COSC 312 Turing Machine

Group 11:
Noah Hobson, Marvin Joshi, Isaac Sikkema

March 28, 2019

Collaborators and Roles

The collaborators, as well as their netids, and the roles for those in Group 11 were the following:

- Noah Hobson (nhobson): Designed and Debuged Error Section of Turing Machine
- Marvin Joshi (mjoshi5): Debuged Turing Machine, Wrote Report
- Isaac Sikkema (isikkema): Designed and Debuged Turing Machine

Problem Description

We decided to design a Turing Machine that was able to take in an Algorithmic Expression, that can contain the following characters, '(', ')', '+', '-', '*', '/', '0', '1', '2', '3', '4', '5', '6', '7', '8', and '9', and output whether or not the expression is valid. Validity is determined by making sure that there is a open parenthesis, '(', that is completed by a closed parenthesis, ')', later on in the input tape. The Turing Machine also checks to make sure that the mathmatical expressions in the tape is valid and follows basic arithmetic rules.

Motivation

Our motivation for designing this Machine came from the fact that arithmetic expressions can be very simple or very complex. Lots of programs around the world, including nearly every compliler, use arithmetic expressions to implement one functionality or another. We thought that an analyzer for these expressions would be an interesting problem with an extermely wide range of use cases.

TM Functional Design

The complete state diagram for our Turing Machine can be seen below in Figure 1.

