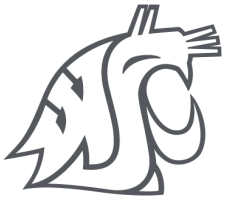

Washington State University

Computer Science 322

Design Specification

David Harkins

Instructor
Neil Corrigan

October 13, 2016

Contents

List of Figures

Revision History

Revision 2, October 13, 2016 : Updated based on the notes provided from the initial draft.

Revision 1, April 1, 2016 : The initial version of the design specification.

Chapter 1

Introduction

The purpose this document is to create a design of the Turing machine simulation program. This document will be used by the developers that will be doing the implementing. With each developer having a copy of this design they will be able to work largely independently. This will increase the amount of developers that can be working on this project at a single time. It will also provide a single uniform guide so each developer is working towards the same common goal.

This design document will provide and explain the design of the Turing machine simulation program. First, this will be a class diagram, made with the unified modeling language, that will be used to specify the classes and their relationships. Next, each class will be described in more detail without using UML. Each class description will have a short description, the associations to other classes, the attributes, and the methods. The next thing will that will be described is the commands. Each command design will provide a description, examples of output, how to call it, and if there are any limitations of when it can be called.

Chapter 2

Architecture

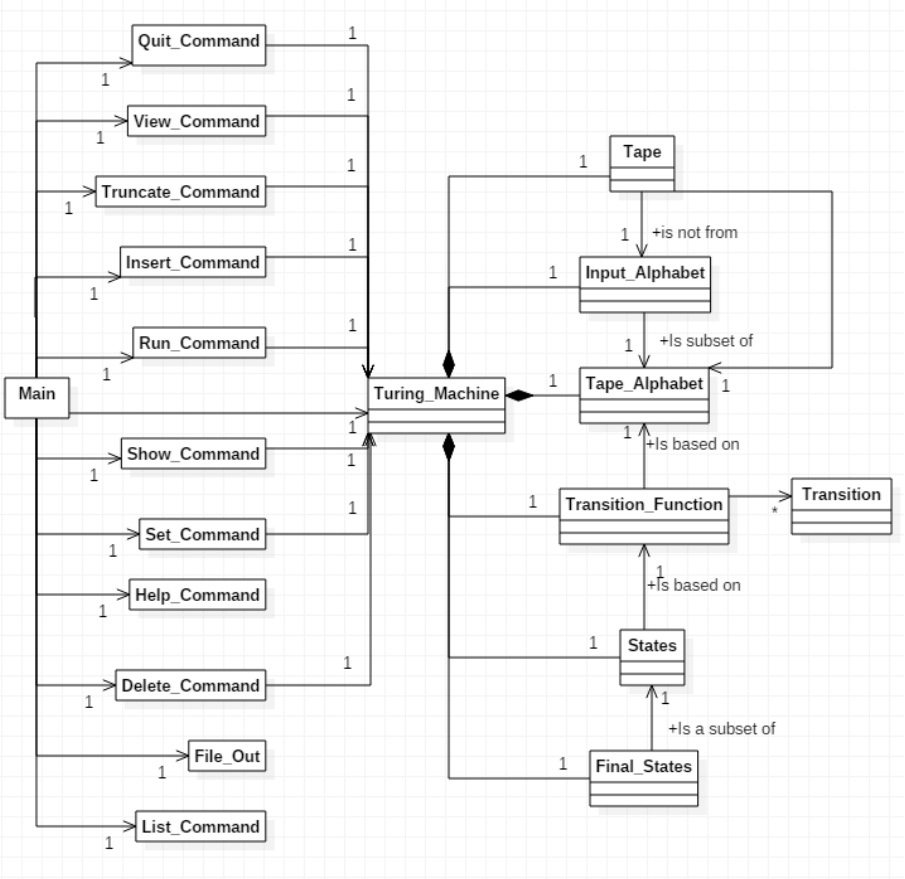

Figure 2.1: The associations between the classes.

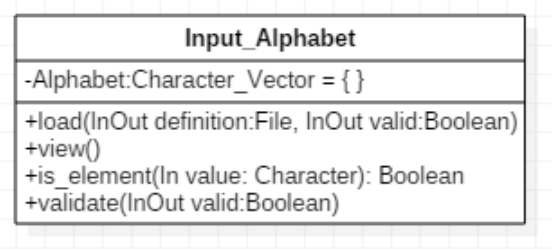

Figure 2.2: Input_Alphabet class diagram.

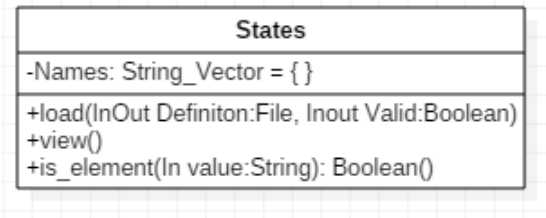

Figure 2.3: States class diagram.

