Source code: https://github.com/davidalexandru1370/LabLFTC/tree/lab4

(Available only on Lab4 branch)

# Documentation

## Finite automata class

- Internal representation for states, transitions, initial states, final states is:
  - InitialState is  a string
  - FinalStates is a list of strings
  - Alphabet is  a list of strings
  - Transitions are represented by a hasmap using as key a pair of State -> Symbol and as value a set of states representing the next states (

Symbol

  - eg: State  ->  nextState

In the constructor we get the file path to read the finite automata input and then file is parsed and data get inserted in program

ToString(): String – this method return a friendly string with internal representation of finite automata. Other helper methods are used to split the responsibilities. This kind of helper methods iterate over all the elements in their collection and append into the result string that collection's data

MatchSequence(sequence: String): String – this method checks if a sequence can be parsed by the finite automata. It goes through every letter until the end of sequence or there is no longer a transition in finite automata. In case it ends into a final state the matched sequence is returned otherwise the empty string is returned

EBNF:

Letter ::= 'a' | 'b' | 'c' | … 'z' | 'A' | 'B' | … | 'Z'

Digit ::= 0 | 1 | … | 9

SpecialSymbol = '+' | '-'

AlphabetCharacter = letter | digit | specialSymbol

State ::= letter

InitialState ::= letter

Alphabet ::= AlphabetCharacter {AlphabetCharacter }

FinalStates ::= state {state }

Transition ::= state AlphabetCharacter state

AllTransitions ::= Transition’\n’{Transition’\n’}