# Testing

## Test Plan **Note: All Turing machines used in testing can be found in the Appendix in page 309, under ‘Turing Machines used in Testing’.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Interface During Editing Tests** | | | |  |  |  |
| **Test ID** | **Description** | **Inputs** | **Expected Result** | | | |
|  | UI loads when program is run. | N/A | The UI will open in a new window and will appear identical to the design shown in the “Form1.cs[Design]” mode. | | | |
|  | Press ‘New Machine’. | Press ‘New Machine’ | A new tab page called “machine1” will open in the editing box and a new blank tape will be created. | | | |
|  | Press ‘Delete Machine’. | Press ‘New Machine’, Press ‘Delete Machine’ | The new machine will be opened, and when closed the tab page will be deleted, and so will the tape at the top of the page. | | | |
|  | Press ‘Open Machine’. | Press ‘Open Machine’,  Select ‘interface\_open.txt’ | The machine of the new tab will be set to 0, one tape will be made, the contents of which will be “1011+1100”, its initial state will be “set”, and the text editor will contain a new tab with the code required to add two binary numbers together. | | | |
|  | Test ‘save machine’. | Open a new machine,  Machine = 2. Read Only Machine  Tape = “10”  Pointer = 3  Initial State = “a”  Code = “a 0 1 r a”  and name ‘interface\_save.txt’ | A new text file called ‘interface\_save.txt’ will be made, containing:  “1  10  3  a  a 0 1 r a” | | | |
|  | Test ‘duplicate machine’ works. | Open ‘interface\_open.txt’, Duplicate Machine. | A new tab will be created, where all of the data from the ‘test’ machine will be carried over to the new machine. | | | |
|  | Change the machine to make sure that all values correctly change to match the newly selected machine. | Open ‘interface\_open.txt’, create new machine and switch back to ‘interface\_open.txt’ tab, then back to the new machine. | The new machine created after opening the test machine will be blank. When the tab is changed back to ‘test’, all of the data from the test machine will be set into the fields. Then, when going back to the new machine, the fields will be set blank. | | | |
|  | Make sure ‘Rename Machine’ works. | Open a new machine, right click and select rename. Rename the file to ‘renamedMachine’. | The name of the tab will now be renamed to ‘renamedMachine’. No other information about this machine should be changed in this process. | | | |
|  | Use the shortcut ‘Ctrl-N’ to create a new machine. | Ctrl-N | A new machine should open, exactly as it does in test 02. | | | |
|  | Clicking “Quit” in the taskbar closes the program. | Open Program, File > Quit | The program should close once this option is pressed. | | | |
|  | Test initial state field update. | Open ‘interface\_open.txt’ and change last line to “c \_ \_ odd”. | The initial state combo box should add the state “c” to the list of states to start from. | | | |
|  | Test initial state field becomes blank when active state is removed. | Open ‘interface\_open.txt’ and remove the code from the editor. | The initial state field should no longer read “a” and should not contain any states when the list is expanded. | | | |
|  | Editing the tape updates the position of the pointer. | Ctrl-N, type ‘pointerTest’ in tape | The pointer will follow the first key of the tape, in this case, ‘p’. | | | |
|  | Test left pointer button. | Open “interface\_move”, press left pointer button | The pointer will move the space of one character to the left. | | | |
|  | Test right pointer button. | Open “interface\_move”, press right pointer button | The pointer will move the space of one character to the right. | | | |
|  | Changing the machine type to a multi-tape machine will add the option to add a new tape. | Ctrl-N, set machine to “multi-tape machine” | The “Add New Tape” button will appear below the Machine combo box. | | | |
|  | Changing the machine type to a multi-track machine will add the option to add a new tape. | Ctrl-N, set machine to “multi-track machine” | The “Add New Tape” button will appear below the Machine combo box. | | | |
|  | Selecting a non-multiple tape machine after selecting a multiple tape machine hides the ‘add’ button. | Ctrl-N, set machine to 3, set machine to 1 | The “Add New Tape” will become hidden when the machine becomes a Read-Only Machine. | | | |
|  | Test adding new multi-tape tape. | Ctrl-N, set machine to 3, press Add New Tape | A blank tape will be added, with the option to close the tape showing and no initial state combo box. The height of the program will increase to accommodate the new tape. | | | |
|  | Test adding new multi-track tape. | Ctrl-N, set machine to 4, press Add New Tape | A blank tape will be added, with the option to close the tape showing as well as the initial state combo box. The height of the program will increase to accommodate the new tape. | | | |
|  | Check tapes can be edited independently of each other. | Ctrl-N, set machine to 3, Add New Tape, Tape 1 = “101”, Tape 2 = “010” | The first tape will read “101” and the second will read “010”. When one tape is being edited, the value of the other does not change at all. | | | |
|  | Check tape pointers can be moved independently of each other. | Open “interface\_multiMove”, move tape 1 pointer left, move tape 2 pointer right | The first tape’s pointer will move one key to the left and the second’s one to the right. Neither pointer if affected when the other pointer’s position is changed. | | | |
|  | Check tape pointers can be moved independently of each other (reversed direction). | Ctrl-N, set machine to 3, Add New Tape, move tape 1 pointer right, move tape 2 pointer left | The first tape’s pointer will move one key to the right and the second’s one to the left. Neither pointer if affected when the other pointer’s position is changed. | | | |
|  | Check initial states can be edited independently of each other. | Open “interface\_multiTrackStates”, select “b” for tape 1, “a” for tape 2. | The first tape’s initial state will become “b” and tape 2’s state will stay the same. When tape 2 is changed to “a”, it remains this value and tape 1’s initial state will remain “b”. | | | |
|  | Check current state and step count labels are shown for every tape when machine is a multi-track. | Ctrl-N, machine = 4, add tape | The standard “steps” and “current state” labels are hidden. Then current state and step count labels are shown on each tape, to the left of the initial state combo boxes. | | | |
|  | Check these individual labels are removed when switching away from a multi-track machine. | Ctrl-N, machine = 4, add tape, machine = 3 | The individual labels are hidden and the standard labels are shown again. | | | |
|  | Check that selecting a Universal Turing Machine shows the option to generate the transition code for the machine. | Ctrl-N, machine = 5 | The button “Generate Universal Code” should appear in the options pane. | | | |
|  | Check that selecting a Universal Turing machine forces the user to have three tapes with no option to close them | Ctrl-N, machine = 5 | The machine should have three tapes, with no close button, no tape-specific labels and no initial state options for all but the first tape. | | | |
|  | Pasting to the text editor while an image file is on the clipboard removes the image from the clipboard. | Search “turing machine” on Google. Copy the first image and its caption, Paste into blank machine. | The information about the picture – its title, the website and URL it came from, should be pasted in. The image itself, however, should not be pasted into the text editor. | | | |
|  | The text editor should be able to undo and redo edits made in the control. | New Machine, type “a 0 0 r a”. Press Ctrl-Z, press Ctrl-Y | After pressing Ctrl-Z, the text written into the editor should be removed. After Ctrl-Y is pressed, the text will be returned to the editor. | | | |
|  | Placing a large number of characters on the tape causes the tape’s (and read/write head’s) size to decrease to fit all characters on the screen. | Open “interace\_longTape.txt”, add 5 1’s to the tape. | After placing the first character, the tape’s size should reduce to the next font size. The size of the read/write head should also decrease to stay in line with the first character in the tape. | | | |
|  | Removing characters causes the tape to increase in size up to the original maximum font size. | Take tape from Test 33, remove 5 characters from tape. | The first four characters removed should not cause the tape’s font size to be increased. After removing the fifth character, the tape’s font size should increase, as the tape is narrow enough now to be below the maximum width for the tape. | | | |
| **Turing Machine Logic Tests** | | | |  |  |  |
| **Test ID** | **Description** | **Inputs** | **Expected Result** | | | |
|  | Write a character beyond the left bound of the tape. | Open “dtm \_write\_left.txt”, Run | The final tape should contain a ‘3’ at the left end of the original tape. No other characters in the tape will be edited. | | | |
|  | Insert a character within the bounds of the tape. | Open “dtm \_write\_middle.txt”, Run | The machine should replace the fifth character in the tape with a ‘3’. The final tape should read “11013101” | | | |
|  | Add a character to the end of the tape. | Open “dtm \_write\_right.txt”, Run | The final tape should contain a ‘3’ at the right end of the original tape. No other characters in the tape will be edited. | | | |
|  | Make sure that the read/write head correctly moves to the right when the command specifies this. | Run “dtm\_right.txt” | The machine should move the read/write head one space to the right and then halt. | | | |
|  | Make sure that the read/write head correctly moves to the left when the command specifies this. | Run “dtm\_left.txt” | The machine should move the read/write head one space to the left and then halt. | | | |
|  | Make sure that the read/write head correctly does not move at all when the command specifies this. | Run “dtm\_stay.txt” | The machine should halt without moving the read/write head. | | | |
|  | Ensure that the final state of the machine after completing the function is consistent with what is written in the transition function. | Step “dtm\_state.txt” once. | After following the transition function, the state of the machine should change from ‘a’ to ‘b’. | | | |
|  | Test that the Non-Deterministic machine correctly chooses a random function out of the matching transition functions given to it. | Run “ndtm\_generic.txt” 5 times. | This machine should travel across the tape from left to right, setting the ones and zeroes on the tape to a one or zero randomly, halting at the right end. Each time, the outcome of the machine should be different without changing any parameters of the machine. | | | |
| **Turing Machine Validation Tests** | | | |  |  |  |
| **Test ID** | **Description** | **Inputs** | **Expected Result** | | | |
|  | Check that a machine cannot be run if one of its tapes has a blank or invalid initial state. | Load “dtm\_generic.txt”, remove initial state, Run. | An error message telling the user to set an initial state should appear. | | | |
|  | When an issue occurs in a specific line, the line number is given by the error message displayed. | Run “validate\_lineNumber.txt” | As the line in which the error occurs in this machine is the first line, the line number in the error message should be 1. | | | |
|  | Ensure a machine cannot be run if two transition functions for the same state use the same read key (except for Non-Deterministic machines). | Run “validate\_sameRead.txt” | An error message telling the user that they cannot have two functions using the same state and read key should appear. | | | |
|  | Make sure that a machine is rejected if there are any duplicate transition functions. | Run “validate\_duplicate.txt” | An error message telling the user that they cannot have duplicate functions should appear. | | | |
|  | Test that a transition missing any one of the required fields for that machine variant is not accepted. | Run “validate\_fieldMissing.txt” | An error message telling the user that they have too few fields in their transition function should appear. | | | |
|  | Test that a transition function requiring too many fields for that machine variant is not accepted. | Run “validate\_fieldExtra.txt” | An error message telling the user that they have too many fields in their transition function should appear. | | | |
|  | Test that a transition function where the move key is not ‘r’, ‘l’ or ‘\_’. | Run “validate\_moveWrong.txt” | An error message telling the user that their move key is not valid will appear. | | | |
|  | Test that a transition function is rejected in a Multi-Tape machine if the number of read, write, or move keys do not match the number of tapes in the machine. | Run “validate\_tapeMissing.txt” | An error message specifying that there is an issue with one of their keys should be shown. | | | |
|  | Make sure a machine cannot be run if any of its tapes contains the default case character. | Run “validate\_defaultKey.txt” | An error message telling the user that they cannot have the default case character in their tape. | | | |
|  | Make sure that a Universal Turing machine is not usable if the instruction tape is not laid out correctly (five fields for each function followed by a ‘|’ character to separate each). | Run “validate\_utm\_instructionTape.txt” | An error message saying that their instruction tape is invalid. | | | |
|  | Reject a Universal Turing machine that contains more than one character in its current state tape. | Run “validate\_utm\_stateTape.txt”, make state tape blank and run. | An error message saying that their state tape is not of the right length. | | | |
|  | Ignore lines that contain “//” at the beginning of lines when checking the program. | Run “validate\_commented.txt” | A transition function with an error in them that contains a “//” at the beginning should not be included in validation. | | | |
|  | Ignore lines that contain “//” at the beginning of lines when running the program. | Run “validate\_commented.txt” with tape “0”. | The first transition function should be ignored when compiling. This causes the program to halt without performing any steps, as there are no matching transition functions to use. | | | |
|  | A single tape Turing machine cannot be run if more than one tape is present. | Open “validate\_dtmTapes.txt”, set machine to 0, Run. | An error message should appear, telling the user that they cannot use more than one tape in this machine. | | | |
| **Interface During Operation Tests** | | | |  |  |  |
| **Test ID** | **Description** | **Inputs** | **Expected Result** | | | |
|  | Ensure that the “TrimEdges” procedure only trims the spaces outside the bounds of the pointer. | Ctrl-O, open “dtm\_trimedges\_leftBound.txt”, Run | The final tape should add two blank characters to the left of the tape, resulting in a final tape of ‘\_\_10101’. | | | |
|  | Check for the other direction. | Ctrl-O, open “dtm\_trimedges\_rightBound.txt”, Run | The final tape should add one blank character to the right of the tape, resulting in a final tape of ‘10101\_’. | | | |
|  | Test that “TrimEdges” will remove unneeded blank characters from the edges of the tape. | Ctrl-O, open “dtm\_trimedges\_left.txt”, Run | The final tape should remove the additional blank space added to the left of it during operation. | | | |
|  | Check for the other direction | Run “dtm\_trimedges\_right.txt” | The final tape should remove the additional blank space added to the right of it during operation. | | | |
|  | Make sure the machine cannot be operated once halted. | Ctrl-O, open “dtm\_generic\_halting.txt”, Run, Reset | Once the machine has run, the ‘run’ and ‘step’ options will become unusable. Once pressing reset, this will be undone. | | | |
|  | Make sure the machine cannot be edited once in operation. | Open “dtm\_generic.txt”, Step | Once compiled, the only options that will be available to use will be the ‘reset’, ‘run’ and ‘step’ buttons, as well as the speed control slider. Text fields, such as the tape and pointer, will not become disabled but they will not be able to be written to. | | | |
|  | Make sure the machine cannot be edited once running. | Open “dtm\_generic.txt”, Run | When running, the only options available will be the ‘pause’ and speed controls. Text fields, such as the tape and pointer, will not become disabled but they will not be able to be written to. | | | |
|  | Changing the speed of the machine should temporarily pause the machine. | Open “dtm\_generic.txt”, Run, Click and drag speed slider. | When choosing a new speed to operate the machine at, the machine should pause to prevent any errors occurring during this process. Holding the speed slider in one position should not cause the machine to pause, as the speed value is not changing at this time. | | | |
|  | After a single-tape machine halts, the debug text box produces information about the tape’s final contents and the machine’s final state and steps taken. | Run “debug\_single.txt” | The debug text box’s contents should be updated. An indication that the machine has halted should be shown, followed by the contents of the tape at the end of its operation, the final state and the number of steps taken by the machine to get to this state. | | | |
|  | After a multi-tape machine halts, the debug text box produces information about the final contents of each tape and the machine’s final state and steps taken. | Run “debug\_multiTape.txt” | The debug text box’s contents should be updated. An indication that the machine has halted should be shown, followed by a numbered list of the contents of the tapes at the end of the machine’s operation. Following this should be the final state of the machine and how many steps taken by the machine to get to this state. | | | |
|  | After a multi-track machine halts, the debug text box produces information about each tape’s final contents, state and step count. | Run “debug\_multiTrack.txt” | The debug text box’s contents should be updated. An indication that the machine has halted should be shown, followed by a numbered list of the contents of the tapes at the end of the machine’s operation. Next to each tape should be the state that tape halted in and the number of steps it took before it halted. | | | |
| **Default Case Functions Tests** | | | |  |  |  |
| **Test ID** | **Description** | **Inputs** | **Expected Result** | | | |
|  | Test the default case (‘\*’) character doesn’t write to the tape. | Open “dtm\_generic\_halting.txt”, Run | The machine should pass over the tape without changing any of the characters in it. | | | |
|  | Test that the default case character will work on any character when used as the read key. | Open “dtm\_defaultcase.txt”, Run | The machine should turn all of the numbers in the tape to ‘0’s. | | | |
|  | Test that the default case character will not work on a character that has a command written for it. | Open “dtm\_defaultcase\_override.txt’,  Run | The machine should turn all of the numbers in the tape to ‘0’s, except for the letter ‘4’, which should remain the same. | | | |
|  | Test that the default case character can accept multiple override commands. | Open “dtm\_defaultcase\_override  \_multiple.txt”, Run | The machine should produce the tape “0004600000”, with override commands taking place at 2, 4, and 5. | | | |
|  | Test that the default case character can accept multiple override commands in Read-Only machines. | Open “rom\_defaultcase\_override  \_multiple.txt”, Run | The machine’s tape should remain the same, and the machine should halt in the state “odd”. | | | |
|  | Test that a command containing all default case characters as its write key doesn’t write to any tape in a multi-tape machine. | Run “mtam\_defaultcase\_write  \_none.txt” | The machine will move the read/write head through the tapes without editing any of its contents. | | | |
|  | Test that a command containing at least one (but not all) default case characters in its write key doesn’t write to a multi-tape, for the tapes where the default case is used. | Run “mtam\_defaultcase\_write  \_patial.txt” | The machine will not write to the top or bottom tapes, and move through them without editing it. The middle tape’s 0s will be replaced with 1s as it moves along it. | | | |
|  | Test that the full default case function will not work when a matching fully-defined function is present. | Run “mtam\_defaultcase\_override  \_full.txt” | The first transition function will be ignored and the middle one will be executed each time; the top tape’s 3s will be replaced with 4s. | | | |
|  | Test that the full default case function will not work in a Multi-Tape if there is a matching partially-defined function present. | Run “mtam\_defaultcase\_override  \_partial.txt” | The first transition function will be ignored and the middle one will be executed each time; the top tape’s 3s will be replaced with 4s. | | | |
|  | Test that a partially-defined function with a lower priority than another will not be accepted, and the higher priority function will be accepted. | Run “mtam\_defaultcase\_override  \_highPriority.txt” | The third transition function should be executed, turning the ‘0’s in the middle tape to ‘3’s and the ‘3’s in the top tape to ‘4’s. | | | |
|  | Test that a matching partially-defined function will not be used if there is a matching fully-defined function present. | Run “mtam\_defaultcase\_override  \_both.txt” | Instead of writing 4s to the top tape, which is defined by the partial function, each tape’s contents will be incremented by one; the bottom tape will become 2s, the middle, 1s, and the top, 4s. | | | |
|  | Test that a command containing the default character in its write key will not write to a tape in a Multi-Track Turing machine. | Run “mtrm\_defaultcase\_write  \_none.txt” | The read/write heads will travel from right to left across each tape. None of the contents of the tapes will be changed. | | | |
|  | Test that the machine will execute a default command if the key on the tape does not have a command written for it. (Multi-Track Machine) | Run “mtrm\_defaultcase.txt” | Because the current state does not contain a matching transition function, the default function will be executed and the 2s on the tapes will be changed to 1s. | | | |
|  | Test that the machine will use separate override character lists for each tape in a Multi-Track machine. | Run “mtrm\_defaultcase  \_seperateLists.txt” | All of the tapes have different states. These states contain defined functions with different read keys to each other. The top tape does not have a defined function for ‘1’ in its current state. Therefore, it must resort to the default case for its state. The same is true for the bottom tape. The middle tape’s state does have a defined function for ‘1’, and should execute this. | | | |
| **Preferences Tests** | | | |  |  |  |
| **Test ID** | **Description** | **Inputs** | **Expected Result** | | | |
|  | Test that changing the minimum speed value updates the slider. | Settings, set Min to 200, OK. | The far-right side of the slider will now update to be 200, as reflected by the counter above the slider. | | | |
|  | Test that changing the maximum speed value updates the slider. | Settings, set Max to 500, OK, slide to left. | The far-left end of the slider will now read 500. | | | |
|  | Test that the minimum speed value cannot be larger than the maximum. | Settings, set min to 1000, set max to 50, OK. | An error message will appear, telling the user this is not allowed. The settings menu will stay up to let the user choose an appropriate value. | | | |
|  | Test that the two speed values cannot be equal. | Settings, set min to 1000, set max to 1000, OK. | An error message will appear, telling the user this is not allowed. The settings menu will stay up to let the user choose an appropriate value. | | | |
|  | Make sure that the preferences won’t be accepted if the default case character given is the blank character (‘\_’). | Settings, set default key to ‘\_’. | An error message will appear, telling the user this is not allowed. The settings menu will stay up to let the user choose an appropriate value. | | | |
|  | Make sure that the preferences won’t be accepted if the default case character given is the blank character (‘ ’). | Settings, set default key to ‘ ’. | An error message will appear, telling the user this is not allowed. The settings menu will stay up to let the user choose an appropriate value. | | | |
|  | Make sure that the preferences won’t be accepted if the default case character field does not have anything in it. | Settings, set default key to nothing. | An error message will appear, telling the user this is not allowed. The settings menu will stay up to let the user choose an appropriate value. | | | |
|  | Make sure the program runs correctly using a custom default case key. | Settings, set default key to ‘8’, run “defaultkey\_custom.txt” | The program should correctly add the unsigned binary integers 1011 and 1100 together to calculate 10111. | | | |
|  | Test that editing the preferences saves them to the “config.ini” file in the project’s Debug folder. | Settings, set max speed to 50, set full speed increment to 75, OK. Open config.ini. | After editing these two variables, the config file should contain a 50 in its first line, and a 75 in its fourth line. | | | |
|  | Test that reopening the program correctly loads in the custom preferences. | Open program, Settings. | The edited values from Test 89 should now be present in the input fields; Max = 50 and Increment = 75. | | | |
