant status, the transition table for equivalent NFA
without epsilon can be built.
```

1.4 Code

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int st;
    struct node *link;
};

void findclosure(int x, int sta)
{
```

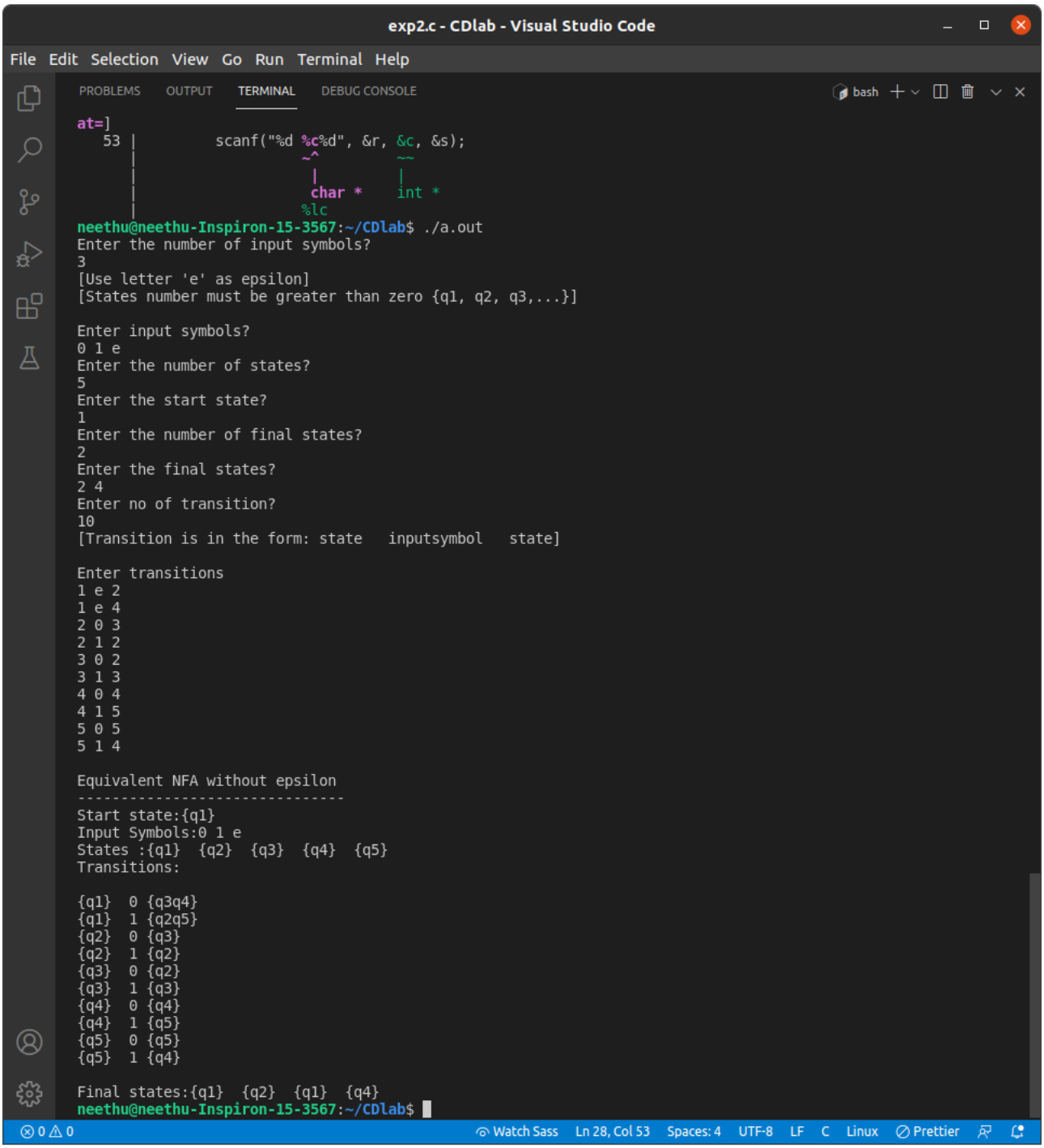
```
    struct node *temp;
    int i;
    if (buffer[x])
        return;
    e_closure[sta][c++] = x;
    buffer[x] = 1;
    if (alphabet[noalpha - 1] == 'e' && transition[x][noalpha - 1] != NULL)
    {
        temp = transition[x][noalpha - 1];
        while (temp != NULL)
        {
            findclosure(temp->st, sta);
            temp = temp->link;
        }
    }
}

void insert_trantbl(int r, char c, int s)
{
    int j;
    struct node *temp;
    j = findalpha(c);
    if (j == 999)
    {
        printf("error\n");
        exit(0);
    }
    temp = (struct node *)malloc(sizeof(struct node));
    temp->st = s;
    temp->link = transition[r][j];
    transition[r][j] = temp;
}

int findalpha(char c)
{
    int i;
    for (i = 0; i < noalpha; i++)
        if (alphabet[i] == c)
            return i;

    return (999);
}

void unionclosure(int i)
{
    int j = 0, k;
    while (e_closure[i][j] != 0)
    {
        k = e_closure[i][j];
        set[k] = 1;
        j++;
    }
```

```
}

void print_e_closure(int i)
{
    int j = 0;
    printf("{");
    if (e_closure[i][j] != 0)
        printf("q%d", e_closure[i][0]);
    printf("}  ");
}

void findfinalstate()
{
    int i, j, k, t;
    for (i = 0; i < nofinal; i++)
    {
        for (j = 1; j <= nostate; j++)
        {
            for (k = 0; e_closure[j][k] != 0; k++)
            {
                if (e_closure[j][k] == finalstate[i])
                {
                    print_e_closure(j);
                }
            }
        }
    }
}

static int set[20], nostate, noalpha, s, notransition, nofinal,
        start, finalstate[20], c, r, buffer[20];
char alphabet[20];
static int e_closure[20][20] = {0};
struct node *transition[20][20] = {NULL};
void main()
{
    int i, j, k, m, t, n;

    struct node *temp;
    printf("Enter the number of input symbols?\n");
    scanf("%d", &noalpha);
    getchar();
    printf("[Use letter 'e' as epsilon]\n");
    printf("[States number must be greater than zero {q1, q2, q3,...}]\n");

