Former-languages-and-compiler-design/Lab4 at main · 915-Nistor-Anca/Former-languages-and-compiler-design (github.com)

Documentation – Finite Automaton

Firstly, the program reads the elements of a FA, then checks if it is a DFA or a NFA. In case it's a DFA, it is given a sequence, and it calls a function which checks whether the sequence is accepted by the FA or not.

FiniteAutomaton class:

__init__(self, file_name): Initializes the FiniteAutomaton class. file_name is the name of the file containing FA information.

read_from_file(self): Reads FA information from a file and initializes the FA attributes accordingly. The first line has of states enclosed in curly braces {} and separated by commas. The second one contains the alphabet enclosed in curly braces {} and separated by commas. The third line has the transitions separated by semicolons. The forth line represents the initial state. The fifth line contains the final states enclosed in curly braces {} and separated by commas.

__str__(self): Displays the details of the FA including states, alphabet, transitions, initial state, and final states.

breakTransition(self): The transitions are saved in a format [(state1, symbol)=state2, ...], which might be hard to work with. This function takes each transition in a list and transforms it into a list of type [state1, symbol, state2]. It first removes the parentheses, then splits everything by commas, resulting in a list like [state1, symbol=state2]. After that, it splits each element of the list by '=', but this creates a list within a list. The function iterates through the outer list to create a new, simplified list having the desired format.

checkSequence(self, sequence): This function checks if the given sequence is accepted by the FA. A sequence is accepted if, starting from the initial state, it can find a transition where the first state is the initial state and the symbol matches the first symbol of the sequence. This logic continues until the last symbol of the sequence, where the program also checks if it is in the list of final states or not. It returns a corresponding message regarding whether the sequence is accepted or not.

checkIfDFA(self): This function checks if the given FA is a DFA or not. It is a DFA if, for each state and for each symbol of the alphabet, it can find at most one transition where the first state is that state and the symbol is that symbol.

The FA should be written in this EBNF format:

```
letter = a | b | c | ... | z | A | B | ... | Z
digit = 0 | 1 | 2 | ... | 9
state = letter {digit}
symbol = lettter | digit
transition = ”(”  state ”,” symbol ”)” ”=” state
initial_state = state
final_state = state

first_line = ”{” state {”,” state} ”}”
second_line = ”{” symbol {”,” symbol} ”}”
third_line = ”{” transition {”,” transition} ”}”
fourth_line = initial_state
fifth_line = ”{” final_state {”,” final_state} ”}”
input_file = first_line ”\n” second_line ”\n” third_line ”\n” fourth_line ”\n” fifth_line
```