|  | If the “config.ini” file is not present, the program should use default values and create a new config file. | Delete “config.ini”, Open program. | When opening the program, a new “config.ini” file should be created. The values should be:   * Maximum Speed: 1000 * Minimum Speed: 1 * Default Case Character: ‘\*’ * Full Speed Interval: 50 | | | |
| **FSM Representation Tests** | | | |  |  |  |
| **Test ID** | **Description** | **Inputs** | **Expected Result** | | | |
|  | The program correctly lays out each state as a node in the FSM using a depth-first algorithm based on the order of occurrence in the text editor.  The program correctly identifies halting states as such, and changes the image of the state in the FSM to reflect this (a double circle). | Open Palindrome Example, Display as FSM. | For this test, the edges will be ignored. The nodes should display correctly, as described below.  The initial state ‘a’ should branch off to three nodes: ‘a1’, ‘a0’ and ‘accepthalt’. These three nodes are evenly spaced and the middle node is vertically in line with the initial state. Beyond the ‘a1’ and ‘a0’ states should be their respective check states. As this is the only connecting state, they will be vertically in line with their parent state. Because ‘a1check’ comes first in the text editor, it is from this node that the ‘acceptreset’ and ‘rejecthalt’ nodes will be placed. The nodes ‘accepthalt’ and ‘rejecthalt’ will have the halting state graphic instead of the standard node graphic, which consists of an extra ring around the state. | | | |
|  | Make sure that the initial state of the machine is placed at the far left of the screen, with an arrow pointing into it. | Open Addition Example, Display as FSM | The initial state, ‘set’, should be placed vertically in the middle of the screen, close to the left edge of the screen. An arrow should come from beyond the left end of the screen and point to the state. | | | |
|  | Draw a straight-line transition correctly, where the label is placed correctly. | Display “fsm\_straightLine.txt” as FSM | A straight, flat line should come from state ‘a’ and point towards state ‘b’. Underneath the line should be a label identifying that the transition will be caused when reading a 1, and will result in a change to 0 and a right move. | | | |
|  | Draw a branch from one state to two states, making sure all lines and labels are displayed correctly. | Display “fsm\_branchingLine\_2.txt” as FSM | A straight line should point diagonally upwards from ‘a’ to ‘b’, and another should point diagonally downwards from ‘a’ to ‘c’. The label for a-b should read “1 | 0 >” and be below the line. The label for a-c should read “0 | 1 >” and be above the line. | | | |
|  | Test that creating a branch from one state to three states correctly draws all lines and labels. | Display “fsm\_branchingLine\_3.txt” as FSM | A straight line should point diagonally upwards from ‘a’ to ‘b’. A straight, flat line should point from ‘a’ to ‘c’, and another line should point diagonally downwards from ‘a’ to ‘d’. The label for a-b should read “1 | 0 >” and be below its line. The label for a-c should read “0 | 1 >” and be below its line. The label for a-d should read “\_ | \_ >” and be above its line. | | | |
|  | Test that a transition that leads to the same state it came from correctly displays as a looping Bezier curve, with the label outside the loop. When the state is level with the initial state, the loop should go below the node. | Display “fsm\_looping\_middle.txt” as FSM | A curved line should come out from state ‘a’, loop down and back up to state ‘a’. Below this line, at its turning point, should be a label containing information about the transition function. | | | |
|  | Repeat test 74, with the looping state above the initial state to make sure the loop goes above the node. | Display “fsm\_looping\_up.txt” as FSM | A curved line should come out from state ‘b’, loop up and back down to state ‘b’. Above this line, at its turning point, should be a label containing information about the transition function. | | | |
|  | Repeat test 74, with the looping state below the initial state to make sure the loop still goes below the node. | Display “fsm\_looping\_down.txt” as FSM | A curved line should come out from state ‘c’, loop down and back up to state ‘c’. Below this line, at its turning point, should be a label containing information about the transition function. | | | |
|  | Test that a transition that leads backwards in the graph will create an arcing Bezier curve from the bottom of the current state to the bottom of the final state (when the sending state is aligned with the destination state in the y-axis). | Display “fsm\_arcing\_middle.txt” as FSM | A curved line should come out from state ‘b’, travel downwards and then travel back up to hit state ‘a’. The information about this transition should be displayed approximately halfway along the line and should be outside the curve. | | | |
|  | Repeat test 77, with the sending state above the destination state, to show that when this is true the arc will travel up instead of down. | Display “fsm\_arcing\_up.txt” as FSM. | A curved line should come out from state ‘b’, travel upwards and then travel back down to hit state ‘a’. The information about this transition should be displayed approximately halfway along the line and should be outside the curve. | | | |
