<https://github.com/thediciman/FLCD/tree/master/Lab%204>

The finite automaton is represented as a class that contains 5 members:

* the set of states, which is a list of strings
* the set of symbols from the alphabet, which is a list of strings
* the initial state, which is a string
* the set of final states, which is a list of strings
* the set of transitions, which maps a pair of type (source, route) to a destination

This class has a static method that allows for a FA to be read from a file, which returns a FA created from the input file if it is valid, or throws an exception otherwise.

The structure of the file is the following:

* first line: the set of states, separated by space
* second line: the set of symbols from the alphabet, separated by space
* third line: the initial state
* fourth line: the set of final states, separated by space
* all of the remaining lines will contain a transition of the form (source\_state, route\_symbol) -> destination\_state

If we were to put this in BNF, it would have the following structure:

fa = states ‘\n’ alphabet ‘\n’ initial\_state ‘\n’ final\_states ‘\n’ transitions

states = state | state ‘ ‘ states

alphabet = symbol | symbol ‘ ‘ alphabet

initial\_state = state

final\_states = state | state ‘ ‘ final\_states

transitions = transition | transition ‘\n’ transitions

transition = ‘(‘ state ‘,’ symbol ‘)’ ‘->’ state

character = letter | digit

string = {character}

letter = uppercase\_letter | lowercase\_letter

uppercase\_letter = "A" | "B" | ... | "Z"

lowercase\_letter = "a" | "b" | ... | "z"

digit = “0” | "1" | ... | "9"

state = string

symbol = string

The FiniteAutomaton class also has the following methods:

* a method that checks if the FA is a DFA, which checks that for all destination states, there is no more than one route that goes to it.
* a method that checks if a sequence is accepted by the FA

Example input file:

s a b k  
0 1  
s  
s a k  
(s, 0) -> s  
(s, 1) -> a  
(a, 1) -> s  
(a, 0) -> b  
(b, 0) -> k