Turing Machine Design Specification

CPTS322 Software Engineering I

Christian Penick

April 29, 2024

Table of Contents

[Revision History 3](#_Toc164953600)

[List of Figures 4](#_Toc164953601)

[Introduction 5](#_Toc164953602)

[Architecture 6](#_Toc164953603)

[Data Dictionary 13](#_Toc164953604)

[Main 13](#_Toc164953605)

[Commands 14](#_Toc164953606)

[Input\_Strings 15](#_Toc164953607)

[Configuration\_Settings 16](#_Toc164953608)

[Turing\_Machine 17](#_Toc164953609)

[Transition\_Function 22](#_Toc164953610)

[Transition 24](#_Toc164953611)

[Tape 26](#_Toc164953612)

[Tape\_Alphabet 30](#_Toc164953613)

[Input\_Alphabet 31](#_Toc164953614)

[States 33](#_Toc164953615)

[User Interface 35](#_Toc164953616)

[Command Line Invocation 35](#_Toc164953617)

[Help Command 35](#_Toc164953618)

[Show Command 36](#_Toc164953619)

[View Command 36](#_Toc164953620)

[List Command 37](#_Toc164953621)

[Insert Command 37](#_Toc164953622)

[Delete Command 37](#_Toc164953623)

[Set Command 38](#_Toc164953624)

[Truncate Command 39](#_Toc164953625)

[Run Command 40](#_Toc164953626)

[Quit Command 41](#_Toc164953627)

[Exit Command 41](#_Toc164953628)

[Files 42](#_Toc164953629)

[References 43](#_Toc164953630)

# Revision History

First edition.

April 9, 2024.

Second edition.

April 29, 2024.

Included a list of figures.

Renamed method **get\_command** from class **Commands** to current name **handle\_user\_command** and expanded its description.

Removed method **is\_first\_cell** from class **Tape**.

# List of Figures

[Figure 1 6](#_Toc164783994)

[Figure 2 7](#_Toc164783995)

[Figure 3 7](#_Toc164783996)

[Figure 4 7](#_Toc164783997)

[Figure 5 7](#_Toc164783998)

[Figure 6 8](#_Toc164783999)

[Figure 7 8](#_Toc164784000)

[Figure 8 9](#_Toc164784001)

[Figure 9 10](#_Toc164784002)

[Figure 10 10](#_Toc164784003)

[Figure 11 11](#_Toc164784004)

[Figure 12 11](#_Toc164784005)

[Figure 13 11](#_Toc164784006)

# Introduction

This document provides a complete object-oriented design that can be used to implement a Turing Machine (TM). The design detailed in this document will be followed by Christian Penick to implement the TM described in Dr. Corrigan’s CPT\_S 322 course.

The remainder of this document will provide a complete class diagram in UML notation to show the architecture of the TM using object-oriented design, a data dictionary to describe each class shown on the class diagram, examples of how the user interface will appear in the TM application, examples of the text files used by the TM application, and a list of references.

# Architecture

A diagram of a machine

Description automatically generated

Figure 1

|  |
| --- |
| Main  Figure 2 |
|  |
| main(in TM\_file\_name: String): Integer |

|  |
| --- |
| Commands  Figure 3 |
| handle\_user\_command(in cmd: Char,  inout input\_strings: Vector\_String,  inout valid: Bool,  in input\_file\_name: String,  inout num\_transitions\_per\_run: Integer,  inout num\_cells\_in\_id: Integer): Integer |

|  |
| --- |
| Input\_Strings  Figure 4 |
| input\_strings: Vector\_String  modified: Boolean |
| load (inout definition: file)  input\_string(in input\_string\_file\_name: String  inout str\_list\_modified: Boolean): Vector\_String |

|  |
| --- |
| Configuration\_Settings  Figure 5 |
| max\_number\_of\_cells: Integer  max\_number\_of\_transitions: Integer |
| set\_truncate(in maximum\_number\_of\_cells: Integer)  set\_transition(in maximum\_number\_of\_transitions: Integer)  get\_truncate\_amount(): Integer  get\_transition\_amount(): Integer |
| Transition\_Function  Figure 6 |
|  |
| load (inout definition: file, inout valid: Boolean)  validate (inout valid: Boolean)  view()  find\_transition(in source\_state: String,  in read\_character: Character  out destination\_state: String  out write\_character: Character,  out move\_direction: Direction,  out found: Boolean) |