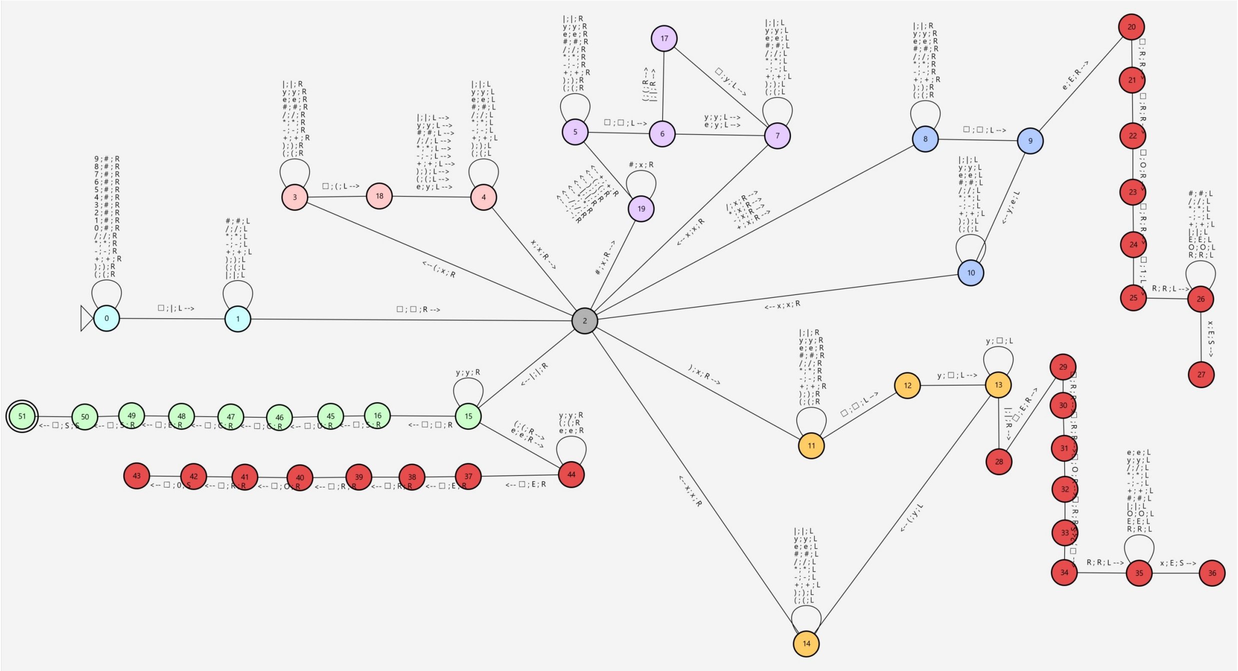

Figure 1: Complete State Diagram of Turing Machine

The following Figures show the breakdown of our Turing Machine. Figure 2 demonstrates how the Turing Machines goes through the tape and changes all the numbers in the tape to the '#' sign. The machine goes through the whole tape and performs this task and once it reaches the end, it adds a '|' to illustrate the beginning of the stack. The tape then goes back to the beginning.

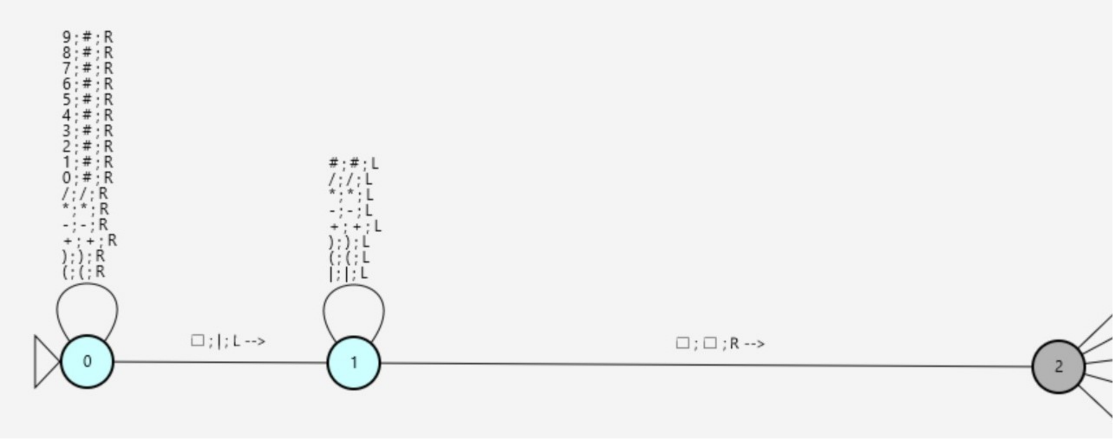

Figure 2: Light Blue Section of Turing Machine

The next figure, Figure 3, is the section of the Turing Machine that takes in a '('. Once it sees a '(', it changes it to an 'x' and goes through the tape until it reaches the first blank space in the tape. Once it reaches that point, it will ouput a '(' and go back through the tape until it sees an 'x' in the tape.

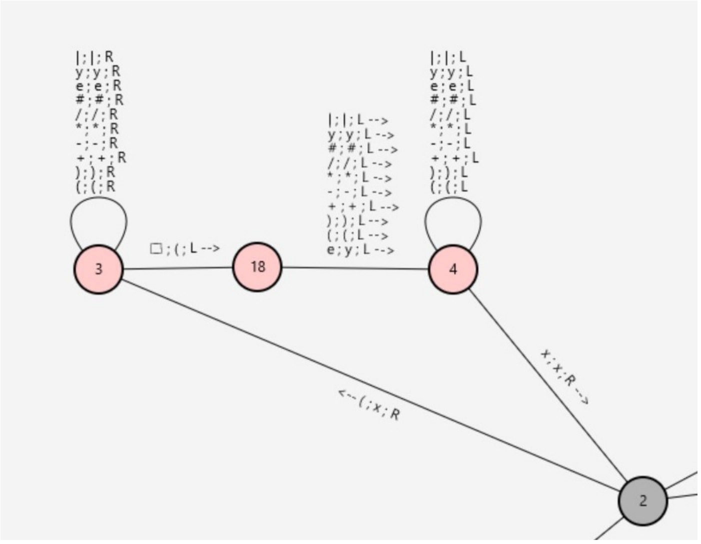

Figure 3: Pink Section of Turing Machine

Figure 4 shows what the machine does when it sees a number, '#'. The machine does the same thing as before and changes the '#' to an 'x'. It then goes to next empty space that it sees, which is the last space in tape, and goes to the left. If there a '(' or '|' at that position, a 'y' is added. A 'y' means that the tape is valid up to the position of the number , '#'. If there is an 'e' or a 'y', the character is changed to a 'y'. The machine then goes back through the tape until it sees an 'x' in the tape.

