UNIVERSITY OF COPENHAGEN

Bachelor's Project

Christian Bendix Fjordstrøm

Conversion and Combination of Different Representations of Regular Langauges

Date: May 14, 2024

Advisor: Andrzej Filinski

Table of content

1 Introduction

Lexical tokens are typically defined using regular expressions which are then converted to finite automata by a lexer generator. Since lexical analysis is a core step in compiling or interpreting a program, students of computer science learn how these conversions are done.

The purpose of this project is to design and implementing a tool in F# for conversion and combination of regular languages in the forms of NFA's, DFA's, regular expressions and regular grammars and combining them by applying operations such as union, intersection, complement or concatenation.

The project thus seeks to help students and teachers by providing a tool that can answer what two regular languages have in common, if they are equivalent, or simply convert between representations.

2 Analysis

As mentioned in the introduction, this project revolves around designing and implementing a tool for conversion and combination of different representations of regular languages. The requirements for the project can thus be broken down into the following parts:

2.1 Input/Output

The syntax for how users define automata as well as the syntax for how users express their desired conversions and combinations needs to be designed. The user should also be able to define which output format they want, and the syntax of the output should be the same as the syntax for the input, such that it can be fed back in. The output also exhibit a ”roundtrip property”, meaning that feeding the output back in should result in the same language. Additionally, there should also be an option for a user to check if a string is recognised by the given regular language.

2.2 Conversion

The program is able to convert regular expressions to NFA's, convert NFA's to DFA's, minimise DFA's and converting automata back to regular expressions, so that users can get the resulting language in whichever format they desire.

2.3 Combination

The standard regular expression operators such as concatenation, repeated concatenation and union as well as union and complement should be applicable to both regular expressions and automata. In addition, the syntax for how to express these combinations must be designed and implemented.

3 Design

3.1 Input

3.1.1 Automata

3.1.2 Grammar

3.2 Conversion

3.2.1 Regular expression to NFA

3.2.2 NFA to DFA

3.2.3 Minimisation of DFA’s

3.2.4 Automata to regular expression

3.3 Combination

3.3.1 Standard regular expression operations

3.3.2 Intersection

3.3.3 Complement

4 Implementation

5 Testing

6 User Guide

7 Conclusion