Super Mario the Turing Machine!

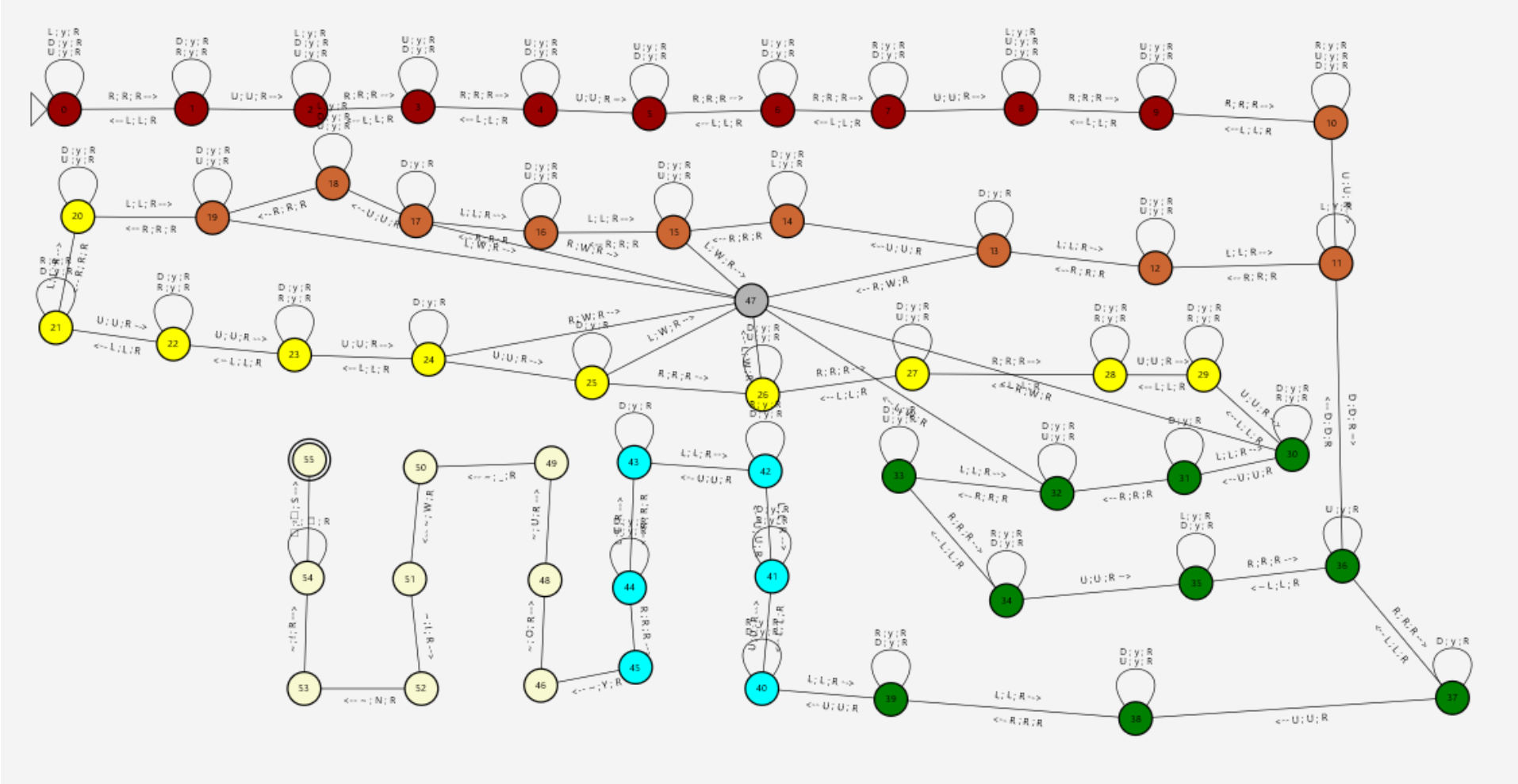
Swasti Mishra, Katie Nuchols, Riya Patel

Summary:
Our Turing machine tests whether a given string would win the first level of Super Mario Bros. Essentially, when a user plays Super Mario, they can either input right, left, up (also known as jump), or down. The amount of times and order in which they input right is very important, though they can usually input up or down without much consequence. However, if the player reaches a pipe, they *must* jump in order to continue progressing right. The same goes for holes- if a user doesn't input up in the state prior to a hole, they will fall in the hole and enter the death/reject state. Further, the player can go left, though in no case will it lead them closer to the success state. In fact, if the player goes left in states that follow a hole, they may fall down the hole and again reach the death/reject state.

