WSU Tri-Cities

Turing Machine Software Project

Cpt_S 322

Jon Churchill
4/29/2013

Table of Contents

List of Figures

Revision History

Version 1.0 – 3/25/2013 – first draft of UML specification.

Version 2.0 – 4/29/2013 – final draft of UML specification

Introduction

This paper is intended to show an overview of class diagrams for a Turing Machine with definitions to describe what they are and how they are used. The intended audience as of right now is the professor of this class, but everything will be written in a manner where someone who doesn't know much about these subjects should understand the basic idea of what is going on. This document will have a general layout of having the UML diagrams for the project in the architecture section, then for each class diagram the data dictionary will go into deeper detail. Then the user interface will be shown so that the user will know what each command does and looks like.

Architecture

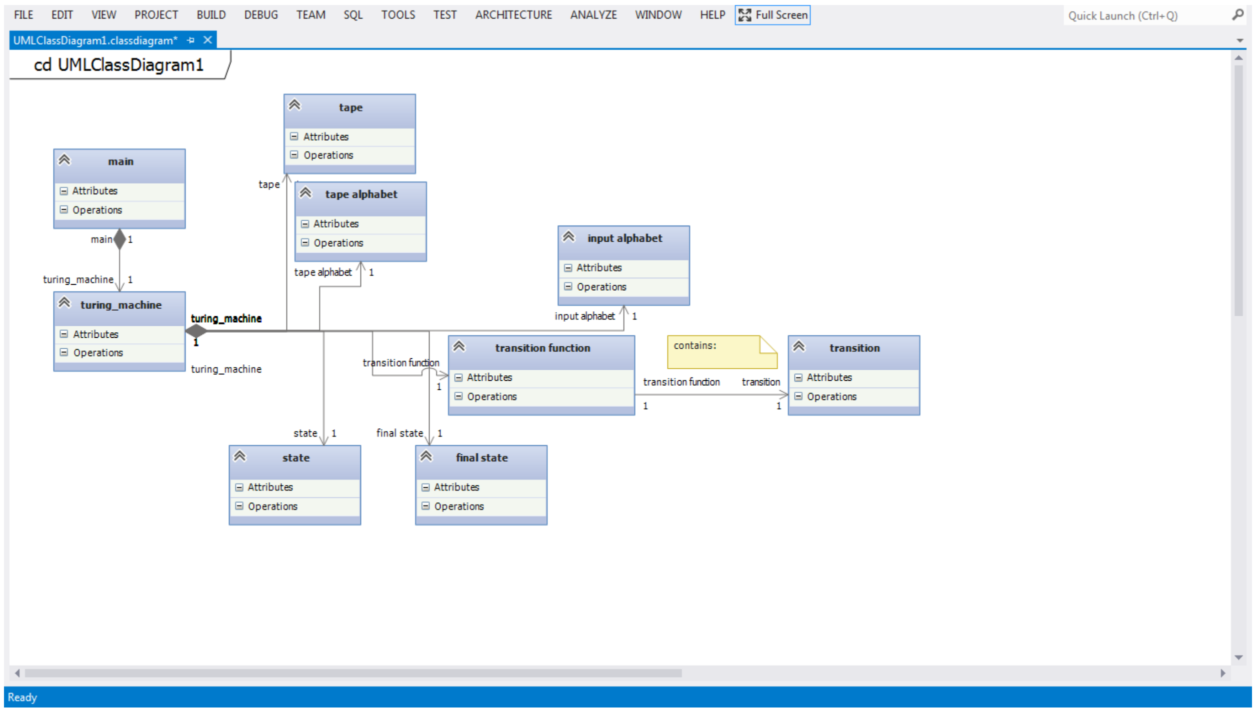


figure 1.

