Finite automata parser

-transitions – list containing tuples representing the transitions for the automata

-alphabet – all the characters of the language

-states – list containing the states for the automata

-initial\_states – list containing the initial states for the automata

-final\_states – list containing the final states for the automata

-verify(input, state):

-recursively verifies if the given input can pass through the automata

-params:

-input: string representing the input to be verified

-state: state that the input is in at the current step of the verification

-return: True if the input matches, False otherwise

**<letter> ::= A | B | … | Z |a |b | … | z**

**<digit> ::= 0 | <non-zero-digit>**

**<non-zero-digit> ::= 1 | … | 9**

**<program> ::= <transitions-decl> “\n” <alphabet-decl> “\n” <states-decl> “\n” <initial-decl> “\n” <final-decl>**

**<transitions-decl> ::= “Transitions \n” <transition> “\n” | <transitions-decl> <transition> “\n”**

**<transition> ::= <state> “,” <value> “,” <state>**

**<value> ::= <letter> | <digit> | + | -**

**<state> ::= <letter> | <digit> | <state> <digit> | <state> <letter>**

**<alphabet-decl> ::= “Alphabet \n” <value> “\n” | <alphabet-decl> <value> “\n”**

**<states-decl> ::= “States \n” <state> “\n” | <states-decl> <state> “\n”**

**<initial-decl> ::= “Initial\_states \n” <state> “\n”**

**<final-decl> ::= “Final\_states \n” <state> “\n”| <final-decl> <state> “\n”**

[GitHub - TudorFernea/FLCDLab2](https://github.com/TudorFernea/FLCDLab2)