4/26/2017

Murry, Kenneth

Computer science major

Design Specification

Interactive Turing Machine



CptS 322 Spring 2017

Table of Contents

[Document Revision History 4](#_Toc480918336)

[List of Figures 5](#_Toc480918337)

[**1. Introduction 6**](#_Toc480918338)

[**2. Architecture 6**](#_Toc480918339)

[**3. Data Dictionary 11**](#_Toc480918340)

[**3.1 Main 11**](#_Toc480918341)

[3.1.1 Description 11](#_Toc480918342)

[3.1.2 Associations 11](#_Toc480918343)

[3.1.3 Attributes 11](#_Toc480918344)

[3.1.4 Methods 11](#_Toc480918345)

[**3.2 FileParser 12**](#_Toc480918346)

[3.2.1 Description 12](#_Toc480918347)

[3.2.2 Associations 12](#_Toc480918348)

[3.2.3 Attributes 12](#_Toc480918349)

[3.2.4 Methods 14](#_Toc480918350)

[**3.3 Turing\_Machine 16**](#_Toc480918351)

[3.3.1 Description 16](#_Toc480918352)

[3.3.2 Associations 16](#_Toc480918353)

[3.3.3 Attributes 16](#_Toc480918354)

[3.3.4 Methods 17](#_Toc480918355)

[**3.4 Tape 18**](#_Toc480918356)

[3.4.1 Description 18](#_Toc480918357)

[3.4.2 Associations 18](#_Toc480918358)

[3.4.3 Attributes 19](#_Toc480918359)

[3.4.4 Methods 19](#_Toc480918360)

[**3.5 Input\_Alphabet 21**](#_Toc480918361)

[3.5.1 Description 21](#_Toc480918362)

[3.5.2 Associations 21](#_Toc480918363)

[3.5.3 Attributes 21](#_Toc480918364)

[3.5.4 Methods 21](#_Toc480918365)

[**3.6 Tape\_Alphabet 21**](#_Toc480918366)

[3.6.1 Description 21](#_Toc480918367)

[3.6.2 Associations 21](#_Toc480918368)

[3.6.3 Attributes 21](#_Toc480918369)

[3.6.4 Methods 22](#_Toc480918370)

[**3.7 Transition\_Function 22**](#_Toc480918371)

[3.7.1 Description 22](#_Toc480918372)

[3.7.2 Associations 22](#_Toc480918373)

[3.7.3 Attributes 22](#_Toc480918374)

[3.7.4 Methods 22](#_Toc480918375)

[**3.8 Transition 23**](#_Toc480918376)

[3.8.1 Description 23](#_Toc480918377)

[3.8.2 Associations 23](#_Toc480918378)

[3.8.3 Attributes 23](#_Toc480918379)

[3.8.4 Methods 23](#_Toc480918380)

[**3.9 States 24**](#_Toc480918381)

[3.9.1 Description 24](#_Toc480918382)

[3.9.2 Associations 24](#_Toc480918383)

[3.9.3 Attributes 24](#_Toc480918384)

[3.9.4 Methods 24](#_Toc480918385)

[**3.10 Final\_States 25**](#_Toc480918386)

[3.10.1 Description 25](#_Toc480918387)

[3.10.2 Associations 25](#_Toc480918388)

[3.10.3 Attributes 25](#_Toc480918389)

[3.10.4 Methods 25](#_Toc480918390)

**3.11 Crash……………………………………………………………………………………..26**3.11.1 Description...……………………………………………………………………...26  
3.11.2 Associations……...……………………………………………………………….26  
3.11.3 Attributes………………………………………………………………………….26  
3.11.3 Methods…………………………………………………………………………...26

[**4. User Interface……………………………………………………………………………27**](#_Toc480918391)

[4.1 Initial Command Line Startup 27](#_Toc480918392)

[4.2 Help Command 27](#_Toc480918393)

[4.3 Show Command 28](#_Toc480918394)

[4.4 View Command 28](#_Toc480918395)

[4.5 List Command 30](#_Toc480918396)

