Lab 5

Github Link

https://github.com/rusuraluca/lftc/tree/main/lftc_lab5

Docs

`class FiniteAutomata`: will read the file when it is initialized, will parse the input file and then populate all the fields in the class

`states`

- [] -> is an array which contains all the possible states

`alphabet`

- [] -> is an array which contains all the possible letters

`transitions`

- {} -> this would represent a map, where the key represents a pair between (state, alphabet_value) and the value represents the projection of the

`initial_state`

- "" -> a simple string would be enough because we can have only one initial_state

`final_states`

- [] -> the array of final states

- `__init__`
  - we initialize the FiniteAutomata with the filename
  - we initialize all the fields
  - we call the read_from_file method
- `read_from_file`
  - we read from the file and populate all the fields
- `print_fa`
  - we print the FiniteAutomata
- `start_menu`
  - we start a menu where we can print any of the fields
- `print_menu`
  - we print the menu
- `check_word_if_integer_constant`
  - we check if a word is an integer constant
- `check_word_if_identifier`
  - we check if a word is an identifier

FA.in

```
M = {
Q = q0, q2, q4, g5, q6
E = _, a, b, c, d, e, f, g, h, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, +, -
RO =
start
q0, _ -> q2
q2, [a, b, c, d, e, f, g, h, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z] -> q2
q0, [1, 2, 3, 4, 5, 6, 7, 8, 9] -> q5
q0, 0 -> q6
q0, [+, -] -> q4
q4, [1, 2, 3, 4, 5, 6, 7, 8, 9] -> q5
q5, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] -> q5
end
q0 = q0
F = q2, q5, q6
```

For a DFA, verifies if a sequence is accepted by the FA.

In order to check if an word is generated by the Finite Automata, we have 2 functions, one for integers and one for identifiers. Considering that we have a dfa, we don't need to check for multiple entrances in the transitions dictionary, only 1

e.g.

- 12-34, check if integer:
  (q0, '') -> (q5, '1') -> (q5, '12') -> at that point, no transition is found, so we return false
- 1234, check if integer
  (q0, '') -> (q5, '1') -> (q5, '12') -> (q5, '123') -> (q5, '1234')

EBNF for FA.in

FA.in ::= "M = {\n" Q "\n" E '\n' RO '\n' q0 '\n' F
Q ::= {state","} state
E ::= {alphLetter","} alphLetter
RO ::= "start\n"