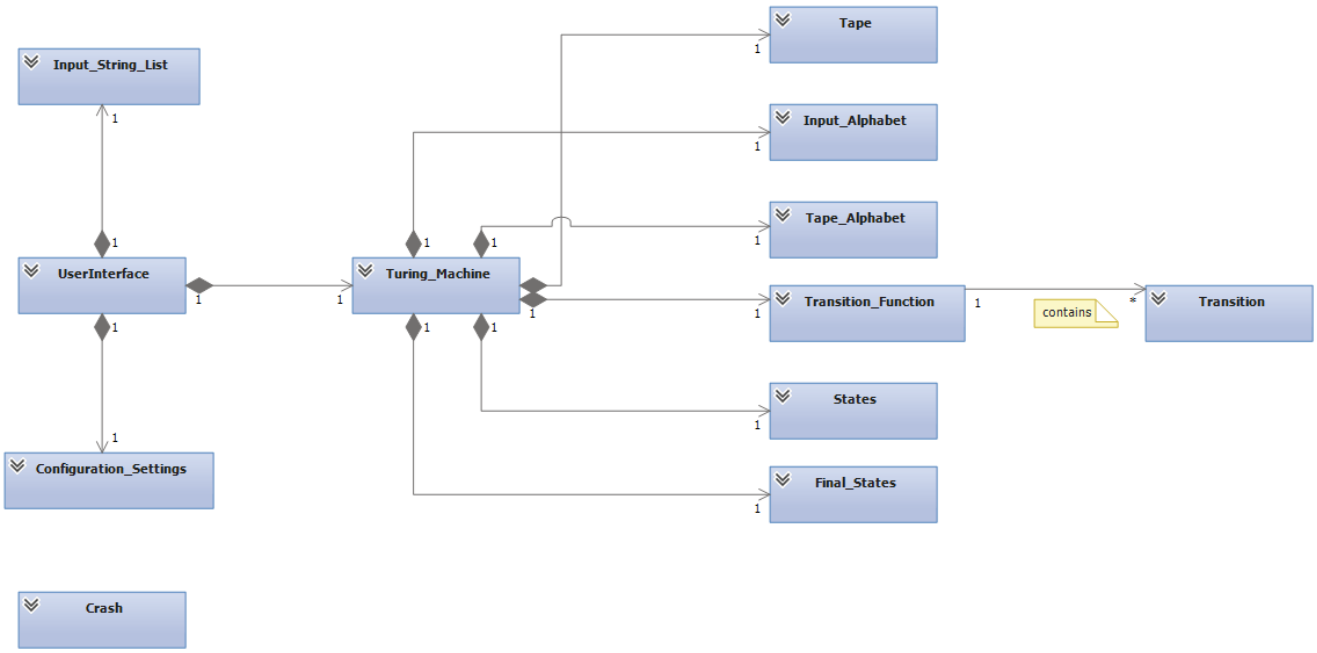
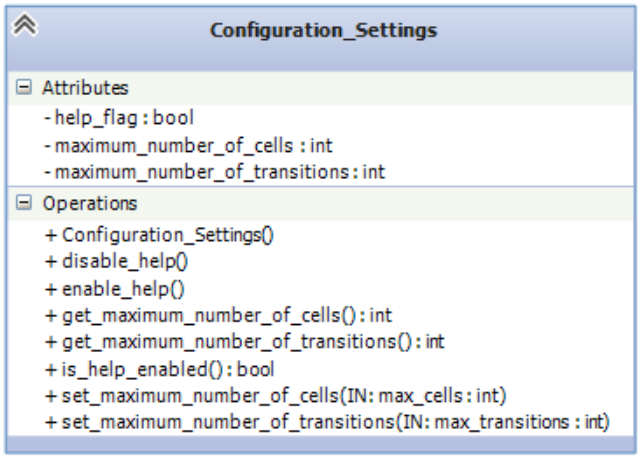
Table of Contents

