Turing Machine Simulator

CPTS 322

Ryan Wilson

March 7, 2014

Table of contents

List of Figures

Revision History

Revision 5-4-14 added page numbers to table of contents, and other minor changes.

First draft no revisions. 3-7-14

1.0 Introduction

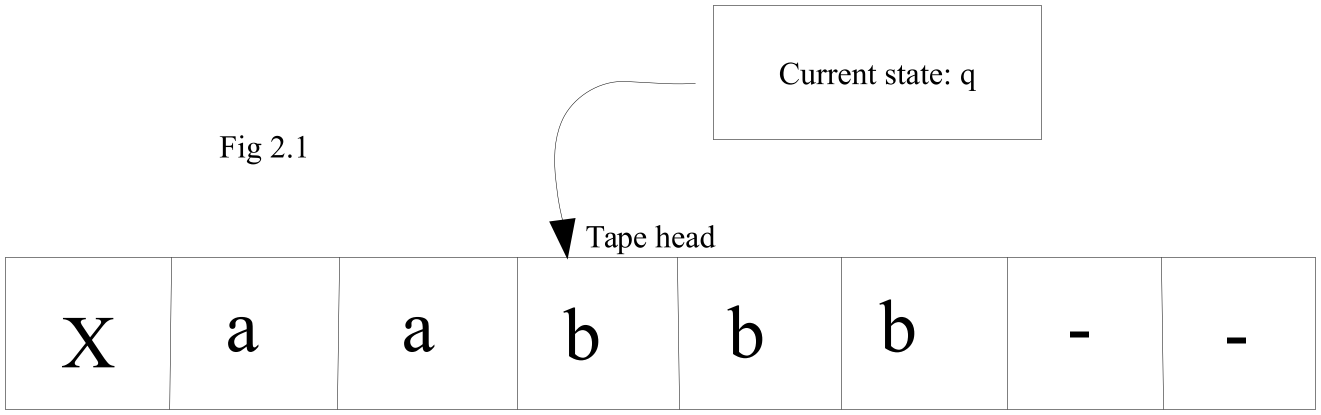
Requirement specifications for a Turing machine computer application intended for professionals in the theoretical studying of what a Turing machine is capable of. To know how this application is going to help, some background on what a Turing machine is and how it works. Followed by the purpose of the Turing machine application, hardware and software environment for development and installation. The environment will be clearly explained, as well as the operation of application. Finally reference and appendix.

2.0 Background

A Turing machine was conceived by Alan Turing in the 1930's, it's purpose is to recognize a language. Its theoretical capabilities are limited only by the imagination of the user, because in theory it is infinitely fast and has infinite storage. A language consists of an alphabet denoted by ∑ which consists of a finite set of characters used to construct strings. A string is a finite sequence of characters from ∑. A language is a set of these strings. An example of a language would be written $L=\{a^n b^n \mid n \geq 1\}$, this language would consist of a number greater than or equal to one a's followed by the same number of b's. The alphabet would be ∑={a,b}, the string w=aaabbb would be accepted by a Turing machine for this language.

A Turing machine also has an infinitely long tape, which is used to store the input string. This tape contains cells, each cell holds a single character from the tape alphabet. The tape alphabet denoted by Γ consists of ∑ and any other characters necessary including a character that represents the blank character. The tape also has a starting cell usually starting with the first character of the string. For simplicity this Turing machine tape is only infinite to the right, and the starting cell is located to the far left cell. The string is inserted to the left and followed by an infinite number of blank characters. The Turing machine can see what's in the current cell, known as the tape head, replace that cell with a character from Γ, and move the tape head one cell to the left or to the right as in figure 2.1. However, if at the first cell and is moved left off the tape that would crash the Turing machine rejecting the input string.