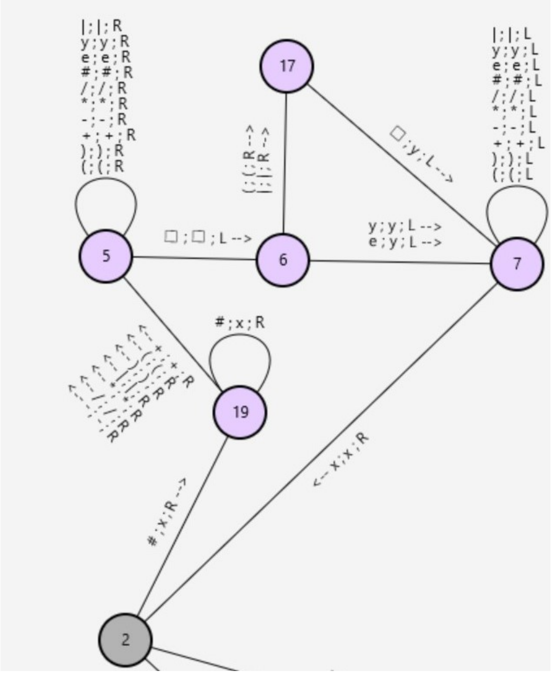

Figure 4: Purple Section of Turing Machine

The next figure, Figure 5, shows what the machine does when it sees a arithmetic expression, +, -, *, or /. The machine does the same thing as before and changes the expression, #, to an 'x'. It then goes to next empty space that it sees, which is the last space in tape, and goes to the left. In order for the input string to be valid, the character following a arithmetic expression must be a 'y'. A 'y' represents that the string is valid up to this point. If the machine does see a 'y', it changes it to an 'e'. An 'e' illustrates that the machine has seen a number followed by a valid mathematical expression. In order the for the input string to be valid, a number must also follow a mathematical expression. As explained in the section above, if a number, '#', is inputed and the last value in the stack is a 'e', the machine changes the 'e' to a 'y', which means the mathmatical expression is valid. The machine then goes back through the tape until it sees an 'x' in the tape. There is an error that can occur here. If the user has inputed subsequent arithmetic expressions, there will be an 'e' at the end of the stack. This machine then goes through the red part of this section and outputs 'ERROR1' and replaces the error in the string with 'E'.