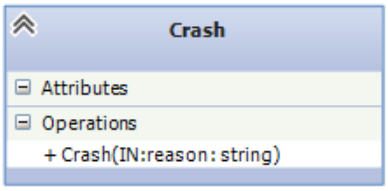
1.0 Introduction

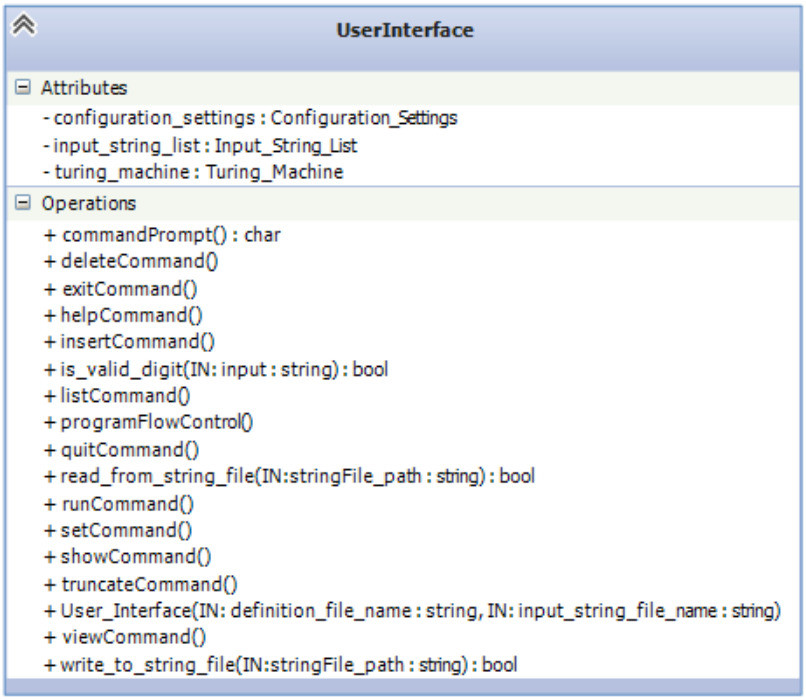
The purpose of this document is to specify the classes that will be used to create this application, and how these classes communicate with one another to accomplish a task. Furthermore, this document specifies what methods will be used by each class that satisfy the requirement specification, and how they are expected to operate. The intended audience of this document is the developers of the application. The remainder of this document will include a UML class diagram, which will show the relationship between classes and methods for each class, a detailed explanation of each class and its methods, the expected functionality of the user interface, and example Turing machine definition files and input string files.

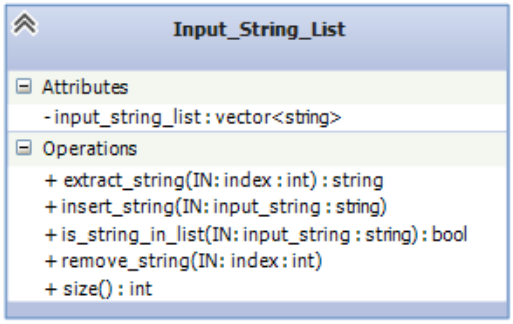
2.0 Architecture

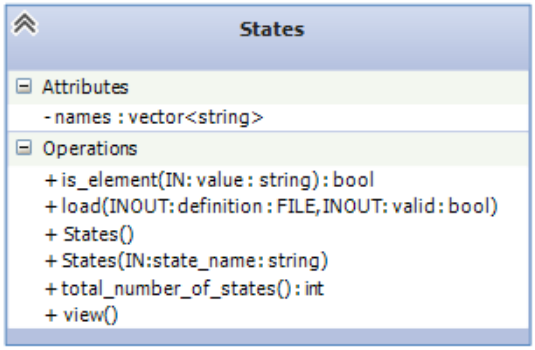


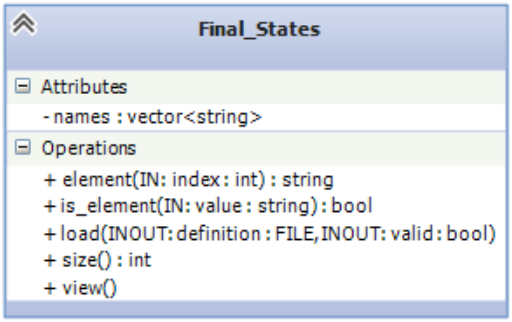

Configuration_Settings
Attributes
- help_flag : bool
- maximum_number_of_cells : int
- maximum_number_of_transitions : int
Operations
+ Configuration_Settings()
+ disable_help()
+ enable_help()
+ get_maximum_number_of_cells() : int
+ get_maximum_number_of_transitions() : int
+ is_help_enabled() : bool
+ set_maximum_number_of_cells(IN: max_cells : int)
+ set_maximum_number_of_transitions(IN: max_transitions : int)

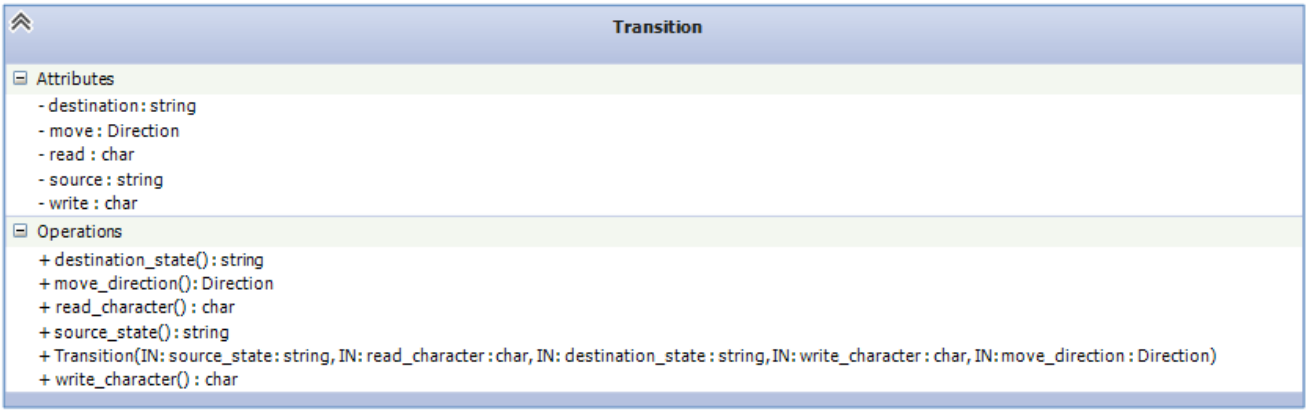

Crash
Attributes
Operations
+ Crash(IN:reason : string)