Fig 2.1



States are how a Turing machine determines what to do when it sees the character in the current cell. There are transitions in or out of these states. Transitions are instructions on what state to transfer into, what character to replace in the current cell, and which way to move the current cell. These states and transitions can be represented with graphs. The figure 2.2 is the graph of the states of a Turing machine that would accept the example language $L=\{a^n b^n \mid n \geq 1\}$. The arrows are the transitions out of the states

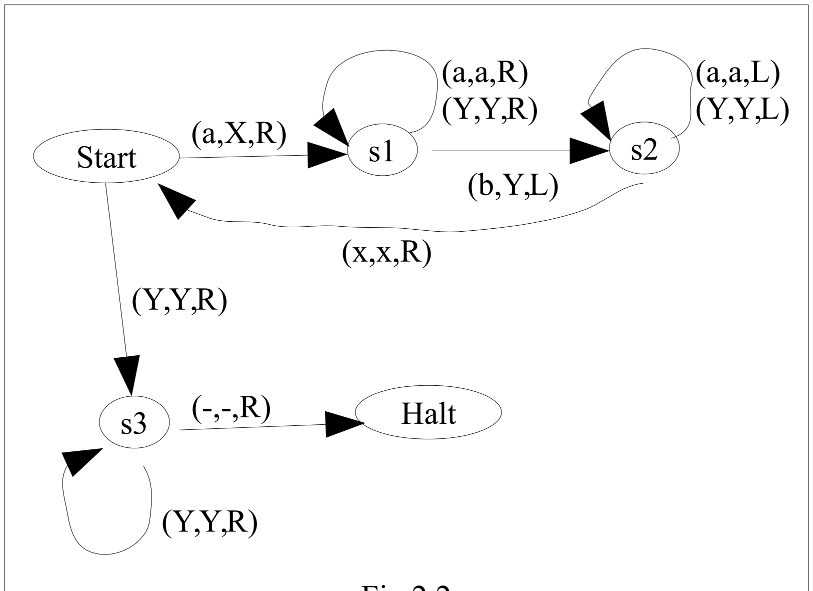


Fig 2.2

labeled from left to right, what character the Turing machine sees the replacement character, and the direction to move the tape head. Following the arrows if the first character is an a the tape head is replaced with a X, then moved to state s1. While in s1 the tape head will be moved to the right until a b is reach. It is replaced with a Y and the current state is changed to s2, and the tape head is moved to the left. It continues left until it sees an X, the state is changed to start, it then moves the tape head to the right if the next character is a Y the state is changed to s3, if an a is seen the state changes to s1 and the process of looking for a b and coming back to find an X repeats. If the current state is s3 the characters being read in are Y's, the tape head is moved to the right. If the blank character is seen the state is changed to the halt state and the string is accepted. This is just one example for a Turing machine, the formal definition is $M=(Q,\sum, \Gamma,\delta,q_0,B,F)$, where

Q is a finite set of states
$\sum$ as a subset of $(\Gamma - \{B\})$ is a finite input alphabet
Γ is a finite tape alphabet
δ is a transition function from $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$
q_0 is a state in Q that is the initial state
B is a character in Γ that is the blank symbol
F is a state in Q that is the final state

The above Turing machine example would look like this.

$Q = \{s0,s1,s2,s3,s4\}$

$\sum = \{a,b\}$

$\Gamma = \{a, b, X, Y, -\}$

$\delta(s0, a) = (s1, X, R)$
$\delta(s0, Y) = (s3, Y, R)$
$\delta(s1, a) = (s1, a, R)$
$\delta(s1, b) = (s2, Y, L)$
$\delta(s1, Y) = (s1, Y, R)$
$\delta(s2, a) = (s2, a, L)$
$\delta(s2, X) = (s0, X, R)$

$\delta(s2, Y) = (s2, Y, L)$
$\delta(s3, Y) = (s3, Y, R)$
$\delta(s3, -) = (s4, -, R)$

$q_0 = s0$

$B = -$

$F = \{s4\}$

Any transitions not included in the definition are undefined.

3.0 Overview

The purpose of this Turing machine application is to allow a user to interactively trace the operation of a Turing machine definition, and run strings on this definition to see if that string is accepted or rejected. This application will need a computer with the Linux operating system, keyboard, and monitor. Users will activate the program by typing on the name of the program followed by the name of a Turing machine definition file, which would be (<some name>.def). This file will contain a definition of a single Turing machine similar to the formal definition with a few difference explained in the environment section. Only a single Turing machine definition can be loaded per instance of the program. There is also an optional input string file, explained in the environment section, with the same name but ending in (.str). Once a Turing machine definition is loaded, along with the optional string file, the user can use the commands to perform operations on the Turing machine explained in the commands section.

