Test Report

construct()

Checking: Whether creating an FSM works. (Should not return null)

Result (Success): Constructing an FSM does not cause null.

nodeConstructor()

Checking: Creating a node with a label

Result (Success): A node is created with the desired label

addNode()

Checking: Ability to add a node to the FSM

Result (Success): Adding a node to an empty FSM and not-empty FSM causes it to exist in the FSM.

setStart()

Checking: Setting the start node in FSM

Result (Success): Before setting the start node, the start node is null (does not exist). Setting the start, makes the desired node the start node.

changeState()

Checking: Changing the acceptability of a node

Result (Success): The default state of a node is false.

edgeConstructor()

Checking: Creating an edge between nodes

Result (Success): The edge between two nodes is created with the desired label.

addEdge()

Checking: Ability to add edge to the FSM.

Result (Success): Adding a node to causes it to exist with a label.

setNodeLabel()

Checking: Changing the label of a node

Result (Success): Setting the label of a node to something else causes the label to change to the current one.

setEdgeLabel()

Checking: Changing the label of an edge.

Result (Success): Setting the label of an edge to something else causes the label to change to the current one.

getMachine()

Checking: The FSM machine stores the information (nodes and edges) correctly.

Result (Success): The machine shows edges and nodes accurately.

Save and Load

After implementing saving and loading, we were able to save the Finite State Machine as a human readable file and load it back from the saved file. The trials showed that both saving and loading was working properly, as wanted.

Traversal

Traversing the Finite State Machine was implemented to visit every node in the machine. The trials on traversal showed that traversing the machine was giving the correct results.