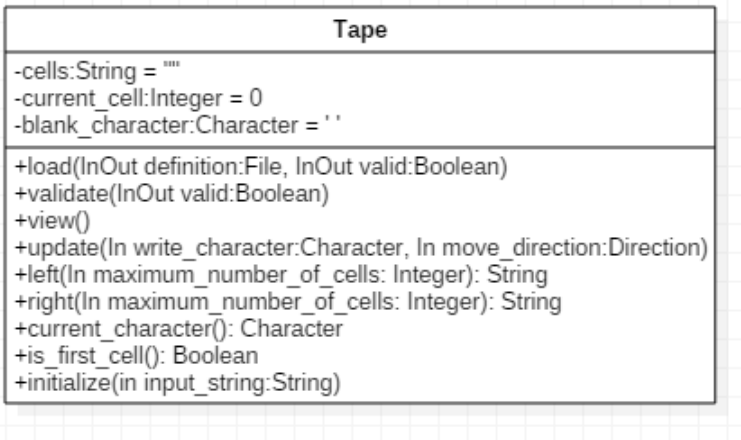

Figure 2.4: Tape class diagram.

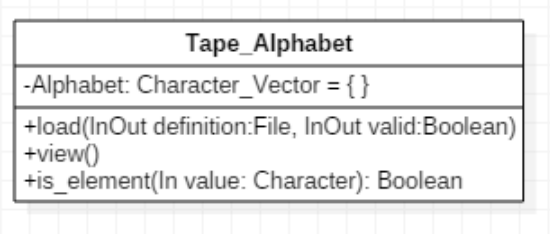

Figure 2.5: Tape_Alphabet class diagram.

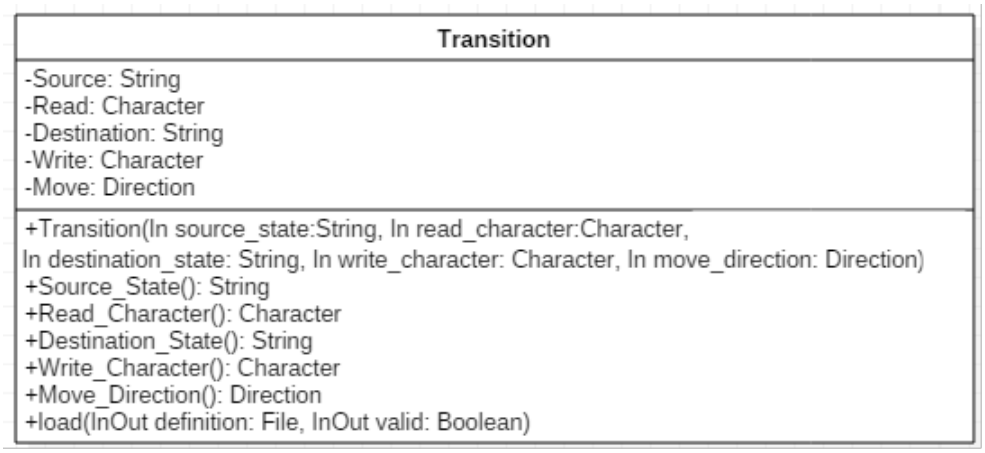

Figure 2.6: Transition class diagram.

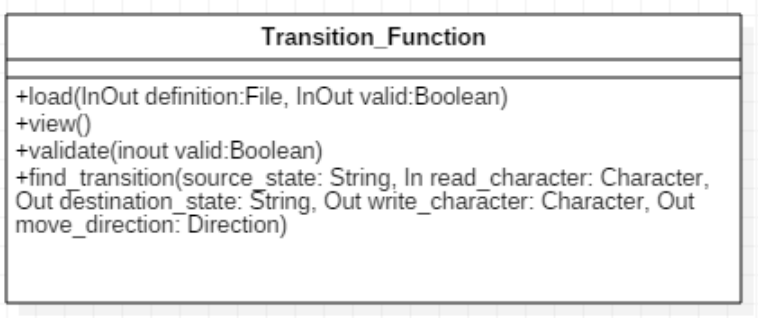

Figure 2.7: Transition_Function class diagram.

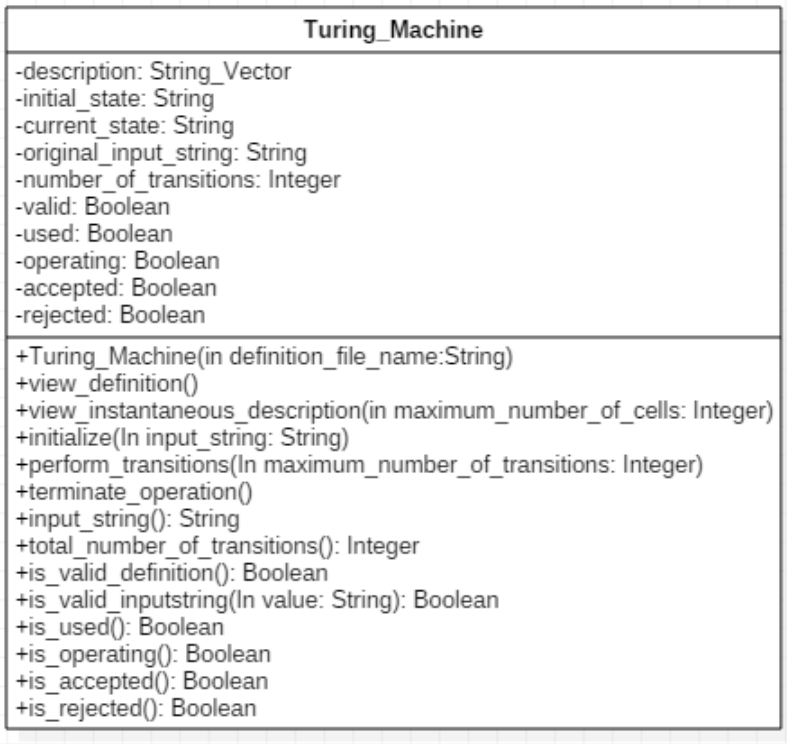

Figure 2.8: Turing_Machine class diagram.

