MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII MOLDOVA
Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică
Departamentul Inginerie Software și Automatică

Zagorodniuc Anastasia FAF-223

Report

Intro to formal languages. Regular grammar. Finite Automata.

of Formal Languages & Finite Automata

Checked by:

Cretu Dumitru, *university assistant*

Chișinău – 2023

Objectives

1. Understand what an automaton is and what it can be used for.
2. Continuing the work in the same repository and the same project, the following need to be added:
    a. Provide a function in your grammar type/class that could classify the grammar based on Chomsky hierarchy.
    b. For this you can use the variant from the previous lab.
3. According to your variant number (by universal convention it is register ID), get the finite automaton definition and do the following tasks:
    a. Implement conversion of a finite automaton to a regular grammar.
    b. Determine whether your FA is deterministic or non-deterministic.
    c. Implement some functionality that would convert an NDFA to a DFA.
    d. Represent the finite automaton graphically (Optional, and can be considered as a ***bonus point***):

Code

Point 1: implemented

My program classifies what grammar is my variant based on Chomsky hierarchy. First, it checks what typeof grammar it is using 4 validation. After that, the program returns the name of the grammar.

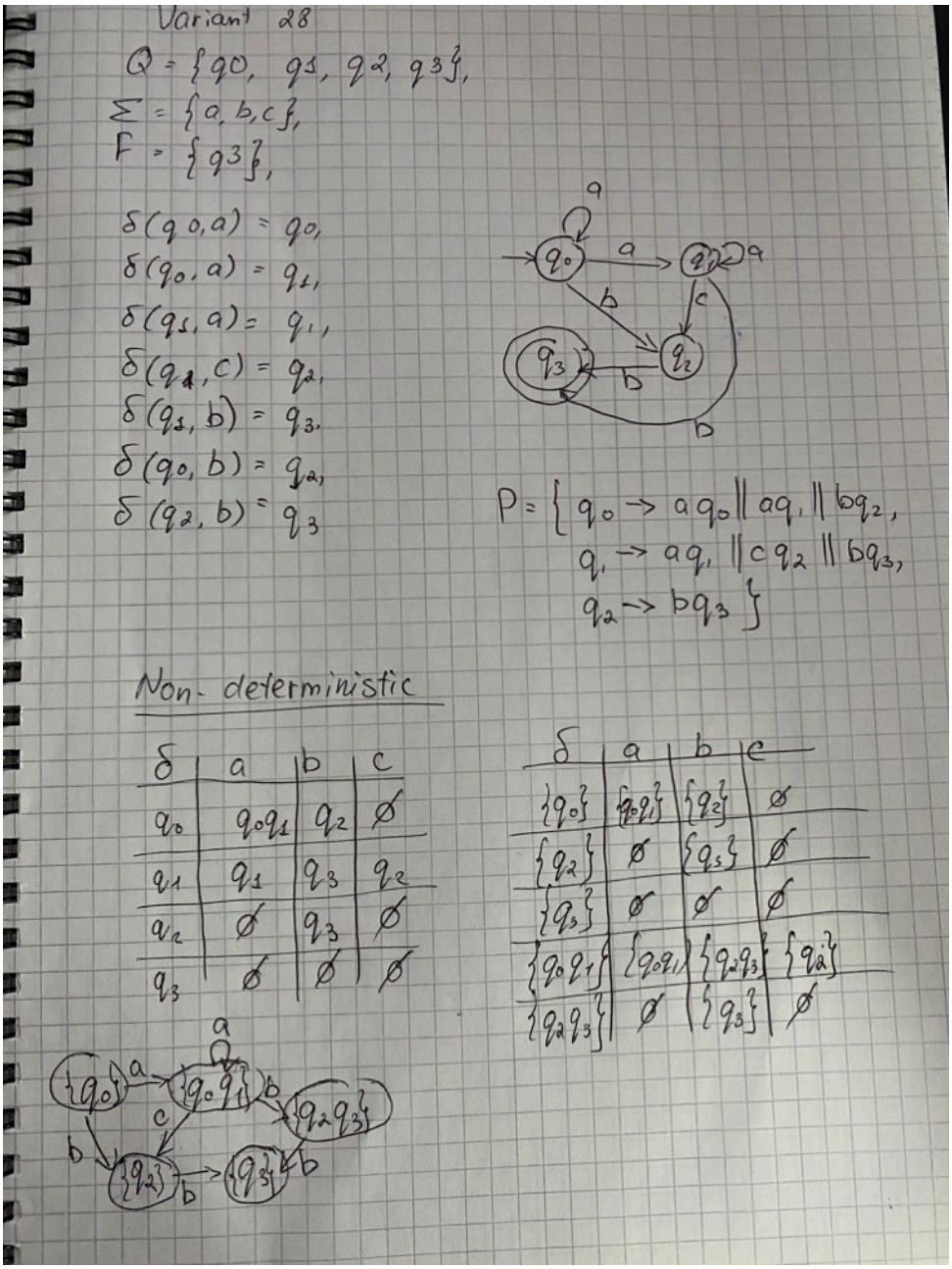
```
private bool IsType0()
{
    // All grammars are Type 0
    return true;
}

private bool IsType1()
{
    // Check if all production rules are of the form α -> β where |α| <= |β|
    foreach (ProductionRule rule in P)
    {
        if (rule.LeftSide.ToString().Length > rule.RightSide.Length)
        {
            return false;
        }
    }
    return true;
}

private bool IsType2()
{
    // Check if all production rules are of the form A -> γ where A is a non-terminal and γ is a string of terminals or non-terminals
    foreach (ProductionRule rule in P)
    {
        if (rule.RightSide.Length == 0 || !VN.Contains(rule.LeftSide) || !IsStringOfTerminalsOrNonTerminals(rule.RightSide))
        {
            return false;
        }
    }
    return true;
}

private bool IsType3()
{
    // Check if all production rules are of the form A -> aB or A -> a where A and B are non-terminals and a is a terminal
    foreach (ProductionRule rule in P)
    {
        if (rule.RightSide.Length == 0 || !VN.Contains(rule.LeftSide) || (rule.RightSide.Length == 1 && !VT.Contains(rule.RightSide[0])))
        {
            return false;
        }
    }
    return true;
}
```

```
public string Classify()
{
    if (IsType3())
    {
        return "Type 3 (Regular)";
    }
    else if (IsType2())
    {
        return "Type 2 (Context-Free)";
    }
    else if (IsType1())
    {
        return "Type 1 (Context-Sensitive)";
    }
    else if (IsType0())
    {
        return "Type 0 (Unrestricted)";
    }
    else
    {
        return "Invalid grammar";
    }
}
```

Output:

```
Enter lab:
lab2
Grammar Classification: Type 3 (Regular)
```

Point 3: implemented



For this point I implemented the visual part using graphs and tables. My FA is non-deterministic because it returns to itself using the same variable. I converted NDFA to DFA using tables and common variables for each q. The last graph shows a DFA

Conclusion:

This laboratory work helped me to understand what Chomsky hierarchy is and how can I implement the solution via coding. I also understood what NDFA and DFA are and learned how to transform NDFA to DFA.