CSAI335 – Theory of Computing (Fall 2025/2026)

Project 01: NFA to DFA Converter

Team Members

Farida Mohamed – [Your ID] [Teammate 2 Name] – [ID] [Teammate 3 Name] – [ID] (Add more if your group has up to 5 members)

Project Description

This project implements a Non-Deterministic Finite Automaton (NFA) to Deterministic Finite Automaton (DFA) Converter. The goal of this project is to take an NFA — represented as a set of states, input symbols, transitions, a start state, and one or more accept states — and automatically generate an equivalent DFA using the Subset Construction Algorithm. In simple terms, the program reads an NFA definition and outputs a DFA that recognizes the same language but behaves deterministically. This process is one of the core transformations in automata theory and forms the foundation of compiler design and lexical analysis.

Input Format

The NFA is defined inside the code as a Python dictionary: nfa = { 'states': {'q0', 'q1', 'q2'}, 'symbols': {'a', 'b'}, 'start': 'q0', 'accept': {'q2'}, 'transitions': { ('q0', 'a'): {'q0', 'q1'}, ('q1', 'b'): {'q2'} } }

Output Format

The output DFA is printed as a dictionary with states, transitions, start state, and accept states. Example output: DFA States: [frozenset({'q0'}), frozenset({'q0', 'q1'}), frozenset({'q2'})] DFA Start State: frozenset({'q0'}) DFA Accept States: [frozenset({'q2'})] DFA Transitions: δ({'q0'}, 'a') -> {'q1', 'q0'} δ({'q0', 'q1'}, 'a') -> {'q1', 'q0'} δ({'q0', 'q1'}, 'b') -> {'q2'}

Inside Mechanism

The Subset Construction Algorithm works as follows: 1. Start State: Create DFA start state as a set containing the NFA’s start state. 2. State Expansion: For each unprocessed DFA state and input symbol, compute all reachable NFA states. 3. Transition Creation: These reachable states form a new DFA state. 4. Iteration: Repeat until all possible new DFA states have been discovered. 5. Accept States: Any DFA state that includes an NFA accept state becomes a DFA accept state.

Programming Language and Tools

- Programming Language: Python - Environment: Jupyter Notebook - Libraries Used: collections (built-in), graphviz (optional for visualization)

Code Screenshot / Output Example

========== NFA to DFA Conversion ========== NFA States: {'q0', 'q2', 'q1'} NFA Start State: q0 NFA Accept States: {'q2'} DFA States: [frozenset({'q0'}), frozenset({'q1', 'q0'}), frozenset({'q2'})] DFA Start State: frozenset({'q0'}) DFA Accept States: [frozenset({'q2'})] DFA Transitions: δ({'q0'}, 'a') -> {'q1', 'q0'} δ({'q1', 'q0'}, 'a') -> {'q1', 'q0'} δ({'q1', 'q0'}, 'b') -> {'q2'}

Conclusion

This project successfully converts any given NFA into its equivalent DFA using the Subset Construction method. It demonstrates the theoretical principles of automata determinization and provides a working Python implementation.