**Finite Automaton Class Documentation**

**Overview**

The FiniteAutomaton class is a Python implementation of a finite automaton (FA), which is a mathematical model used in computation theory to simulate a finite state machine. This class provides methods to create, manage, and analyze the components of an automaton, such as states, transitions, and alphabets. It supports both deterministic and non-deterministic finite automata (DFA and NFA).

**Class Attributes**

**\_\_init\_\_**

Initializes an empty finite automaton with the following attributes:

* **states**: A set of all states in the automaton.
* **alphabet**: A set of symbols (characters) in the automaton's alphabet.
* **transitions**: A dictionary representing state transitions, where each key is a state and each value is another dictionary mapping symbols to a set of destination states.
* **initial\_state**: The starting state of the automaton (initially None).
* **final\_states**: A set of final (accepting) states.

**Methods**

**Adding and Managing States**

* **add\_state(state, is\_final=False)**
  + Adds a state to the automaton.
  + Parameters:
    - state: Name of the state.
    - is\_final (optional): Boolean indicating whether the state is a final state (default is False).
* **set\_initial\_state(state)**
  + Sets the initial state of the automaton.
  + Parameter:
    - state: Name of the state to be set as the initial state.
* **get\_states()**
  + Returns the set of all states in the automaton.
* **get\_final\_states()**
  + Returns the set of all final states.
* **get\_initial\_state()**
  + Returns the initial state of the automaton.

**Adding and Managing Transitions**

* **add\_transition(from\_state, symbol, to\_state)**
  + Adds a transition between two states using a symbol from the alphabet.
  + Parameters:
    - from\_state: Starting state of the transition.
    - symbol: Symbol used for the transition.
    - to\_state: Destination state of the transition.
* **write\_transitions()**
  + Returns a string representation of all transitions in the automaton.
  + Format: from\_state --symbol--> to\_state.
* **get\_alphabet()**
  + Returns the set of symbols in the automaton’s alphabet.

**Automaton Analysis**

* **check\_if\_deterministic()**
  + Checks if the automaton is deterministic (DFA).
  + Returns True if every state has at most one transition for each symbol, otherwise False.
* **is\_accepted(input\_string)**
  + Determines if a given input string is accepted by the automaton.
  + Parameters:
    - input\_string: A string consisting of symbols from the automaton’s alphabet.
  + Returns:
    - True if the input is accepted, False otherwise.

**Input and Output Operations**

* **parse\_fa\_file(fa\_file)**
  + Reads automaton components from a file.
  + File format:
    1. Line 1: List of states separated by semicolons (;).
    2. Line 2: Initial state.
    3. Line 3: Final states separated by semicolons (;).
    4. Lines 4+: Transitions in the format: from\_state symbol to\_state.
* **save\_to\_files()**
  + Saves the automaton’s components to separate files:
    1. states.txt: List of states.
    2. transitions.txt: List of transitions in the format: from\_state symbol to\_state