4.0 Environment

The application will operate as a console application on a PC using the Ubuntu Linux operation system with the Gnu g++ compiler. It will be implemented in the c++ programing language using the c++ standard library.

4.1 Input and output devices

Input devices include the keyboard, Turing machine definition file, and input string file. Output devices included the monitor and the input string file. The keyboard is going to be used for inputing commands and responses for actions involved after selecting a command. The Turing machine definition file (<some name>.def) will be loaded upon application start up, and is necessary for operation of program. The input string file being both an input and output device will be loaded if it exist in the same directory as the

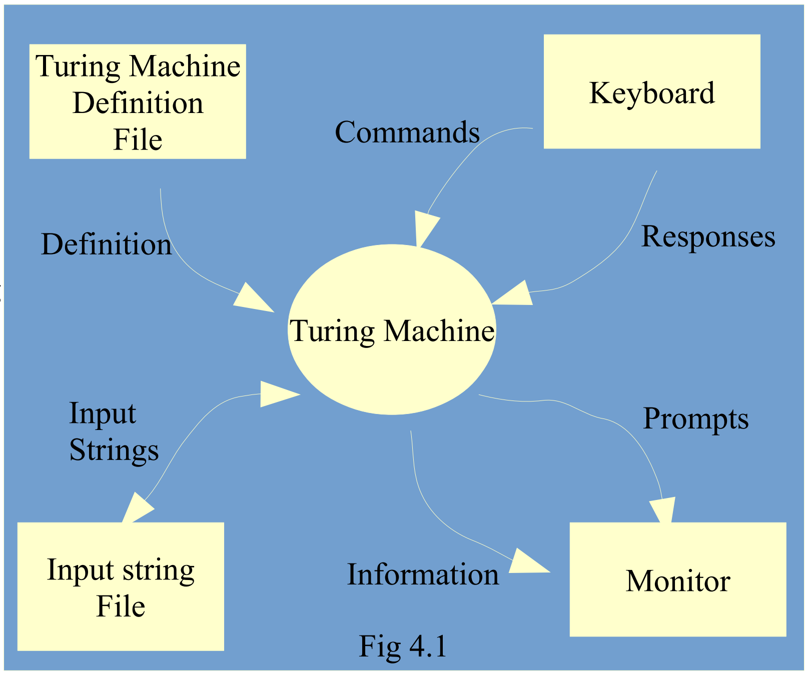


Fig 4.1

definition file, and will be saved after exiting the program if any strings were added or deleted from the input string list. The monitor will be used for displaying prompts, and information about the program and Turing machine during operations. Figure 4.1 shows a diagram of these interactions.

4.2 Turing machine definition file

The Turing machine definition file will be constructed using a text editor like note pad or vii if on Linux, but any plain text editor will work. Something that save the file with special formating instructions included in the file will not, and cause the program to display an error message and exit. Any name can be given to the file, but it must end with (.def), so TM.def, alpha.def, and a.def are acceptable. Any problems with the definition file will cause the program to display possible error messages and exit. Keywords will be used to specify different parts and must be in order. These keywords in order are States:, Input_alphabet:, Tape_alphabet:, Transition_function:, Initial_state:, Blank_character:, and Final_states:. The colon and underscore is included in the keywords, but keywords may be in upper, lower, or any combination of case. The file may begin with a description which is considered anything before States:. It can be written how ever the user wishes as long as it doesn't contain any of the keywords. White space is the delimiter of this file, and is considered spaces, tabs, and new lines in any combination. This definition file is considered free format, meaning that as long as there is white space between entries it will work. An example of this is, if inputing "States: s0 s1" is the same as "States: s0 <tab> s1" or "States: <new line> s0 <new line> s1".

Starting with the keyword States:, this is where the user would input all the names of states for the Turing machine. The names of the states have no limit on length, and is a string of upper or lower case letters, digits, or underscores. All state names are case sensitive and must be unique. Some examples are "state1", "state_2", "s3", "alpha", or any other names wished. Invalid state names would be "/state", "state 1", "[state]", or any other that violates the rules. With "state 1", because of the space it's not really an invalid name, but would be considered two state names.

