**Linux Command-line**

**Turing Machine**

**Simulator**

**Software Requirements**

**Specification**

Date: 25 February 2018

Author: Emmanuel Bonilla

Course: Computer Science 322, Software Engineering

**TABLE OF CONTENTS**

List of Figure Page 3 Revision History Page 4

1. Introduction Page 5
   1. Objective Page 5
   2. Intended Audience Page 5
   3. Layout Page 5
2. Background Page 6
   1. Mechanical example Page 7
3. Overview Page 8
4. Environment Page 9
   1. Input and Output Devices Page 9
   2. Turing Machine Definition File Page 6
   3. Input String File Page 14
5. Operation Page 15
   1. Invocation Page 15
      1. Command Line Page 15
      2. Configuration Setting Page 15
      3. Opening Turing Machine Page 16
   2. Commands Page 17
      1. Help User Page 17
      2. Show Status Page 18
      3. View Turing Machine Page 19
      4. List Input String Page 20
      5. Insert Input String Page 20
      6. Delete Input String Page 20
      7. Set Transitions Page 21
      8. Truncate Instantaneous Descriptions Page 21
      9. Run Turing Machine Page 22
      10. Quit Turing Machine Page 22
      11. Exit Application Page 22
   3. Termination Page 22
      1. Closing Turing Machine Page 22

References Page 23

Appendix Page 23

**LIST OF FIGURES**

1-1 Mechanical representation of Turning Machine page 7

2-2 Instantaneous Description page 8

2-3 Display messages page 8

4-1 Keywords page 9

4-2 Valid Definition File page 12

4-3 Erroneous File page 13

4-4 Input String Example page 14

5-1 File Input Example page 15

5-2 Config settings page 16

5-3 Invocation commands page 16

5-4 Help Display page 17

5-5 Show Display page 18

5-6 View Display Example page 19

5-7 List Display Example page 20

5-8 Run Display Example page 22

**REVISION HISTORY**

|  |  |  |
| --- | --- | --- |
| *Description* | *Rational* | *Date* |
| Initial release specifications | Initial Release | 25 February 2018 |
| Corrections to initial release | Multiple spelling, grammar, missing information errors | 3 March 2018 |

**1.0 INTRODUCTION**

* 1. **Objective**

This Turing Machine simulator is designed to work the Linux operating system only. The simulator’s user interface is limited to the command-line.

* 1. **Intended Audience**

The simulator is produced for computer science 322 project, and personal use.

* 1. **Layout**

***Background:*** This is the understanding of what a Turing Machine is, and how one works in theory.

***Overview:*** Describes the reason for the Turing Machine application. Explain the hardware and software environment for the development and installation of the application. Also, how to run the application, and interactively trace the operation of a

Turing Machine.

***Environment:*** Describes the input and output devices for the Turing Machine. As well the Turing Machine definition file, input string file, with descriptions of each.

***Operation***: Describes how to use the Turing Machine application from beginning to end of operation. Including detailed descriptions of commands and settings.

**2.0 BACKGROUND**

A Turing machine is an abstract mathematical model of a computer. The machine is designed to operate on infinite memory tape, that contains symbols from the machine definition file alphabet. Based on the symbol and current state, the Turing machine has the option to write a new symbol, move left, move right, or stay in the existing position.

A Turing Machine is specified by M = (Q, Σ, Γ, δ, q0, B, F), where

Q is a finite set of states

Σ ⊆ (Γ – {B}) is a finite input alphabet

Γ is a finite tape alphabet

δ is a transition function from Q x Γ → Q x Γ x {L, R}

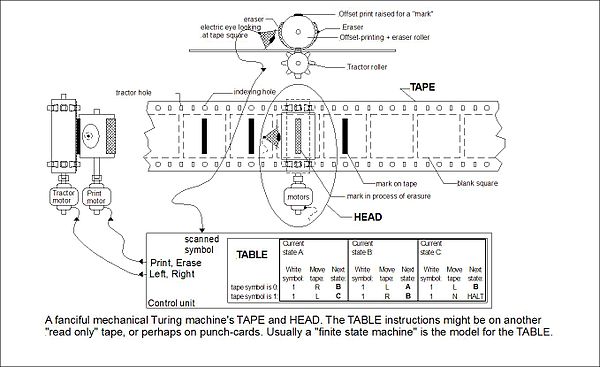
q0 ∈is the initial state

B ∈Γ is the blank symbol

F ⊆ Q is the set of final states

There are endless possibilities and requirements for a Turing Machine. Please refer to fig (1-1) on page 7 for an example of a Turing machine as a mechanical Device.