|  |
| --- |
| Transition  Figure 7 |
| Source: String  Read: Character  Destination: String  Write: Character  Move: Direction |
| transition(in source\_state: String,  in read\_character: Character  in destination\_state: String  in write\_character: Character,  in move\_direction: Direction)  source\_state(): String  read\_character(): Character  destination\_state(): String  write\_character(): Character  move\_direction(): Direction |

|  |
| --- |
| Turing\_Machine  Figure 8 |
| description: String\_Vector  initial\_state: String  current\_state: String  original\_input\_string: String  number\_of\_transitions: Integer  valid: Boolean  used: Boolean  operating: Boolean  accepted: Boolean  rejected: Boolean |
| Turing\_Machine(in definition\_file: String)  view\_definition()  view\_instantaneous\_description(in maximum\_number\_of\_cells: integer)  initialize(in input\_string: String)  perform\_transitions(in maximum\_number\_of\_transitions: integer)  terminate\_operation()  input\_string(): String  total\_number\_of\_transitions(): Integer  is\_valid\_definition(): Boolean  is\_valid\_input\_string(in value: String): Boolean  is\_used(): Boolean  is\_operating(): Boolean  is\_accepted\_input\_string(): Boolean  is\_rejected\_input\_string(): Boolean |

|  |
| --- |
| Tape  Figure 9 |
| cells: String= “”  current\_cell: integer = 0  blank\_character: Character = ‘’ |
| load(inout definition: file,  Inout valid: Boolean)  validate(inout valid: Boolean)  view()  initialize(in input\_string: String)  update(in write\_character: Character,  in move\_direction: Direction)  left(in maximum\_number\_of\_cells: integer): String  right(in maximum\_number\_of\_cells: integer): String  current\_character(): Character  is\_first\_cell(): Boolean |

|  |
| --- |
| Input\_Alphabet  Figure 10 |
| alphabet: Character\_Vector = {} |
| load(inout definition: file,  inout valid: Boolean)  validate(inout valid: Boolean)  view()  is\_element(in value: Character): Boolean |

|  |
| --- |
| Tape\_Alphabet  Figure 11 |
| alphabet: Character\_Vector = {} |
| load (inout definition: file,  Inout valid: Boolean)  view()  is\_element(in value: Character): Boolean |

|  |
| --- |
| States  Figure 12 |
| names: String\_Vector = {} |
| load (inout definition: file,  inout valid: Boolean)  view()  is\_element(in value: String): Boolean |

|  |
| --- |
| Final\_States  Figure 13 |
| Names: String\_Vector = {} |
| load (inout definition: file,  inout valid: Boolean)  view()  is\_element(in value: String): Boolean |

|  |
| --- |
| TM\_utils  Figure 13 |
|  |
| check\_unique\_set(in element\_list: Vector,  in current\_component: String,  inout valid: Boolean)  uppercase(in value: String): String  parse\_string\_to\_int(in value: String): Integer |

# Data Dictionary

**Class**

## Main

**Description**

The class **Main** is responsible for taking the command line arguments and invoking the **Turing\_Machine** class with the string provided at command line. **Main** also houses the **Commands** class to take commands from the user.

**Associations**

The class **Main** has the association **performs** with the class **Commands**, allowing **Main** to send messages to **Commands**.

**Attributes**

None.

**Methods**

**main**(TM\_file\_name: String): Integer

The method **main** takes the name of the TM definition file/TM input string file from the command line for use in the TM class. The method returns 0 on success, or 1 otherwise.

**Class**

## Commands

**Description**

The class **Commands** sends messages to the appropriate classes to add input strings, update configuration settings, or perform an action on the **Turing\_Machine** class.

**Associations**

The class **Commands** has an association **specify** with the class **Input\_Strings** allowing **Commands** to alter the list of input strings.

The class **Commands** has an association **assign** with the class **Configuration\_Settings** allowing **Commands** to alter the configuration settings: number of transitions to perform, and number of cells to show in instantaneous transition.