[4.6 Insert Command 31](#_Toc480918397)

[4.7 Delete Command 31](#_Toc480918398)

[4.8 Set Command 32](#_Toc480918399)

[4.9 Truncate Command 33](#_Toc480918400)

[4.10 Run Command 34](#_Toc480918401)

[4.11 Quit Command 34](#_Toc480918402)

[4.12 Exit Command 35](#_Toc480918403)

[**5. Files 36**](#_Toc480918404)

[5.1 Turing Machine Definition File 36](#_Toc480918405)

[5.2 Input String File 37](#_Toc480918406)

[**6. References 38**](#_Toc480918407)

# Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Revision Date | Description | Rationale |
| 1.0 | 4/10/2017 | Initial Revision | Formally specify design requirements |
| 1.1 | 4/23/2017 | Added needed methods | Adding needed methods for class FileParser |
| 2.0 | 4/26/2017 | Added classes Crash to design | Adding utility classes |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# List of Figures

Figure # Title Page #

2.1 The associations between classes 7

2-2 Main class diagram 8

2-3 FileParser class diagram 9

2-4 Turing\_Machine class Diagram 10

2-5 Tape class diagram 10

2-6 Input\_Alphabet class diagram 11

2-7 Tape\_Alphabet class diagram 11

2-8 Transition\_Function class diagram 11

2-9 Transition class diagram 12

2-10 States class diagram 12

2-11 Final\_States class diagram 12

## Introduction

The purpose of this document is to create a design of an interactive Turing Machine application. This document will be used as a reference to the software developer that will be doing the implementation of this application. This design specification document will provide and explain the design of the interactive Turing Machine application.

This document will start with an architectural unified modeling language(UML) diagram that will illustrate the relationship between class. Next, each class will be described in more detail using a UML class diagram. This document will go on to a data dictionary that will have the associations to other classes, the attributes, and the methods for each class in the architecture design. The last section of this document will be describing the commands. Each command design will provide a description of the command, examples of what output is expected from the command and examples of invalid command input from the user. This document will conclude with an example of a valid definition file and valid input string file.

