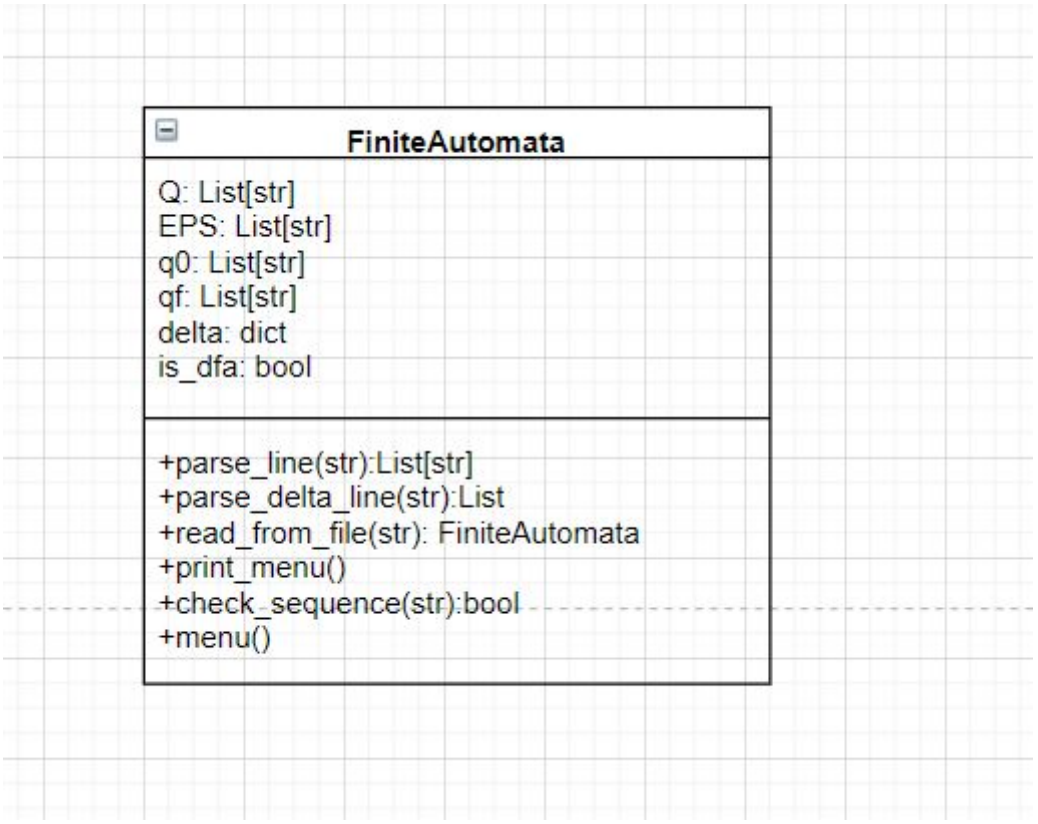
Laboratory 4

https://github.com/IoanaBaciu24/LFTC



For this laboratory, the class FiniteAutomata can be instantiated using a static method that reads from the file the data and creates an instance (read_from_file). It is important that the structure of the file is this:

1. Each field (apart from delta and is_dfa) are on separate lines, in the order from the diagram
2. For delta, once Q, EPS, q0 and qf are read, the program will read line by line until EOF and will add the transitions in a dictionary. As this process happens, it will check whether the FA is deterministic or not, namely if there is a left hand side of a transition that repeats itself, and will set the parameter accordingly.
3. The transitions are kept in a dictionary of the form {(state, valie) : [next_state]}. In case of a non deterministic FA, the list corresponding to each key will have more than one element.

The checking is done using check_sequence. The algorithm is straight-forward, it goes through the sequence as long as there is a sequence and it is correct. The algorithm was tested on the FA from the lab, and the 2 FAs that replaced the regex expressions in the scanner.

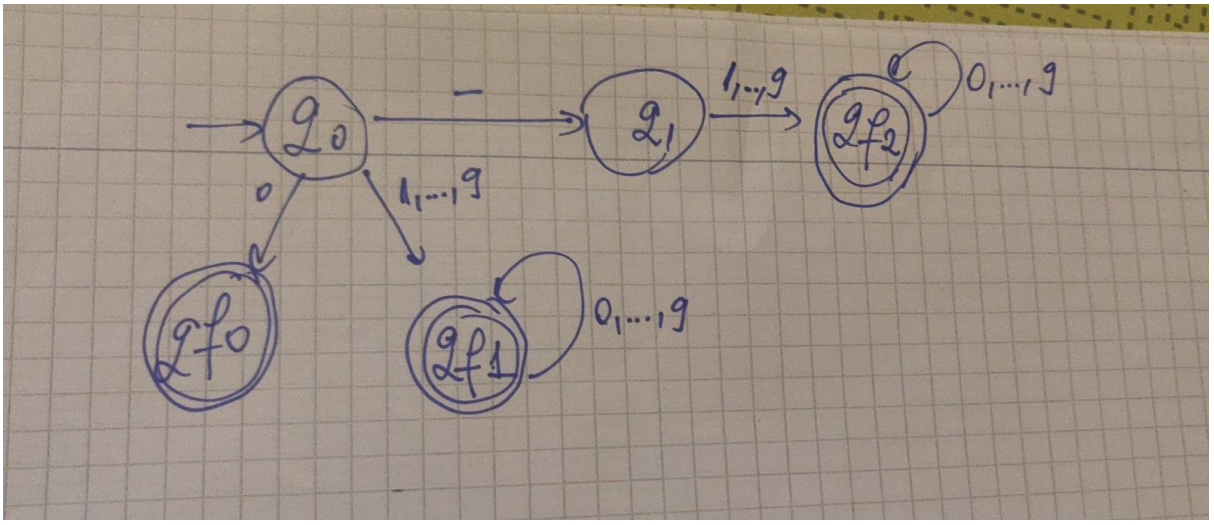
FA.in ::= Q \n EPS \n q0 \n qf \delta
<Q> ::= <state>’,’ | <state>
<state> ::= <letter> | <letter><digit>
<letter> ::= ‘a’ | .. | ‘z’ | ‘A’ |...|’Z’
<digit> ::= ‘0’|...|’9’

<EPS>::=<alphabet_unit>’,’|<alphabet_unit>
<alphabet_unit> ::= ASCII symbol (the user chooses the content of the alphabet)

<q0>::=<state>
<qf>::= <state>’,’ | <state>

<delta> ::= <transition> ‘\n’ | <transition>
<transition> ::= <state>,<alphabet_unit> = <state>

The finite automata for integer constants



The finite automata for identifiers: