Github Link:
https://github.com/cs-ubbcluj-ro/lab-work-computer-science-2024-raresdan/tree/main/2-Finite-Automata/Lab4

Documentation:

Class: `Transition`

Represents a state transition in a finite automaton.

Attributes:

- `from_` **(str): The state where the transition starts.**
- `to` **(str): The state where the transition leads.**
- `on` **(str): The input symbol that triggers the transition.**

Class: `FiniteAutomataData`

Data container for finite automaton properties.

Attributes:

- `states` **(List[str]): List of all states in the automaton.**
- `initial_state` **(str): The initial state of the automaton.**
- `final_states` **(List[str]): List of final (accepting) states.**
- `alphabet` **(List[str]): List of symbols in the automaton's alphabet.**
- `transitions` **(List[Transition]): List of all state transitions.**

Class: `FiniteAutomata`

Represents a finite automaton, including its configuration and operations.

Loads the finite automaton configuration from a JSON file.

- **Parameters:**
  - `file_path` **(str): Path to the JSON file containing automaton data.**

Attributes:

- `states` **(Set[str]): Set of states in the automaton.**
- `initial_state` **(str): The initial state.**
- `final_states` **(Set[str]): Set of final (accepting) states.**
- `alphabet` **(Set[str]): Set of symbols in the alphabet.**
- `transitions` **(List[Transition]): List of state transitions.**

Methods

`menu()`

Displays the menu options for interacting with the finite automaton.

- **No Parameters**
- **No Return Value**

`choice() -> int`

Prompts the user for a menu option and validates the input.

- **Returns:**
  - `int`**: The selected menu option, or** `-1` **if input is invalid.**

`display()`

Displays the finite automaton's properties based on user selection from the menu.

- **Behavior:**
  - **Option 1: Displays all states.**
  - **Option 2: Displays the initial state.**
  - **Option 3: Displays all final states.**
  - **Option 4: Displays the alphabet.**
  - **Option 5: Displays all transitions.**
  - **Option 6: Check sequence**
- **Returns:**
  - `int`**: Returns** `-1` **for "Exit",** `0` **for valid options,** `1` **for invalid options.**

`is_accepted(sequence: str) -> bool`

Checks whether a given input sequence is accepted by the finite automata.

- **Parameters:**
  1. `sequence` **(str): The input string to validate.**
- **Returns:**
  1. `bool`**:** `True` **if the sequence is accepted,** `False` **otherwise.**
- **Algorithm:**
  1. **Starts at the initial state.**
  2. **For each symbol in the input sequence:**
     - **Finds a matching transition based on the current state and symbol.**
     - **Moves to the next state if a transition exists; otherwise, rejects the sequence.**
  3. **After processing all symbols, checks if the final state is in the set of accepting states.**

BNF Representation

```
<FA_file> ::= "{"
            "\"states\": " <state_list> ","
            "\"initial_state\": " <state> ","
            "\"final_states\": " <state_list> ","
            "\"alphabet\": " <symbol_list> ","
            "\"transitions\": " <transition_list>
            "}"

<state_list> ::= "[" <state> { "," <state> } "]"
<state> ::= "\"" <identifier> "\""

<symbol_list> ::= "[" <symbol> { "," <symbol> } "]"
<symbol> ::= "\"" <character> "\""

<transition_list> ::= "[" <transition> { "," <transition> } "]"
<transition> ::= "{"
              "\"from\": " <state> ","
              "\"to\": " <state> ","
              "\"on\": " <symbol>
              "}"

<identifier> ::= (alphabetic | "_") { (alphanumeric | "_") }
<character> ::= alphanumeric
```