Additionally, If the player inputs down in state 12 (a special pipe, signified by the "A" and down arrow on it) they will teleport to the pipe in state 37, signified by the up arrow on the pipe. If a string passes all of the conditions of our Turing machine, it means that the same input would yield success in this Super Mario level.

Swasti Mishra came up with the idea for the machine and proofread and edited the document, Katie Nuchols translated the idea to the software and found accept/reject strings, and Riya Patel wrote and edited the document.

Turing Machine Functional Design



Most moves will print the same thing on the tape that was inputted (e.g., an “R” input will print an “R” to the tape). However, an input of up (“U”) or down (“D”) where that move is not necessary will print a “y” to the tape to indicate that the move was not needed. If the player makes a move that results in Mario dying, for example, moving right when they should have jumped over a hole, a “W” will print to the tape as evidence that the player lost the game. And when the player runs successful input through the machine, the message “YOU_WIN!” will print at the end of the tape. An annotated version of this machine is in the appendix, next to pictures of what each state represents.

Input Specification

Users can input “R”, “U”, “D”, and “L”. Input “R” represents Mario walking right, “U” represents Mario jumping, “D” represents Mario crouching, and “L” represents Mario walking left. A lowercase “r” represents moving the tape to the right. Inputting an “R”, “U”, or “D” from the start state would move the tape to the right, while an input of “L” would loop the tape in the current state.

We chose these input specifications based on the hardware that Super Mario Brothers was originally built for- the controller really only had four buttons, so we think it’s fair to exclude all other input.

Correct Execution and Sample Outputs

The following are some examples of strings that would result in a success state (reaching state 55), or halting. If the player were to lose the game, they would either end in a reject state (state 47) or our Turing machine would continue to wait for further input. The start state is state zero.

Acceptable Strings

Input: RURRURRURRURRURRRURRRUUUURRRUUURRRURRURUUUURR
Output: RURRURRURRURRURRRURRRUUUURRRUUURRRURRURUUUURRYOU_WIN!

Input: UDUDUDRURLRRURRURRURRURRRURRRUUUURRRUUURRRURRURUUUURLRR
Output: yyyyyyRURLRRURRURRURRURRRURRRUUUURRRUUURRRURRURUyUUURLRRYOU_WIN!

Input: UDUDUDRLRURRUDRLRRURRUDRLRURUUUURR
Output: yyyyyyRLRURRUyRLRRURRUDRLRURUUUURRYOU_WIN!

Rejected Strings

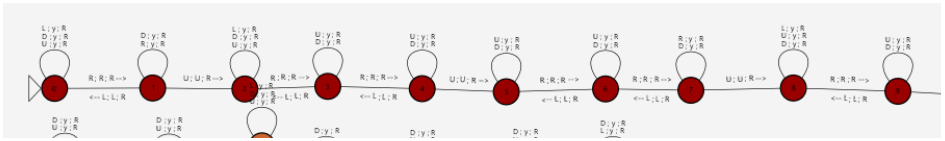
Input: RURRURUURURLLRLRRRRLRURUUUURDDR
Output: RURRURyyRURLyRLRRyyLRURyyyyRyyW

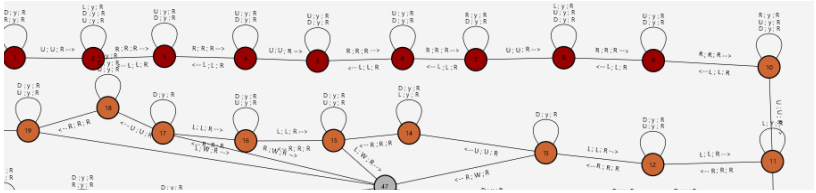
Input: RLRURLRLRURURRURRURRURRRUUUURL
Output: RLRURLRLRyRURRURRURRURRRUyyyRW