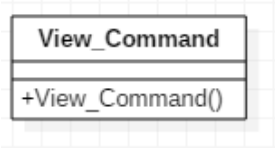

Figure 2.9: View_Command class diagram.

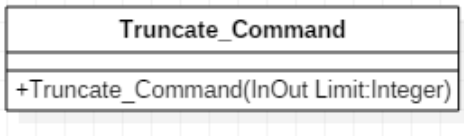

Figure 2.10: Truncate_Command class diagram.

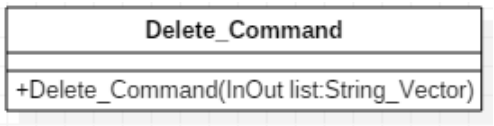

Figure 2.11: Delete_Command class diagram.

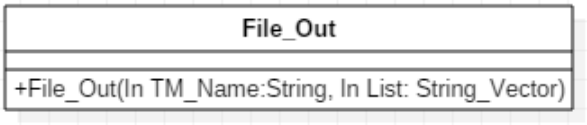

Figure 2.12: File_Out class diagram.

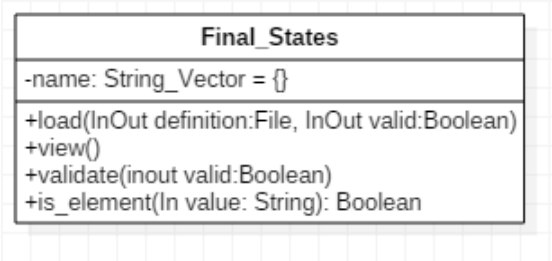

Figure 2.13: Final_States class diagram.

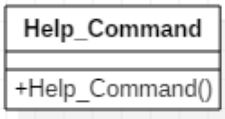

Figure 2.14: Help_Command class diagram.

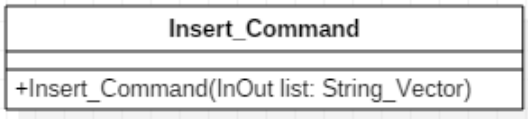

Figure 2.15: Insert_Command class diagram.

Figure 2.16: List_Command class diagram.

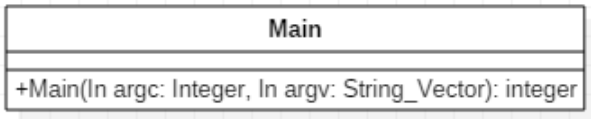

Figure 2.17: Main class diagram.

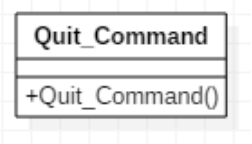

Figure 2.18: Quit_Command class diagram.

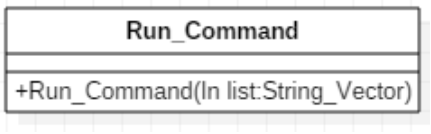

Figure 2.19: Run_Command class diagram.

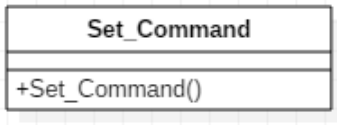

Figure 2.20: Set_Command class diagram.

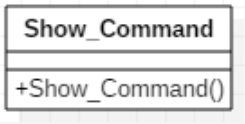

Figure 2.21: Show_Command class diagram.

Chapter 3

Data Dictionary

3.1 Turing_Machine

3.1.1 Description

The Turing_Machine class is used to control the simulation of a Turing Machine. It is composed of the elements required to run a Turing machine and it handles the interaction between them.

3.1.2 Associations

The class Turing_Machine is composed of a single one of each of the following; Tape, Input_Alphabet, Transition_Function, States, and Final_States.

3.1.3 Attributes

description: String_Vector

The attribute description stores the description that was loaded from the Turing Machine definition file.

initial_state: String

The attribute the initial_state of the Turing machine that was loaded from the Turing Machine definition file.

current_state: String

The attribute the current_state that the Turing machine is on in the simulation.

original_input_string: String

The attribute original_input_string contains the original input string that was initially put on the Turing Machine tape.

number_of_transitions:Integer

This attribute stores the number of transitions that the Turing Machine has run.

valid: Boolean

This attribute holds the state for if the Turing_Machine is a valid Turing Machine.

used: Boolean

This attribute stores if the Turing_Machine has been used before.

operating: Boolean

This attribute stores if the Turing_Machine is currently operating on a input string.

accepted: Boolean

This attribute stores if the Turing_Machine has accepted the input string,

rejected: Boolean

The attribute rejected is used to store if the Turing_Machine was rejected or not.