**Turing Machine as a Mechanical Device**



**Fig 1-1**

**3.0 OVERVIEW**

The Linux Command-line Turing Machine Simulator is designed to work with the users input. The display is strictly for the Linux command terminal and written with Microsoft C++ language. The user must provide two files (definition file & string file) for the user to trace the operation of the Turing machine on an input string with the run command and the instantaneous description (ID).Run command allows the user to trace operation of the Turing Machine on an input string using ID’s. The instantaneous description is described in three parts. First part is an integer followed immediately with a period. This is to display the number of transitions currently performed on input string. Second is the current state started and finished with brackets “ [ ] ”. This will display the current state on the tape head. Third is the input string that is truncated with “ < > ” symbols. This is to display the input string before and after the current state if available. Please refer to example 2-2.

|  |
| --- |
| Fig 2-2 Instantaneous Description |
| 0. [s0] aabb |
| 10. <baabaa[s4]bababa> |
| 74. <XYYXY[s5] |

The Final state will display the string inputted and the number of transitions for both cases accepted, rejected or user quit. Accepted means the Turing Machine has successfully transitioned thru the entire input string and has reached the final state. Rejected means the Turing Machine cannot transition and so it stops. User Quit means the user quit the operation before termination. Please refer to example 2-3 for display messages.

|  |
| --- |
| Fig 2-3 Display Messages |
| ‘aaabbb’ was accepted in 23 transitions |
| ‘abababab’ was rejected in 97 transitions |

**4.0 ENVIRONMENT**

**4.1 Input and Output Devices**

There are three input devices for this application; users keyboard, Turing Machine definition file, and input string file (optional). There are two output devices. The user computer monitor and the input string file created by the application. The user inputs the Turing Machine definition file with the keyboard using the insertion command and monitor. All other commands can also be inputted on the command line using the keyboard.

**4.2 Turing Machine Definition File**

The Turing Machine Definition file is a text file with a “.def” extension. For the definition file to be valid it must have seven key words in an ordered list (fig 4-1). Note that the transition\_function requires more than one function.

|  |  |
| --- | --- |
| Fig 4-1 Keywords | |
| Keywords | **Meaning** |
| States: | Q |
| Input\_alphabet: | Σ |
| Tape\_alphabet: | Γ |
| Transition\_function: | δ\* |
| Initial\_state: | q0 |
| Blank\_character: | B |
| Final\_states: | F |
| Turing Machine is specified by  M = (Q, Σ, Γ, δ, q0, B, F), | |

Keywords are required to match fig 4-1 including underscore, and colon at the end of the word. Keywords are not case sensitive and do not need to be in separate lines. However, they must not be duplicated in the definition file. If the file is accepted it? It will display an appropriate message and prompt the command line. Unaccepted files will display an appropriate message and terminate. Reference fig 4-2 for a valid definition file.

STATES:

*States* is equal to Q (fig 4-1). Since Q must include q0, then Q must include one state to be accepted. Upper-case or Lower-case letters, digits, and underscores are valid in the string. Each state is case sensitive and must be unique. If any other symbols are added or the state is not unique Q will be rejected. There is also no limit to the length of state or number of states.

INPUT\_ALPHABET:

*Input\_alphabet* is equal to Σ (fig 4-1). Between Σ and Γ the input alphabet will consist only of printable characters from the ASCII character set. There will only be five exception characters. These are \ , [ , ] , < , > . Every character in Σ must also be in Γ, except for the blank symbol B. It must not be in Σ.

