Github repo: https://github.com/915-Petruta-Razvan/LFTC_Labs

Documentation

FiniteAutomaton

Details: this class helps us read a finite automaton's information from a file (states, alphabet, transitions, initial state and the final states) and also, for a deterministic finite automaton, it can be used to check if a sequence is accepted by the finite automaton.

Methods:

- PrintStates(): it prints the set of states
- PrintAlphabet(): it prints the alphabet
- PrintTransitions(): it prints all the transitions between states
- PrintInitialState(): it prints the initial state
- PrintFinalStates(): it prints the set of final states
- IsWordValid(word: string): given a sequence, it checks if it is accepted by the finite automaton. From the sequence, it goes character by character, starting from the initial state and checking if there is a transition to other state using as symbol the current character. If we cannot find a transition matching the current character, the sequence is not valid. Otherwise, it updates the current state based on the found transition and at the end we check that the current state is a final one, in which case our sequence is valid, otherwise it is not.
- PrintListOfStrings(name: string, list: List<string>): function used to print a list of strings in a nice format
- InitFromFile(): it reads the information about the finite automaton from the file and initialise the necessary filelds
- GetLineRightHandSide(line: string): function used in the InitFromFile(). When reading a line, we need to take the right hand side of the equal to initialise our fields
- IsDeterministicFiniteAutomaton(): checks that the finite automaton is a valid one (it has the states, the alphabet, the final states, the initial state and that the initial and final states are among the states) and after checks if it is a deterministic finite automaton (from a state we can go to a maximum of one state using the same symbol)
- GetTokenFromFA(word: string): given a string, it returns the sequence from index 0 that is accepted by the finite automaton

Transition

Details: this class is used to represent a transition, having 3 fields: FromState, ToState and the Symbol.

Menu

Details: this class is used to create a menu for the user to interact with. After reading a finite automaton, the user can print the states, alphabet, transitions, initial state, final states and can input a word and check if it is accepted by the automaton as a valid sequence.

EBNF format of the finite_automaton.txt

character = “0” | ”1” | ... | ”9” | ”a” | ”b” | ... | ”z” | ”A” | ”B” | ... | ”Z”
states_line = “states={” character {“,”character} ”}\n”
alphabet_line = “alphabet={“ character {“,”character} “}\n”
transition = “(“ character “,” character “,” character “)”
transitions_line = “transitions={“ transition {“|”transition} “}\n”
initial_state = “initial_state={“ character “}\n”
final_states = “final_states={“ character {“,”character} “}”
finite_automaton_file = states_line alphabet_line transitions_line initial_state final_states

Scanner

Modified methods:

- CheckIntConstant(): instead of using the regex, we are using the finite automaton defined for an int constant in the file int_constantFA.txt
- CheckIdentifier(): instead of using the regex, we are using the finite automaton defined for an identifier in the file identifierFA.txt