09.11.2021

Andrei-Ovidiu Muntean
Computer Science, year 3, group 935/1

FLCD Lab 4

written in Python 3.9

Github

Please check lab3 folder for the integration of the FA with the Scanner
This pdf is available in lab4 folder
https://github.com/netfree/lftc

Implementation

State class representation

a state has a `name` and a field `final` specifying that it is final

FiniteAutomata class representation

```
self.states: Dict[str, State]
self.alphabet: List[str]
self.transitions: Dict[State, Dict[str, State]]
self.initial_state: State
```

Constructor

We keep the states as a dictionary for quick access
For each State, we have a dictionary of transitions with the key, the symbol, and the value of the next State

We have a method that parses the finite automata and checks whether a sequence of characters is accepted or not by the FA

We also have multiple `to_string` methods for all the fields

Accept sequence method

```
def accept_sequence(self, seq: str):
    current_state = self.initial_state
    for letter in seq:
        if letter in self.transitions[current_state].keys():
            current_state = self.transitions[current_state][letter]
        else:
            return False
    if current_state.final:
        return True
    return False
```

The accept sequence method parses the sequence and, for each symbol, tries to find the next state. If there is no transition for the symbol, it returns False.
If we have successfully consumed the sequence and we've reached a final state, we return True, otherwise False.

FiniteAutomataParser

It is a helper class that reads a FA from a file

Input file in EBNF form:

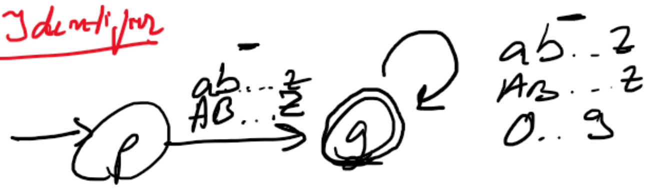
```
letter = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K"
| "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" |
"X" | "Y" | "Z" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |
"j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" |
"v" | "w" | "x" | "y" | "z"
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
special =  "?" | ":" | "." | " " | "!" | "@" ... | "#" | "<" | ">"

fa = “set_of_states\n” {state “\n”} “finite_alphabet\n” {symbol “\n”}
“transitions
\n” {transition “\n”} “initial_state\n” initialstate “\n”
“final_states\n” {state}
symbol = letter|digit|special
state = {letter|digit}
transition = state “ ” symbol “ ” state
```

09.11.2021

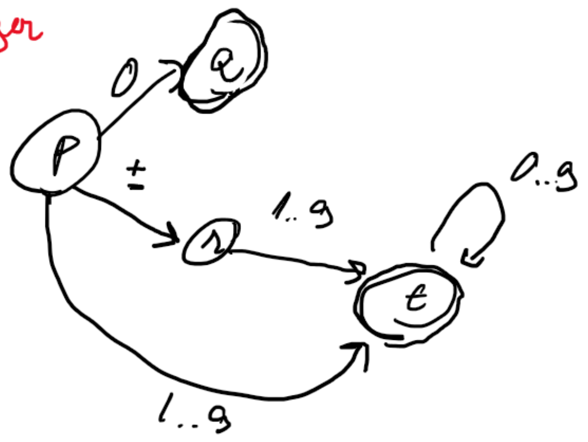
Andrei-Ovidiu Muntean
Computer Science, year 3, group 935/1

FA visual representation:

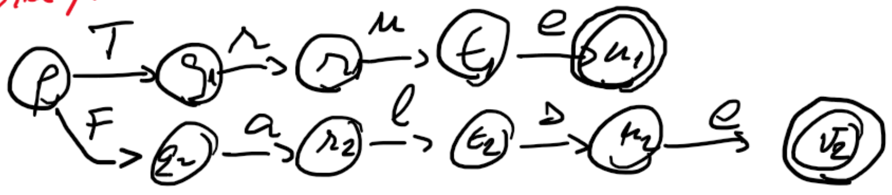
Identifier



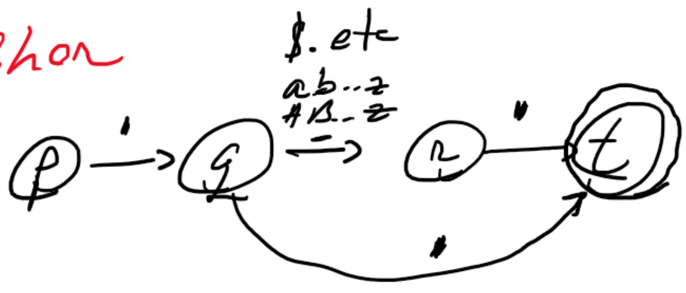
Integer



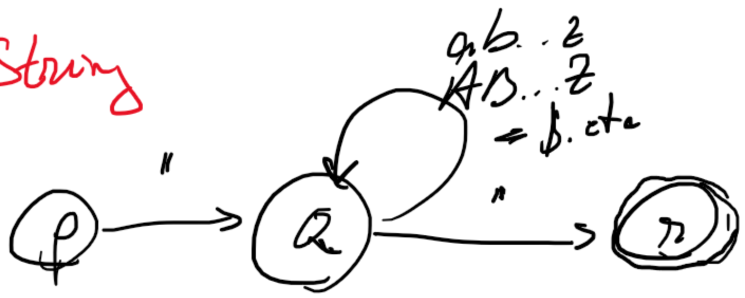
True/False



Char



String


p
q
r
"
ab..z AB..Z _$.etc
"