UserInterface
Attributes
- configuration_settings : Configuration_Settings
- input_string_list : Input_String_List
- turing_machine : Turing_Machine
Operations
+ commandPrompt() : char
+ deleteCommand()
+ exitCommand()
+ helpCommand()
+ insertCommand()
+ is_valid_digit(IN: input : string) : bool
+ listCommand()
+ programFlowControl()
+ quitCommand()
+ read_from_string_file(IN:stringFile_path : string) : bool
+ runCommand()
+ setCommand()
+ showCommand()
+ truncateCommand()
+ User_Interface(IN: definition_file_name : string, IN: input_string_file_name : string)
+ viewCommand()
+ write_to_string_file(IN:stringFile_path : string) : bool


Input_String_List
Attributes
- input_string_list : vector<string>
Operations
+ extract_string(IN: index : int) : string
+ insert_string(IN: input_string : string)
+ is_string_in_list(IN: input_string : string) : bool
+ remove_string(IN: index : int)
+ size() : int


States
Attributes
- names : vector<string>
Operations
+ is_element(IN: value : string) : bool
+ load(INOUT: definition : FILE, INOUT: valid : bool)
+ States()
+ States(IN:state_name: string)
+ total_number_of_states() : int
+ view()


Final_States
Attributes
- names : vector<string>
Operations
+ element(IN: index : int) : string
+ is_element(IN: value : string) : bool
+ load(INOUT: definition : FILE, INOUT: valid : bool)
+ size() : int
+ view()


Transition
Attributes
- destination: string
- move : Direction
- read : char
- source : string
- write : char
Operations
+ destination_state() : string
+ move_direction() : Direction
+ read_character() : char
+ source_state() : string
+ Transition(IN: source_state : string, IN: read_character : char, IN: destination_state : string, IN: write_character : char, IN: move_direction : Direction)
+ write_character() : char

Transition_Function

Attributes

- transitions : vector<Transition>

Operations

+ destination_state(IN: index: int) : string
+ is_defined_transition(IN: source_state : string, IN: read_character : char, OUT: destination_state : string, OUT: write_character : char, OUT: move_direction : Direction) : bool
+ is_source_state(IN: state : string) : bool
+ is_unique_transition(IN : i_source, IN : i_read, IN : i_destination, IN : i_write, IN : i_move) : bool
+ load(INOUT: definition : FILE, INOUT: valid : bool)
+ read_character(IN: index : int) : char
+ size() : int
+ source_state(IN: index : int) : string
+ view()
+ write_character(index : int) : character

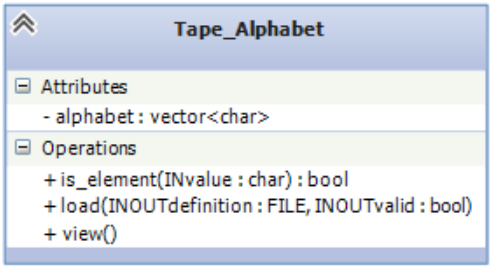
Turing_Machine

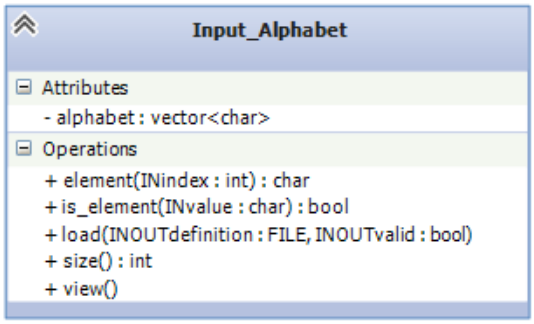
Attributes

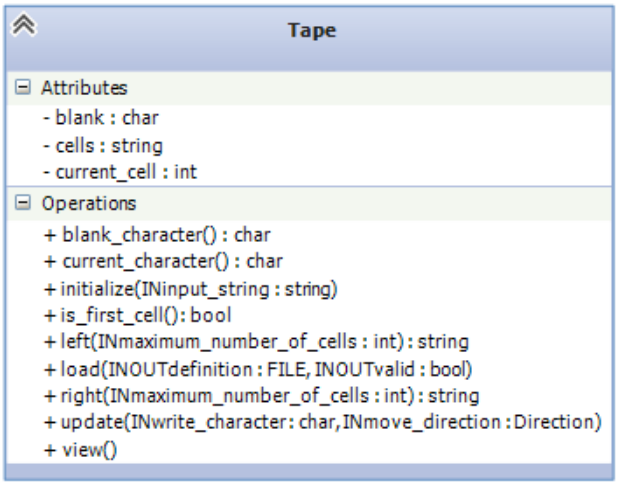
- accepted : bool
- current_state : string
- description : vector<string>
- Final_States final_states
- initial_state : string
- Input_Alphabet input_alphabet
- number_of_transitions : int
- operating : bool
- original_input_string : string
- rejected : bool
- States states
- Tape tape
- Tape_Alphabet tape_alphabet
- Transition_Function transition_function
- used : bool
- valid : bool