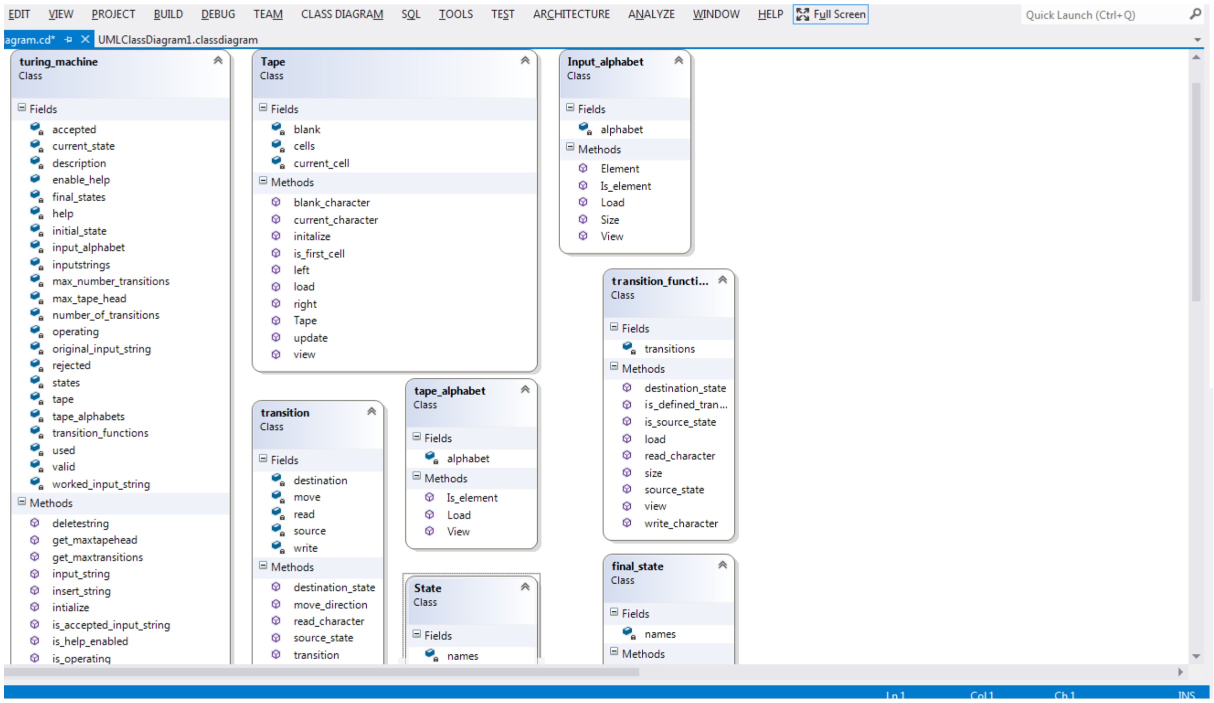
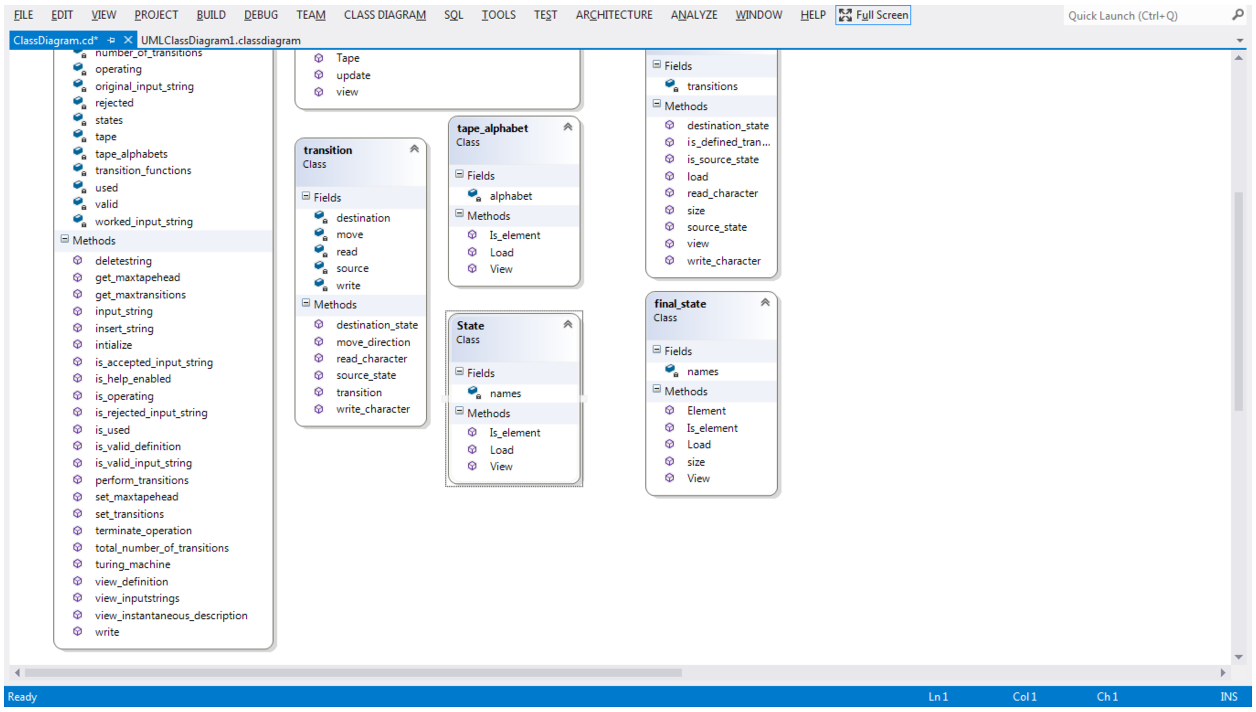