**Attributes**

None.

**Methods**

**handle\_user\_command()**

The method **handle\_user\_command()** reads user input from the console, validating to ensure that a single character was entered. Based off the character, the method then sends the appropriate message to perform the given command.

Character mapping

|  |  |
| --- | --- |
| Characters | Command |
| ‘d’, ‘D’ | Delete input string |
| ‘e’, ‘E’ | Set max number of transitions |
| ‘h’, ‘H’ | Display list of commands |
| ‘i’, ‘I’ | Insert input string |
| ‘l’, ‘L’ | View input strings |
| ‘q’, ‘Q’ | Quit TM running on input string |
| ‘r’, ‘R’ | Run TM on input string |
| ‘t’, ‘T’ | Truncate number of cells in ID |
| ‘v’, ‘V’ | Display TM definition and description |
| ‘w’, ‘W’ | Show TM info |
| ‘x’, ‘X’ | Exit TM application |

**Class**

## Input\_Strings

**Description**

The class **Input\_Strings** stores the contents of the input string file. These contents may be modified. All input strings must be composed of characters from the set of characters in the input alphabet. If the original list of input strings is altered in any way, it will overwrite the input string file that was used for initialization.

**Associations**

None.

**Attributes**

**input\_strings: Vector\_String**

The attribute input\_strings stores the strings from the string input file. This vector of strings may be modified. Modifying this vector of strings sets the Boolean **modified** to true.

**modified: Boolean**

The attribute **modified** is used to signify whether the list of input strings has been modified.

**Methods**

**load (inout input\_string: file)**

The method **load** loads the input string file into the attribute **input\_strings**. Each line of the input string file is an input string that must composed of characters from the input alphabet. The empty string is specified by reserved character ‘\’ appearing on a line by itself. If any line is empty or contains a character not in set of input alphabet, the line will be discarded and an error message will be displayed on the console. Duplicate input strings are discarded. There is no limit on the length of string in the input string file or on the number of strings. If any modifications are made that cause the list of input strings stored in the class **Input\_Strings** to differ from the list in the input string file, the Boolean **modified** is set to true.

**Class**

## Configuration\_Settings

**Description**

The class **Configuration\_Settings** is used to store the maximum number of characters to display for the instantaneous description and to store the maximum number of transitions to perform for the Turing Machine.

**Associations**

None.

**Attributes**

**max\_number\_of\_cells: Integer = 32**

The attribute **max\_number\_of\_cells** is used to store the maximum number of cells to display for the instantaneous description.

**max\_number\_of\_transitions: Integer = 1**

The attribute **max\_number\_of\_transitions** is used to store the maximum number of transitions that the Turing Machine may perform at once.

**Methods**

**set\_truncate(in maximum\_number\_of\_cells: Integer)**

The method **set\_truncate** is used to set the attribute **max\_number\_of\_cells**. The method’s parameter must be a non-negative number. If the parameter is zero or less, the attribute **max\_number\_of\_cells** is not set.

**set\_transition (in maximum\_number\_of\_transitions: Integer)**

The method **set\_transition** is used to set the attribute **max\_number\_of\_transitions**. The method’s parameter must be a non-negative number. If the parameter is zero or less, the attribute **max\_number\_of\_transitions** is not set.

**get\_truncate\_amount(): Integer**

The method **get\_truncate\_amount** returns the value of **max\_number\_of\_cells.**

**get\_transition\_amount(): Integer**

The method **get\_transition\_amount** returns the value of **max\_number\_of\_transitions.**

**Class**

## Turing\_Machine

**Description**

The class **Turing\_Machine** contains the necessary classes (as described in the *Associations* section) that allow the class to operate like a theoretical TM. To this end, the **Turing\_Machine** class consists of a tape, input alphabet, tape alphabet, transition functions and states. The **Turing\_Machine** class can load a TM definition, initialize the tape, perform transitions, terminate operation, store the acceptance/rejection of an input string, output its definition, output its instantaneous description, output its original input string, and output the number of transitions performed.

**Associations**

**Turing\_Machine** contains a class **Tape**, allowing the TM to keep track of the contents of the tape, gather relevant information for transitions, and perform transitions on the tape.