TAPE\_ALPHABET:

*Tape\_alphabet* is equal to Γ (fig 4-1). Between Σ and Γ the input alphabet will consist only of printable characters from the ASCII character set. There will only be five exception characters, which are \ , [ , ] , < , > . Every character in Σ must also be in Γ, except for the blank symbol B. it must not be in Σ.

TRANSITION\_FUNCTION:

*Transition\_function* is equal to δ (fig 4-1). Between tape\_alphabet and *initial\_state* are considered part of the section. Each function has five rules to be accepted.

1. Transition must define a direction to move the tape head left or right. Using L or R in upper or lower-case characters.
2. At most, one transition from a given state on a given tape character may be defined.
3. Blank symbol B must be a tape character in gamma.
4. Initial state of q must be in state Q, and any state in Q may be defined to be a final state in F.
5. If a state is defined to be a final state in F, no transition from that state will be allowed.

INITIAL\_STATE:

*Initial\_state* is equal to q0 (fig 4-1). Between *initial\_state* and the *blank\_character* q0 must only contain one state and be separated by a blank space. q0 can consist of upper and lower-case letters, digits, and underscores. If any other symbols are added or the state is not included into Q. Then q0 will be rejected.

BLANK\_CHARACTER:

*Blank\_character* is equal to B (fig 4-1). Between *blank\_character* and *final\_states*, the letter can only be separated by blank space and be in the ASCII character set. There will only be five exception characters which are \ , [ , ] , < , > .

FINAL\_STATES

*Final\_states* are equal to F (fig 4-1). Final\_states specify that you are currently in a key word. Or that you are at the end of the file. Upper-case or Lower-case letters, digits, and underscores are valid in the string. Each state is case sensitive and must be unique. If any other symbols are added or F is not included in Q, then it will be rejected. There is also no limit to the length of state or number of states.

|  |
| --- |
| Fig 4-2 Valid Definition File |
| This Turing machine accepts the language of one or more a’s followed by the same number of b’s  STATES: s0 s1 s2 s3 s4  INPUT\_ALPHABET: a b  TAPE\_ALPHABET: a b X Y –  TRANSITION\_FUNCTION:  S0 a s1 X R  S0 Y s3 Y R  S1 a s1 a R  S1 b s2 Y R  S1 Y s1 Y R  S2 a s2 a R  S2 X s0 X R  S2 Y s2 Y R  S3 Y s3 Y R  S3 - s4 - R  INITIAL\_STATE: s0  BLANK\_CHARACTER: -  FINAL\_STATES: s4 |

|  |
| --- |
| Fig 4-3 Erroneous File |
| STATES: states not declared in δ, q0 and F  INPUT\_ALPHABET: a b  TAPE\_ALPHABET: b X Y – a is declared in Σ but not Γ  TRANSITION\_FUNCTION:  S0 a s1 X R  S0 Y s3 Y R  S1 a s1 a missing move direction (L or R)  S1 b s2 Y R  S1 Y s1 Y R  S2 a s2 a R  S2 X s0 X R  S2 Y s2 Y R  S3 Y s3 Y R  S3 - s4 - R  INITIAL\_STATE: s0  BLANK\_CHARACTER: -  FINAL\_STATES: |

**4.3 INPUT STRING FILE**

The Turing Machine input string file is a text file with an “.str” extension. For the string to be valid the input string file must pass four checks. First each line of the input string file is an input string composed of characters from Σ. Second λ (empty string) is specified in the file as \ and must appear on its own line. Third duplicated strings are not allowed and will be discarded. Four if any other line is empty or contains a character not in Σ, the line will be discarded. If any of the checks fail the simulator will display an appropriate error message per the check. Please refer to fig 4-4 to find examples of valid and invalid strings.

|  |
| --- |
| Fig 4-4 Input String Example |
| Σ = { a, b, c }  aaaabacbcabcacbacba Valid input string  acbcde Invalid, { d, e } not in Σ  aaacbc\abc Invalid, \ must appear in its own line  \\ Invalid, { \ } not in Σ  \ Valid empty string |

There are no limits to the length of an input string, on the number of input strings or the number of strings. If the input string read from the file is valid, then it is stored in the string list.