|  | Repeat test 77 with the sending state below the destination state to show that the arc will still travel down. | Display “fsm\_arcing\_down.txt” as FSM. | Same expected result as Test 100. | | | |
|  | Make sure that read-only machines only contain read-key and move-key information in the labels of the transition lines. | Open Parity Example, Display as FSM. | The resulting diagram should contain labels on edges that only contain information about the read-key of the function and its move key. | | | |
|  | Ensure that the labels of diagrams for multi-tape machines contain the read-keys, write-keys, and move-keys for every tape in the transition function. | Display “fsm\_multitape.txt” as FSM. | The labels should have read key, write key and move key fields of length 3, with the values correctly corresponding to the transition function in the text editor. | | | |
|  | Make sure that Non-Deterministic machines are not allowed to be represented as a diagram when attempted. | Display “fsm\_nondeterministic.txt” as FSM. | An error message should appear, telling the user that they cannot display this machine as an FSM. | | | |
|  | Make sure that Universal Turing machines are not allowed to be represented as a diagram when attempted. | Display “fsm\_universal.txt” as FSM. | An error message should appear, telling the user that they cannot display this machine as an FSM. | | | |
|  | When two transition functions come from the same state and end in the same state, the two should be collected and put into one transition label with one edge. | Display “fsm\_overlap.txt” as FSM | The new label for this transition function should consist of lists for the read, write and move keys. The order of each item corresponds to the other lists. In this example, the read keys are [0,1], the write keys [1,0] and the move keys [>,>]. Therefore, when a 0 is read, 1 is written and the read/write head moves right. When a 1 is read, 0 is written and the read/write head moves right. | | | |
|  | When two transition functions come from the same state and end in the same state, and neither states result in the character changing, the write key section of the label should be blank. | Display “fsm\_overlap\_noChange.txt” as FSM. | The resulting diagram should contain a label to its edge which contains the two read keys that follow this transition, 0 and 1, and the output side should contain a right arrow and left arrow corresponding to the two transition functions. No write key should be displayed in the label. | | | |
|  | When two transition functions come from the same state and end in the same state, and all of these states write the same character to the tape, the write key section should contain only one occurrence of the character. | Display “fsm\_overlap\_sameWrite.txt” as FSM. | The label for this transition should contain only one instance of the key that is to be written to the tape. | | | |
|  | When two transition functions come from the same state and end in the same state, and all of these states move the tape in the same direction, the move key section should contain only one occurrence of the character. | Display “fsm\_overlap\_sameMove.txt” as FSM. | The label for this transition should only contain one instance of the direction to be moved along the tape. | | | |
|  | Make sure the results in Test 104 and Test 105 do not interfere with each other. | Display “fsm\_overlap\_both.txt” as FSM. | The label for this transition should contain only one instance of the key that is to be written to the tape, and only one instance of the direction to be moved along the tape. | | | |
|  | The arcing Bezier curves from transition functions should not overlap or interfere with looping Bezier curves, when the sending states are in the Y-Axis plane. This neatens the image and makes sure that it is more legible. | Open Addition Example, Display as FSM. | To ensure clarity when looking at the FSM, the lines should overlap as little as possible. When this Turing machine is displayed, the arcing transition that resets the machine back to the ‘set’ state should travel over all of the looping transitions that are in its way. | | | |
|  | Scrolling along the window horizontally refreshes the edges of the diagram without causing screen-tearing. | Open Subtraction Example, turn off maximised view, Display as FSM. | When the FSM goes outside the bounds of the window, the scroll bar at the bottom allows the user to see the rest of the FSM. When scrolling, the edges should refresh to minimise the errors that occur during this process. | | | |
| **Universal Turing Machine Tests** | | | |  |  |  |
| **Test ID** | **Description** | **Inputs** | **Expected Result** | | | |
|  | Using ‘Convert to Universal’ correctly transfers the information from a Deterministic machine onto the tape of a Universal machine. | Open “dtm\_utm.txt”, Convert to Universal | The resulting set of tapes on the UTM should consist of (from top to bottom):   * The original DTM tape, * The initial state of the DTM (a). * The transition functions from the DTM, with the states simplified to single characters and each function separated by a ‘|’ character. | | | |