    printf("\nEnter input symbols?\n");
    for (i = 0; i < noalpha; i++)
    {
        alphabet[i] = getchar();
        getchar();
    }
    printf("Enter the number of states?\n");
```

```
scanf("%d", &nostate);
printf("Enter the start state?\n");
scanf("%d", &start);
printf("Enter the number of final states?\n");
scanf("%d", &nofinal);
printf("Enter the final states?\n");
for (i = 0; i < nofinal; i++)
    scanf("%d", &finalstate[i]);
printf("Enter no of transition?\n");
scanf("%d", &notransition);
printf("[Transition is in the form: state inputsymbol state]\n", notransition);

printf("\nEnter transitions\n");
for (i = 0; i < notransition; i++)
{

    scanf("%d %c%d", &r, &c, &s);
    insert_trantbl(r, c, s);
}

printf("\n");

for (i = 1; i <= nostate; i++)
{
    c = 0;
    for (j = 0; j < 20; j++)

    {
        buffer[j] = 0;
        e_closure[i][j] = 0;
    }
    findclosure(i, i);
}
printf("Equivalent NFA without epsilon\n");
printf("-------------------------------\n");
printf("Start state:");
print_e_closure(start);
printf("\nInput Symbols:");
for (i = 0; i < noalpha; i++)
    printf("%c ", alphabet[i]);
printf("\nStates :");
for (i = 1; i <= nostate; i++)
    print_e_closure(i);

printf("\nTransitions:\n");

for (i = 1; i <= nostate; i++)
{

    for (j = 0; j < noalpha - 1; j++)
    {
        for (m = 1; m <= nostate; m++)
```

```
                set[m] = 0;
            for (k = 0; e_closure[i][k] != 0; k++)
            {

                t = e_closure[i][k];
                temp = transition[t][j];
                while (temp != NULL)
                {

                    unionclosure(temp->st);
                    temp = temp->link;
                }
            }
            printf("\n");
            print_e_closure(i);
            printf("%c\t", alphabet[j]);
            printf("{");
            for (n = 1; n <= nostate; n++)
            {
                if (set[n] != 0)
                    printf("q%d", n);
            }
            printf("}");
        }
    }
    printf("\n\nFinal states:");
    findfinalstate();
    printf("\n");
}
```

1.5 Output



1.6 Result

Implemented the program to convert NFA with epsilon-transition to NFA without epsilon-transitions using C language in Ubuntu 20.04 and the above outputs were obtained.