Team members

Nagham Samir 221000697

Minnah Tariq 211001751

Mona Gomaa 221000415

Nada Gamal 221000829

Farida Hazem 221000214

Submitted Respectively To:

Dr. Zeinab Mohamed

Eng. Donia Khaled

Contents

1. Introduction

The Regex to NFA Visualizer is an interactive tool that transforms regular expressions (regex) into their corresponding nondeterministic finite automaton (NFA) using the Thompson Construction algorithm. It not only performs the conversion but also offers a visual representation of the resulting NFA, enabling users to explore its states and transitions. Designed for students, educators, and enthusiasts, the tool serves as a valuable resource for learning and understanding how regular expressions and automata work.

1.1. Objective

- **Conversion:** Transform a user-provided regular expression into its equivalent nondeterministic finite automaton (NFA).

- **Visualization:** Display the resulting NFA graphically to enhance users' understanding of the regex-to-automaton transformation process.

- **Interactive Features**: Enable users to input regular expressions, initiate the conversion, and explore the visual output in an intuitive and user-friendly way.

- **Export:** Offer the ability to save the generated NFA as a PNG image for use in documentation or further analysis.

2. Methodology

2.1. Tools and Technologies

The development of this project involved the use of the following tools and technologies:

- **Python**: Serves as the primary programming language for implementing the core logic behind regex-to-NFA conversion and visualization.

- **Tkinter**: Utilized for building the application's graphical user interface.

- **PIL (Pillow):** A Python imaging library employed for handling image manipulation and supporting the export functionality.

- **Regex-to-NFA Algorithm**: Implements the Thompson Construction algorithm to convert regular expressions into NFAs.

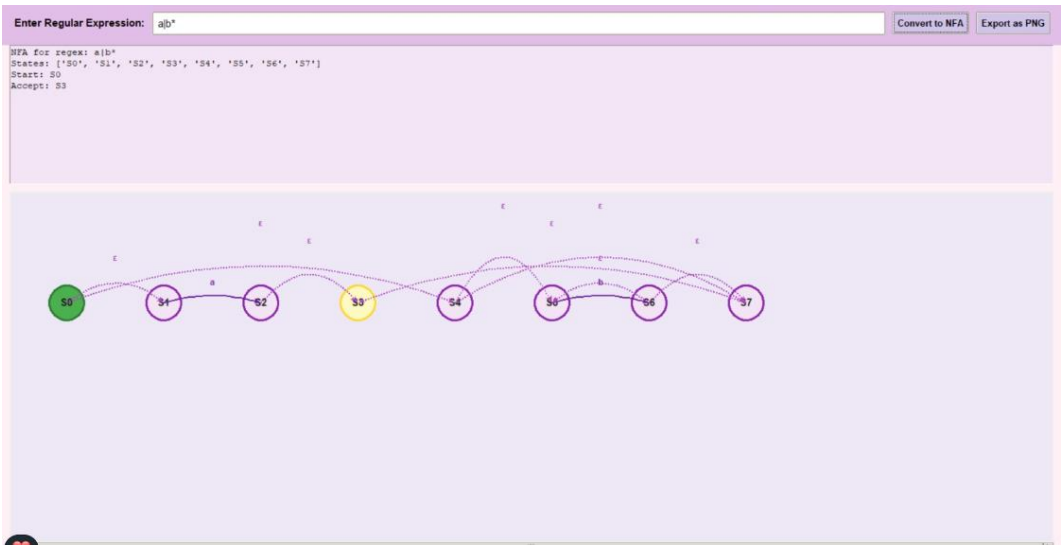
- **Subprocess Module**: Facilitates the execution of external Python scripts and processes within the application.

2.2.Features of the Application

2.2.1. Welcome Screen

When the application is launched, users are welcomed with a *Welcome Screen* designed to create a friendly and engaging first impression. This screen features a prominent "Welcome ✿" message that warmly greets the user. A logo is displayed to enhance the visual appeal of the application; however, if the logo image is unavailable, a fallback text message appears in its place. At the center of the screen is a *"Start Exploring"* button, inviting users to move forward to the main interface where they can enter regular expressions and generate the corresponding NFAs.