The alphabet keywords Input_alphabet: and Tape_alphabet: must consist of characters from the ASCII character set, with the exception of '\', '[', ']', '<', '>' and white spaces. Because ∑ is a subset of Γ every character in ∑ must be contained in Γ. The blank symbol B must be in Γ and must not be in ∑.

The keyword Transition_function: is for all transition functions. An example input in the definition file would look like this "Transition_function: s0 a   s1 X R". Any number of transition may be defined, and any undefined transitions from Q x Γ will cause the Turing machine to crash rejecting the string. Transition functions must use states in Q and tape characters from Γ, and define which direction to move the tape head with L or R in upper or lower case. Turing machine is deterministic, so at most one transition from a given state on a given tape character. So a transition out of s1 with an a will only do one thing. No transitions out from any states defined to be a final states. There must be only one initial state q_0, and must be a state in Q. Finally any state in Q may be defined as a final state in F. Some examples of acceptable definition files are as follows:

tm.def

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     s3 Y R
s3 Y     s3 Y R
s3 -     s4 – R

INITIAL_STATE: s0

BLANK_CHARACTER: -

FINAL_STATES: s4
```

tm.def

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 s3 Y R s3 Y s3 Y R s3 - s4 – R INITIAL_STATE: s0 BLANK_CHARACTER: - FINAL_STATES: s4

Some formates that wouldn't be accepted are:

tm.def

```
        This Turing machine accepts the language of one or more a's followed by the same number of
b's.

INPUT_ALPHABET: a b

STATES: s0 s1 s2 s3 s4

TAPE_ALPHABET: abXY-

TRANSITION_FUNCTION:
s0a      s1 X R
s0 Y     s3 Y R
s1 a     s1 a R
s1 b     s2 Y L
```

```
s1Y     s1 Y R
s2 a    s2 a L
s2 X    s0 X R
s2 Y    s3 Y R
s3 Y    s3 Y R
s3 -    s4 – R

INITIAL_STATE: s7

BLANK CHARACTER: -

FINAL_STATES:s4
```

This file would be rejected because input_alphabet: comes before states:, the tape_alphabet: has no spaces between characters, the first transition has no space between s0 and a, the initial_state: isn't in Q, the blank_character is missing the underscore, and there is no space between final_states: and s4. Any problems in countered while reading the file will cause the program to display as many error messages as possible and exit the program.

4.3 Input string file

The input string file must have the same name as the definition file except instead of .def it ends with (.str), TM.str, alpha.str, and a.str are examples. This file is completely optional and is not required for operation of the application. Each line of the file is considered to be one input string, and may be of any length. This means that there is no new line or carriage returns in it, but may wrap to the next line on some text editors. Blank lines are invalid, to represent the empty string the \ is used. Only one \ can be used on the line, and a \ with any other characters is also invalid. The characters used in this file must come from ∑. Duplicate entries are also invalid, and all invalid lines will be discarded with a message displaying errors. The application will not exit upon these errors just not read in that line. There's no limit of the amount of strings a user can use in this file. The input strings are read into and maintained in a list by the application which allows users to insert or delete strings from this list. If any changes occur to this list, the entire list of input strings will be written to file when application is terminated. If no file exists when the application exits a file with the name of the Turing machine will be created with the extension (.str). If the file exists it will be overwritten and original file will be lost. An example file could look like this:

tm.str