Operations

+ final_states_have_transitions() : bool
+ initialize(input_string : string)
+ input_string() : string
+ is_accepted_input_string() : bool
+ is_operating() : bool
+ is_rejected_input_string() : bool
+ is_used() : bool
+ is_valid_definition() : bool
+ is_valid_final_states() : bool
+ is_valid_input_alphabet() : bool
+ is_valid_input_string(value : string) : bool
+ load(INOUTdefinitionFile : FILE, INOUTvalid : bool)
+ perform_transitions(maximum_number_of_transitions : int)
+ read_definition_file(INOUT : definitionFile)
+ terminate_operation()
+ total_number_of_transitions() : int
+ Turing_Machine(definition_file_name : string)
+ Turing_Machine()
+ view_definition()
+ view_instataneous_description(maximum_number_of_cells : int)


Tape_Alphabet
Attributes
- alphabet : vector<char>
Operations
+ is_element(INvalue : char) : bool
+ load(INOUTdefinition : FILE, INOUTvalid : bool)
+ view()


Input_Alphabet
Attributes
- alphabet : vector<char>
Operations
+ element(INindex : int) : char
+ is_element(INvalue : char) : bool
+ load(INOUTdefinition : FILE, INOUTvalid : bool)
+ size() : int
+ view()


Tape
Attributes
- blank : char
- cells : string
- current_cell : int
Operations
+ blank_character() : char
+ current_character() : char
+ initialize(INinput_string : string)
+ is_first_cell(): bool
+ left(INmaximum_number_of_cells : int) : string
+ load(INOUTdefinition : FILE, INOUTvalid : bool)
+ right(INmaximum_number_of_cells : int) : string
+ update(INwrite_character: char, INmove_direction : Direction)
+ view()

3.0 Data Dictionary

3.1 Class:

Configuration_Settings

Description:

The application will have three configuration settings, initially set to default values. The configuration settings will include whether or not Help messages are provided to user for all prompts, maximum number of transitions to perform at a time, and maximum number of cells to left and right of tape to display in instantaneous description. These settings may be changed by the user during operation, but will not be saved when the application is terminated.

Associations:

The Configuration_Settings will be a component of the user interface.

Attributes:

help_flag: bool = false

The attribute help_flag will be initially false, meaning that no help messages will be provided to user.

maximum_number_of_transitions: int = 1

The attribute maximum_number_of_transitions will initially be one.

maximum_number_of_cells: int = 32

The attribute maximum_number_of_cells will default to 32.

Methods:

Configuration_Settings()

The constructor will initialize the attributes to their default values.

enable_help()

The method enable_help will set the help attribute to true.

disable_help()

The method disable_help will set help attribute to false.

is_help_enabled(): bool

The method help_status will return the value of the help attribute.

set_maximum_number_of_transitions(IN max_transitions: int)

The method set_maximum_number_of_transition will set the appropriate attribute to the value provided to the method.

get_maximum_number_of_transitions(): int

The method set_maximum_number_of_transition will return the value of the appropriate attribute.

set_maximum_number_of_cells(IN max_cells: int)

The method set_maximum_number_of_cells will set the appropriate attribute to the value provided to the method.

get_maximum_number_of_cells(): int

The method set_maximum_number_of_cells will return the value of the appropriate attribute.

3.2 Class:

Final_States

Description:

Any state in the Turing machine may be defined as a final state, but the final states must have no transitions. When an input string enters a final state, the string is accepted by the Turing machine.

Associations:

The class Final_States is a component of the class Turing_Machine, receiving messages from the Turing machine.

Attributes:

names: string vector = {}

The attribute names is a vector of containing strings which will be used to store the names of the final states of the Turing machine. This attribute will be set when the Turing machine definition file is loaded.

Methods:

is_element(IN value: string): bool

The method is_element return true if given an input string(final state), the string(final state) exists in the names vector, otherwise, it returns false.

load(INOUT definition: FILE, INOUT valid: bool)

The method load reads the final states from the turing machine definition file. If the final states are not valid (not in the set of states or invalid format) or not printable, or the next keyword does not follow it in the file, and error message is displayed and valid is set to false.

view()

The view method displays the final states of a Turing machine.

size(): int

The size method returns the total number of final states.

element(IN index: int): string

The element method returns the name of a final state, given a specified index.

3.3 Class:

Input_Alphabet

Description:

The input alphabet of a Turing machine consist of printable characters from the ASCII character set, with the exception of the reserved characters, \, [,], <, and >. Lambda is specified by the reserved character \. The opening and closing square brackets will be used to enclose the state name when displaying the instantaneous description, and the characters, < and > will be used to truncate left and/or right side of the tape head.

Associations:

The class Input_Alphabet is a component of the class Turing_Machine, receiving messages from the Turing machine.

Attributes:

alphabet: character vector = {}

The attribute alphabet is a vector containing characters specified by the Turing machine definition file.

