

TONCEA ION-ALIN

GROUP 937/2

LAB4

<https://github.com/TonceaAlin/FLCDlab/tree/master/lab4>

Finite Automata is represented as a class with the respect to the definition of a FA

```
class FiniteAutomata:
    def __init__(self, Q, E, q0, F, S):
        self.Q = Q
        self.E = E
        self.q0 = q0
        self.F = F
        self.S = S
```

Transition functions are represented like this:

```
S = (X, 1) Y; (Y, 2) Y; (Y, 1) Z; (X, 2) Z;
```

being held in a Map with the members between parenthesis as a key and the value after as value. This has the meaning: '(first element' has the cost 'second element)' to the 'destination;'

The others elements are read from the file and are being kept in lists as strings

After the handling part:

1. Checking that the FA is deterministic: done by checking if in the final states exists a key which has more than one destination

```
def isDFA(self):
    for key in self.S.keys():
        if len(self.S[key]) > 1:
            return False
    return True
```

2. Checking the acceptance for a sequence received as input. This is done by parsing the finite automata like a oriented graph. We start from the initial state and we check if in the end we achieved a final state

```
def isAccepted(self, seq):  
    if self.isDFA():  
        crt = self.q0  
        for symbol in seq:  
            if (crt, symbol) in self.S.keys():  
                crt = self.S[(crt, symbol)][0]  
            else:  
                return False  
        return crt in self.F  
    return False
```

EBNF for the FA

FA : Description

FA = "Q" | "E" | "q0" | "F" | "S"

Description = states | alphabet | initialState | finalStates |
transitions

States = letter{,letter}

letter = a..z|A..Z

alphabet = (letter|digit){,letter|digit}

digit = 0..9

initialState = letter

finalStates = letter{,letter}

transitions = {(letter, alphabet) letter;}

