Automata Theory Project

28-feb-2022

Project Members

Kashif Hussain 10682

Usama Zulfiqar 10686

Raja Hassan Ali 10689

Project Topic: Simulation of Deterministic Finite State Automaton

Project Description:

A "Finite State Machine" (abbreviated FSM), also called "State Machine" or "Finite State Automaton" is an abstract machine which consists of a set of states (including the initial state and one or more end states), a set of input events, a set of output events, and a state transition function. A transition function takes the current state and an input event as an input and returns the new set of output events and the next (new) state. Some of the states are used as "terminal states". The operation of an FSM begins with a special state, called the start state, proceeds through transitions depending on input to different states and normally ends in terminal or end states. A state which marks a successful flow of operation is known as an accept state.

What do we want to make?

We want to recognize the meaning of very small sentences with an extremely limited vocabulary and syntax. In our model, User will input a sentence and our model will recognize whether it's Grammatically correct or incorrect. We are working on present tenses only, whose patterns will be like:

"Kashif is a boy" = Present Simple Tense.

"He is a boy"= Present Simple Tense.

"Kashif was boy"= error state.

"Kashif is working on a project" = Present Continuous

'Kashif has Done the project' = Present Perfect

We will recognize Present Simple, Present Perfect and Present Continuous tenses by using NLTK library.

Mathematical Model

A non-deterministic finite state machine or acceptor deterministic finite state machine is a quintuple (Σ,S,s_0,δ,F), where:

Σ is the input alphabet (a finite, non-empty set of symbols).

S is a finite, non-empty set of states.

s0 is an initial state, an element of S.

δ is the state-transition function: δ : S x Σ → S

(in a nondeterministic finite state machine it would be δ : S x Σ → ℘(S), i.e., δ would return a set of states). (℘(S) is the Power set of S)

F is the set of final states, a (possibly empty) subset of S.

What would our program be like?

The program should check the validity of that string and displays the states that is encountered by the input string.

Inputs to the program:

1. The number of states that the DFA will contain.

2. Here we take the Σ the set of alphabets constant as 0, 1 they might be altered or increased changing the course of the program.

3. Now we define the next states for every state for both alphabets 0 and 1; and if a state is final state or not.

4. q0 ∈ Q the initial state is taken as input and we have our NFA ready.

5. Next we take one string after another and check the validity of that string and the program shows the number of states the machine traversed for that input string.

Program output:

1. Acceptance of the input string.

2. Comment about the input string.

3. The state reached by the input string.