Author:

Justin Gong|JGON722|795439069

Aaron Zhang|TZHA934|123934091

Class name: FiniteStateTable

Overview

The FiniteStateTable class is used to represent a finite state machine(FST). It will create a two-dimensional array named FST. The class contain a built in structure for each cell of the FST, and can provide helpful functions that can let you get and set the value of the finite state table cells.

Example Code:

```
class Program
    {
        public delegate void Action();
        static void Main(string[] args)
        {
            /* 2 by 2 Finite State Table(FST) Example
             *
             *                   state
             *               | S1  | S2  |
             *          _____|_____|_____|
             *               | S2  |     |        Next State
             *      e     a  | X   |     |        Transition Action
             *      v   _____|_____|_____|
             *      e        |     |  S1 |        Next State
             *      n     b  |     |  Y  |        Transition Action
             *      t   _____|_____|_____|
             *
             *          the initial state of this table is S2
             */

            string[] States = { "S1", "S2" };
            string[] Events = { "a", "b" };
            void Action_X() { Console.WriteLine("Action_X"); };
            void Action_Y() { Console.WriteLine("Action_Y"); };

            var ExampleFST = new FiniteStateTable(Events, States, "S2");

            //set the cell of FST
            ExampleFST.SetNextState("S1", "a", "S2");
            ExampleFST.SetAction("S1", "a", Action_X);
            ExampleFST.SetNextState("S2", "b", "S1");
            ExampleFST.SetAction("S2", "b", Action_Y);

            //example of using the FST
            //getting the initial state of the FST
            string currentExampleFSTState = ExampleFST.GetInitalState();
            Console.WriteLine("The inital state is {0}", currentExampleFSTState);

            //if event "b" occured, Action_Y will be performed, and state will be updated
            ExampleFST.GetAction(currentExampleFSTState, "b")();
            currentExampleFSTState = ExampleFST.GetNextState(currentExampleFSTState, "b"); //update the current state
            Console.WriteLine("The current state is {0}", currentExampleFSTState);

            //if event "b" occurred, No action and state will not be updated
            ExampleFST.GetAction(currentExampleFSTState, "b")();
            currentExampleFSTState = ExampleFST.GetNextState(currentExampleFSTState, "b"); //update the current state
            Console.WriteLine( "The current state is {0}", currentExampleFSTState);

            //checking if a cell on the FST have a next state associated with it
            if (ExampleFST.IsValidCell(currentExampleFSTState, "b"))
            {
                Console.WriteLine("Cell State 'S1' and Event 'b' has a next state");
            }
            else
            {
                Console.WriteLine("Cell State 'S1' and Event 'b' doesn't have a next state");
            }

            /*The example displays the following
             *
             *      The initial state is S2
             *      Action_Y
             *      The current state is S1
             *      The current state is S1
             *      Cell State 'S1' and Event 'a' doesn't have a next state
             *
             */
        }
    }
```

Constructors

FiniteStateTable(string[] Inputs, string[] States, string initialState)	Initialises a new instance of a Cell_FST structure to the specific Inputs and States and InitialState

Fields

FST	cell_FST[,] Array. Two dimensions array where the first index of the array correlate to the event of the FST cell, and the second index of the array corollate to the state of the FST cell.
Inputs	String[] Array. Represents the string of Events of a FST. Private variable.
States	String[] Array. Represents the string of States of a FST. Private variable.
initialState	String. Represents the initial state of the Finite state machine, defaults to first element of State array when given invalid input or not set. Private variable.
isEmpty_StatesNInputs	bool. Whether the instance of the Finite State Table contain the Inputs and States array. Private variable.

cell_FST Fields

isEmpty_nextState	bool. Whether the next state of a cell is empty, default to true.
isEmpty_action	bool. Whether the next action of a cell is empty, default to true.
nextState	string. The next state of a cell on the FST
action	Action Delegate. The action associated with state transition of a cell

Input Variables

state	string. The state of the FST cell
input	string. The event of the FST cell
nextState	string. The next state of the FST cell
action	Action Delegate. Action the machine will do

Public Functions

SetNextState(string state, string input, string nextState)	Sets nextState field of the FST cell
GetNextState(string state, string input)	Returns a string contain the next state of a FST cell, if there is no next state for the particular cell or invalid cell, return the input parameter state.
SetAction(string state, string input, Action action)	Sets the Action to perform between state transition of a particular Finite State Table cell.
GetAction(string state, string input)	Returns an Action delegate associated with the state transition. Returns Action_NULL() if there are no the FST cell doesn't have an action or invalid input.
SetInitialState(string state)	Sets the initial state of the Finite State Table. if specified state is invalid will set the value to the first member of the States array. //Might need to add a flag to see if initial state has been set
GetInitialState()	Returns a string, the initial state of the Finite State Table as a string
IsValidCell(string states, string input)	Returns a bool variable based on if a particular cell of the FST have a next state associated with it. Return false if the cell does not have a next state associated with it.

Private Functions

SetIsEmpty()	Sets the isEmpty and isTransistionCondition variables to avoid index out of range.
GetCell(int stateIndex, int inputIndex)	Returns a cell_FST structure(event,state) of a specific cell in the cell table. Return an empty FST cell if given invalid input parameter.
SetNextState(int stateIndex, int inputIndex, int nextStateIndex)	Sets the next state in the finite state machine cell. Will take the specific index of the state, input and next state.
GetNextState(int stateIndex, int inputIndex)	Returns a string, the next state in the Finite State Machine cell as a string. Return the input parameter state if the input is invlid.
SetAction(int stateIndex, int inputIndex, Action action)	Sets the action associated with the state transition of a certain cell of the Finite State Machine.
GetAction(int stateIndex, int inputIndex)	Returns the Action delegate associated with the state transition of a cell in the FST. Return Action_NULL() if there are no the FST cell doesn't have an action or invalid input.
GetInputIndex(string input)	Returns an int, the index of the input event in the cell as an Integer in the Input array. Return -1 if input is not a member of the Input array.
GetStateIndex(string state)	Returns an int, the index of the state event in the cell as an integer in the State array. Return -1 if input is not a member of the State array.
GetStateTransationIndex(string state)	Returns an int, the index for a specified state in the Input array, only used when the transition condition for at least one cell of the FST is a state-based state transition rather than a input based state transition. Return -1 if input parameter state is not a member of the Input array.
Action_NULL()	Default for transitions that does not have an action associated.