```
a
aaabbb
ababab
\
bbaa
cds

\\
\ab
```

aaabbb
a a abbb

The first 5 strings in this file would be inserted into the list, the rest would be discarded. The string "cds" doesn't contain characters from ∑. The double \ and the \ followed by characters are not allowed, as well as the blank line. The aaabbb is already inserted into the list, and there is spaces in the last string.

5.0 Operation

Starting the program is done from the command line using the name of the application followed by the name of the Turing machine. After the program is loaded three configuration settings are established with their default values. These values can be changed while the application is running, but will not be saved upon termination of the program. These settings are help, transitions to perform, maximum cells. The help setting is for user help, displaying advice on what the prompt on the monitor is looking for. The default value for this is off. The transitions to perform is the maximum number of transitions to perform before prompting if the user wishes to continue. The default value for this setting is one. Maximum cells is the maximum number of cells to display on the monitor to the left and right in the instantaneous description. The default value is 32. The instantaneous description is how the current state of the Turing machine running on the current input string.

5.1 Invocation

5.1.1 Command line

On the command line the user would type in the name of the application followed by the name of the Turing machine definition file with out the .def extension. An example is "TMapp alpha" where TMapp is the name of the program and alpha is the name of the Turing machine definition file with out the .def extension. The definition file can be relative or absolute, meaning it could be in the same directory as the Turing machine application as in the example, or the full path can be included like "/home/TmDefinitionFiles/TMapp". An application usage message will be displayed if other arguments are included, and execution will be terminated. If there is a input string file that must be in the same directory as the definition file, or the program will load as if there is no input string file. Only a valid definition file will be accepted other wise the program will terminate, refer to section 4.2 "Turing machine definition file" for information on valid definition files and examples. If a valid definition file is loaded the program will display the message "Definition file loaded successfully" and display the prompt "Command:". This is where all commands will be entered followed by pressing the enter key.

5.1.2 Configuration settings

The three default configuration settings will be loaded at the start of the program, and may be changed using commands explained in more detail in the commands section(5.2). The help setting is to help the user with command prompts. The default setting is off with only on as an alternative. After turning it on messages will be displayed before the prompt. An example of this would be all the commands, the character to use, and a short description of what the command does and is used for. More examples in the commands section(5.2).

The transitions to perform setting will set the maximum number of transitions to perform. After invoking the command to change this setting a prompt displaying the current number will be displayed, then the user can enter a new number which will change it. Once changed the Turing machine will then run up to that many transitions on the input string before displaying a prompt asking if they wish to continue. The input string may be rejected or accepted during this time, and will display the number of transitions performed on that string upon rejection or acceptance. More examples in the commands section(5.2).

The maximum cells setting is the number of cells to display to the user in the instantaneous description. If the input string has more characters than this setting the < and > are used to show that there are more characters than what's being displayed. Using the command to change this setting will show the current value of the setting, and prompt the user for a new number. More examples of what this looks like are in the commands section(5.2).

5.2 Commands

The commands will be entered after the prompt and followed by pressing the enter key. With all commands if nothing is entered where a command or user input should, and the enter key is pressed that will bring back the command prompt. As an example while in the delete string command the user may change their mind, or forget which string number they wish to delete, in this case pressing the enter key with no input will bring the command prompt back up and wait for another command. Any invalid commands or entries will cause an error message to display, and bring the command prompt back up.

5.2.1 Help user

The help user command invoked with the character 'H', toggles the help setting on or off, default is off. Once invoked every prompt will have a help message displayed before the prompt it self. The command prompt will display all the commands, the character to use, and a short description of what the command does and is used for. After using one of the other commands like 'I' for input, the help message will tell the user "This is where you enter a string for insertion into the list", and will be followed by a few examples. Once toggled off the help messages are no longer displayed to the user.

5.2.2 Show status

The show status command invoked with the character 'W', will show the status of the application. This will include course name, instructors name, authors name, version number, the settings and their value, and current state of the Turing machine. An example is:

```
Course: CPTS 322 spring 2014
Instructor name: Neil Corrigan
Author name: Ryan Wilson
Version 1.01

Help is disabled.
Max number of transitions set to: 1
Showing 32 characters to the left and right of instantaneous discription.