**5.0 OPERATION**

**5.1 Invocation**

**5.1.1 Command Line**

To run the Turing Machine simulator, the user must enter the definition file name and input file name. Files need to be in the working directory. For a valid input, the user must enter file and input name without any extension (.def or .str). Also, a blank space must be entered between the definition file name and input file name. Please refer to Fig 5-1 for an example

|  |
| --- |
| Fig 5-1 File Input Example |
| Files in working directory: teslaTM.def and inputstring.str  ./teslaTM inputstring |

If the files names are valid the simulator will read both files and load them to the application. Once complete, a display will prompt the user for success or failure. Only if successful will the application continue, and display “command: “.

**5.1.2 Configuration Settings**

There are only two configuration settings within the simulator and are set to default at the beginning of the operation. The first setting is the maximum number of transitions to be performed at a time. The second setting is the maximum number of cells to the left and right of the tape head, to display in an instantaneous description (ID). Default for maximum transitions is 1, for maximum number of cells is 32.

The number of transitions setting is designed to let the user change how many transitions are to be completed on an input string. This is displayed as an integer with an immediate period.

The maximum number of cells setting is designed to let the user change how many values are displayed to the left and right of the head. If there are more than the specified number of characters, the simulator will truncate with the “ < ” and “ > “.

Neither settings are saved upon closing the application and will be set back to default. Please refer to Fig 5-2 for an example.

|  |
| --- |
| Fig 5-2 Config settings |
| Max # of transitions: 50 Max # of cells: 5 |
| 50. <babaa [s6] baaba> |
| Max # of transitions: 1 Max # of cells: 1 |
| 51. <b [s6] a> |

**5.1.3 Opening Turing Machine**

Once the application confirms a valid file (reference 5.1.1) it will display a command prompt “command: “. The user will input one of eleven invocation commands. Refer to Fig 5-3

|  |  |
| --- | --- |
| Fig 5-3 Invocation commands | |
| Command | **Invocation** |
| Delete | D or d |
| Exit | X or x |
| Help | H or h |
| Insert | I or i |
| List | L or l |
| Quit | Q or q |
| Run | R or r |
| Set | E or e |
| Show | W or w |
| Truncate | T or t |
| View | V or v |

If any command other than what is in the invocation, this includes a blank response, the application will return to the command prompt.

**5.2 Commands**

**5.2.1 Help User**

Invocation: h or H (only)

*Help* user command will display each of the eleven commands with a brief description of each purpose. Refer to Fig 5-4. No other command from the user is required. The application will return to command prompt after displaying Fig 5-4.

|  |  |
| --- | --- |
| Fig 5-4 Help Display | |
| (D)ELETE:  E(X)IT:  (H)ELP:  (I)NSERT:  (L)IST:  (Q)UIT:  (R)UN:  S(E)T:  SHO(W):  (T)RUNCATE:  (V)IEW: | Delete input string from list  Exit application  Display Commands  Insert input string into list  List input strings  Quit operation of TM on input string  Run TM on input string  Set maximum number of transitions to perform  Show status of application  Set truncation length of instantaneous description  View Turing Machine |

**5.2.2 Show Status**

Invocation: w or W (only)

*Show* user command is designed to display nine different categories related to the application. There are two main details status of the application and configuration settings. Refer to Fig 5-5 for an example.

Status of the application include: Course, Semester, Year, Instructor, Author, and Application Version, Turing Machine Name, Status of Turing Machine. The Turing Machine Name is copied from the definition file. The application will remove the “.def “ extension before displaying.

Status of the Turing machine has three options to be displayed. One, Turing Machine has never been run on an input string. Second, Turing Machine is currently running on an input string. Third, Turing Machine has completed operation on an input string.