**Turing\_Machine** contains a class **Input\_Alphabet**, allowing the TM to fill the tape with characters and validate input strings.

**Turing\_Machine** contains a class **Tape\_Alphabet**, allowing the TM to define transition functions and validate the input alphabet.

**Turing\_Machine** contains a class **Transition\_Function**, allowing the TM to store and validate the transition functions of the definition file to perform transitions on the tape.

**Turing\_Machine** contains a class **States**, allowing the TM to hold all the available states.

**Attributes**

**description: String\_Vector**

The attribute **description** is a vector of strings, allowing the TM to store and view its description from the definition file.

**initial\_state: String**

The attribute **initial\_state** is a string used to store and view the initial state of the TM.

**current\_state: String**

The attribute **current­\_state** is a string used to store the current state of the TM. This allows the current state to be viewed and to find the correct transition function.

**original\_input\_string: String**

The attribute **original\_input\_string** is a string used to store the original state of the TM. This allows the original state of the TM to be viewed.

**number\_of\_transitions: Integer**

The attribute **number\_of\_transitions** is used to store and view the number of transitions the TM has performed.

**valid: Boolean**

The attribute **valid** is used to check whether the various components of the TM are valid. These checks are performed via passing the Boolean **valid** to validation functions that certain classes associated with the class **Turing\_Machine** use.

**used: Boolean**

The attribute **used** is set to true if operation on an input string has been completed (the input string was accepted, rejected, or operation of the TM on that was input string was quit).

**operating: Boolean**

The attribute **operating** is used to store whether the TM is currently operating upon an input string or not.

**accepted: Boolean**

The attribute **accepted** is used to store whether the TM accepted the original input string or not.

**rejected: Boolean**

The attribute rejected is used to store whether the TM rejected the original input string or not.

**tape: Tape**

The attribute **tape** is an object of the class **Tape**, allowing the TM to make use of its tape

and the blank character.

**input\_alphabet: Input\_Alphabet**

The attribute **input\_alphabet** is an object of the class **Input\_Alphabet**, allowing the TM to validate input strings.

**tape\_alphabet: Tape\_Alphabet**

The attribute **tape\_alphabet** is an object of the class **Tape\_Alphabet**, allowing the TM to validate the input alphabet and validate the transition functions.

**transition\_function: Transition\_Function**

The attribute **transition\_function** is an object of the class **Transition\_Function**, allowing the TM to store zero or more transition functions.

**states: States**

The attribute **states** is an object of the class **states**, allowing the TM to store states.

**final\_states: Final\_States**

The attribute **final\_states** is an object of the class **final\_states**, allowing the TM to know if an input string has been accepted.

**Methods**

**Turing\_Machine(in definition\_file\_name: String)**

The method **Turing\_Machine** is a constructor taking the name of a TM definition file passed as a string. This method will call the **load** and **validate** method of the various classes contained in the **Turing\_Machine** class to initialize the TM and ensure the given TM definition is valid.

**view\_definition()**

The method **view\_definition** uses the **view** method of the various classes contained in the **Turing\_Machine** class to view the TM definition.

**view\_instantaneous\_description(in maximum\_number\_of\_cells: Integer)**

The method **view\_instantaneous\_description** uses the **current\_state** attribute and **tape** class to view the current state of the TM along with **maximum\_number\_of\_cells** symbols to the left of the tape head and **maximum\_number\_of\_cells** to the right of the tape head, including the tape head. If more than **maximum\_of\_cells** of tape characters exist to the left and right of the tape head, then the reserved symbols “<” and “>” are used to signify truncation.

**initialize(in input\_string: String)**

The method initialize sets the beginning of the TM’s tape to the characters in the string **input\_string**.

**perform\_transitions(in maximum\_number\_of\_transitions: Integer)**

The method **perform\_transitions** can be used while the TM is operating. This method will check the current state and character at the tape head position and find an appropriate transition function to use. This method performs up to **maximum\_number\_of\_transitions**. If the tape head is on the first cell, and the transition function requires the tape head to move to the left, the TM will crash. If TM is in a state and a character is read from tape for which no transition is defined, the TM will crash. If the TM crashes, the TM rejects the original input string. If the TM enters a final state, the TM accepts the original input string. The method **perform\_transitions** may be used until the TM accepts/rejects the original input string or until the TM application runs out of storage which will terminate the TM application. If the next transition will cause the TM to crash, then display the **instantaneous description** right before the crash.