Turing machine “alpha” is running input string “aaabbb”
```

Total number of transitions: 35

The first section will not change much, but the other two will change depending on the current value of the setting, and current state of the Turing machine. The current state of the Turing machine could be one of the possibilities. The Turing machine “alpha” has never ran an input string, the Turing machine “alpha” is running on a string, or Turing machine “alpha” has accepted or rejected the input string. The status is not saved upon exiting of the program, nor is the status of a previous status view.

5.2.3 View Turing machine

The view command, invoked by entering the character 'V', shows the current Turing machine loaded. This includes the description included in the file before the keyword states:, The states Q, alphabets sigma and gamma, transitions, start state, blank symbol, and final state. An example is

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}

Sigma = {a,b}

Gamma = {a, b, X, Y, -}

Delta(s0, a) = (s1, X, R)
Delta(s0, Y)= (s3, Y, R)
Delta s1, a) = (s1, a, R)
Delta(s1, b) = (s2, Y, L)
Delta(s1, Y) = (s1, Y, R)
Delta(s2, a) = (s2, a, L)
Delta(s2, X) = (s0, X, R)
Delta(s2, Y) = (s2, Y, L)
Delta(s3, Y) = (s3, Y, R)
Delta(s3, -) = (s4, -, R)

q_0 = s0

B = -

F = {s4}

5.2.4 List input strings

The list input strings command, invoked by entering the character 'L', displays the list of input strings. No special wrapping will be done by the program if the string is longer than the size of the console, all that is handled by the operating system. Each string in the list will be displayed on a new line, except where wrapping occurs. If the list is empty a message will be displayed stating that fact. All strings in the list will be numbered starting with 1, this is to be used for selecting a string from the list in other commands. An example is:

```
1. aabb
2. aaaabbbb
3. aab
4. abababa
5. aaaaabbbbbbbbbbbbbbbbbbbbb
6. bbbbbbbbbbbbbbbbaaaaaaaaaaa
```

The program will not make any attempt to display the list of input strings in a pages format or allow for scrolling, all of that will be handled by the console and operating system.

5.2.5 Insert input string

The insert input string command, invoked by entering the character 'I', allows the user to add additional strings to the input string list. After invoking this command the program will prompt for a input string. An example would be:

```
Command: I

Insert input string: aaabbbbb

String “aaabbbbb” added to list.
```

If the string already exists or the string couldn't be entered, a error message is displayed, and returns to wait for commands.

5.2.6 Delete input string

The delete command, invoked by entering the character 'D', will bring up a prompt asking for the string number related to the list for deletion. An example is:

```
Command: d

String number for deletion: 3

String “aaabbb” was deleted from list.
```

If there were an error that didn't allow the deletion, or the number entered isn't in the list a message would be displayed, and would return to wait for a new command.

5.2.7 Set Transitions

The set transitions command, invoked by entering the character 'E', is used to set the maximum number of transitions to complete before prompting the user if they wish to continue on that string. The current value is displayed and prompts for the user to enter a number.

```
Command: e

Set Transitions[1]: 25

Maximum number of transitions set to 25.
```

5.2.8 Truncate instantaneous description

The truncate instantaneous description command, invoked by entering the character 'T', is how the user changes that setting's value. If the value was set to 3 the instantaneous description may look like:

```
8. <aab[s2]bba>
```

Eight is the number of transitions, the < shows that there are characters to the left, and > show there are more characters to right.

5.2.9 Run Turing machine

The run command, invoked by entering the character 'R', runs a string on the loaded Turing machine. If no string is running pressing the command will prompt the user for a string number. If there is a string running, and the command is used, then the current string will be run through transitions. The maximum transitions that are run are determined by the maximum transitions setting. An example of program input and output would be:

```
Command: r

Enter input string number: 2

0. [s0]aaabbb

Running: r

1. X[s1]aabbb

Running: r

2. Xa[s1]abbb

Running: e

Maximum number of transitions[1]: 50

Maximum number of transitions set to 50.

Running: r
```

```
16. XXXYYY[s4]

Turing machine accepted input string “aaabbb” in 16 transitions.
```

5.2.10 Quiting Turing machine

The quit command, invoked by entering the character 'Q', quits running on a current string. This does not exit the program. An example would be:

```
Command: r

Enter input string number: 2

0. [s0]aaabbb

Running: r

1. X[s1]aabbb

Running: r

2. Xa[s1]abbb

Running:q

String “aaabbb” not accepted or rejected, user quit in 2 transitions.
```

If quit command is used when not operating on a string, an error message will be displayed and return to wait for commands.

5.2.11 Exit applications

The exit command, invoked by entering the character 'X'. terminates the application. If any changes to the input string file were made the program will save them to a file with the name of the Turing machine and the .str extension. A message displaying if the file was saved successfully or not. No warnings will be given before application terminates.

5.3 Termination

5.3.1 Closing Turing machine

After closing the Turing machine with the exit command the computer will return to the operating system's command prompt. Here a user can restart the application with a new Turing machine definition.

References

Appendix