|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | |
| Automata & Logic Software Engineering 1 | | | | |
| 17-10-2020 | | | | |
| Jane Buzykina 392854 | |  |  |  |

Table of Contents

[1](#_Toc53517655)

[Introduction 3](#_Toc53517656)

[Assignment 1: Parse + Tree 3](#_Toc53517657)

[Assignment 2: Truth Table 3](#_Toc53517658)

[Assignment 3: Simplification 3](#_Toc53517659)

[Assignment 4: Normalization 3](#_Toc53517660)

[Assignment 5: Nandify 3](#_Toc53517661)

[Software Design 3](#_Toc53517662)

[GUI 3](#_Toc53517663)

[Testing 3](#_Toc53517664)

[Conclusions and Improvements 3](#_Toc53517665)

# Introduction

This report will guide a reader through the process of completing the assignments for the ALE1 course in Fontys. It consists of 9 sections. The first five explain in detail the chosen approach as well as encountered challenges during the implementation of each of the assignments. Next three parts five an overview of the whole program, namely its Software Design, GUI, and Testing. The report finishes with the conclusions and improvements part.

# Assignment 1: Parse + Tree

The goal of this phase was to make a foundation for the application, which should include main functions such as parsing the input and producing a binary tree based on it. To start with development, I first needed to decide on technologies for this project. Initially, I wanted to write a web app creating interface using React or Vue.js and backend in C#. However, due to time constraints, I decided to go with C# Win Forms, a UI framework with which I had the most experience. As a result, I could spend more time thinking about actual logic.

Having all the formal parts settled, I began to think about where to start. The first step was creating a Binary Tree data structure. The idea was the following: Binary Tree has information about its root; the root has information about its left and right child; right and left children have information about their children and so on... All the nodes were divided into types like leaf node, disjunction node, conjunction node and represented by its type class inherited from the Node abstract class. Why abstract and not interface? Simply because all its children had some implementations in common.

Now that the Binary Tree structure was there, I had to come with a way to initialize it by recursively parsing input from a user. While testing my middle versions of the parsing algorithm, I had an application breaking too many times due to wrongly formatted input. This inconvenience became a motivation to add a step of validation before parsing. During my automata course, I got familiar with regex expressions used to validate the patterns. The only problem with those was that in theory regex cannot keep track of the number of repetitions of some symbol, which was a big drawback because then parenthesis could not be counted. However, after some research online, I found an article about balancing groups that can be used in C# regex expressions. A balancing group is just like a counter, thus the number of opening and closing parentheses can be compared now. This finding helped me to come with the working validation regex. I verified its correctness by testing on few different expressions and moved on to the parsing.

Parsing was quite straightforward to me. The idea was to read input char by char, first building the left branch of the tree when it was done build the right one. Once there were no more nodes to process, the binary tree would be done. This was achieved using recursion, which acted just as a pre-order tree walk. After a lot of successful testing, this assignment was completed.

# Assignment 2: Truth Table

Big O notation running time. Reaching pointers, which I got surprised about.

# Assignment 3: Simplification

Hey hey my fun assignment.

http://www.32x8.com/index.html

# Assignment 4: Normalization

Not sure what to say here

# Assignment 5: Nandify

# Software Design

# GUI

# Testing

# Conclusions and Improvements

This course is in my personal top of the Fontys courses. It was as challenging as enjoyable. During the course, I had a chance not only to learn more, but also to apply the knowledge I got previously from my premaster courses. Implementing it, gave me a deeper understanding of the learned theory. The course upgraded my logical thinking skills and will be a nice addition to my CV.

Overall, I am proud of the created solution. However, the existing solution still can be improved. The running time should be optimized in order to process more complex expressions. Additionally, I would modify existing GUI. As for me, it looks too old-fashioned.