HW3

BURAK BAHAR

2380137

1

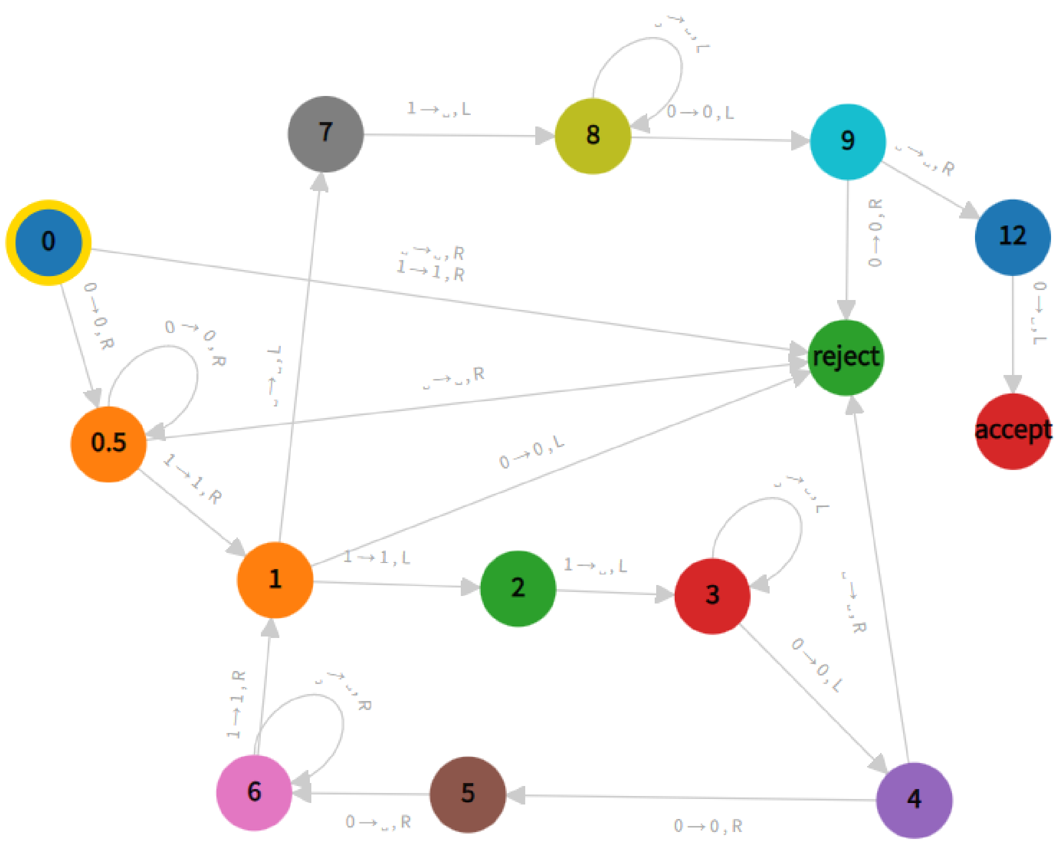

Figure 1: O^N1^N

0 : **If word starts with 1 reject.**
If there are 0**'s go to state** 0.5 **with the first** 0**.**

0.5 : **If word only has** 0**'s exhaust them and reject.**
Else go to right until reaching the first 1 **and go to state** 1**.**

1 : **If the** 1 **that we reached is the last one go to state** 7 **to check whether the remaining** 0 **is the last one or not.**
If machine reads 0**, this means there is a** 0 **after** 1**. So reject.**
If this 1 **is not the last one go to state 2.**

2 : **Write blank in place of 1 and go to state** 3**.**

3 : **Go to left until finding a** 0**. When** 0 **is found go to state** 4**.**

4 : **Check the left of** 0**. If it is empty(this is the last** 0 **and there is at least a** 1**) reject since in state** 1 **we checked if there are more than one** 1**. If there are remaining** 0**'s in the left side go to state** 5**.**

5 : **Write blank in place of the** 0**.**

6 : **Go to right until finding a 1 since the last** 1 **that we made blank was not the last one.**