**terminate\_operation**()

The method **terminate\_operation** will terminate operation of the TM on an input string before completion. This method will set the Boolean **operating** to false, and ensure that the Booleans **operating**, **accepted**, and **rejected**, are all set to false. **The** method **terminate\_operation** is used to help implement the **Quit (Q)** command for the user interface.

**input\_string(): String**

The method **input\_string** returns the original input string the TM began operation on. This method will be used in the user interface **Show(W)** command.

**total\_number\_of\_transitions(): Integer**

The method **total\_number\_of\_transitions** returns the number of transitions the TM has performed on the current input string. This method will be used in the user interface **Show(W)** command.

**is\_valid\_definition(): Boolean**

The method **is\_valid\_definition** returns the state of the class **Turing\_Machine**’s Boolean **valid.** Because this internal Boolean **valid** is passed to methods of objects of the **Turing\_Machine** class, the method **is\_valid\_definition** returns whether the TM definition file provided a valid TM definition or not.

**is\_valid\_input\_string(in value: String): Boolean**

The method **is\_valid\_input\_string** is used to determine if a given string contains characters only within the set of TM’s input\_alphabet. This method will return **true** if the string is solely composed of characters from the TM’s input alphabet, otherwise, it will return **false.** This method makes use of the internal object of type **Input\_Alphabet** to validate the given input string.

**is\_used(): Boolean**

The method **is\_used** returns the value of the internal Boolean **used** to indicate that the TM has accepted, rejected, or quit while running on an input string.

**is\_operating(): Boolean**

The method **is\_operating** returns the value of the internal Boolean **operating**. This method is used in the user interface command logic for **Quit (Q)**.

**is\_accepted\_input\_string(): Boolean**

The method **is\_accepted\_input\_string** returns the value of the internal Boolean **accepted**. This method is used in the user interface command **Run (R)** to display the result of the TM.

**is\_accepted\_input\_string(): Boolean**

The method **is\_rejected\_input\_string** returns the value of the internal Boolean **rejected**. This method is used in the user interface command **Run (R)** to display the result of the TM.

**Class**

## Transition\_Function

**Description**

The class **Transition\_Function** manages a vector of the TM’s transition functions. The class will initialize the vector via reading the TM definition file and will validate the transitions to ensure no transitions are defined to leave the final state and so that any state has at most one transition function that reads a given tape character.

**Association**

The association **is based on** with the class **Tape\_Alphabet** ensures that a transition function may only use characters from the TM’s tape alphabet.

The association **is based on** with the class **States** ensures that a transition function may only use states with the TM’s set of states.

The association **does not leave** with the class **Final\_States** ensures that a transition function may not leave a final state.

The association **contains** with the class **Transition** allows the **Transition\_Function** class to store the transitions of the TM.

**Attributes**

**transitions: Transition\_Vector = {}**

The attribute **transitions** is a vector of objects of the class **Transition** containing the transition functions of the TM.

**Methods**

**load(inout definition: File,**

**inout valid: Boolean)**

The method **load** reads the transition functions from the TM definition file. If a component of a transition is missing, or if a character read/write component of the definition is a string or is part of the reserved characters set, an error is displayed to the console, and the Boolean **valid** is set to false, otherwise, **valid** is set to true.

**validate(inout valid: Boolean)**

The method **validate** sets the Boolean **valid** if a transition uses states not in the TM’s set of states, uses tape characters not in the TM’s tape alphabet, or has multiple transition functions defined for a single state/read tape character pair.

**view()**

The method **view** displays the transition functions of the TM.

**find\_transition(in source\_state: String,**

**in read\_character: Character**

**out destination\_state: String**

**out write\_character: Character,**

**out move\_direction: Direction,**

**out found: Boolean)**

The method **find\_transition** takes in all the components required to define a transition function and sets the Boolean **found** to true if a transition function definition with the components exists in the TM’s set of transition functions. Otherwise, it sets the Boolean **found** to be false.