figure 2.



number_of_transitions
operating
original_input_string
rejected
states
tape
tape_alphabets
transition_functions
used
valid
worked_input_string
Methods
deletestring
get_maxtapehead
get_maxtransitions
input_string
insert_string
intialize
is_accepted_input_string
is_help_enabled
is_operating
is_rejected_input_string
is_used
is_valid_definition
is_valid_input_string
perform_transitions
set_maxtapehead
set_transitions
terminate_operation
total_number_of_transitions
turing_machine
view_definition
view_inputstrings
view_instantaneous_description
write

Tape
update
view

transition
Class
Fields
destination
move
read
source
write
Methods
destination_state
move_direction
read_character
source_state
transition
write_character

tape_alphabet
Class
Fields
alphabet
Methods
Is_element
Load
View

State
Class
Fields
names
Methods
Is_element
Load
View

Fields
transitions
Methods
destination_state
is_defined_tran...
is_source_state
load
read_character
size
source_state
view
write_character

final_state
Class
Fields
names
Methods
Element
Is_element
Load
size
View

figure 3.

Data Dictionary

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 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.

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:integer = 0

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

Blank:character = ' '

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

Methods:

load(inout definition:file, inout valid:boolean)

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

View()

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

initialize(in input_string:string)

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

update(in write_character:character, in move_direction:direction)