Methods:

load(INOUT definition: FILE, INOUT valid: bool)

The method load reads the input alphabet from the turing machine definition file. If the input alphabet is not valid or not printable, or the next keyword does not follow it in the file, and error message is displayed and valid is set to false.

View()

The method view displays the input alphabet of the Turing machine.

size(): int

The method size returns the size of the attribute input_alphabet.

element(IN index: int): char

The method element returns a character from the attribute input_alphabet given an input index.

is_element(IN value: char): bool

The method is_element returns true if given an input character, the character exists in the input alphabet, otherwise, it returns false.

3.4 Class:

States

Description:

The Turing machine contains a set of states which will be used to trace the input string. States may be named as a string of upper or lower case letters, digits, or underscores, and must be unique and will be case sensitive. There will be no limit on the length of a state name or the number of states.

Associations:

The class States is a component of the class Turing_Machine, receiving messages from the Turing machine.

Attributes:

names: string vector = {}

The attribute names is a vector containing strings which will be used to store the names of states of the Turing machine. This attribute will be set when the Turing machine definition file is loaded.
Methods:

is_element(IN value: string): bool

The method is_element returns true if given an input string(state), the string (state) exists in the names vector, otherwise, it returns false.

load(INOUT definition: FILE, INOUT valid: bool)

The method load reads the states from the turing machine definition file. If the states are not valid (invalid format/duplicate) or not printable, or the next keyword does not follow it in the file, and error message is displayed and valid is set to false.

view()

The view method displays the states of a Turing machine.

States()
This will be the default constructor. Nothing to be initialized.

States(IN state_name:string)
This constructor initializes a new state given the input state.

total_number_of_states(): int
This method will return the total number of states.

3.5 Class:

Input_String_List

Description:

The list of strings of a Turing machine will contain a list of valid input strings that are able to be processed by the Turing machine. The list could be initially empty or not. There will be no limit on the length of the input string or number of input strings. The list will be able to be modified by either deleting or inserting new input strings. The entire list of input strings is written to an input string file when the application is terminated if the list is modified.

Associations:

The class Input_String_List is a component of the class User_Interface, receiving messages from the User_Interface.

Attributes:

input_string_list: string vector = {}

The attribute input_string_list will contain all of the valid input strings.

Methods:

insert_string(IN input_string: string)

The method insert_string will insert a new string into the input_string_list attribute. It is assumed that the string is valid.

remove_string(IN index: int)

The method remove_string will remove the string at position index from the attribute input_string_list.

is_string_in_list(IN input_string: string): bool

The method is_string_in_list will search for the specified string, and return true if a match is found, else it will return false.

extract_string(IN index: int): string

The method will return the string at location index from the attribute input_string_list.

size(): int

This method will return the size of the list.

3.6 Class:

Tape_Alphabet

Description:

The input alphabet of a Turing machine consist of printable characters from the ASCII character set, with the exception of the reserved characters, \, [,], <, and >. Lambda is specified by the reserved character \. The opening and closing square brackets will be used to encapsulate the state name when displaying the instantaneous description, and the characters, < and > will be used to truncate left and/or right side of the tape head. The tape alphabet will be obtained from the Turing machine definition file.

Associations:

The class Tape_Alphabet is a component of the class Turing_Machine, receiving messages from the Turing machine.

Attributes:

alphabet: character vector = {}

The attribute alphabet is a vector containing characters specified by the Turing machine definition file.

Methods:

load(INOUT definition: FILE, INOUT valid: bool)

The method load reads the input alphabet from the turing machine definition file. If the tape alphabet is not valid or not printable, or the next keyword does not follow it in the file, and error message is displayed and valid is set to false.

view()

The method view displays the input alphabet of the Turing machine.

is_element(IN value: char): bool

The method is_element return true if given an input character, the character exists in the tape alphabet, otherwise, it returns false.

3.7 Class:

Transition_Function

Description:

A Transition_Function in the Turing machine consists of a source state, character to be read, destination state, character to write, and the direction to move the tape head. On an input string, the string is processed one character at a time, assuming the string is valid, and will look for the appropriate transition function to determine what state to move next, and what character is to be written in the tape cell.

Associations:

The class Transition_Function is a component of the class Turing_Machine, receiving messages from the Turing machine.

Attributes:

transitions: Transition vector

The attribute transition will contain all of the transition function available to be used when processing an input string.

Methods:

load(INOUT definition: FILE, INOUT valid: bool)

The method load reads the transition functions from the turing machine definition file. If the transition are not valid (invalid format) or not printable, or the next keyword does not follow it in the file, and error message is displayed and valid is set to false.

view()

The method view will display all of the transition functions.

size(): int

The method size will return the total number of transitions in the attribute.

source_state(IN index: int): string

The method source_state will return the source state of a specified transition.

read_character(IN index: int): char

The method read_character will return the read_character of a specified transition.

destination_state(IN index: int): string

The method destination_state will return the destination state of a specified transition.

write_character(IN index: int): char

The method write_character will return the write_character of a specified transition.

is_defined_transition(IN source_state: string, IN read_character: char,
OUT destination_state: string,
OUT write_character: char,
OUT move_direction: Direction): bool

The method is_defined_transition will determine, given input parameters that define a transition, if that transition exists in the vector of transitions contained in the transitions attribute. If the transition exists in the attribute, then the method will assign values to the OUT parameters and return true, else false is returned.

is_source_state(IN state: string): bool

The method is_source_state, given string state parameter, determines if it is a source state or not. Returns true if it is a source state, else it returns false.

is_unique_transition(IN i_source: string, IN i_read: char, IN i_destination: string, IN i_write: char, IN i_move: DIRECTION): bool

This function will determine if, given the input transition, is unique.