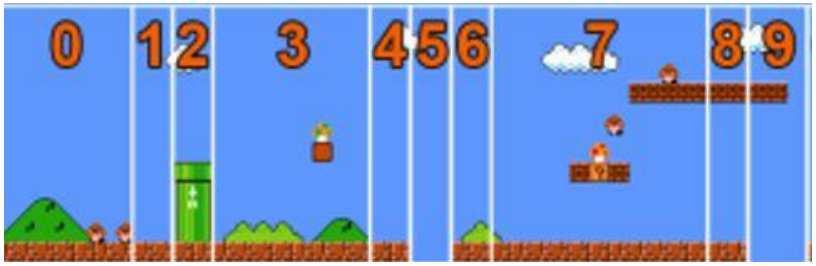
Input: RDRURRURRURRURRURRRURRRUUUURRRUUURRRURD
Output: RyyURRURRURRURRURRRURRRUUUURRRUUURRRURD

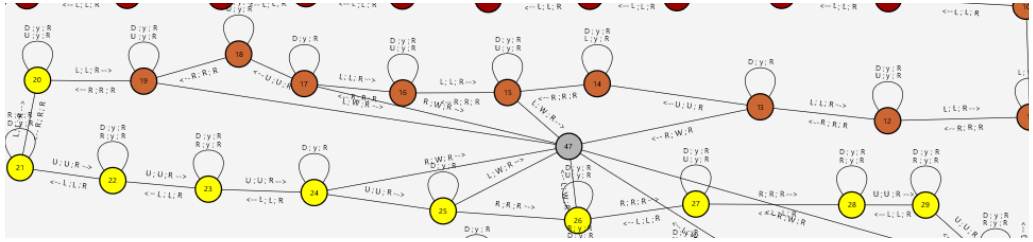
Appendix

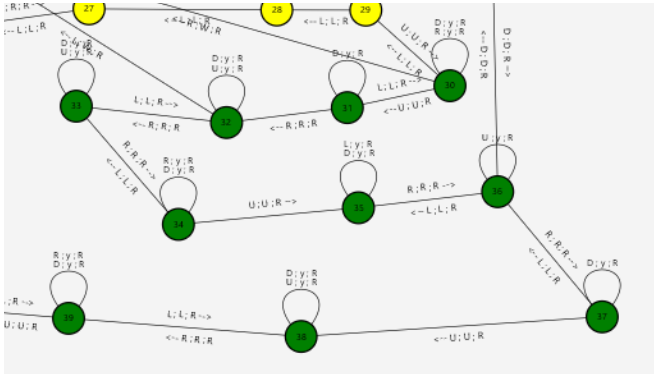
We've also included pictures of the 1-1 Level next to our machine, because there are a lot of states and our machine can be confusing to look at.



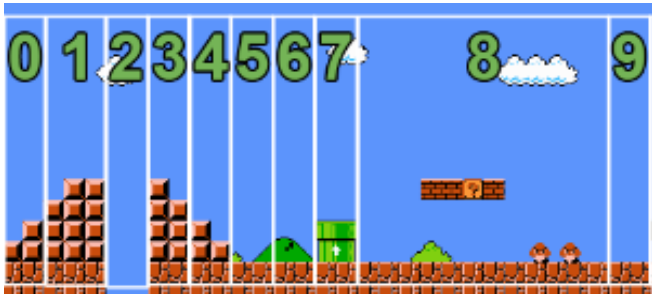


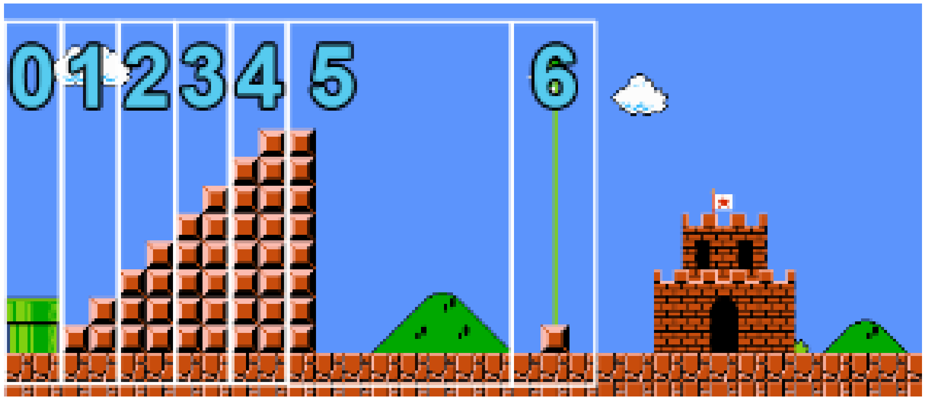







0
1
2
3
4
5
6
7
8
9


0
1
2
3
4
5
6
7
8
9


0 1 2 3 4 5
6