## Architecture

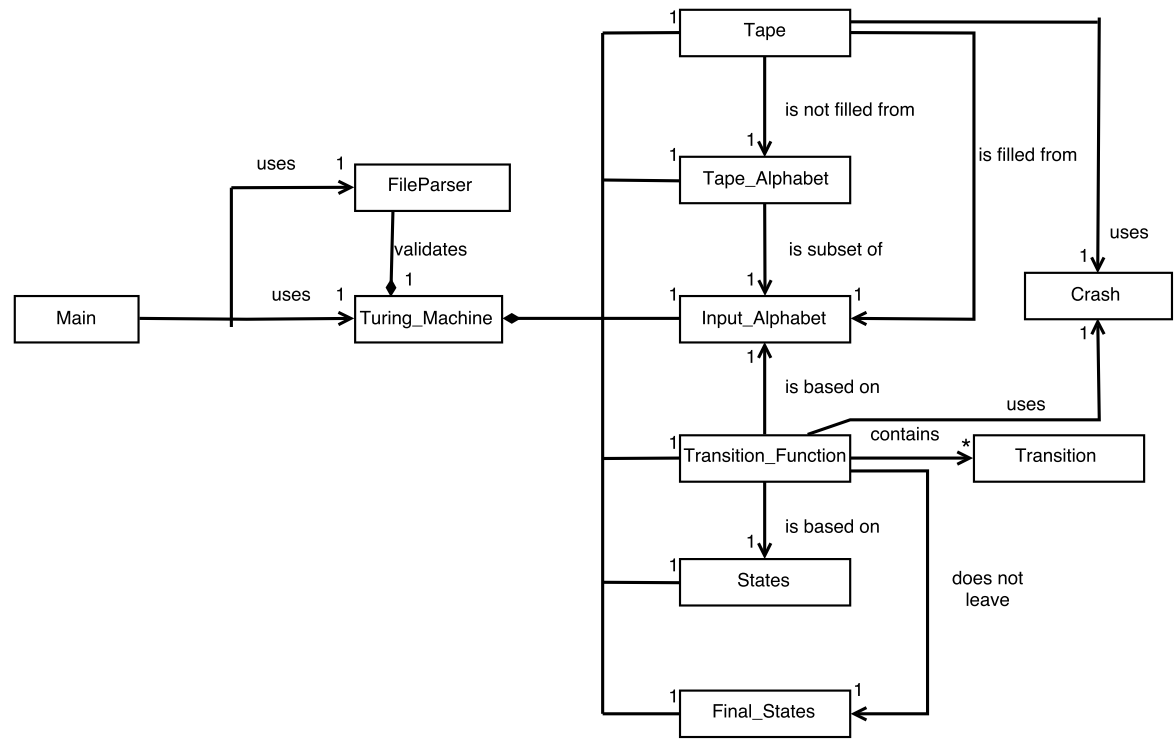


Figure 2.1: The associations between classes

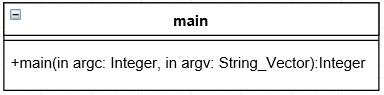


Figure 2-2: Main class diagram

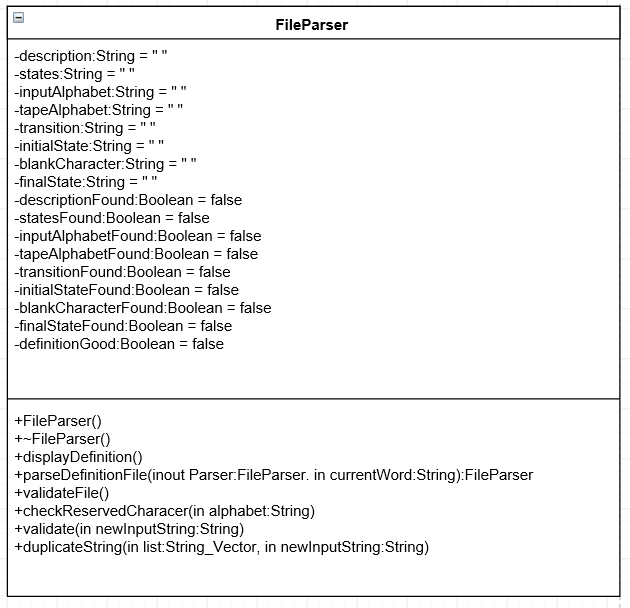


Figure 2-3: FileParser class diagram

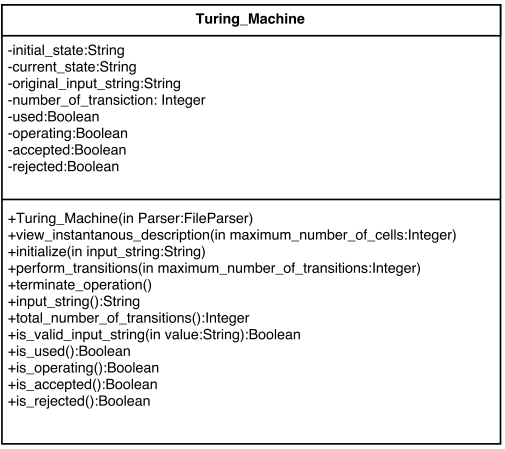


Figure 2-4: Turing\_Machine class diagram

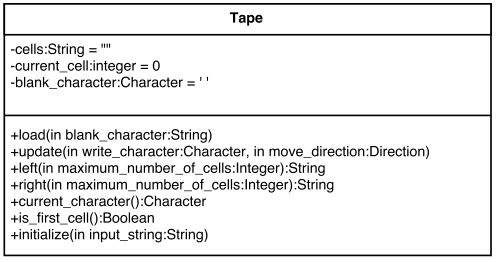


Figure 2-5: Tape class diagram

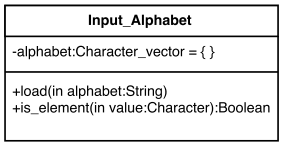


Figure 2-6: Input\_Alphabet class diagram

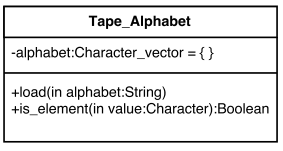


Figure 2-7: Tape\_Alphabet class diagram

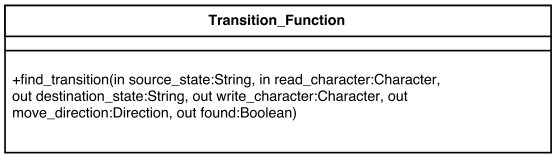


Figure 2-8: Transition\_Function class diagram

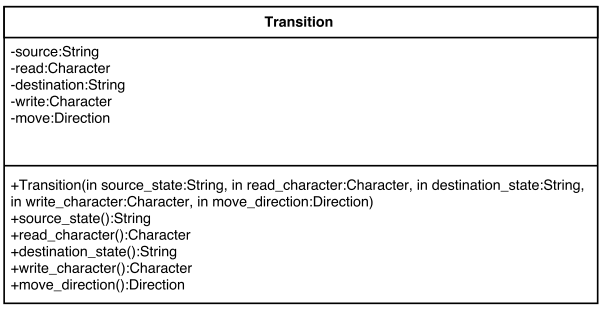


Figure 2-9: Transition class diagram

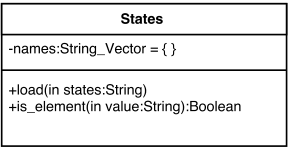


Figure 2-10: States class diagram

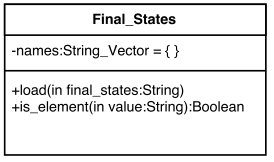


Figure 2-11: Final\_States class diagram

## Data Dictionary

## Main

## Description

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

## Associations

The class **Main** is associated with a single **FileParser** class and a single Turing\_Machine class.

## Attributes

None

## Methods

**main(in argc: Integer, in argv: String\_Vector):Integer**

The method **main** attempts to load the Turing machine definition file that is named after the user passed in parameter. If the Turing machine file is valid and able to be loaded the method continues. If the definition file is invalid or not able to be loaded, an error message shall be displayed to the terminal and the application shall exit. The method **main** will attempt to load the input string file. The method **main** will continue to a switch statement that will hold to commands of the Turing machine.

## FileParser

## Description

This class is the main work horse in validating and storing the definition file to be loaded into the Turing machine class.