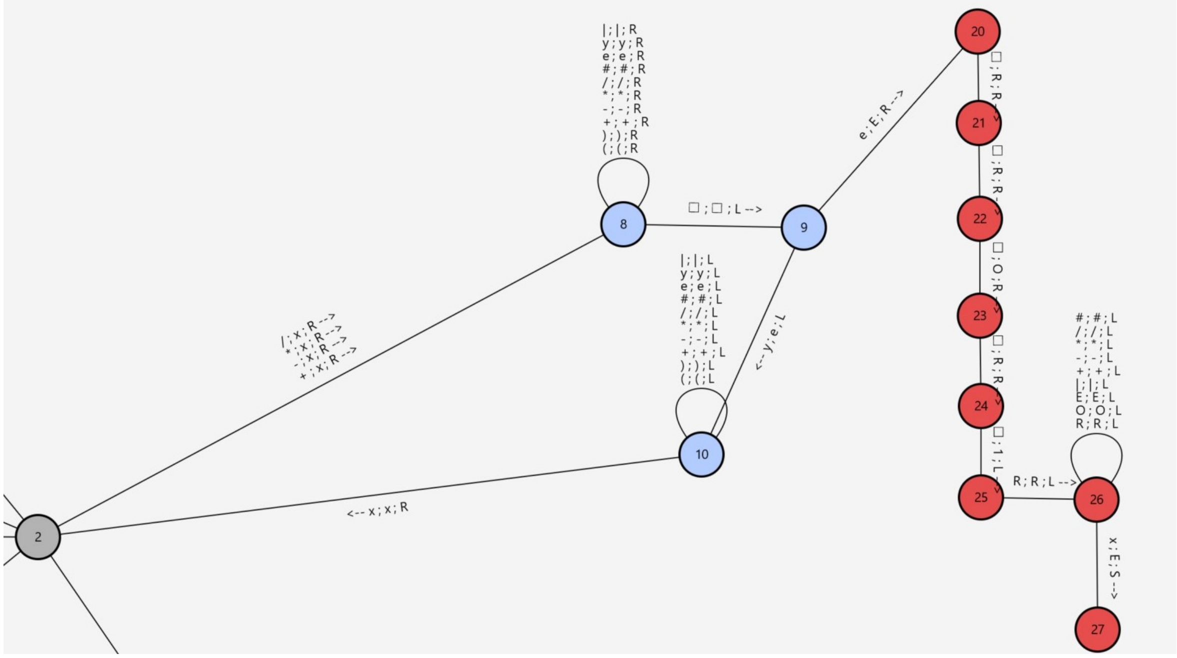

Figure 5: Dark Blue Section of Turing Machine

Figure 6 represents the section of the Turing Machine that sees a ')'. Once it sees a ')', it changes it to an 'x' and goes through the tape until it reaches the first blank space in the tape, and goes to the left. Once we reach this part of the machine, if the input string is valid, the only characters in the stack should be either '(' or 'y'. If the character to the left is a 'y', it changes it to a blank space and goes backwards. It repeats this step for as many y's that are following it. If the next character is a '(', it changes the '(' to a 'y' and goes back through the tape until it sees an 'x'. Another type of error can occur here. If there are any extra parenthesis at the end of the input string, the machine goes the red part of this section and outputs 'ERROR2' and replaces the error in the string with a 'E'.