The method update first determines if the update of the Turing machine tape is possible. The method returns if a left move is specified from the first cell. If a right move is specified from the last cell, a blank character is appended to the tape. 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.

```
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);
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 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 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 cells to the right of the current cell up to the rightmost nonblank character. If the string is truncated from the tape, the reserved character '>; will be added to the end of the string.

current_character():character

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

blank_character():character

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

in_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.

Class: Input_alphabet, Tape_alphabet

Description:

the input alphabet will consist of only printable characters from the ASCII character set, with a few characters reserved for system use, such as the \, which is for the blank character, [] which is for enclosing the state name, and <> which is used to truncate tape characters so you only see characters near the state. The whitespace character shall be excluded from the input and tape alphabets. Every character in the input alphabet must be in the tape alphabet but not every character in the tape alphabet will be in the input alphabet such as the blank symbol which can not be in the input alphabet.

Associations:

The class Input_alphabet and Tape_alphabet are components of the class Turing_Machine, receiving messages delegated to it by the Turing machine.

Attributes:

alphabet:character_vector = {}

The attribute alphabet gets the alphabet that is going to be used, for the tape_alphabet every character is going to be in there, with the input_alphabet some will be excluded.

Methods:

load(inout definition:file, inout valid:boolean)

This will load the file, and input the input_alphabet and tape_alphabet into the corresponding variables.

View()

This will let you view what is on the definition file.

size():integer //input_alphabet only

This is the number of characters in the input file.

element(in index:integer):character //input_alphabet only

This is the character at a defined spot in the input file.

is_element(in value: character):boolean

This will see if the character is in the file.

Class: States, Final_States

Description:

The states and final states classes will be named as a string of upper or lower case letters, digits, or underscores. They will be case sensitive and each name shall be unique. There is no limit on the length of a state name or on the number of states.

Associations:

The class States and Final_States are components of the class Turing_Machine, receiving messages delegated to it by the Turing machine.

Attributes:

names:string_vector = {}

This creates the name of the state for a given node.

Methods:

load(inout definition:file, inout valid:boolean)

This will load the file, and input the state names.

View()

This will let you view what is on the definition file.

size():integer //Final_States only

This is the number of characters in the input file.

element(in index:integer):string //Final_States only

This is the state at a defined spot in the input file.

is_element(in value: string):boolean

This will see if the state is in the file, or what the final state is.

Class: Transition

Description:

This class deals with reading and writing the characters and moving the tape head to the left or right. It also has a destination state and a source state.

Associations:

The class Transition is a component of Transition_Function, receiving information sent from the Transition_Function and then doing to required operations.

Attributes:

source_string

This is the starting state.

read_char

This is the character that is being read at the current position

destination:string

This is the state that the transition will go to

write:char

This is the character to be written if the tape head is in a state to write a character.

move:direction

This is the direction that the tape head will move when it transitions.

Methods:

transition(in source_state:string, in read_character, in destination_state:string, in write_char:char, in move_dir:direction)

This is a constructor for the transition class.

source_state():string

This is the starting state of the Turing machine.

read_char():char

This function reads in a character from the definition file.

destination_state():string

This is the final state of the Turing machine.

write_char():char

This function will write a character if the state it is in requires it.

move_dir():direction

This function will tell the Turing machine which direction to move the tape head.

Class: Transition_Function

Description:

The Transition function tells the program which function to call, such as to move the tape head or write a character. It can define any number of transitions. If any transition is undefined it will cause the Turing machine to crash during execution. It must specify direction to move the tape head, left or right, using the characters l or r or (L or R), upper or lower case. At most there will be one transition from a given state on a given tape character.

Associations:

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

Attributes:

none.

Methods:

load(inout definition:file, inout valid:boolean)

Loads the input from the TRANSITION_FUNCTION: part of the file.

View()

Views the part of the file under the TRANSITION_FUNCTION.

size():integer

Returns the size of the given object.

source_state(in index:int):string

Loads in the state of the indicated index.

read_char(in index:int):char

Reads the character in the selected index.

destination_state(in index:int):string

Sets the final state from the selected index.

write_char(in index:int):char

Writes a character at the given index.

is_defined_transition(in source_state:string, in read_char:char, out destination_state:string, out write_char:char, out move_direction:direction):boolean

This reads in the whole transition function line from the file, and then sets the corresponding values.

is_source_state(in state:string):boolean

Query's if it is in the source state.

Class: Turing_Machine

Description:

The Turing_Machine class is basically the base class of which all other classes with the exception of the class Transition get their information. It sets the current state, the initial state, the number of transitions and so on. If there is any problem with definition file, the application will provide error messages on the monitor to the user. Applications will not use erroneous files, it requires the user to correct through a text editor.

Associations:

All other classes are components of this class Turing_Machine, with the possible exception of the class Transition.

Attributes:

inputstrings:string_vector

This is a vector of strings from the input file.

worked_input_string:string

This is the modified input string.

max_number_transitions:int

The inputted max transitions set by the user, default is 1.

max_tape_head:int

The maximum shown spaces to the right and left of the tape head.

help:bool

For the help function, this is default set to off.

description:string

This is the text description at the beginning of the input file.

initial_state:string

The starting state from the file.

current_state:string

The current state that the Turing machine is in.

original_input_string:string

The starting input string from the .str file.

number_of_transitions:integer

The total number of transitions the Turing machine is capable of.

valid:boolean

Checks to see if the current state is valid.

used:boolean

Checks to see if something is being used.

operating:boolean

Checks to see if the Turing machine is currently operating.

accepted:boolean

Checks to see if the Turing machine accepted the input.

rejected:boolean

Checks to see if the Turing machine rejected the input.

Methods:

Turing_Machine(in definition_file_name:string)

default constructor for the Turing machine.

view_definition()

Shows the user the contents of the definition file.

view_instantaneous_description(in max_number_cells:int)

Shows the user everything that is contained within the < and >.

initialize(in input_string:string)

Starts the Turing machine program with an inputed string.

perform_transitions(in max_num_of_transitions:int)

Runs the program a number of times that has been inputed.

terminate_operation()

Stops the operation.

input_string():string

Gets an input string.

total_num_of_transitions():int

Sets the total number of transitions you want the Turing machine to do.

is_valid_definition():boolean

Checks to see if the definition is valid.

is_valid_input_string(in value:string):boolean

Checks to see if the input string is valid.

is_used():boolean

Checks to see if its being used.

is_operating():boolean

Checks to see if the Turing machine is in operation.

is_accepted_input_string():boolean

Checks to see if the input string was accepted.

is_rejected_input_string():boolean

Checks to see if the input string was rejected.

enable_help:boolean

This function returns true if help is enabled or false if disabled.

deletestring(int value):boolean

This function will delete the line number entered if it is set to true.

view_inputstrings

This function will display the input string.

insert_string(string insertstring):boolean

This function will put user entered information into the input file.

get_maxtransitions:unsigned long

This is a getter for the amount of max transitions.

set_transitions(unsigned long maxtrans)

This is a setter for the number of transitions the user wants to perform at a time.

set_maxtapehead(unsigned long maxtapehead)

This is a setter for the number of spaces displayed to the right and left of the tapehead.

get_maxtapehead:unsigned long

This is a getter for the set_maxtapehead.

is_help_enabled:boolean

This function is a check to see if the help is on or not.

write(string definition_file_name)

This is the function to write to the .str file.

User Interface

Command line invocation:

```
./TM
```

Help Command:

```
Command: h
```

```
- Delete – delete input string from list
- Exit – exit application
- Help – help user with prompts or not
- Insert – insert input string into list
- List – list input strings
- Quit – quit operation of TM on input string
- Run – run TM on input string
- Set – set maximum number of transitions to perform
- Show – show status of application
- Truncate – truncate instantaneous descriptions
- View – view TM
command:
```

Show Command:

```
command: w
```

```
cpt_s 322
spring 2013
Neil B. Corrigan
Jon Churchill
version 1.0
help = true
transitions = 1000
```

```
cells to left and right of tape head = 32

status: running – input_string = aabb, transitions = 10.

command:
```

View Command:

command: v

```
This Turing machine accepts the language of one or more A's follwed by the same number of B's

M =  (Q, sigma, gamma, delta, q0, B, F)

command:
```

List Command:

command: l

```
1) a

2) b

command:
```

Insert Command:

command: I

```
insert: d

error: input string not in sigma, discarding.

command:
```

Delete Command:

command: d

```
Delete: 1
```

```
string deleted.

Delete: 100

error –

command:
```

Set Command:

command: e

```
maximum number of transitions[100]:xyz

        error, value unchanged

        command:
```

Truncate Command:

command: t

```
Truncate: 4

command:
```

Run Command:

command: r

```
input string number: 5

        0. [s0]aabbabbaa>

        100. <bxaabyaaab[s3]xyyabbabyy>

        command: R

        200. abyx[s2]abbb

        command: R

        225. abyx[s5]ax

        input string aabbabbaaaaa accepted in 225 transitions

        command:
```

Quit Command:

command: q

input string aabbabbaaaaa is not accepted or rejected in 200 transitions

command: q

error, not running on an input string

command:

Exit Command:

command: x

Exiting: file could not be written

Files

tm.def

```
This Turing machine accepts the language of one or more A's follwed by the same number of B's

STATES: s0 s1 s2 s3 s4

INPUT_ALPHABET: a b

TAPE_ALPHABET: a b X Y -

TRANISTION_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
```

tm.str

a

ab

\

aaabb

aaaaaaaaaabbbbbbbbbb

aabb

aaaaaabbbbbb

ba

aba

bb

References

Appendix