3.1.4 Methods

Turing_Machine(In definition_file_name:String)

This function accepts a file name and appends ".TM" to it and attempts to load the file. If it is unable to load the file it displays an error and returns. If the file is found and is able to be read, this function will read the name of the description of the Turing Machine from the file, and the open file will be passed to the following classes in this order. First States, Input_Alphabet, Tape_Alphabet, Transition_Function, back to this function where it will read the initial state and pass the file to Tape_Alphabet, and finally to Final_States.

view_definition()

This function prints out the definition of the Turing Machine that was loaded from the Turing Machine definition file. To do this the function calls the correct display function for each component contained within it.

view_instantaneous_description(in maximum_number_of_cells: Integer)

This function displays the instantaneous description to the user. When displaying the max number of cells to display on each side that are displayed is limited to the maximum_number_of_cells that is passed as a parameter.

initialize(In input_string:String

This function sets the current input string to input_string if operating is set to false. Next, the initial state of the Turing Machine's tape is displayed to the user. If operating is set to true, an error message is displayed and the function returns.

perform_transitions(In maximum_number_of_transitions: Integer)

This function call will simulate the Turing Machine that is loaded in from the Turing Machine definition file on the set input string. The simulation can only happen if operating is set to true, accepted and rejected are both set to false. The function will continue to simulate the Turing Machine until the amount of transitions reaches the maximum_number_of_transitons unless the Turing machine rejects or accepts the input string before the maximum_number_of_transitions is met. If the Turing Machine is accepted or rejected, operating is set to false, and a message is displayed to the user.

terminate_operation()

This function stops the operation on the current input string by setting the operating variable to false. This makes the Turing_Machine unable to continue and forces the user to start a new simulation on a input string.

input_string(): String

This function returns the current input string that the Turing_Machine is operating on.

total_number_of_transitions(): Integer

This function returns the amount of transitions that have been completed on the currently selected input string.

is_valid_inputstring(In value:String)

This function returns if the input string is valid. This function checks that every character in the input string is contained within the input_alphabet. If any character contains a character that is reserved or is not in the input_alphabet it returns false. If it contains only characters that are in the input alphabet it returns true.

is_valid_definition(): Boolean

This function returns if the validation by the components, that are contained within the Turing_Machine class, all validated correctly.

is_used(): Boolean

This function returns if the Turing_Machine has been used before.

is_operating(): Boolean

This function returns the value of operating.

is_accepted(): Boolean

This function returns the value of accepted.

is_rejected(): Boolean

This function returns the value of rejected.

3.2 Tape

3.2.1 Description

The tape of a Turing machine consists of an ordered sequence of cells, indexed starting at 0, which may grow to any size needed up to the limit of storage during operation of the machine on an input string. Each cell contains a character in the tape alphabet. An input string is stored in the lowest numbered tape cells at the beginning of operation, and all other tape cells initially contain the blank character. The current cell starts at the first cell on the tape. In performing a transition of the Turing machine, the character contained in the current cell may be read and written, and the current cell may be moved one cell to the left or right. The tape exists only as part of a Turing machine.

3.2.2 Associations

The class Tape is a component of the class Turing_Machine, receiving messages delegated to it by the Turing machine. The association not filled with the

class Input_Alphabet used to validate that the blank character for initialization and extension of the Turing machine tape is in the tape alphabet.

3.2.3 Attributes

cells:String = ""

The attribute cells is a dynamically growing character string containing the Turing machine tape. In preforming an update, the tape may be extended by appending a blank character.

current_cell:Integer = 0

The index of the current cell on the Turing machine tape is stored in the attribute current_cell.

blank_character:Character = ' '

The blank character used, to initialize and extend the Turing machine tape, is contained in the attribute blank_character.

3.2.4 Methods

load(InOut definition:File, InOut valid:Boolean)

The method load reads the blank character from the Turing machine definition file. If the blank character is reserved or not printable, or the next keyword does not follow it in the file, an error message is displayed and valid is set to false.

validate(InOut:Boolean)

The method validate determines if the blank character of the Turing machine is in the tape alphabet, but not in the input alphabet. If the blank character is in the input alphabet, or is not in the tape alphabet, an error message is displayed and valid is set to of false.

view()

The method view displays the blank character of the Turing machine.

initialize(In input_string:String)

The method initialize sets the Turing machine tape to the input string followed by a blank character, replacing the previous contents of the tape. The current cell is set to the first cell on the tape, indicated by the index 0.

update(In write_character:Character, In move_direction:Direction)

The method update first determines if the update of the Turing machine tape is possible. The method returns if a left move is specified from the first cell. If a right move is specified from the last cell, a blank character is appended to the tape. Assuming that the update may be performed, the character to write on the tape is stored in the current cell, replacing the previous character in that cell. To move the current cell one cell to the left, the index is decremented, or to move the current cell one cell to the right, the index is incremented.

left(In maximum_number_of_cells:Integer):String

The method left returns a character string up to the maximum number of cells from the Turing machine tape to the left of the current cell, excluding that cell. The length of the string will be less than the maximum if there are fewer cells to the left of the current cell. If the string is truncated from the tape, the reserved character '¡' will be added to the beginning of the string.

right(in maximum_number_of_cells:Integer):String

The method right returns a character string up to the maximum number of cells from the Turing machine tape to the right of the current cell, including that cell. The length of the string will be less than the maximum if there are fewer cells to the right of the current cell up to the rightmost nonblank character. If the string is truncated from the tape, the reserved character '¿' will be added to the end of the string.

current_character():Character

The method current_character returns the character contained in the current cell on the Turing machine tape.

is_first_cell():Boolean

The method is_first_cell returns a value of true if the current cell on the Turing machine tape is the first cell, indicated by the index 0. Otherwise, it returns a value of false.