## Associations

This class is associated to the Main class and to a single Turing machine class providing the data that is need for the Turing Machine.

## Attributes

**description:String**

The attribute **description** will be used to store the description of the Turing machine from the definition file.

**states:String**

The attribute **states** will be used to store the states of the Turing machine from the definition file.

**inputAlphabet:String**

The attribute **inputAlphabet** will be used to store the input alphabet of the Turing machine from the definition file.

**tapeAlphabet:String**

The attribute **tapeAlphabet** will be used to store the tape alphabet of the Turing machine from the definition file.

**transition:String**

The attribute **transition** will be used to store the transitions of the Turing machine from the definition file.

**initialState:String**

The attribute **initialState** will be used to store the initial state of the Turing machine from the definition file.

**blankCharacter:String**

The attribute **blankCharacter** will be used to store the blank character of the Turing machine from the definition file.

**finalState:String**

The attribute **finalState** will be used to store the final state of the Turing machine from the definition file.

**descriptionFound:Boolean**

The attribute **descriptionFound** will be used to store whether description was found in the definition file.

**statesFound:Boolean**

The attribute **statesFound** will be used to store whether state was found in the definition file.

**inputAlphabetFound:Boolean**

The attribute **inputAlphabetFound** will be used to store whether input alphabet was found in the definition file.

**tapeAlphabetFound:Boolean**

The attribute **tapeAlphabetFound** will be used to store whether tape alphabet was found in the definition file.

**transitionFound:Boolean**

The attribute **transitionFound** will be used to store whether transitions was found in the definition file.