3.8 Class:

Transition

Description:

A Transition in the Turing machine consists of a source state, character to be read, destination state, character to write, and the direction to move the tape head. On an input string, the string is processed one character at a time, assuming the string is valid, which may cause the Turing machine to change to a different state.

Associations:

The class Transition is a component of the class Transition_Function, receiving messages from the Transition_Function.

Attributes:

source: string

The attribute source will contain the source state.

read: char

The attribute read will contain the character to be read.

destination: string

The attribute destination will contain the name of the destination state.

write: char

The attribute write will contain the character to be written when this specific transition occurs.

move: Direction

The attribute move will contain the direction to move the tape head.

Methods:

Transition(IN source_state: string, IN read_character: char,
IN destination_state: string,
IN write_character: char,
IN move_direction: Direction)

The constructor Transition will initialize all of its attributes given the arguments provided to it.

source_state(): string

The method source_state will return the source state.

read_character(): char

The method read_character will return the value of the attribute read_character.
destination_state(): string

The method destination_state will return the value of the attribute destination_state.

write_character(): char

The method write_character will return the value of the attribute write_character.

move_direction(): Direction

The method move_direction will return the value of the attribute move_direction.

3.9 Class:

User_Interface

Description:

The Turing machine will have a user interface that will prompt a user for a command. Depending on the input command, the appropriate information will be displayed on the console. The user interface will only accept one letter commands, whose description can be displayed by enabling help while running the application.

Associations:

The class User_Interface will contain the instance of the Turing_Machine, and will communicate with it to execute the commands specified.

Attributes:

configuration_settings: Configuration_Settings

The attribute configuration_settings will contain the three configuration settings, initially with default values, and will be able to be modified by the user.

turing_machine: Turing_Machine

The attribute turing_machine is a Turing Machine that will be used to process an input string given a valid Turing machine definition file.

input_string_list: Input_String_List

The attribute input_string_list will maintain a list of all valid input strings that are read from the input string file, and added by the user during operation.

Methods:

User_Interface(IN definition_file_name: string, IN input_string_file_name: string)

The constructor User_Interface will initialize the attribute turing_machine with the definition_file_name. If the input_string_file_name exists, the strings will be read and loaded into the input_string_list.

write_to_string_file(IN stringFile_path: string)

The method will write to a string file when the user decides to exit the application. If the string name is empty, meaning no file exists, the file is created and the contents of the input string list are written to the file. If the file does exists, the contents of the file are replaced with the contents of the input string list.

programFlowControl()

The method programFlowControl will direct control to a method given the users input command.

```
while(exitFlag == false) {
      switch(CommandPrompt())
      {
         case 'D':
            this->DeleteCommand();
            break;
         case 'X':
            cout << "Input string file successfully written" << endl;
            exitFlag = true;
            break;
         case 'H':
```

```
            this->HelpCommand();
            break;
        case 'I':
            this->InsertCommand();
            break;
        case 'L':
            this->ListCommand();
            break;
        case 'Q':
            this->QuitCommand();
            break;
        case 'R':
            this->RunCommand();
            break;
        case 'E':
            this->SetCommand();
            break;
        case 'W':
            this->ShowCommand();
            break;
        case 'T':
            this->TruncateCommand();
            break;
        case 'V':
            this->ViewCommand();
            break;
        case '\t':
            break;
        default:
            cout << "Invalid Command" << endl;
            break;
    }
}
```

commandPrompt(): char

The method CommandPrompt will prompt the user for an input command and will parse the input appropriately.

helpCommand()

The method HelpCommand will toggle help user with prompts or not.

showCommand()

The method ShowCommand will display the status of the application.

- Information displayed (order does not matter and the format is up to the developer)
  - Course, Semester, year, instructor, Author(me), and version of Application
- Includes configuration settings:
  - whether or not help is provided
  - maximum number of transitions to perform at a time
  - maximum number of cells to the left and right of tape head to display in ID (instantaneous

    description)
- Name of TM (without extension .DEF)
- status of TM
- If TM is running
  - Show input string and total number of transitions that have been performed
- If TM has completed operation
  - last input string
  - whether it was accepted or rejected, or operation was terminated before normal completion.
  - Total number of transitions performed

viewCommand()

The method ViewCommand will display the Turing machine definition file contents.

listCommand()

The method ListCommand will display a list of input strings along with its index (starting at 1).

insertCommand()

The method InsertCommand will allow the user to insert a string into the input_string_list. Must verify that the input string is valid by checking the input alphabet.

deleteCommand()