3.3 Input_Alphabet

3.3.1 Description

The list of characters that are allow for the user to enter. This must be a subset of the Tape_Alphabet. The association is class Tape is not from Input_Alphabet, the

3.3.2 Associations

This class is a component of the class Turing_Machine, receiving messages delegated to it by the Turing machine.

3.3.3 Attributes

Alphabet:Character_Vector = { }

This attribute is used to store the list of Input_Alphabet characters that were loaded from the Turing Machine definition file. The characters contained must be a subset of the ones contained in the Tape_Alphabet and also does not include the blank character.

3.3.4 Methods

load(InOut definition:File, InOut valid:Boolean)

This function attempts to load the Input_Alphabet from the File. If there is a failure on reading or parsing the file, the function will set valid to false, display the error, and return. If the function is able to read and parse the file, the function will return.

view()

This function displays the contents of Alphabet.

is_element(In value: Character): Boolean

This function checks if the parameters character is contained within Alphabet.

validate(InOut valid:Boolean)

This function compares the elements loaded from the Turing Machine definition file to the ones in Input_Alphabet

3.4 Tape_Alphabet

3.4.1 Description

The list of characters that can be written to the type cells in Tape.

3.4.2 Associations

This class is a component of the class Turing_Machine, receiving messages delegated to it by the Turing machine.

3.4.3 Attributes

Alphabet: Character_Vector = {}

The attribute Alphabet is used to store the characters that can be placed on the tape cells.

3.4.4 Methods

load(InOut definition:File, InOut valid:Boolean)

This method is used to load new Tape_Alhpabet characters from a file. This stops when a keyword is found or the end of the file is reached. All errors will be displayed to the user. If there are any errors the function will set valid to fail thereby rejecting the Turing Machine Definition file.

is_element(In value: Character): Boolean

This method is used to check if the character is within the alphabet.

3.5 Transition_Function

3.5.1 Description

The transition function is responsible providing easy access to the class Transition from other classes. It is responsible for loading the transitions, selecting the correct one for the next transition if available, validating the transitions, and printing out the transitions.

3.5.2 Associations

This class is a component of the class Turing_Machine, receiving messages delegated to it by the Turing machine. This class has a is based on association with the class States.

3.5.3 Attributes

None.

3.5.4 Methods

load(InOut definition:File, InOut valid:Boolean)

This method is passed a File and it reads it until, it finds the next section's tag, a error in parsing the data happens, or it reaches the end of the file.

view()

This method is used to print out the value of the transition. This is used by class Show_Command though the class Turing_Machine.

validate(InOut valid:Boolean)

This method is used to validate the transitions. It checks that each transition is based on the class States.

find_transition(source_state:String, In read_character: Character, Out destination_state: String, Out write_character: Character, Out move_direction: Direction)

This method searches all of the class transition until it: finds a matching source and read character; or it searched all of the Transitions and was unable to find one. If the class transition_Function is able to find a Transition that matches it will return the Transition. If the class Transition_Function is unable to find a transition that matches it returns an error.

3.6 Transition

3.6.1 Description

This class is designed to hold a single Turing Machine definition transition. All other classes that want to use it must access it by using the class Transition_Function.

3.6.2 Associations

This class has one to many association with the class Transition_Function.

3.6.3 Attributes

Source: String

This attribute contains the name of the current state that the Turing machine must be in to use this transition.

Read: Character

This attribute contains the name of the state that is currently the current tape cell.

Destination: String

This attribute contains the name of the state that the Turing Machine will transfer to if it uses this transition.

Write: Character

This attribute contains the character that will be written to the class Tape where the Turing Machine is currently pointing. This happens before the transition moves the current_cell that is contained in the class Tape.

Move:Direction

This attribute contains the direction that the tape head will move after the character is written to the class Tape. It can only be to the right or to the left.

3.6.4 Methods

load(InOut definition:File, InOut valid:Boolean)

This method is used to load the Transitions from the Turing Machine definition file. The method will stop searching the file a new keyword or the end of the file. It displays any errors out to the user.

Transition(In source_state:String, In read_character:Character, In destination_state: String, In write_character: Character, In move_direction: Direction)

Source_State():String

This method is used to get the current value of Source attribute that is contained in the class Transition.

Read_Character(): Character

This method is used to get the current value of Read attribute that is contained in the class Transition.

Destination_State(): String

This method is used to get the current value of State attribute that is contained in the class Transition

Write_Character(): Character

This method is used to get the current value of write attribute that is contained in the class Transition

Move_Direction(): Direction

This method is used to get the current value of Direction attribute that is contained in the class Transition.

3.7 States

3.7.1 Description

The possible states the Turing machine can be in at any given time.