**initialStateFound:Boolean**

The attribute **initialStateFound** will be used to store whether initial state was found in the definition file.

**blankCharacterFound:Boolean**

The attribute **blankCharacterFound** will be used to store whether blank character was found in the definition file.

**finalStateFound:Boolean**

The attribute **finalStateFound** will be used to store whether final state was found in the definition file.

**definitionGood:Boolean**

The attribute **definitionGood** will be used to store whether the definition file is valid.

## Methods

**FileParser()**

The method **FileParser** is the default constructor that takes no parameters and sets the initial value of all the attributes in the **FileParser** class. This method shall set all String attributes to empty strings, definitionGood attribute to true and all other Boolean attributes to false.

**displayDefintion()**

The method **displayDefintion** will display the definition file to the screen for the user to view in the format described in section 4.4.

**parserDefinitionFile(inout Parser:FileParser, in currentWord:String):FileParser**

The method **parserDefinitionFile** is the main work horse for the **FileParser** class it takes in a pointer to a **FileParser** object that can be read and written too, the current word being read from the definition file, and will return a pointer to the **FileParser** class. This method takes in the current word and stores it in a temporary attribute. The method take that temporary attribute and converts it to all uppercase letters in an attempt to look for the tags word: States: , Input\_Alphabet: , Tape\_Alphabet: , Transition\_Function: , Initial\_State: , Blank\_Character: , Final\_States:. If a tag is found then it is known that what proceeded the tags was the previous information (e.g. If States: tag is found the description has been found and the descriptionFound attribute can be set to true.). If a tags is found yet the previous take was not found by checking if it’s found attribute was still false then an error has occurred and a message will be displayed to the screen and the definitionGood attribute will be set to false in order for the Turing machine application to terminate because of an invalid definition file.

**checkReservedChar(in string alphabet)**

The method **checkReservedChar** checks if a reserved character is found in the read in string.

**validateFile()**

The method **validateFile** will evaluate the data stored in the String attributes of the **FileParser** object. This method will check if the input alphabet, tape alphabet, and blank character do not contain a reserved character by calling a message to checkReservedChar() method. This method will check if the initial state is in the set of finite states. This method will check if the final state is in the set of finite states. This method will check if the blank character is in the tape alphabet. This method will check if every character in the input alphabet is in the tape alphabet. This method will check if every transitions is a valid transition by checking if every current is in the finite set of states, the character being read is in the tape alphabet, that the destination states in in the finite set of states, the character to write is in the tape alphabet, and if the direction to move is a valid direction ‘R’ or ‘L’. If any of these validations are to be found invalid a message will be displayed to the screen and the definitionGood attribute will be set to false to exit the Turing machine application.

**validate()**

The method **validate** will validate the new user input string and check that the new user defined input string is not Erroneous.

**duplicateString()**

The method **duplicateString()** will validate if the users new input string is a duplicate input string.

## Turing\_Machine

## Description

The **Turing\_Machine** class is used to control the application of a Turing machine. It is composed of the essentials required to run a Turing machine and it handles the interaction between them.

## Associations

The class **Turing\_Machine** is composed of one of each of the class: Tape, Input\_Alphabet, Transition\_Function, States, and Final\_States.

## Attributes

**initial\_state:String**

The attribute **initial\_state** will store the initial state loaded from the **FileParser** class.

**curret\_state:String**

The attribute **current\_states** will store the current state of the Turning machine.

**original\_input\_string:String**

The attribute **original\_input\_string** will store the original input string that was initially put on the Turing machine tape.

**number\_of\_transiction:Integer**

The attribute **number\_of\_transiction** will store the number of transitions that the Turing machine has run.

**used:Boolean**

The attribute **used** will store if the Turing machine has been used.

**operating:Boolean**

The attribute **operating** will store if the Turing machine is currently operating on an input string.

**accepted:Boolean**

The attribute **accepted** will store if the Turing machine accepted the input string.

**rejected:Boolean**

The attribute **rejected** will store if the Turing machine rejected the input string.