2.2.2. Main NFA Generation Screen/ Input - output Format



The *Main NFA Generation Screen* serves as the core interface where users can input a regular expression and visualize its corresponding NFA. This screen is composed of several well-organized components designed to facilitate an interactive and informative user experience:

1. Regex Input Section

- A text entry field allows users to input their regular expressions.
- The system supports a wide range of regex operators, including:
    - Union (|)
    - Concatenation (.)
    - Kleene Star (*)
    - Kleene Plus (+)
    - Optional (?)
    - Character Classes ([])

2. Conversion Button

- A button labeled **"Convert to NFA"** triggers the conversion process.
- The input regex is processed using the **Thompson Construction algorithm**, generating its corresponding NFA.

3. NFA Visualization Area

- The generated NFA is displayed on a canvas with the following graphical elements:
    - **States**: Shown as labeled circles (e.g., S0, S1).
    - **Transitions**: Arrows connecting the states, each labeled with the input symbol.
    - **Start State**: Highlighted in **green** to indicate the entry point of the automaton.
    - **Accept State**: Highlighted in **yellow**, marking the final state where the automaton halts on a successful match.

4. Regex and NFA Details Display

After conversion, additional information is presented to help users understand the structure and behavior of the generated NFA:

- **Input Regex**: Displayed above the canvas for reference.
- **List of States**: All states involved in the NFA are shown with their identifiers.
- **Start State**: Clearly identified to show where the automaton begins.
- **Accept State**: Indicated as the final destination for matching strings.

5. Export Functionality

- An **"Export as PNG"** button allows users to save the NFA visualization as a PNG image.
- This feature is useful for documentation, presentations, or further study.

This structured layout ensures users can easily input regex, view the resulting automaton, understand its components, and preserve their work.

3. Inside Mechanism

The regex_to_nfa function is responsible for converting a regular expression into an equivalent Nondeterministic Finite Automaton (NFA) using **Thompson's Construction algorithm**. The process is broken down into several logical steps:

1. Expanding Character Classes

The function first scans the regex for character classes like [a-z]. These are expanded into their equivalent OR pattern—for example, [a-c] becomes (a|b|c)—to standardize processing.

2. Adding Explicit Concatenation

Since regex doesn't always use an explicit operator for concatenation, a custom . operator is inserted between characters where concatenation is implied. For example, ab(c|d) becomes a.b.(c|d).

3. Infix to Postfix Conversion

To simplify parsing, the regex is converted from infix to postfix notation using the **Shunting Yard Algorithm**. This allows for easier stack-based processing during NFA construction by respecting operator precedence.

4. Building the NFA

The postfix expression is parsed to construct the NFA:

- **Basic Characters**: A single-character NFA is created with a start and an accept state.
- **Concatenation (.)**: Joins two NFAs by linking the accept state of the first to the start of the second.
- **Union (|)**: Creates a new start and accept state with epsilon (ε) transitions to/from the original NFAs.
- **Kleene Star (*)**, **Plus (+)**, **Optional (?)**: Builds loops and conditional paths using ε-transitions for repetition and optionality.

Each operation modifies the NFA structure using stack-based logic and results in a single, final NFA object.

5. Final Output

The function returns an NFA object containing the start and accept states. These states are interconnected with labeled transitions and epsilon links to represent the logic of the original regex.

6. Supporting Classes

- **State**: Represents a node in the automaton, with transition mappings and a list of ε-transitions.
- **NFA**: Holds the entry (start) and exit (accept) states for the built automaton.

4. Conclusion

The **Regex to NFA Visualizer** project provides an interactive and user-friendly tool for visualizing the transformation of regular expressions into nondeterministic finite automatons. By supporting various common regex operators and offering detailed visual feedback, this tool is an excellent resource for anyone studying or working with finite automata and regular expressions. The export functionality further enhances its utility, making it a valuable tool for learning, teaching, and documenting regex-based concepts.