Github: https://github.com/Taveeh/Formal-Languages-Compiler-Design/tree/Lab4

Finite Automata

The input file of the Finite Automata is as follows:

1st line: States

2nd line: Alphabet

3rd line: Initial State

4th line: Final States

On each other line is a transition, until the EOF

FA.in file:

EBNF:

file = states_line "\n" alphabet_line "\n" intial_state_line "\n" final_states_line "\n" {transition_line "\n"}

states_line = state_name {", " state_name}

alphabet_line = character {", " character}

initial_state_line = state_name

final_states_line = state_name {", " state_name}

transition_line = state_name ", " character ", " state_name

state_name = letter{letter}

character = letter | digit | symbol

letter = "a" | "b" | … "z" | "A" | "B" | … | "Z"

digit = "0" | "1" | … | "9"

symbol = "-" | "+" | "_" | "*" | "/"

Data structure used for the FiniteAutomata: A class with an array of states, an array representing the alphabet, a value for the initial state, an array for the final states and a dictionary for the transitions saved as: key is a pair (state, value) and value is a state, meaning from the state in key to the state in value we have transition value

Verification if a sequence is accepted by the finite automata is done as follows:

Starting from the initial state, we go character by character in the sequence, and if there is a value in the transition dictionary with value (current_state, character), we continue with the

current_state as the resulting value. Otherwise stops and returns false. If the automaton is not DFA, returns false from the start.