## Methods

**Turing\_Machine(in Parser:FileParser)**

The method **Turing\_Machine** accepts a **FileParser** object and will set the attributes of the following class: Tape class blank\_character attribute, Input\_Alphabet class alphabet attribute, Tape\_Alphabet class alphabet attribute, States class names attribute, Final\_States class names attribute.

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

The method **view\_instantanous\_description** will display the instantaneous description to the screen. When displaying the max number of cells to display on each side that are displayed specified by maximum\_number\_of\_cells

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

The method **initialize** will set the current input string to input\_string if the operating attribute is set to false. If operating is set to true, an error message is displayed and the method return else the initial state of the Turing machine tape is displayed.

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

The method **perform\_transitions** will simulate the Turing machine that is loaded in from the **FileParser** on the set input string. This can only happen if operating is set to true, accepted is set to false, and rejected are set to false. This method will continue to run the Turing machine until the amount of transition reaches the maximum\_number\_of\_transitions unless the Turing machine accepts or rejects the input string before the maximum number of transition is met. If the input string is accepted or rejected, operating will be set to false, and a message will be displayed to the user.

**terminate\_operation()**

The method **terminate\_operation** will stop the operation on the current input string by setting the operating attribute to false. This will make the Turning machine unable to continue, forcing the user to select a new input string or quit the application.

**input\_string():String**

The method **input\_string** returns the current input string that the Turing machine is operating on.

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

The method **total\_number\_of\_transitions** will return the number of transition that have been performed on the current user selected input string.

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

The method **is\_valid\_input\_string** checks that every character in the input string is contained in the input alphabet. If the string contains only valid character that are in the input alphabet is will return true else if it finds an invalid character it will returns false.

**is\_used():Boolean**

The method **is\_used** will return a true or false if the Turing machine has been used.

**is\_operating():Boolean**

The method **is\_operating** will return a true or false if the Turing machine is operating on a input string.

**is\_accepted():Boolean**

The method **is\_accepted** will return the value of accepted

**is\_rejected():Boolean**

The method **is\_rejected** will return the value of rejected

## 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 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.

## Associations

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

The association **is not filled** with the class **Input\_Alphabet** is used to validate that the blank character for initialization and extension of the Turing machine tape is not in the tape 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 Turing machine tape is in the tape alphabet.

## 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**.

## Methods

**load(in blank\_character:String)**

The method **load** loads the blank character passed into the attribute **Tape** class attribute **blank\_character.**

**validate(inout valid: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.

**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, excluding 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. 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.

## Input\_Alphabet

## Description

The class **Input\_Alphabet** stores the set of characters that are allow for the user to enter and stored in the **Input\_Alphabet** class. The input alphabet **is a subset of Tape\_Alphabet**.

## Associations

The class **Input\_Alphabet** is a component of the class **Turing\_Machine**, getting messages assigned to it by the Turing machine.

## Attributes

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

The attribute **alphabet** will be used to store the input alphabet characters that were loaded from the definition file and stored in the **FileParser** class.

## Methods

**load(in alphabet:String)**

The method **load** will load the input alphabet into the attribute **alphabet**.

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

The method **is\_element** checks if the passed in character is contained within alphabet.

## Tape\_Alphabet

## Description

The class **Tape\_Alphabet** class stores the set of characters that can be written to the cells of the **Tape** class.

## Associations

The class **Tape\_Alphabet** is a component of the class **Turing\_Machine**, getting messages assigned to it by the Turing machine.

## Attributes

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

The attribute **alphabet** will be used to store the tape alphabet characters that were loaded from the definition file and stored in the **FileParser** class.

## Methods

**load(in alphabet:String)**

The method **load** will load the tape alphabet into the attribute **alphabet**.

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

The method **is\_element** checks if the passed in character is contained within the attribute **alphabet**.

## Transition\_Function

## Description

The class **Transition\_Function** provides access to the class **Transition** from other classes. It is responsible for loading the transitions, selecting the correct one for the next transition.