**Class**

## Transition

**Description**

A TM utilizes transitions functions to leave one state and enter another state given the current state, character at current tape head position, destination state, character to write to current position of tape head, and direction to move tape head. A state may only have one transition function leaving it as this TM is deterministic. If TM is in a state and a character is read from tape for which no transition is defined, TM will crash. When TM crashes, it terminates execution without accepting input string. When TM reaches a final state or halt state, it terminates and accepts the original input string. No transitions are defined out of a final state.

**Association**

None.

**Attributes**

source: String

The attribute **source** is used to store the source state that the TM must exist in currently.

**read: Character**

The attribute **read** is used to store the character that must be at the TM tape head’s current position.

**destination: String**

The attribute **destination** is used to store the destination state that the TM must enter.

**write: Character**

The attribute **write** is used to store the character that must be written at the TM tape head’s current position.

**move: Direction**

The attribute **move** is used to store the direction the TM tape head must move.

**Methods**

**Transition(in source\_state: String,**

**in read\_character: Character,**

**in destination\_state: Character,**

**in write\_character: Character,**

**in move\_direction: Direction)**

The method **Transition** is a constructor that takes in the components of a TM transition function and sets its attributes to the respective parameters.

**source\_state(): String**

The method **source\_state** returns the class attribute **source\_state**. This is used for finding the correct transition function to use and to view the transition functions of the TM.

**read\_character(): Character**

The method **read\_character** returns the class attribute **read\_character**. This is used for finding the correct transition function to use and to view the transition functions of the TM.

**destination\_state(): String**

The method **destination\_state** returns the class attribute **destination\_state**. This is used for finding the correct transition function to use and to view the transition functions of the TM.

**write\_character(): Character**

The method **write\_character** returns the class attribute **write\_character**. This is used for finding the correct transition function to use and to view the transition functions of the TM.

**move\_direction(): Direction**

The method **move\_direction** returns the class attribute **move\_direction**. This is used for finding the correct transition function to use and to view the transition functions of the TM.

**Class**

## Tape

**Description**

The tape of a TM 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 TM, 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 TM.

**Associations**

The class Tape is a component of the class **Turing\_Machine**, receiving messages delegated to it by the TM.

The association **is not filled from** with the class **Input\_ Alphabet** is used to validate that the blank character for initialization and extension of the TM tape is not in the input alphabet.

The association **is filled from** with the class **Tape\_Alphabet** is used to validate that the blank character for initialization and extension of the TM tape is in the tape alphabet.

**Attributes**

**cells: String = “”**

The attribute **cells** is a dynamically growing character string containing the TM tape.

In performing an update, the tape may be extended by appending a blank character.

**current cell: Integer = 0**

The index of the current cell on the TM tape is stored in the attribute **current \_cell**.

**blank character: Character = ‘’**

The blank character used to initialize and extend the TM tape is contained in the attribute **blank \_character**.

**Methods**

**load(inout definition: File, inout valid: Boolean)**

The method **load** reads the blank character from the TM 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 valid: Boolean)**

The method **validate** determines if the blank character of the TM is in the tape alphabet but not 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 false.

**view()**

The method view displays the blank character of the TM.

**initialize (in input \_string: String)**

The method **initialize** sets the TM 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 TM 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.

Pseudocode on following page.

pseudocode:

method update (in write character: Character,

in move direction: Direction) is

begin

if move direction = L and current cell - 0 then

return;

end if;

if move\_direction = R and current\_cell = cells. length() - 1 then

cells. append (blank\_character) :

end if;

cells (current\_cell] := write\_character;

if move direction = L then

current\_cell := current\_ cell - 1:

else

current\_cell := current\_ cell + 1;

end if;

end update;

**left(in maximum\_number\_of\_cells: integer): String**

The method **left** returns a character string of up to the maximum number of cells from the TM 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 of up to the maximum number of cells from the TM 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 TM tape.

**is\_first\_cell(): Boolean**

The method **is\_first\_cell** returns a value of true if the current cell on the TM tape is

the first cell, indicated by the index 0. Otherwise, it returns a value of false.