3.7.2 Associations

This class is a component of the class Turing_Machine, receiving messages delegated to it by the Turing machine. The class Final_States has an is a subset of association with the class States.

3.7.3 Attributes

Names: String_Vector = { }

This attribute contains all of the state names. They are loaded from a Turing Machine definition file and validated while they are loaded.

3.7.4 Methods

load(InOut definition:File, InOut valid:Boolean)

This method is called by the Turing_Machine class that it is contained inside. This method will attempt to load Turing Machine states from the file until when a keyword is found or the end of the file is reached. All errors will be displayed to the user. If there are any errors the function will set valid to fail thereby rejecting the Turing Machine Definition file.

view()

This method is used to display the current states that are attributes of the current class instance of States.

is_element(In value:String): Boolean()

This method is used to check if a state is contained in the current class instance of State.

3.8 Final_States

3.8.1 Description

The state that if the Turing Machine transitions into cannot leave, the Input String is accepted, and the class Turing_machine sets operating to false.

3.8.2 Associations

This class is a component of the class Turing_Machine, receiving messages delegated to it by the Turing machine. This class has an is a subset of association with the class States.

3.8.3 Attributes

name: String_Vector = { }

This attribute contains all of the states that are considered final states. All of the Final_States must be contained in the class States as well.

3.8.4 Methods

load(InOut definition:File, InOut valid:Boolean)

This method is called to load new final states into the class Final_States. This stops when a keyword is found or the end of the file is reached. All errors will be displayed to the user. If there are any errors the function will set valid to fail thereby rejecting the Turing Machine Definition file.

view()

This method is used to display all of the current final states in a formatting of the formal definition of the Turing Machine class.

validate(InOut valid:Boolean)

This method is used to make sure that each of the Final_States in contained in the class States.

is_element(In value String): Boolean)

This method is used to check if a state is a final state. The class Turing_Machine uses this to check if the current transition is into a final state.

3.9 Quit_Command

3.9.1 Description

When the Turing machine class is currently running on a input string the class the Quit_Command will terminate the running on the input string. The Turing machine will neither accept or reject the input string is the Quit_Command is used. If the command is used, and the Turing machine is not running on a input string, a error will be displayed, and Quit_Command will return.

3.9.2 Associations

This class is associated with the class Main, receiving messages delegated to it by the user. The class Quit_Command is associated with a single Turing_Machine class.

3.9.3 Attributes

None.

3.9.4 Methods

Quit_Command()

The Quit_Command function will terminate the running of a Turing machine on a input string. If there is no input string running or the Turing machine has accepted, rejected, or never been used before an error will be displayed and the function will return.

3.10 View_Command

3.10.1 Description

The View_Command displays the Turing machine definition file that is contained inside of a Turing machine. Before the function can be called, the program has insured that the Turing machine definition is valid.

3.10.2 Associations

This class is associated with the class Main, receiving messages delegated to it by the user.

The class View_Command is associated with a single Turing_Machine class.

3.10.3 Attributes

None.

3.10.4 Methods

View_Command()

This function prints the currently stored Turing Machine definition from a Turing machine.

3.11 Truncate_Command

3.11.1 Description

The Truncate_Command gets user input and updates the truncation setting. The value that the user enters must be an integer and greater than zero, if it is not an error, is displayed and the function returns without changing the setting.

3.11.2 Associations

This class is associated with the class Main, receiving messages delegated to it by the user.

The class Truncate_Command is associated with a single Turing_Machine class.

3.11.3 Attributes

None.

3.11.4 Methods

Truncate_Command(InOut limit: Integer)

This function gets user input and updates the truncation setting. The value of the user input must be a integer and greater than zero. If it is not, an error message is displayed and the function returns without changing the setting. If the user enters nothing, the function returns without changing the setting or displaying any message.

3.12 Insert_Command

3.12.1 Description

The class Insert_Command allows the user to enter a new input string. The input string that will be inserted can be used to run on the Turing Machine. If the user wishes to specify the empty string they must enter the backslash character.

3.12.2 Associations

This class is associated with the class Main, receiving messages delegated to it by the user.

The class Insert_Command is associated with a single Turing_Machine class.

3.12.3 Attributes

None.

3.12.4 Methods

Insert_Command(InOut list: String_Vector)

This function will user input, and if the input is not empty, it will validate the input against the current Turing_Machine. If the input is empty, the function will return with no display or change in list. If the input is invalid for the Turing Machine definition, an error is displayed and the function returns. If the input is valid, the new input string is added to the end of the list. If the user enters the backslash character will be replaced with an empty string before validation occurs.

3.13 Run_Command

3.13.1 Description

This class Run_Command will start the Turing Machine running on a selected input string.

3.13.2 Associations

This class is associated with the class Main, receiving messages delegated to it by the user. The class Run_Command is associated with a single Turing_Machine class.

3.13.3 Attributes

None.

3.13.4 Methods

Run_Command(In list:String_Vector)

This function takes a list of input strings as a parameter and prompts for user input. The input that the function asks for is a number that is assigned to a input string. The number can be found using the list command on the command prompt. If the user enters nothing, the function will return without any message or without starting the Turing Machine. If the input is an invalid input string number, or the Turing Machine is already running on a input string, an error will be displayed, and the function will return. If the input is a valid string number, the input string that the user selected will be given to Turing_Machine, and the function will return.

