https://github.com/StefanCsPurge/Formal-Languages-and-Compiler-Design

Finite Automata (FA)

Python class with fields:

- inputFile – given as constructor parameter
- Q – list of all states
- E – list containing the alphabet
- q0 – the initial state
- F – list containing the final states
- Tr – list containing all the transitions
  Each transition is a (sourceState, terminalSymbol, destinationState) tuple.

All the necessary components are read from the given input file (using the readFA_fromFile() method) when the class is instantiated.

The method **isSequenceAccepted(sequence)** checks if the given sequence can be formed using the FA by iterating the chars of the sequence. For each character, the algorithm tries to get the new state by using the transitions list. If a transition is not found for the current character, it immediately returns false. Otherwise, if it successfully reaches the final state after the final character, it returns true.

FA.in file – contains on each line the required components for the FA (Q,E,q0,F,Tr), mainly using the space as a separator.

Example of correct FA.in:

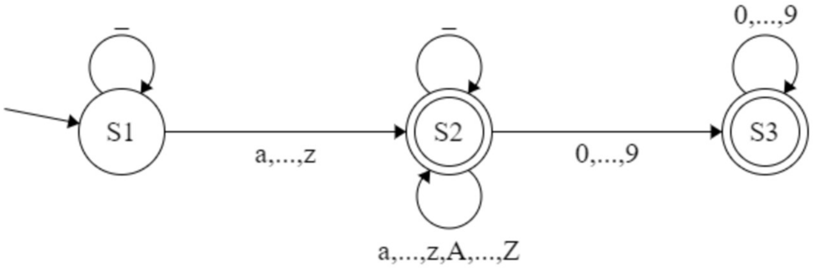
```
Q = A B C
E = 1 2
q0 = A
F = C
Tr =
(A,1) -> C
(A,2) -> B
(B,1) -> C
(C,2) -> B
```

Detection of <identifier> and <integer constant> in the scanner program

The above defined *FiniteAutomata* class can be used to replace regex matching.

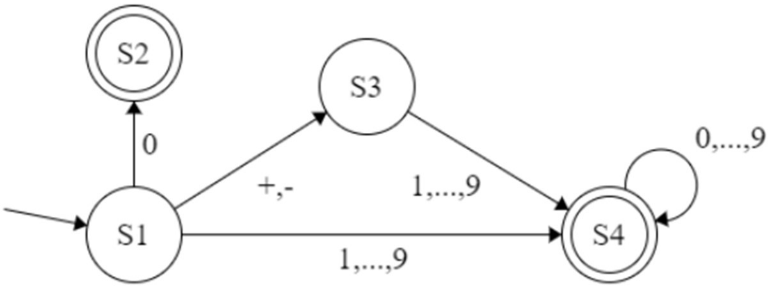
Identifier Python regex: `^_*[a-z](_|[a-zA-Z])*[0-9]*$`

Identifier FA:
(detailed in the
FA-identifier.in
input file)



Integer constant Python regex: `^(0|[+-]?[1-9][0-9]*)$`

Integer constant FA:
(detailed in the
FA-int-constant.in
input file)


S2
0
S3
+,-
1,...,9
0,...,9
S1
1,...,9
S4