The method DeleteCommand will delete an input string from the input string list. The string to be deleted will be identified by entering the string number.

setCommand()

The method SetCommand will the maximum number of transitions to perform.

truncateCommand()

The method TruncateCommand will truncate the instantaneous description by editing the maximum number of cells.

runCommand()

The method RunCommand will allow the user to trace operation of a Turing machine on an input string selected from the input string list. If the Turing machine is not running, the user is prompted for input string number upon which to run Turing machine. If the user selects a non existing input string, and error is displayed and command is terminated. If the input is valid, along with every instantaneous description, the number of transitions are also displayed.

is_valid_digit(IN input:string):bool
This method will determine if the input consists of all digits. Returns true if the input is all digits.

quitCommand()

The method QuitCommand will quit operation of the Turing machine on an input string. A message should indicate that the Turing machine has not accepted or rejected the input string. The total number of transitions should be displayed, along with the input string. If the Turing machine is not running on an input string, an error message is displayed.

exitCommand()

The method will provides no opportunity for the user to confirm or cancel termination of application. If an input string was inserted or deleted from the list by a command during the session, the entire list is written to the input string file, therefore, this will replace any original file. A message should be provided to user indicating that the input string file was successfully written, or an error message is displayed.

3.10 Class:

Turing_Machine

Description:

The Turing Machine will control the flow of information to and from various objects that the class contains to perform the desired operations that the user requests.

Associations:

The class Turing_Machine will contain a Tape, Input_Alphabet, Tape_Alphabet, Transition_Function, Transition, States, Final_States, User_Interface, Input_String_List, and Configuration_Settings class, sending messages to these classes to perform the desired actions of the application. The Turing_Machine is a component of the User_Interface.

Attributes:

tape: Tape

The attribute Tape 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 input string.

input_alphabet: Input_Alphabet

The attribute input_alphabet will contain the input alphabet accepted for input strings.

tape_alphabet: Tape_Alphabet

The attribute tape_alphabet will contain the tape alphabet.
transition_function: Transition_Function

The attribute transition_function will contain a vector of all legal transition.

states: States

The attribute states will contain a group of states defined by the turing machine definition file.

final_states: Final_States

The attribute final_states will contain a group of final states.

description: string vector

The attribute description will contain a description of the specified Turing machine.

initial_state: string

The attribute initial_state will contain the name of the initial state.

current_state: string

The attribute current_state will contain the name of the current state.

original_input_string: string

The attribute original_input_string will contain a copy of the original input string.

number_of_transitions: int

The attribute number_of_transitions will be used to keep track of the number of transitions performed on an input string.

valid: bool

The attribute valid will be used to signal whether the specified Turing machine is valid or not.

used: bool

The attribute used will be used to signal if the original_input_string has been used.

operating: bool

The attribute operating will be used to signal if the Turing machine is operating or not.

accepted: bool

The attribute accepted will be set to true if the input string was accepted.

rejected: bool

The attribute will be set to true if an input string was rejected.

Methods:

Turing_Machine(IN definition_file_name: string)

The constructor will take in a file name to obtain information to initialize its attributes.

view_definition()

The method will display all of its contents pertaining to the definition of the Turing machine.

view_instantaneous_description(IN maximum_number_of cells: int)

The method will display the instantaneous description displaying maximum_number_of_cells to the left and right of the current cell, possibly truncating using the reserved characters.

initialize(IN input_string: string)

The method will initialize its attributes to prepare to process an input string, and call its attribute tape to

initialize the tape to the input string followed by a blank character. Also, it will maintain a copy of the input string using the attribute original_input_string.

perform_transitions(IN maximum_number_of_transitions: int)

The method will call the tape and transition_function attribute to perform maximum_number_of_transitions. This will be done by identifying if there is a valid transition, and then performing the operations requested by the transition to update the state.

terminate_operation()

The method will terminate operation on the current input string if it is running. A message will be displayed that says the the Turing machine has not accepted or rejected the input string. The total number of transitions will be displayed, along with the input string. If the Turing machine is not running on an input string an error is displayed.

input_string(): string

The method returns the value of input_string.

total_number_of_transitions(): int

The method returns the value of total_number_of_transitions.

is_valid_definition(): bool

The method returns true if the Turing machine definition file was valid, else it returns false.

is_valid_input_string(IN value: string): bool

The method will determine if the given string is valid, if it is valid, it will return true, else false. This will be determined by comparing each character to the input alphabet.

is_used(): bool

The method returns the value of the attribute used.

is_operating(): bool

The method returns the value of the attribute operating.

is_accepted_input_string(): bool

The method determines returns the value of accepted attribute.

is_rejected_input_string(): bool

The method returns the value of the attribute rejected.

read_definition_file(string definitionFile): bool
This method will be the driver for beginning to read the Turing machine definition file. This method will create the valid variable and call the load method to start reading the definition file, and will call all of components of Turing machine to read their part of the file.

load(ifstream &definitionFile, bool &valid)
This method will read the description from the Turing machine definition file. If successful it will set valid to true.

is_valid_input_alphabet(): bool
This method will verify that the input alphabet is part of the tape alphabet.

is_valid_final_states(): bool
This method will verify that the final states are a part of the set of states.

final_states_have_transition(): bool
This method will check if final states have transitions. If they do, true will be returned.

