Ministry of Education, Culture and Research of Moldova

Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Software Engineering and Automation Department

Laborotory Work

Formal Languages and Compiler Design

Student: Wu Xenia

Group: FAF-203

Professor: Cojuhari Irina

Vdovicenco Alexandr

Chisinau, 2022

Variant 24

Laboratory tasks:

1. Present the automaton in form of graph. Is this automaton deterministic or no? Why?

2. Convert a Finite Automaton to the Regular Grammar.

3. Transform nondeterministic finite automaton (NFA) into a deterministic automaton (DFA). Present the DFA in form of graph.

4. Present 1 string based on the input alphabet $\sum$ that will be not accepted by the automaton. For showing use the Multiple Run option from menu Input from JFLAP.

5. Convert NFA from your variant to DFA on paper, writing all transitions and drawing converted automata.

6. Write program which converts nondeterministic finite automata (NFA) to deterministic finite automata (DFA)

7. Display converted automata in form of graph or transition table

8.* For the finite automaton $AF=(Q, \sum, \delta, q0, F)$ present five strings that will be accepted. The length of strings should be more then n+2, where n is the number of stares from Q. For showing use the Multiple Run option from menu Input from JFLAP. Elaborate the program that for the obtained DFA will verify if string is accepted or no.

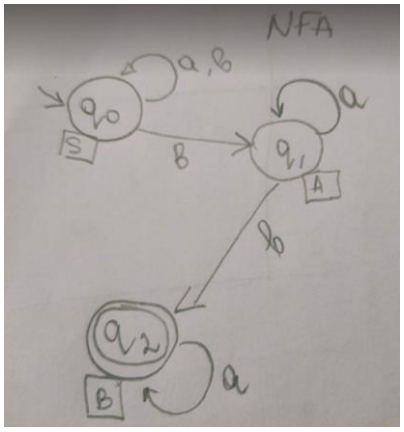
9.* For each string x, present the configuration sequence for the acceptance of this string. To show the acceptance use the option Step with Closure and Fast Run from menu Input from JFLAP.

10.* For each obtained 5 strings build the decomposition x=uvw, using the pumping lemma.

11.*Obtained DFA minimize

**The tasks colored in gray, are optional*

Ex. 1



Ex. 2

$V_N = \{S, A, B\}, V_T = \{a, b\}$

$P = \{S \rightarrow aS|bS|bA$

$A \rightarrow aA|bB$

$B \rightarrow aB|\varepsilon\}$

Ex. 3

NFA table:

State	a	b
q0	[q0]	[q0,q1]
q1	[q1]	[q2]
q2	[q2]	[]

DFA table:

State	a	b
q0	{q0}	{q0q1}
q0q1	{q0q1}	{q0q1q2}
q0q1q2	{q0q1q2}	{q0q1q2}

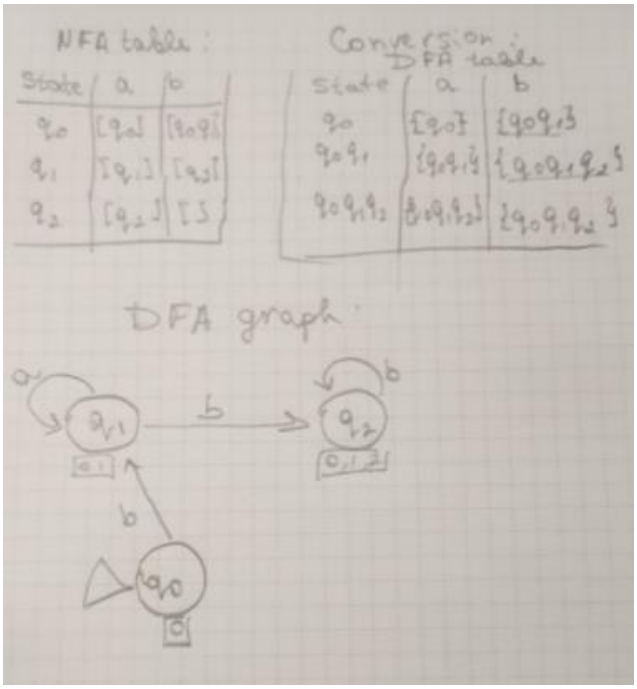
Final state: {q0q1q2}

Ex. 4

Some strings that will not be accepted by this FA:

aba, aaba

Ex. 5



Ex. 6

```
import pandas as pd

NFA = {}
states_brut = input("Define the states (separe them by space)\n")
states = states_brut.split(" ")
print("Q = {}".format(states))
nr_states = len(states)
paths_brut = input("Define the paths (separe them by space)\n")
paths = paths_brut.split(" ")
print("S = {}".format(paths))
nr_paths = len(paths)
finals_brut = input("Define the final state/s (separe them by space)\n")
finals = finals_brut.split(" ")
print("F = {}".format(finals))
nr_finals = len(finals)

print("For each state, complete the NFA table")
for i in range(nr_states):
    state = input("state name : ")
    NFA[state] = {}
    for j in range(nr_paths):
        path = input("path : ")
        print("Enter end state from state {} travelling through path {} : ".format(state, path))
        reaching_state = [x for x in input().split()]
        NFA[state][path] = reaching_state

print("\nNFA :- \n")
print(NFA)
```

```
print("\nPrinting NFA table :- ")
nfa_table = pd.DataFrame(NFA)
print(nfa_table.transpose())

newbies_list = []
DFA = {}
keys_list = list(NFA.keys())
# Computing first row of DFA transition table
DFA[keys_list[0]] = {}
for y in range(nr_paths):
    var = "".join(NFA[keys_list[0]][
                    paths[y]])
    DFA[keys_list[0]][paths[y]] = var
    if var not in keys_list:
        newbies_list.append(var)
        keys_list.append(var)
# Computing the other rows of DFA transition table
while len(newbies_list) != 0:
    DFA[newbies_list[0]] = {}
    for _ in range(len(newbies_list[0])):
        for i in range(len(paths)):
            aux = []
            for j in range(len(newbies_list[0])):
                aux += NFA[newbies_list[0][j]][paths[i]]
            s = ""
            s = s.join(aux)
            if s not in keys_list:
                newbies_list.append(s)
                keys_list.append(s)
            DFA[newbies_list[0]][paths[i]] = s

    newbies_list.remove(newbies_list[0])

print("\nDFA :- \n")
print(DFA)
print("\nPrinting DFA table :- ")
Table_DFA = pd.DataFrame(DFA)
print(Table_DFA.transpose())

dfa_states = list(DFA.keys())
dfa_terminals = []
for x in dfa_states:
    for i in x:
        if i in finals:
            dfa_terminals.append(x)
            break

print("\nFinal states of the DFA are : ", dfa_terminals)
```

Ex. 7

```
Printing DFA table :-
        a    b
A       A   AB
AB     AB  ABC
ABC   ABC  ABC

Final states of the DFA are :  ['ABC']
```