## Associations

The class **Transition\_Function** is a component of the class **Turing\_Machine**, getting messages assigned to it by the Turing machine. The class **Transition\_Function** is based on the classes **States** and **Input-Alphabet**. The class **Transition\_Function** has an association with the class **Final\_States** in which transition function **does not leave** a final state.

## Attributes

This class may need an attribute to store the transition stored in the FileParser. This decision will need to be made during implementation.

## Methods

**find\_transition(in source\_state:String, in read\_character: Character, out**  
**destination\_state: String, out write\_character: Character, out move\_direction:**  
**Direction)**

The method **find\_transition** will search all of the class transition until it finds a matching source and read character, or if it has searched all of the Transitions and was unable to find a transition. If the class **Transition\_Function** can find a transition that matches it will return the Transition. If the class **Transition\_Function** is incapable of finding a transition that matches it rejects the input string and crashes the Turing machine application.

## Transition

## Description

The class **Transition** will hold a single Turing machine transition. Any other class that need to access this class must access it by using the class **Transition\_Function**.

## Associations

The class **Transition** has a zero to many association with the class **Transition\_Function**.

## Attributes

**source:String**

The attribute **source** contains the name of the current state that the Turing machine must be in to use the transition.

**read:Characer**

The attribute **read** contains the current character that must be in the cell of the tape.

**destination:String**

The attribute **destination** contains the name of the state in which to transition too.

**write:Character**

The attribute **write** contains the character that will be written to the current tape cell before moving to the next tape cell.

**move:Direction**

The attribute **move** contains the direction that the tape head will move after the **write** character has been written.

## Methods

**Transition(in source\_state:String, in read\_character:Character, in destination\_state: String, in write\_character: Character, in move\_direction: Direction)**

The method **Transition** is the constructor for the **Transition** class taking the parameters of the **source\_state**, **read\_character**, **destination\_state**, **write\_character**, and **move\_direction**.

**source\_state():String**

The method **source\_state** is used to get the current value stored in the **source** attribute.

**read\_character(): Character**

The method **read\_character** is used to get the current value stored in the **read** attribute.

**destination\_state(): String**

The method **destination\_state** is used to get the current value stored in the **destination** attribute.

**write\_character(): Character**

The method **write\_character** is used to get the current value stored in the **write** attribute.

**move\_direction(): Direction**

The method **move\_direction** is used to get the current value stored in the **move** attribute.

## States

## Description

The class **States** will store the possible states that the Turing machine can be in at any given time.

## Associations

The class **States** is a component of the class **Turing\_Machine**, getting messages assigned to it by the Turing machine.

## Attributes

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

The attribute **names** will store all the states names.

## Methods

**load(in states:String)**

The method **load** will load the states into the attribute **name**.

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

The method **is\_element** checks if the passed in value is contained within the attribute **names**.

## Final\_States

## Description

The class **Final\_States** will contains the state that the Turing Machine cannot leave, input string is accepted, and the Turing\_Machine is set to not operating on a input string.

## Associations

The class **Final\_States** is a component of the class **Turing\_Machine**, getting messages assigned to it by the Turing machine. The class **Final\_States** has an **is a subset** association with the class **States**.

## Attributes

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

The attribute **names** will store all the final states names.

## Methods

**load(in states:String)**

The method **load** will load the final states into the attribute **name**.

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

The method **is\_element** checks if the passed in value is contained within the attribute **names**.

## Crash

## Description

The class **Crash** is used to send the message to halt the Turing Machine.

## Association

The class **Crash** is associated with the classes **Tape** and **Transition\_Function** in that it is used the classes **Tape** and **Transition\_Function**.

## Attributes

None.

## Methods

**crash(string reason)**

The method **crash** accepts a message and is caught by the Turing Machine which can read the message.

## User Interface

## Initial Command Line Startup

Example of running the initial command line startup(valid).

$tm definition  
Turing Machine Loaded Successfully!

Command:

Example of running the initial command line startup(invalid).

$tm definition  
Invalid Definition File: Please check format

$

## 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 ttransitions to perform

Sho(w) Show status of application

(T)runcate Truncate instantaneous descriptions

(V)iew View Turing machine

Command:

## Show Command

Example of using the show command.

