

# CS660 Project Report – Turing Machine

**Ge Gao**

gg5j@virginia.edu

## 1. Overview

I develop a demo of deterministic Turing machine. It can display the running process of any user-specified Turing machine by loading a Turing Machine specification file. The data structure that records the specification is initialize when the specification file is loaded, during which the check for grammar soundness is performed. After the user then input the string into the text field at the bottom of the window and “run”, the Turing Machine start the recognizing process to determine whether the input can be accepted by the Turing Machine. The demo can display each configuration of the Turing Machine during the recognizing process until it reaches the accept or reject configuration. Each configuration is presented by a multi-color string in the middle, with red colored character and state symbol showing the current position of read/write head. Also the demo gives the transition function currently used to perform the state transition in the Turing Machine. At the end of the running process, the outcome of whether the input string belongs to the language decided by the Turing Machine is shown on the screen.

## 2. Instructions

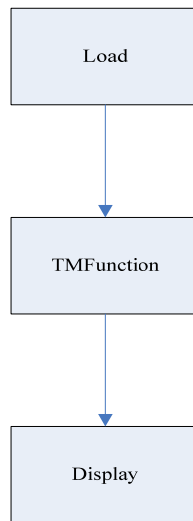
To use our demo, you should:

1. First, click the “LOAD” button on the toolbar and select the Turing Machine specification file from the dialogue.
2. Second, input string you want to examine in the text field at the bottom of the window.
3. Thirdly, click “RUN” on the toolbar.

I provide sample Turing Machine specification files: TMState1\_02n.txt and TMState2\_w#w.txt, which are for TM recognizes the language  $\{0^{2n}\}$  and  $\{w#w \mid w \text{ is in } \{0, 1\}^*\}$ , respectively.

## 3. Project Details Description

The demo is developed based on Java JApplet. It primarily is composed of 3 modules: the loading module, Turing Machine function module and the screen display module. The framework of our demo is shown as follows:



## 1. Input Module

In this module, I implement the class used to load the Turing Machine specification file into the data structures. Most of these functions are implemented in our input class. I decide to use the transition function set to represent the Turing Machine. The initial state, accept state and reject state are defined as “q1”, “qaccept”, “qreject” respectively. I use the Class StateStructure which has 5 fields to represent the 5 variables in a transition function: current state, current tape content, next state, content after modification, L/R action. In the TM specification file, different transition function will be departed by “\n” and different variables of a departed function will be departed by”,” with no “ “ between different variable. Class input is initialized with the name of the state transition source file. The most important member methods is public StateStructure ReadOneState() which read one line of string from the source file if it does not reach the end of file and use the string to initialize a StateStructure object. The method public int initializeStateStructure(StateStructure[] S) take the ReadOneState() method and initialize the State Structure array to represent the Turing Machine. In addition, I provide an easy way to check the correctness of the input state transition source file. public boolean Checkduplicate(StateStructure[] TMStateInfo) are used to check there are no duplicated pair of current state and current tape content in the source file which means error in a deterministic Turing Machine state transition source file.

## 2. Turing Machine function Module

This Module is used to represent the Turing Machine transition function set, choose the right function to use under certain current state and current tape content, and update the new current state, current tape content and RW-head position. I use boolean matchState(String input, String state) to judge whether a specific transition function match the current state and current tape content. If one function is found, the public boolean NextState(String Language, int CurrentIndex, String state) is used to make the whole update that I described above. It is also possible that no

suitable transition function in the function set are suitable under current input and tape content. Under this circumstance, I will report an warning to the user through a JOptionPane and let him decide what to do. It is very likely, the user forget to input some transition function in the source file. He can reload the Turing Machine and run the demo again.

### 3. Display Module

This Module takes charge of displaying useful running state of the demo as well as that of the Turing Machine. A message box will be pop out whenever a load error happens or no suitable state transition functions are found. More importantly, I need to display the content in the tape, the function to use under current circumstances and the right transition function to use. I use the protected void ShowLanguage(String s1, Graphics g, StateStructure TMState) to display the contents on the tape. I choose the current tape content to be always displayed in the center of the screen. Current state is inserted left to the current tape content. As a result, the contents of the tape are displayed in a dynamic animation with the current tape content always in the center of the JApplet. The action of RW head movement are expressed through the swing of current tape content. I choose a different color to display the current tape content and current state in order to make such information more notable. I use protected void ShowRule(Graphics g, StateStructure TMState) to display the transition function to use right below the displayed tape content. If the Turing Machine reaches the halt configuration, “accept” or “reject” are output below the tape content.