https://github.com/fiamardar/FLCD -> GitHub Repository Link

Lab 4

Finite Automata class

```
class FiniteAutomata:
    """
    Definition: A finite automaton (FA) is a 5-tuple
    M = (Q,Σ,δ,q0,F) where:
    • Q - finite set of states (|Q|<∞) ex: Q = {q0, q1, q2}  => the line contains: q0,q1,q2
    • Σ - finite alphabet (|Σ|<∞)      ex: ∑ = {0, 1}        => the line contains: 0,1
    • q0 - initial state q0 ∈ Q        ex: q0 = {q0}         => the line contains: q0
    • F⊆Q - set of final states        ex: F = {q2}          => the line contains: q2
    • δ - transition function : δ:Q×Σ→P(Q)
        • we will read from file line by line the transitions
                    ex: q0 ->(1) q1    => the line contains: q0,q1,1 (initial state, final state, value)

    """
    _file = None
    _set_of_states = []
    _alphabet = []
    _initial_state = None
    _set_of_final_states = []
    _set_of_transitions = []
```

Stores the set of states, the alphabet, the initial state, the set of final states and the set of transitions. It also receives the name of the file from where we want to read the FA.

It contains the function to check if a sequence is accepted by the FA:

```
def check_sequence(self, sequence):
    """
    Function checks if a sequence is accepted by the FA.
    It is accepted if and only if the last state of the sequence is also one of the final states
    and if the input is empty (if we find corresponding transitions for all the elements in the sequence)

    :param sequence: The sequence we want to check
    :return: True, if the sequence is accepted by the FA, False otherwise
    """
    # First set the current state to the initial state
    current_state = self._initial_state

    # Parse the sequence character by character
    for character in sequence:

        # returns the transition which goes from current state with current value
        transition = self.get_corresponding_transition(current_state, character)

        if transition is None:
            return False

        current_state = transition.get_final_state()

    if current_state in self._set_of_final_states:
        return True
    return False
```

FA.in file:

Lexic:

States (including startstates, initialstates, endstates and finalstates):

- a sequence representing the name of the state
  (ex: could be any letter P, Q, R or any sequence P1, P2, P3)

Sintactical rules:

The program will contain at least 5 lines:

- The first line will contain all the states separated by space
- The second line will contain the alphabet, the symbols are separated by space
- The third line will contain the initial state
- The fourth line will contain the final states separated by space
- The next lines until the end of the file will represent each transition:
  - A transition line contains the starting state of the transition, followed by space, followed by the end state of the transition, followed by space, followed by the symbol value of the transition

EBNF:

program = line1 “\n” line2 “\n” line3 “\n” line4 “\n” transitionslines

line1 = STATE | STATE “ “

line2 = symbol | symbol “ “

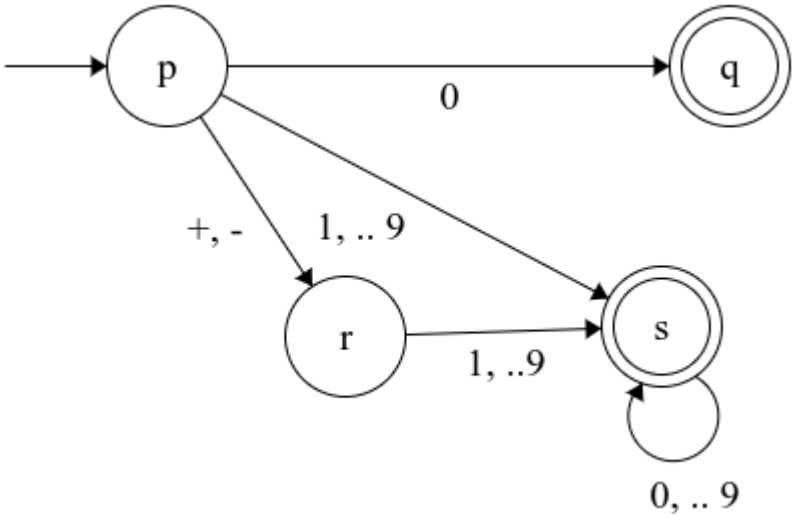
line3 = INITIALSTATE

line4 = FINALSTATE | FINALSTATE “ “

transitionslines = transition | transition “\n”

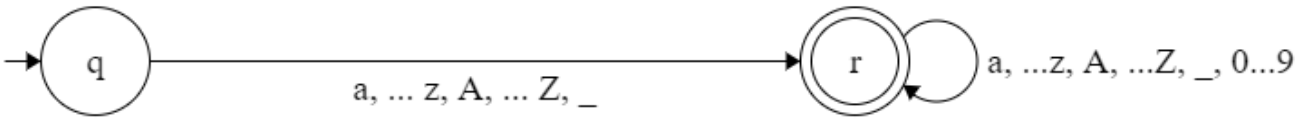
transition = STARTSTATE “ “ ENDSTATE “ “ symbol

FA_integer.in (The file with FA for integer)




```
p q r s
+ - 0 1 2 3 4 5 6 7 8 9
p
q s
p q 0
p s 1
p s 2
p s 3
p s 4
p s 5
p s 6
p s 7
p s 8
p s 9
p r +
p r -
r s 1
r s 2
r s 3
r s 4
r s 5
r s 6
r s 7
r s 8
r s 9
s s 0
s s 1
s s 2
s s 3
s s 4
s s 5
s s 6
s s 7
s s 8
s s 9
```

FA_identifier.in (The file with FA for identifier)




```
q r
a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K
L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9 - _
q
r
q r a
q r b
q r c
q r d
q r e
q r f
q r g
q r h
q r i
q r j
q r k
q r l
q r m
q r n
q r o
q r p
q r q
q r r
q r s
q r t
q r u
q r v
q r w
q r x
q r y
q r z
q r A
q r B
q r C
q r D
q r E
q r F
q r G
q r H
q r I
q r J
q r K
q r L
q r M
q r N
q r O
q r P
q r Q
q r R
q r S
q r T
q r U
q r V
```

```
q r W
q r X
q r Y
q r Z
q r _
r r a
r r b
r r c
r r d
r r e
r r f
r r g
r r h
r r i
r r j
r r k
r r l
r r m
r r n
r r o
r r p
r r q
r r r
r r s
r r t
r r u
r r v
r r w
r r x
r r y
r r z
r r A
r r B
r r C
r r D
r r E
r r F
r r G
r r H
r r I
r r J
r r K
r r L
r r M
r r N
r r O
r r P
r r Q
r r R
r r S
r r T
r r U
r r V
r r W
r r X
r r Y
r r Z
r r _
r r -
r r 0
r r 1
r r 2
r r 3
```

```
r r 4
r r 5
r r 6
r r 7
r r 8
r r 9
```