Command: w

Course: CptS 323

Semester: Spring

Year: 2017

Instructor: Neil B. Corigan

Author: Kenneth M Murry

Version: 1.0

Maximum Transitions: 1

Maximum Cells: 32

Name of Turing machine: tm

Status: TM is currently running on an input string

Transitions: 10

Input String: aaaabbbb

Command:

## View Command

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, s3, s4}

Σ = {a, b}

Γ = {a, b, X, Y, -}

δ(s0, a) = (s1, X, R)

δ(s0, Y) = (s3, Y, R)

δ(s1, a) = (s1, a, R)

δ(s1, b) = (s2, Y, L)

δ(s1, Y) = (s1, Y, R)

δ(s2, a) = (s2, a, L)

δ(s2, X) = (s0, X, R)

δ(s2, Y) = (s2, Y, L)

δ(s3, Y) = (s3, Y, R)

δ(s3, -) = (s4, -, R)

q! = s0

B = -

F = {s0}

Command:

## List Command

Example of using the list command.

Command: l

1. a

2. ab

3. \

4. aaabb

5. aaaaaaaaaabbbbbbbbbb

6. aabb

7. aaaaaabbbbbb

8. ba

9. aba

10. bb

Command:

## Insert Command

Example of using the insert Command (valid).

Command: i

Input String: abbb

String Inserted into List!

Command:

Example of using the insert Command (Invalid)

Command: i

Input String: abc

Erroneous String!

Command:

Example of using the insert Command (duplicate input string)

Command: i

Input String: ab

Duplicate Input String!

Command:

## Delete Command

Example of using the delete command(valid)

Command: d

Input String Number: 1

String Deleted!

Example of using the delete command(invalid)

Command: d

Input String Number: 0

Non-existent line number

Command:

## Set Command

Example of using set command(valid)

Command: e

Maximum Number of Transitions[1]: 5

Maximum Transitions Changed to 5!

Command:

Example of using set command(invalid)

Command: e

Maximum Number of Transitions[5]: -1

Invalid Entry: Must be a positive number

Command: e

Maximum Number of Transitions[5]: a

Invalid Entry: Must be a positive number

Command:

## Truncate Command

Example of using truncate command(valid)

Command: t

Maximum Number of Cells[32]: 10

Maximum Cells Changed to 10!

Command:

Example of using truncate command(invalid)

Command: t

Maximum Number of Cells[10]: -1

Invalid Entry: Must be a positive number

Command: t

Maximum Number of Cells[10]: foo

Invalid Entry: Must be a positive number

Command:

## 4.10 Run Command

Example of using run command

Command: r

Input String Number: 3

0. [s0]aab

10. aab[s5]aX

Command: r

10. aab[s5]aX

20. aab[s5]aX

aabaab is accepted in 20 transitions

Command:

## 4.11 Quit Command

Example of quit command (still running)

Command: q

Input string aab not accepted or rejected in 1 transitions

Example of quit command (not running)

Command: q

Turing Machine is currently not running on an input string!

Command:

## 4.12 Exit Command

Example of exit command(valid)

Command: x

Successfully Wrote Input String File!

Example of exit command(invalid)

Command: x

Error: Failed To Create File!

## Files

## Turing Machine Definition File

The following is an example of a valid Turing machine definition file format:

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:

s0 a s1 X R

s0 Y s3 Y R

s1 a s1 a R

s1 b s2 Y L

s1 Y s1 Y R

s2 a s2 a L

s2 X s0 X R

s2 Y s2 Y L

s3 Y s3 Y R

s3 - s4 – R  
  
INITIAL\_STATE: s0  
  
BLANK\_CHARACTER: -  
  
FINAL\_STATES: s4

## Input String File

The following is an example of a valid Turing machine input string file format:

a

ab

\

aaabb

aaaaaaaaaabbbbbbbbbb

aabb

aaaaaabbbbbb

ba

aba

bb

## References

* Neil Corrigan – General lecture and notes