**Class**

## Tape\_Alphabet

**Description**

The tape alphabet of a TM is a set of printable characters from the Ascii character set except for the reserved characters “\”, “[“, “]”, “<”, “>”. The characters in the tape alphabet must be unique as the tape alphabet is a set. White space characters are excluded from the tape alphabet. The tape alphabet includes the input alphabet and a blank character. The tape alphabet is the only set of characters that may appear on the TM tape. The tape alphabet is used to validate the input alphabet and to validate transition functions. A tape alphabet does not have a maximum size.

**Associations**

The class **Tape\_Alphabet** is a component of the class **Turing\_Machine**, receiving messages delegated to it by the TM.

**Attributes**

**alphabet: Character\_Vector = {}**

The attribute **alphabet** is a vector of characters containing the tape alphabet of the TM.

**Methods**

**load(inout definition: File,**

**inout valid: Boolean)**

The method **load** reads the tape alphabet from the TM definition file. If any of the tape alphabet characters that are read in are reserved characters 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.

**view()**

The method **view** displays the tape alphabet of the TM.

**is\_element(in value: Character): Boolean**

The method **is\_element** returns true if the character passed to it is in the tape alphabet, otherwise it returns false.

**Class**

## Input\_Alphabet

**Description**

The input alphabet of a TM is a set of printable characters from the Ascii character set except for “\”, “[“, “]”, “<”, “>”. Furthermore, the input alphabet of a TM is a subset of the tape alphabet and does not contain the blank character. Input strings that the TM may run on are composed of characters from the input alphabet. An input alphabet does not have a maximum size and may only be defined once. The characters in the input alphabet must be unique as the input alphabet is a set. White space characters are excluded from the input alphabet. The input alphabet is used to validate transition functions.

**Associations**

The class **Input\_Alphabet** is a component of the class **Turing\_Machine**, receiving messages delegated to it by the TM.

The association **is subset of** with the class **Tape\_Alphabet** is used to ensure the TM’s input alphabet is a subset of the tape alphabet.

**Attributes**

**Alphabet: Character\_Vector = {}**

The attribute **alphabet** is a vector of characters containing the input alphabet of the TM.

**Methods**

**load(inout definition: File,**

**inout valid: Boolean)**

The method **load** reads the input alphabet from the TM definition file. If any of the input alphabet characters that are read in are reserved characters 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. Otherwise, **valid** is set to true.

**view()**

The method **view** displays the input alphabet of the TM.

**validate(inout valid: Boolean)**

The method **validate** is used to ensure that the input alphabet is a subset of the tape alphabet. If this condition is not met, **valid** is set to false. Otherwise, **valid** is set to true. Checking that the input alphabet is a subset of the tape alphabet is accomplished through sending messages to the class **Tape\_Alphabet** method **is\_element**.

**is\_element(in value: Character): Boolean**

The method **is\_element** returns true if **value** is in the input alphabet. Otherwise, it returns false.

**Class**

## States

**Description**

A TM begins in an initial state, with transition functions allowing the TM to travel into other states. The acceptance of an input string is defined at the TM entering a final state. The set of states includes the initial state and final state(s).

**Associations**

The class **States** is a component of the class **Turing\_Machine**, receiving messages delegated to it by the TM.

**Attributes**

**names: String\_Vector = {}**

The attribute **names** is a vector of strings containing the states of the TM. There is no limit on the length of a state name or the number of states.

**Methods**

**load(inout definition: File,**

**inout valid: Boolean)**

The method **load** reads the states from the TM definition file. A state is named as a string of upper- and lower-case letters, digits, 0-9, underscores, and/or hyphens. A state is case sensitive and must be unique. If previous conditions do not hold, or the next keyword does not follow it in the file, an error message is displayed and the Boolean **Valid** is set to false. Otherwise, **valid** is set to true.

**view()**

The method **view** displays the states of the TM.

**is\_element(in value: String): Boolean**

The method **is\_element** returns true if the state passed to it as a string is in the set of states, otherwise it returns false.

**Class**

Final\_States

**Description**

A TM has zero or more final states which cause it to halt and accept the original input string when entered. Any state in the TM’s set of states may be defined to be a final state in a set of final states. States in the set of final states must be unique.