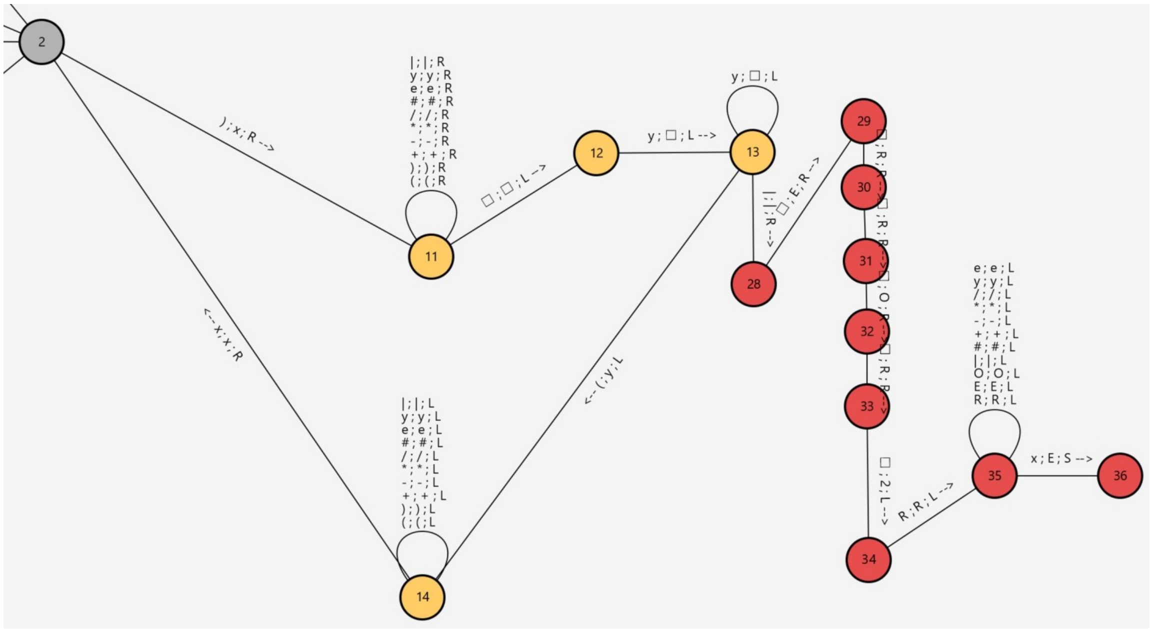

Figure 6: Yellow Section of Turing Machine

The last figure represents end of the Turing Machine. If the input string was valid, the only character remaining in the stack should be 'y's'. The machine should also be starting at the '|'. In this part, the machine goes through the stack and goes to the next space as long as it sees a 'y'. Once it sees a blank space, the machine prints out the word 'SUCCESS' and moves to the accept state. The last type of error can occur here. If the user entered in any unclosed parenthesis in the input string, the stack would not contain only character 'y'. The machine then goes to the red section of this section and outputs 'ERROR0'.

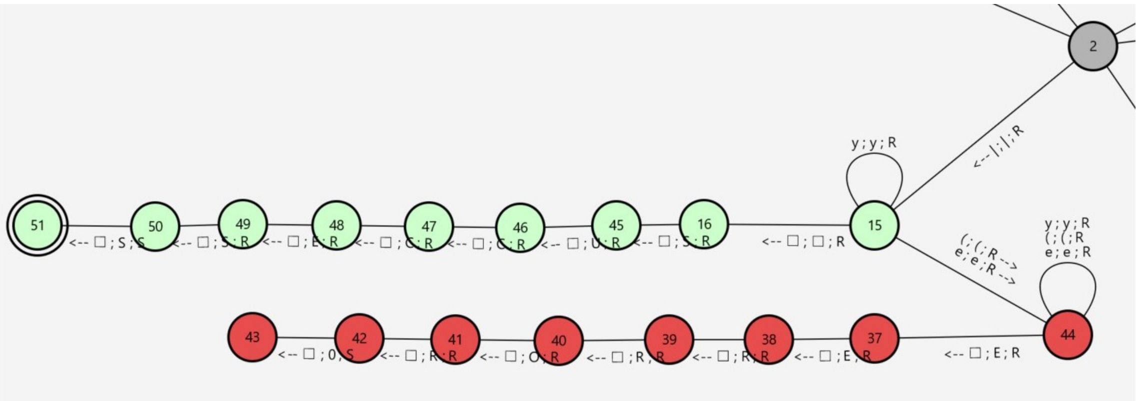

Figure 7: Green Section of Turing Machine

Error Codes

Error Code	Error Meaning
ERROR0	Unmatched Parenthesis in Middle of Input
ERROR1	Two or More Subsequent Arithmetic Expressions (+, -, $*$, /)
ERROR2	Extra Parenthesis at End of Input

Input Specification

You can enter any string into the tape where each character $c \in \{(,),+,-,*,/,0,1,2,3,4,5,6,7,8,9\}$. The machine will run from it's start state (0) to either it's accept state (51) or any of the error states $\in \{27, 36, 43\}$. The machine will output whether or not the input is a valid arithmetic expression. If it is valid, it will output "SUCCESS" to the tape. Otherwise, it will output "ERROR" and the error code, and it will replace the problematic character in the input with an 'E'. An example of a valid expression would be "(1+(2/3)*4)(56+7)" the output on this would be "xxxxxxxxxxxxxxxxx|yy SUCCESS". An invalid expression would be "1++2" and it's output would be "xxE# |ERROR1".