DFA/NFA Tool Development

A JFLAP-inspired Automata Tool

Sameer Khan | Christian Gonzalez

TABLE OF CONTENTS

PROBLEM STATEMENT

- **Objective**: Develop a tool similar to JFLAP for creating and interacting with finite automata.

- **Core Functionalities**:
  - Create deterministic and nondeterministic finite automata (DFA/NFA).
  - Save and load automata configurations in JSON format.
  - Test string membership.

- **Goal**: Provide a graphical, user-friendly interface for automaton visualization and manipulation.

SOLUTION OUTLINE

Key Components:

1. **System Design**: Flexible architecture with seamless front-end and back-end integration.
2. **User Interface (UI)**: Interactive, drag-and-drop design.
3. **Automaton Representation**: Graph-based data structure.
4. **Membership Testing**: Deterministic and nondeterministic algorithms.
5. **File Handling**: JSON-based save/load functionality.

UI DESIGN

- **Interactive Features**:
  - Add and delete states.
  - Define start and accept states.
  - Draw transitions using drag-and-drop.
- **Canvas Display**:
  - States as nodes.
  - Transitions as labeled edges with input symbols.
  - Visual clarity of automaton structure.
- **Challenge**: Overlapping transitions (example in demo).
- **Graph-Based Model**:
  - States as nodes.
  - Transitions as edges labeled with input symbols.
- **State Details**:
  - Unique identifier.
  - Boolean flag for accept state.
  - List of outgoing transitions.
- **Transition Details**:
  - Input symbol.
  - Destination state.

MEMBERSHIP TESTING

- **DFA Testing**:
  - Linear process through states.
  - Unique transition per input.
  - **Time Complexity**: O(n).

- **NFA Testing**:
  - Tracks multiple possible states.
  - Explores all transitions using BFS.
  - **Time Complexity**: O(n * m).

SAVE & OPEN FUNCTIONALITY

- **Format**: JSON-based configuration.
- **Saved Details**:
  - States and their properties.
  - Transitions.
- **Operations**:
  - Save: Converts automaton structure to JSON.
  - Load: Parses JSON to rebuild automaton.
- **Time Complexity**: O(s + t), where s = states, t = transitions.

IMPLEMENTATION FEATURES

- **Automaton Operations**:
  - Add State: O(1).
  - Define Transition: O(1).
  - Membership Testing:
    - DFA: O(n).
    - NFA: O(n * m).
  - Save/Load: O(s + t).

- **Interactive Functions**:
  - Add states and transitions.
  - Test strings for acceptance.

DEMO

FINAL THOUGHTS

- **Key Challenge**: Visual clarity.
  - Overlapping transitions.
- **Proposed Solution**: Layout algorithm for better positioning.
- **Bug Example**:
  - Self-loop display inconsistencies.
- **Achievements**:
  - Fully functional DFA/NFA tool.
  - Efficient algorithms for membership testing.
  - User-friendly, interactive interface.
- **Time Complexity**:
  - DFA operations: Linear.
  - NFA operations: Higher complexity due to nondeterminism.
- **Future Improvements**:
  - Enhanced visual layout.
  - Restrict DFA/NFA-specific rules in the interface.

Q&A

Link to Github Project

https://github.com/sameerkhansf/JFLAPAutomataBuilder


slidesgo