Ternovan Darius-Daniel, group 937/1
Github: https://github.com/ternovandarius/FLCD/tree/master/Lab%205

The FAReader class is initialised with a filename. It opens that filename, reads the Finite Automaton as long as it is formatted as described below, and then offers an explicit command-based menu that allows users to see details about the automaton, as well as check if a given sequence is accepted by the FA.

While the reading of the FA is fairly simple, the transitions need to be explained. Since the transitions come in last in the input document, it will keep checking on each line for the pairs. If there already is a pair of the first two elements on the line, it will add the third element to the value of the corresponding key. If not, it will create a key-value pair, with the two elements being the key and the value being an array with the third element.

Another thing that might need explaining is the way we check if a sequence is accepted. This requires the user to input their sequence, then calls the method “accepted”. In accepted, we have a variable called “state”, which is initialised with the value of the initial state of the FA. Then, for each character in the sequence, we check if there exists a key (state, character) in the transitions hash. If there isn’t, then the sequence is not accepted and we return False. If there is, we take the first value out of the value array of the key (there will be no issue with only taking the first value, as the requirement says that we should only check if a sequence is accepted for DFAs which will always have one value in the value array). If the method has checked all of the characters without returning False, we check if the current state is in the list of final states of the FA. If it is, we return True, else False.

Below you will find the required format for the input file.

Fa.in:

StatesSet NewLine Alphabet NewLine InitialState NewLine FinalState NewLine Transitions

StatesSet := (“a” | “b” | “c” | … | “A” | “B” | “C” | … | “Z”) {“ “ StatesSet}

NewLine := “\n”

Alphabet := (“0” | “1” | … | “9” | “a” | “b” | … | “A” | “B” | … | “Z”) {“ “ Alphabet}

InitialState := “a” | “b” | “c” | … | “A” | “B” | “C” | … | “Z”

FinalState := (“a” | “b” | “c” | … | “A” | “B” | “C” | … | “Z”) {“ “ FinalState}

Transitions := Transition {“ “ Transition}

Transition := (“a” | “b” | “c” | … | “A” | “B” | “C” | … | “Z”) “ “ (“0” | “1” | … | “9” | “a” | “b” | … | “A” | “B” | … | “Z”) “ “ (“a” | “b” | “c” | … | “A” | “B” | “C” | … | “Z”)