|  | Pressing ‘Generate Universal Code’ produces a set of transition functions to compute the DTM. | Generate Universal Code for converted “dtm\_utm.txt” | This process should produce a long set of transition functions and place them in the text editor. The initial state of the machine should be set to “q0”. This list would be unrealistic to check, so the most effective way to verify its function would be to test if it works (see Test 111). | | | |
|  | Running the generated code will correctly emulate the DTM on the tape. | Run UTM for “dtm\_utm.txt”. | As a DTM, this machine is designed to simply travel from the left end of the tape to the right, then travel back and halt once it reaches the left end again. If the machine has been translated correctly, the same should occur in the UTM, just with a lot more steps. | | | |
|  | Altering the transition functions on the instruction tape of the UTM (while not changing the alphabet or states lists) will still yield a correct result when running using the same UTM code as before. | Get converted “dtm\_utm.txt” machine, change instruction tape to “dtm\_utm\_newTape.txt”, run. | As a DTM, the new instruction set is designed to go through a sequence of ones and zeroes, invert them (change one to zero and vice versa), then travel back to the beginning of the tape hand halt. When using the new tape, without re-generating the transition function set, the UTM should compute the same result. This proves that the UTM is truly universal, and doesn’t just puppeteer the original machine. | | | |
|  | Read-Only machines can be converted to Universal Turing machines. | Convert “rom\_utm.txt” to Universal. | The resulting tapes on this UTM should be the same as in test 90. | | | |
|  | If there are more than 52 individual states in a machine, it cannot be converted to a Universal Turing machine. | Convert “utm\_53states.txt” to Universal. | An error message will appear, stating that the user cannot use that many states in a UTM. | | | |
| **Miscellaneous Tests** | | | |  |  |  |
| **Test ID** | **Description** | **Inputs** | **Expected Result** | | | |
|  | Choosing an example from to open will successfully load it into the program. | Open Addition Example | This should load in the Turing machine located in the Debug folder from the file “add.txt”. | | | |
|  | Selecting the ‘info’ option from the menu will load a window that gives information about the program. | Select “Info” in menu | Pressing this option should open a window containing information about the program and how to effectively use it. | | | |
|  | Test that ‘full speed’ mode cannot be toggled while the program is running. | Open “dtm\_generic.txt”, Run machine. | Once the machine is running, the “Run at Full Speed” checkbox should be greyed out and inaccessible. | | | |
|  | ‘Full Speed’ mode can be toggled when the machine is paused. | Open “dtm\_generic.txt”, Step machine. | The machine is compiled but paused, and the “Run at Full Speed” checkbox should be usable. | | | |
|  | Test that ‘full speed’ mode operates at the correct increment. (default value) | Open “dtm\_generic.txt”, Run machine at full speed. | When the machine runs, the number of steps taken should increment in steps of fifty. | | | |
|  | Test that ‘full speed’ mode operates at the correct increment. (custom value) | Settings, set Interval to 500, Run “dtm\_generic.txt” at full speed. | When the machine runs, the number of steps taken should increment in steps of five hundred. | | | |
|  | Ensure that using Full Speed mode improves the execution speed of machines. | Open Addition example, use a long tape. Run with and without Full Speed mode. | There will be additional code added for this test to time how long the machine is operating for. A breakpoint will be triggered when the program halts, and we should find that the time taken for the Full speed run was less than without Full Speed mode enabled. | | | |
|  | Turning on ‘full speed’ mode disables the ability to modify the speed value of the machine. | Check ‘Full Speed’ | The slider used to modify the speed of the machine should become greyed out and unusable. | | | |
|  | Turning off ‘full speed’ mode re-enables the ability to modify the speed value of the machine. | Uncheck ‘Full Speed’ | The slider used to modify the speed of the machine should become usable again. | | | |
|  | Make sure that ‘Clear Debug Box’ will make the debug box’s contents blank. | Run “dtm\_generic\_halting.txt”, Reset, Clear Debug Box | The button should remove the message displayed after the machine halts. | | | |
|  | Disabling the debug box causes it not to be written to after a machine halts. | Uncheck “Enable Debug Box”, Run “dtm\_generic\_halting.txt”. | After the machine halts, the debug box remains blank. | | | |
|  | Test that the “Line:” label in the options pane displays the line number currently selected in the text editor. | New Machine, add some text to the first line, add some text to the second line | When editing the first line, the line number should read ‘1’. When editing the second line, the line number should be ‘2’. | | | |