Turing_Machine()
The constructor will initialize the attributes number_of_transitions, used, valid, operating, accepted, and rejected.

3.11 Class:

Tape

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 cel 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 on cell to the left or right. The tape exists only as part of a Turing machine.

Associations:

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

Attributes:

cells: string = ” ”

The attribute cells is a dynamically growing character string containing the Turing machine tape. Whenever necessary, it may be extended by appending a blank character.

current_cell: int = 0

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

blank: char = ' '

The blank character of the Turing machine is contained in the attribute blank.

Methods:

load(INOUT definition: FILE, INOUT valid: bool)

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, and error message is displayed and valid is set to 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: char, 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. If no storage is available for this character, an out of storage error will be thrown, 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 : int) : string

The method left returns a character string of 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 : int) : string

The method right returns a character string of 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 cell 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(): char

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

blank_character(): char

The method blank_character returns the blank character of the Turing machine.

is_first_cell(): bool

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.

3.12 Class:

Crash

Description: This class will be used for exception handling.
Attributes: No attributes
Methods:

Crash(IN reason:string)
This constructor takes in a string reason when a exception occurs.

4.0 User Interface

4.1 Command Line Invocation

$./TMAPP A
Successfully Loaded!

$./TMAPP
Error!
Usage Message

4.2 Help Command

Command: h
Help Enabled!

Delete	D	Delete input string from list
Exit	X	Exit application
Help	H	Help user with prompts or not
Insert	I	Insert input string into list
List	L	List input strings
Quit	Q	Quit operation of Turing machine on input string
Run	R	Run turing machine on input string
Set	E	Set maximum number of transitions to perform
Show	W	Show status of Application
Truncate	T	Truncate instantaneous description
View	V	View turing machine

Command:

4.3 Show Command

Command: w
Course : Cpts 322 - Software Engineering Principles I
Semester : Spring
Year : 2015
Instructor : Neil B. Corrigan
Author : Andres Herrera
Version : 1.0

Help enabled : NO
Max Number of transitions to perform at a time : 5
Max number of cells displayed on tape head (Left & Right) : 32

Touring Machine Name : TM
Status : TM has completed operation on an input string during session
Last input String : AABB Accepted 20 Transitions

Command:

4.4 View Command

Command : v
Description : This turing machine accpets the language of one or more a's followed by
the same number of b's.

Q = {s0, s1, s2, s3, s4}
Sigma = {A, B}
Gamma = {A, B, X, Y, -}

Transitions Function
Delta(s0, A) = (s1, X, R)
Delta(s0, Y) = (s3, Y, R)
Delta(s1, A) = (s1, A, R)
Delta(s1, B) = (s2, Y, L)
Delta(s1, Y) = (s1, Y, R)
Delta(s2, A) = (s2, A, L)
Delta(s2, X) = (s0, X, R)
Delta(s2, Y) = (s2, Y, L)
Delta(s3, Y) = (s3, Y, R)
Delta(s3, -) = (s4, -, R)

Initial state = s0
Blank character = -
F = {s4}

Command :

4.5 List Command

Command : l
1. A
2. AB
3. \
4. AAABB
5. AAAAAAAAAABBBBBBBBBB
6. AABB
7. AAAAAABBBBBB
8. BA
9. ABA
10. BB

Command :

4.6 Insert Command

Command : i
Input String : AABB
Input String Inserted!

Command :

4.7 Delete Command

Command : d
Input String Number : 3
String Deleted

Command :

4.8 Set Command

Command : e
Maximum number of Transitions[1] : 4
Settings changed!

Command :

4.9 Truncate Command

Command : t
Maximum number of cells[32] : 50
Success!

Command :

4.10 Run Command

Command : r
Input string number : 4
0. [s0]AAABB
5. XA[s1]AYB

Command :

4.11 Quit Command

Command : q
Input string AAABBBB not accepted or rejected in 32 transitions

Command :

4.12 Exit Command

Command : x
Input string file successfully written
$

5.0 Files

5.1 Turing Machine Definition File

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

STATES: s0 s1 s2 s3 s4

INPUT_ALPHABET: a b

TAPE_ALPHABET: a b X Y -

TRANSITION_FUNCTION:
Delta(s0, A) = (s1, X, R)
Delta(s0, Y) = (s3, Y, R)
Delta(s1, A) = (s1, A, R)
Delta(s1, B) = (s2, Y, L)
Delta(s1, Y) = (s1, Y, R)
Delta(s2, A) = (s2, A, L)
Delta(s2, X) = (s0, X, R)
Delta(s2, Y) = (s2, Y, L)
Delta(s3, Y) = (s3, Y, R)
Delta(s3, -) = (s4, -, R)

INITIAL_STATE: s0

BLANK_CHARACTER: -

FINAL_STATES: S4

5.2 Input String File

a
ab
\
aaabb
aaaabbbb
aabb
aaabbb
ba
aba
bb

References

All information was obtained from class notes.

Appendix

No additional material.