**Associations**

The class **Final\_States** is a component of the class **Turing\_Machine**, receiving messages delegated to it by the TM.

The association **is subset of** with the class **States** is used to validate that the set of final states is a subset of the states of the TM.

**Attributes**

**names: String\_Vector = {}**

The attribute **names** is a vector of strings containing the final state(s) of the TM.

**Methods**

**load(inout definition: File,**

**inout valid: Boolean)**

The method **load** reads zero or more final states from the TM definition file. If the next keyword does not follow in the file, an error message is displayed and valid is set to false.

**validate(inout valid: Boolean)**

The method **validate** is passed a Boolean **valid** from the **Turing\_Machine** class which is set to true if the final states contained in the String\_Vector of final states are unique and exist within the TM’s set of states. Otherwise, this method sets the Boolean **valid** to false. Checking if the final states are in the set of states is accomplished through sending messages to the class **States** method **is\_element**.

**view()**

The method **view** displays the final states of the TM.

**is\_element(in value: String): Boolean**

The method **is\_element** returns true if the state passed in is in the set of final states, otherwise it returns false.

**Class**

TM\_Utils

**Description**

The class **TM\_Utils** provides general utility functions used across many classes in the Turing Machine application. The class also defines several constants to output Greek characters to the console and color output on the console.

**Associations**

None.

**Attributes**

None.

**Methods**

**check\_unique\_set(in element\_list: Vector,**

**in current\_component: String,**

**inout valid: Boolean)**

The method **check\_unique\_set** takes a vector of elements and checks if every element of the vector is unique. If this condition is not met, the Boolean **valid** is set to false. This function is used in every class that requires its set of elements to be unique.

**uppercase(in value: String): String**

The method **uppercase** takes a string and returns the uppercase equivalent to that string. This function is used to convert keywords in the TM definition file to ease the parsing process in each class’s **load** method.

**parse\_string\_to\_int(in value: String): Integer**

The method **parse\_string\_to\_int** takes in a string and checks to make sure it’s a positive integer. This eases command line parsing for the Truncate(T) and Set(E) commands.

# User Interface

## A black screen with yellow and white text Description automatically generatedCommand Line Invocation

**A screen shot of a computer

Description automatically generated**

## Help Command

**A screenshot of a computer program

Description automatically generated**

## Show Command

**A screenshot of a computer

Description automatically generated**

## A computer screen shot of a black screen with white text Description automatically generatedView Command

## List Command

**A black rectangular object with white text

Description automatically generated**

## A black background with white text Description automatically generatedInsert Command

## Delete Command

**A black screen with white text

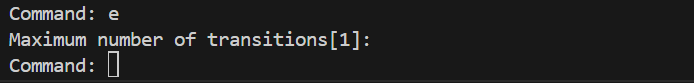
Description automatically generated**

**A black screen with white text

Description automatically generated**

## Set Command

A black background with white text

Description automatically generatedA black screen with white text

Description automatically generated

A black screen with white text

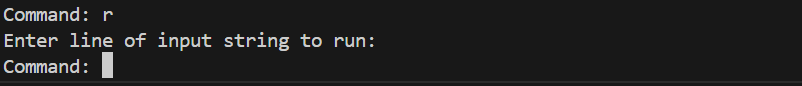
Description automatically generated

## A black rectangle with white dots Description automatically generatedA black screen with white text Description automatically generatedA black rectangular object with white lines Description automatically generatedA black rectangle with white dots Description automatically generatedTruncate Command

## A black screen with white text Description automatically generatedRun Command

**A black screen with white text

Description automatically generatedA black background with white text

Description automatically generatedA black background with white text

Description automatically generated**

## Quit Command

A black screen with white text

Description automatically generated

A black background with white text

Description automatically generated

## Exit Command

A black background with white text

Description automatically generatedA black background with white text

Description automatically generated

# Files

Example of Turing Machine definition file

A screenshot of a computer

Description automatically generated

Example of input string file

A screenshot of a computer

Description automatically generated

# References

Corrigan, Neil. *Dr. CPT\_S 322 Class Notes*. 2024.

Penick, Christian. *Turing Machine Requirements Specification*. 2024.