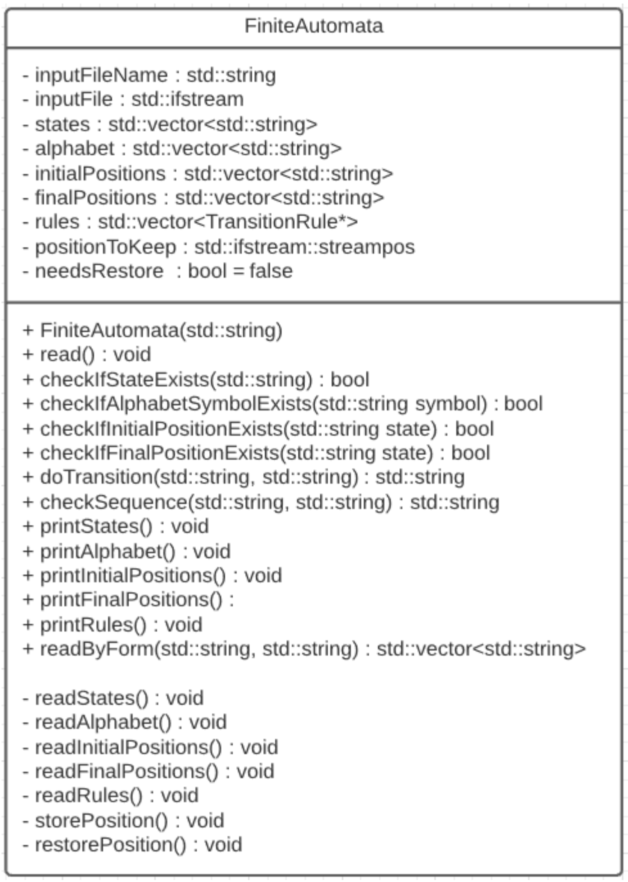
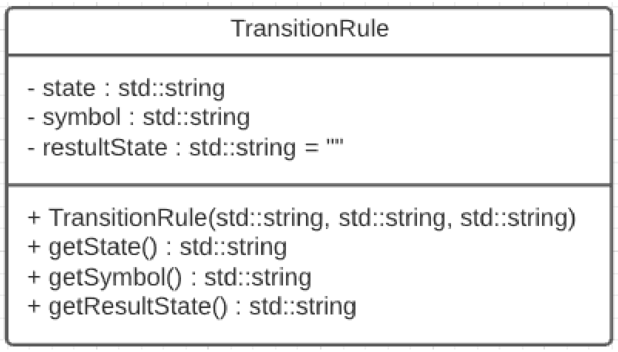
Finite Automata

(lab 4)

Class diagrams:


FiniteAutomata
- inputFileName : std::string
- inputFile : std::ifstream
- states : std::vector<std::string>
- alphabet : std::vector<std::string>
- initialPositions : std::vector<std::string>
- finalPositions : std::vector<std::string>
- rules : std::vector<TransitionRule*>
- positionToKeep : std::ifstream::streampos
- needsRestore : bool = false
+ FiniteAutomata(std::string)
+ read() : void
+ checkIfStateExists(std::string) : bool
+ checkIfAlphabetSymbolExists(std::string symbol) : bool
+ checkIfInitialPositionExists(std::string state) : bool
+ checkIfFinalPositionExists(std::string state) : bool
+ doTransition(std::string, std::string) : std::string
+ checkSequence(std::string, std::string) : std::string
+ printStates() : void
+ printAlphabet() : void
+ printInitialPositions() : void
+ printFinalPositions() :
+ printRules() : void
+ readByForm(std::string, std::string) : std::vector<std::string>
- readStates() : void
- readAlphabet() : void
- readInitialPositions() : void
- readFinalPositions() : void
- readRules() : void
- storePosition() : void
- restorePosition() : void


TransitionRule
- state : std::string
- symbol : std::string
- restultState : std::string = ""
+ TransitionRule(std::string, std::string, std::string)
+ getState() : std::string
+ getSymbol() : std::string
+ getResultState() : std::string

Documentation:

Input file:

-the first line is for the possible states. What is on the left of the equal sign does not really matter. Is just a descrption for the readear, but on the right side, there should be states separated by a '|'.

-the second line is like the first one, except that here should be the symbols

-the third line is the same as the previous ones, this time denoting initial positions

-the forth line demotes final positions, with the same layout as the previous

-on the fifth line, we have the rules. Every rule should be on it's onw line, and should have the following format: *„(%s1, %s2) -> %s3”* (where *%s1* - state, *%s2* – symbol, *%s3* – result state). This form is given as paramter to the function *readByForn(std::string, std::string)*. First string refers to form (currently being *„(%s, %s) -> %s”*, but can be changed if needed), second string is the input in which we check after the given form.

Using the program:

-when running, a menu with numbers from 0 to 6 with their corresponding meaning is displayed. Just enter a number and press *enter*.

FiniteAutomata class:

- positionToKeep and needsRestore are not really used
- rules – a vetctor of TransitionRules instances that stores the possible transitions of ours automata
- the class contains methods wor reading the states, alphabet (meaning the symbols
  which form the alphabet of the FA), initial positions, final position and rules of transition.
- the class contains methods for checking is a symbol/state is already stored in the class
- the class contins methods for printing the elements of an automaton: *printStates(), printAlphabet(), printInitialPositions(), printFinalPositions()* and *printRules().*Though, this methods directly print the information. They do not return anything,
- *doTransition(std::string state, std::string symbol)* checks if there is any rule ragarding the given *state* and *symbol*, and if found, returns the resulting state. If not, an empty string is returned
- *checkSequence(std::string startState, std::string symbolSequence)* – is mainly for ckecking if the sequence given that starts from the *startState* is accepted by the FA or not. If it is accepted, it return the ending state. If it is not accepted, it returns the empty string.
- *read()* method, when called, reads the file containing the input

TransitionRule class:

- helpful class to represent the possible transitions and also for storing them in FiniteAutomata class
- only has a constructor and the getters