Link to source code:
https://github.com/Akitektuo/University/tree/master/3rd%20year/FLCT/lab/Lab5

Link to maximum grade source code:
https://github.com/Akitektuo/University/tree/master/3rd%20year/FLCT/lab/Lab5Max

Finite automata documentation

The finite automata class takes as a parameter the file name that contains the definition of the automata. It has properties for accessing the set of states, initial state, set of final states, set containing the alphabet and transitions.

Transition classes

The Transition class contains the position (source and destination) with its respective value, overrides the *toString()* method in order to format the data and has a validate function which checks if the position is in the set of states and value to be part of the alphabet. This class is further used in Transitions which holds a map of Pair of source and value, each with a list of transitions. Has an add that checks for existing entries based on the key, if an entry is found, the transition is added to the list and the *isDeterministic* flag is set to false, otherwise for the given key, the transition is added wrapped in a list. There is also a method to obtain a transition based on source and value and a validate function that triggers the validate function of each transition contained in the map.

Is deterministic

The function calls the transitions' respective function to verify if the finite automaton is deterministic or nondeterministic.

Initialize

The function wraps the finite automata in a try-catch block to print all *InvalidFiniteAutomataException*s and calls the functions *readProperties* and *validate*.

Read properties

The function reads the file line by line and extracts the states, initial state, final states, alphabet and transitions.

Validate

The validate function checks if the initial state and final states are part of the set of states and calls the validation function of the transitions class. If an issue is detected, an *InvalidFiniteAutomataException* is thrown.

Is accepted

The function takes a sequence represented as a String and if the automaton is nondeterministic, false is returned, otherwise it returns the result of the *isAcceptedRecursive* function.

Is accepted recursive

The recursive function takes as parameters the current state of the automaton, the sequence and the current index of the sequence that is being evaluated. If the index exceeds the sequences' last index, it returns the result of the current state being part of the final states. If not, the symbol is extracted and the available transitions are obtained based on the current state and symbol. If they are null, then false is returned, otherwise it is returned the value of any transition that is accepted as recalling the function with the transition's destination, same sequence and the index incremented by one.

EBNF

file = states initialState finalStates alphabet transitions

states = word { word }
initialState = word
finalStates = word { word }
alphabet = word {word}
transitions = transition {transition}

transition = word word word
word = character {character}
character = “a” | “b” | ... | “z” | “A” | “B” | … | “Z” | “0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9”