Configuration settings include: Maximum number of transitions, max number of left and right cells of the tape head.

|  |  |
| --- | --- |
| Fig 5-5 Show Display | |
| Course:  Semester and Year:  Instructor:  Author:  Version:  Configuration  Setting:  TM Name:  Status of TM: | Computer Science 322  Fall, 2018  Dr. Neil Corrigan  Emmanuel Bonilla  1.0  Max transitions: 1  Max L and R cells: 1  AlanTM  AlanTM has never run on input string |

**5.2.3 View Turing Machine**

Invocation: v or V (only)

*View* user command is designed to display the definition of the currently loaded Turing Machine on the monitor in a form readable by the user. Fig 5-6 is an example of a form readable to the user.

|  |
| --- |
| Fig 5-6 View Display Example |
| 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, R)  δ(s2, X) = (s0, X, R)  δ(s2, Y) = (s2, Y, R)  δ(s3, Y) = (s3, Y, R)  δ(s3, -) = (s4, -, R)  q0 = s0  B= -  F = {s4} |

**5.2.4 List Input Strings**

Invocation: l or L (only)

*List* user command is designed to display a list of input strings upon which Turing Machine may be operated. Each input string will appear on its own line. The numbers will start at 1 and continue sequentially. If the list is empty, the application will produce an appropriate error message.

|  |
| --- |
| Fig 5-7 List Display Example |
| [1] ababa  [2] bbb  [3] aa  [4] \ |

**5.2.5 Insert Input String**

Invocation: i or I (only)

*Insert* input string command allows users to enter an input string and append it to the list. The application will prompt the user for an input string. For the string to be valid it must be check three conditions. First if all characters are in the input alphabet (Σ). Second, ensure the string is not already in the list. Third, there are no additional white spaces. The application will display an appropriate message weather the input is valid or invalid.

**5.2.6 Delete Input String**

Invocation: d or D (only)

*Delete* input string command allows the user to delete an input string from list. Input string is indicated by the number only, not by typing the string. If user entry is valid, string will be deleted, and list will be renumbered, even if the string is being evaluated. The application will display an appropriate message whether the input is valid or invalid.

**5.2.7 Set Transitions**

Invocation: e or E (only)

*Set* transitions command allows the user to change the setting for maximum number of transitions on an input string. Valid necessary input can be a positive integer or blank space. Blank space will be treated as empty and transition value will not change. Invalid inputs are negative numbers, decimals, or other characters. The application will display an appropriate message whether the input is valid or invalid.

**5.2.8 Truncate Instantaneous Descriptions**

Invocation: t or T (only)

*Truncate* command allows the user to change the setting for the maximum number of cells to the left and right of the tape head to display an ID. A valid users input can be a positive integer or blank space. Blank space will be treated as empty and transition value will not change. Invalid inputs are negative numbers, decimals, or other characters. The application will display an appropriate message whether the input is valid or invalid.

**5.2.9 Run Turing Machine**

Invocation: r or R (only)

*Run* command allows the user to trace operation of Turing Machine on an input string using ID’s. The application prompts the user for a positive string number. The string number represents the string to be operated in the application. If user input is not in the list or not a positive integer an appropriate message will appear. If valid input application will display message. Refer to Fig 5-8 for an example.

|  |
| --- |
| Fig 5-8 Run Display Example |
| Input string number: 3  0. [s0] aabb  7. XX [s4] YY  aabb accepted in 7 transitions |

**5.2.10 Quit Turing Machine**

Invocation: q or Q (only)

*Quit* command allows the user to terminate operation of Turing Machine on input string before completion. If the Turing Machine is operating on a string, a message will be displayed. The message will display “input string”, if the Turing Machine has accepted or rejected the input string, and the number of transitions performed before quitting. An appropriate error message will be displayed if Turing machine is not operating on a string.

**5.2.11 Exit Application**

Invocation: x or X (only)

*Exit* command allows the user to terminate execution of application. If the input string was changed in anyway, the entire list is written to the input string file, replacing the original file. A display message will appear if the file was successful or a failure.

**5.3 Termination**

**5.3.1 Closing Turing Machine**

Once the exit command is complete (5.2.1), all dynamic memory is freed, and the application is terminated. No values are stored at all.

**REFERENCES**

FIG 1-1

Turing machine. (2018, February 26). Retrieved February 27, 2018, from https://en.wikipedia.org/wiki/Turing\_machine

**APPENDIX**