3.14 Show_Command

3.14.1 Description

The show command lists the status of the program and Turing machine. The information displayed is based on user settings and the current status of the Turing Machine. The differences in the Turing machine status are: it has run before; is currently running; or was accepted; rejected or the user terminated it.

3.14.2 Associations

This class is associated with the class Main, receiving messages delegated to it by the user. The class Show_Command is associated with a single Turing_Machine class.

3.14.3 Attributes

None.

3.14.4 Methods

Show_Command()

The function Show_Command() displays the status of the Turing Machine. The elements that are listed are the course name, semester, year, instructor, author, version number, max transitions, max cells to the left and right, the name of the Turing Machine file, and the Turing machine status. If the Turing machine has never been run before, a message is displayed to that effect. If the Turing machine is currently running, the input string it is running on is displayed as well as the number of transitions already preformed. If the Turing Machine has accepted, rejected, or the user has terminated operation, the number of transitions and reason the operation was stopped are displayed.

3.15 Set_Command

3.15.1 Description

This Set_Command is used to change the maximum number of transitions for the Turing_Machine to perform before the operation pauses.

3.15.2 Associations

This class is associated with the class Main, receiving messages delegated to it by the user. The class Set_Command is associated with a single Turing_Machine class.

3.15.3 Attributes

None.

3.15.4 Methods

Set_Command()

This function displays the current setting of the maximum transitions and waits for user input. If the user enters nothing, the function returns without changing the setting or displaying a message. If the user does not enter a number greater than 0, and that is a integer, an error is displayed, and the function returns. If the input is valid, the maximum transitions is changed, a message is displayed, and the function returns.

3.16 Help_Command

3.16.1 Description

This class Help_Command displays the usage to the user. This displays all of the valid commands for the users to use and a short description of each.

3.16.2 Associations

This class is associated with the class Main, receiving messages delegated to it by the user.

3.16.3 Attributes

None.

3.16.4 Methods

Help_Command()

The Help_Command displays the usage to the user.

3.17 Delete_Command

3.17.1 Description

This class Delete_Command is used to remove input strings from the input string list.

3.17.2 Associations

This class is associated with the class Main, receiving messages delegated to it by the user. The class Delete_Command is associated with a single Turing_Machine class.

3.17.3 Attributes

None.

3.17.4 Methods

Delete_Command(InOut list:String_Vector)

This function prompts for the user to enter a valid input string index number. If the user enters nothing, the function returns without a message. If the user enters a number that is not above zero, or that is not a integer, an error is displayed and the function returns. If the user enters a valid input string, index it is deleted, a message is displayed, and the function returns.

3.18 File_Out

3.18.1 Description

This class File_Out is used to offload the input string list when the application is closing. The file will only be written out to disk if there were changes made to the file (Invalid input string was removed, the user added a string, or user deleted a string).

3.18.2 Associations

This class is associated with the class Main, being called when the application terminates.

3.18.3 Attributes

None.

3.18.4 Methods

File_Out(In TM_Name:String, In List:String_Vector)

This function saves the entire input string list a file named after the parameter TM_Name. This will overwrite the original input string list if it exists, or will create a new one if it does not. If it is unable to save the input string file to disk, an error will be displayed, and the application will terminate. If the function was able to save the file to disk, a message is displayed and the application terminates.

3.19 List_Command

3.19.1 Description

This class List_Command displays the current list of input strings to the user. If there are no input strings a message will be displayed saying such.

3.19.2 Associations

This class is associated with the class Main, receiving messages delegated to it by the user.

3.19.3 Attributes

None.

3.19.4 Methods

List_Command(In List: String_Vector)

This function displays each input string on its own line with its input string index before it, and the function returns. If there are no input strings a message is displayed saying such and the function returns.

3.20 Main

3.20.1 Description

This class Main is the top level entry point for the application. It is in control of starting all of the other classes.

3.20.2 Associations

The class Main is associated with a single Turing_Machine class, a single Quit_Command, a single View_Command, a single Truncate_Command, a single Insert_command, a single Run_Comannd, a single Show_Command, a single Set_Command, a single Help_Command, a single Delete_Command, a single File_Out, and a single List_Command.

3.20.3 Attributes

None.

3.20.4 Methods

Main(argc: integer, argv: String_Vector): integer

This function attempts to load the Turing Machine definition file named after the user passed parameter. If the Turing Machine file is able to be loaded, and it is valid, the function continues. If it was not able to be loaded, or it was invalid, an error message is displayed, and the application terminates. Next, the function attempts to load the input string file. On failure or success, the application continues to a loops that prompts for a command from the user, and calls the correct function to handle it.

Chapter 4

User Interface

4.1 Command Line Invocation

Command line invocation with a valid file

Example of running the application with a command line argument (Normal):

```
$ TM Definition
Turing successfully loaded.
Command:
```

Command line invocation with a invalid file

Example of running the application with a command line argument (Error):

```
$ TM Definition_error
Error unable to load Turing Machine definition file.
$
```

4.2 Help Command

Example of using the help command:

```
Command: h

(D)elete   - Delete input string from list.
E(x)it     - Exit application.
(H)elp     - Help user.
(I)nsert   - Insert input string into list.
(L)ist     - List input strings.
(Q)uit     - Quit operation of Turing machine on input string.
(R)un      - Run Turing machine on input string.
```

```
S(e)t      - Set maximum number of transitions to perform.
Sho(w)     - Show status of application.
(T)runcate - Truncate instantaneous description.
(V)iew     - View Turing machine.

Command:
```

4.3 Show Command

```
Example of using the show command:

Command: w

Course name: CPT_S 322
Semester: Spring
Year: 2016

Instructor: Neil Corrigan
Author: David Harkins

Version number: 1
Max transitions: 10
Max cells to left and right: 10
Name of TM: A
TM Status: Completed
The input string aaabb has been accepted in 7 transitions.

Command:
```

4.4 View Command

The view command is used to view the currently loaded Turing machine definition file.

```
Example of using the view command:

Command: v

This Turing machine accepts the language
of one or more a's followed by the same number of b's.

Q = {s0, s1, s2, s4}

Sigma = {a, b}

Gamma = {a, b, X, Y, -}

Delta(s0, a) = (s1, X, R)
Delta(s0, Y) = (s3, Y, R)
Delta(s1, a) = (s1, a, R)
Delta(s1, b) = (s2, Y, R)
Delta(s1, Y) = (s1, Y, R)
Delta(s2, a) = (s2, a, R)
```

```
Delta(s2, X) = (s0, X, R)
Delta(s2, Y) = (s2, Y, R)
Delta(s3, Y) = (s3, Y, R)
Delta(s3, -) = (s4, -, R)

Q0 = s0

B = -

F = {s4}


Command:
```

4.5 List Command

```
Example of using the list command:

Command: l

1. a
2. ab
3. \
4. aaabb
5. aaaaaaaaaaabbbbbbbbbb
6. aabb
7. aaaaaabbbbbb
8. ba
9. aba
10. bb

Command:
```

4.6 Insert Command

```
Example of using insert command:

Command: i

Input string: abbb
String inserted into list!

Command:
```

4.7 Delete Command

Delete command with a valid input string index

```
Example of using the delete command (Normal):

Command: d

Input string number: 1
String deleted!

Command:
```

Delete command with a invalid input string index

```
Example of using the delete command (Error):

Command: d

Input string number: -1
Unable to delete, invalid input string index.

Command:
```

4.8 Set Command

The set command with valid input.

```
Example of using the set transition command (Normally):

Command: e

Maximum number of transitions [1]: 20
Number of transitions changed to 20.

Command:
```

The set command with invalid input.

```
Example of using the set transition command (With error):

Command: e

Maximum number of transitions [1]: -1
Error, unable to change transitions. The new value must be greater or equal to 1

Command:
```

4.9 Truncate Command

The truncate command with valid input

```
Example of using the truncate command (Normal):

Command: t

Maximum number of cells [32]: 10
Setting changed!

Command:
```

The truncate command with invalid input

```
Example of using the truncate command (Error):

Command: t

Maximum number of cells [32]: -1
Error, unable to change max cells. The new value must be greater or equal to 1

Command:
```

4.10 Run Command

Using the run command with a valid string input index number

```
Example of using the run command (Normal):

Command: r

Input string number: 1
0. [s0]aaabb
7. XXXXY-[s4]
Input string aaabb was accepted in 7 transitions.

Command:
```

Using the run command with a invalid string input index number

```
Example of using the run command (Error):

Command: r

Input string number: -1
Invalid input string index.
```

```
Command:
```

4.11 Quit Command

Quit command running on input string

```
Example of using the quit command (Normal):

Command: q

Input string aaabb was not accepted or rejected in 112 transitions.

Command:
```

Attempt to Quit on a Turing machine not running on a input string

```
Example of using the quit command (Error):

Command: q

Unable to stop running on input string because Turing machine
was not running on input string.

Command:
```

4.12 Exit Command

Exit command with save normally.

```
Example of using the exit command (Normal):

Command: x

Input string file successfully saved to disc.

$
```

Exit command with save Error.

```
Example of using the exit command (Error):

Command: x

Input string file was unable to be saved to disc.

$
```

Chapter 5

File

5.1 Turing Machine Definition File

The following is an example of the Turing machine file format.

```
Example of using the view command:

Command: v

This Turing machine accepts the language
of one or more a's followed by the same number of b's.

STATES: = s0 s1 s2 s4

INPUT_ALPHABET: = a b

TAPE_ALPHABET: = a b X Y -

TRANSITION_FUNCTION:
s0 a  s1 X R
s0 Y  s3 Y R
s1 a  s1 a R
s1 b  s2 Y R
s1 Y  s1 Y R
s2 a  s2 a R
s2 X  s0 X R
s2 Y  s2 Y R
s3 Y  s3 Y R
s3 -  s4 - R

INITIAL_STATE: = s0
```

```
BLANK_CHARACTER: = -

FINAL_STATES: s4


Command:
```

5.2 Input String File

The following is an example of the input string file format.

```
a
ab
\
aaabb
aaaaaaaaaaabbbbbbbbbb
aabb
aaaaaabbbbbbb
ba
aba
bb
```

Chapter 6

References

- Neil Corrigan - General notes, lecture and help.