7 : **This is a state we reach when we read the last** 1**. So we make the last** 1 **blank while going to state 8.**

8 : **Going left until finding a** 0**. We know there is at least a** 0 **since we check this condition at state** 4**.After finding the** 0 **we go to state** 9**.**

9 : **This state checks whether the** 0 **we reached is the last one or not. If it is the last one go to state** 12**.**

12 : **What this state does is to write blank in place of the last** 0 **and accepts.**

Reject : **Rejected words end up in here.**

Accept : **Words is processed fully and in the language.**

1.1 Examples

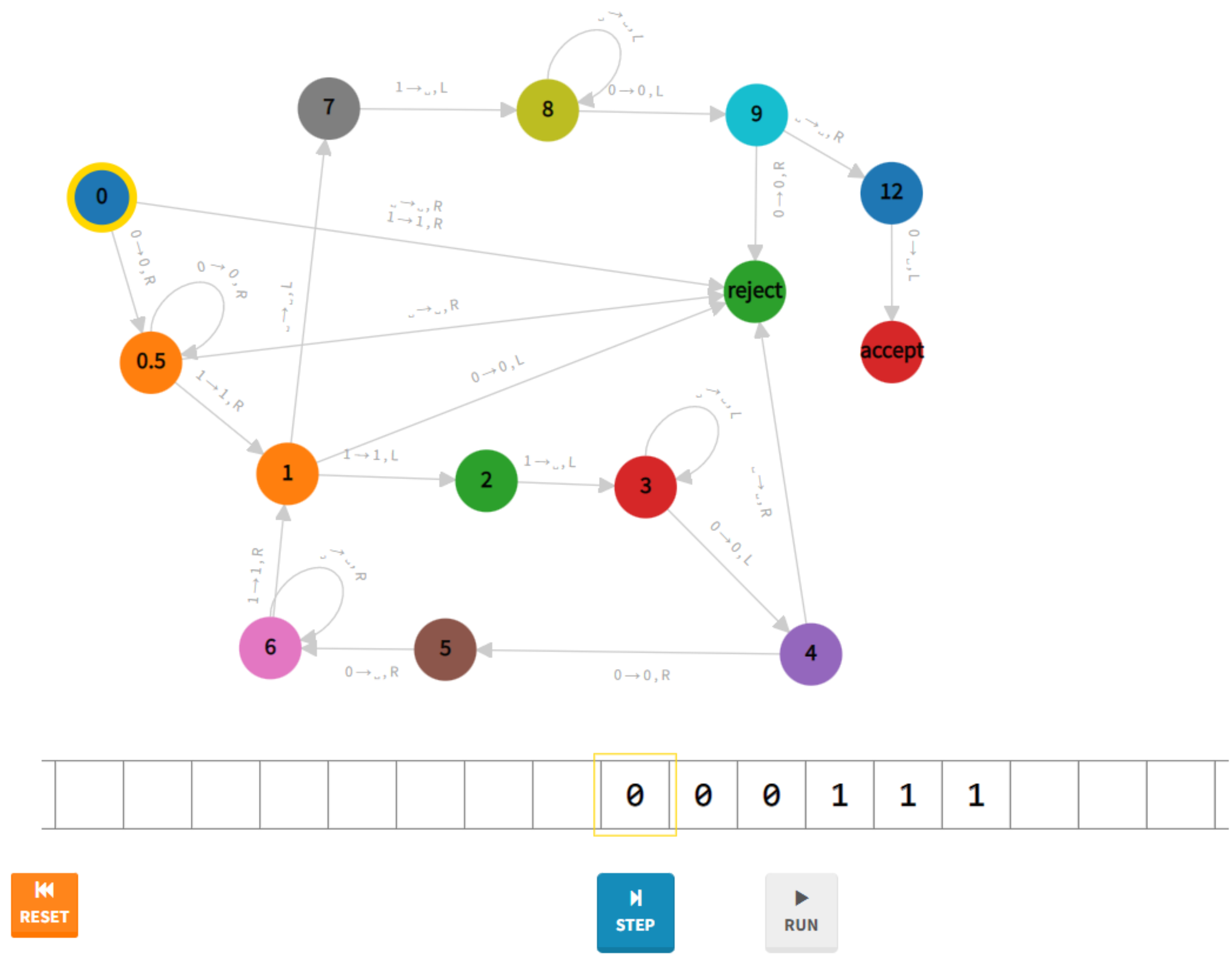

Figure 2: 000111 Start

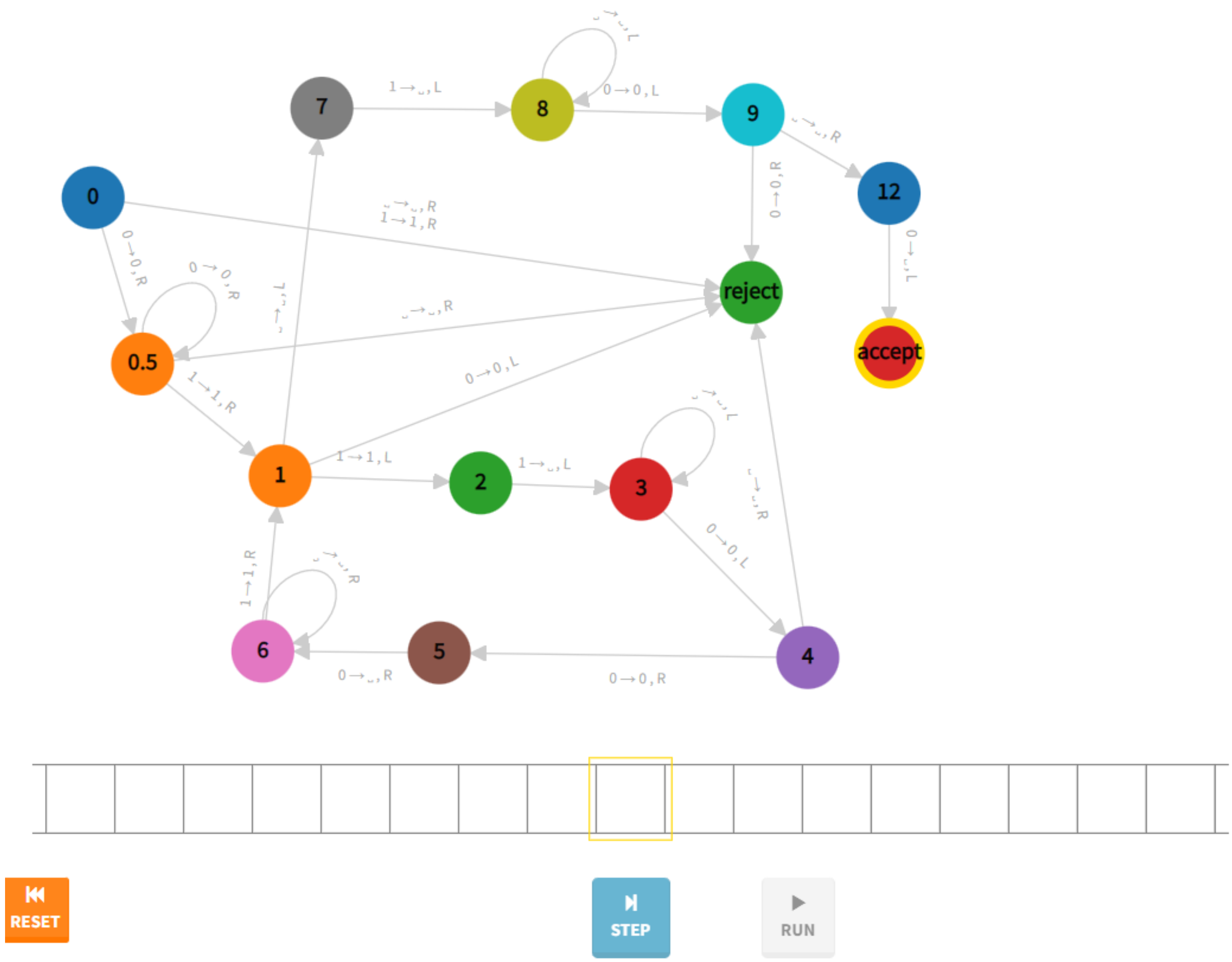

Figure 3: 000111 End

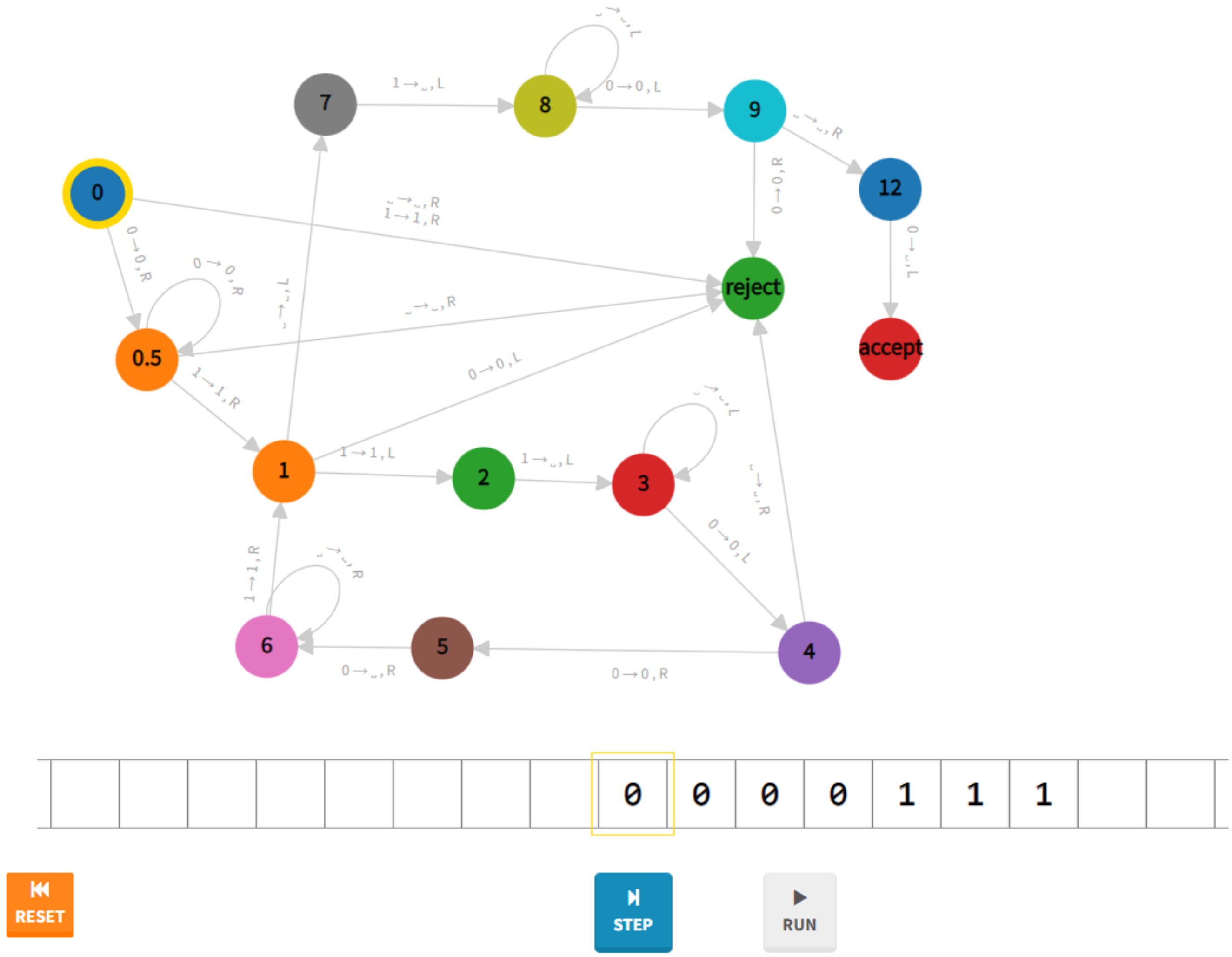

Figure 4: 0000111 Start

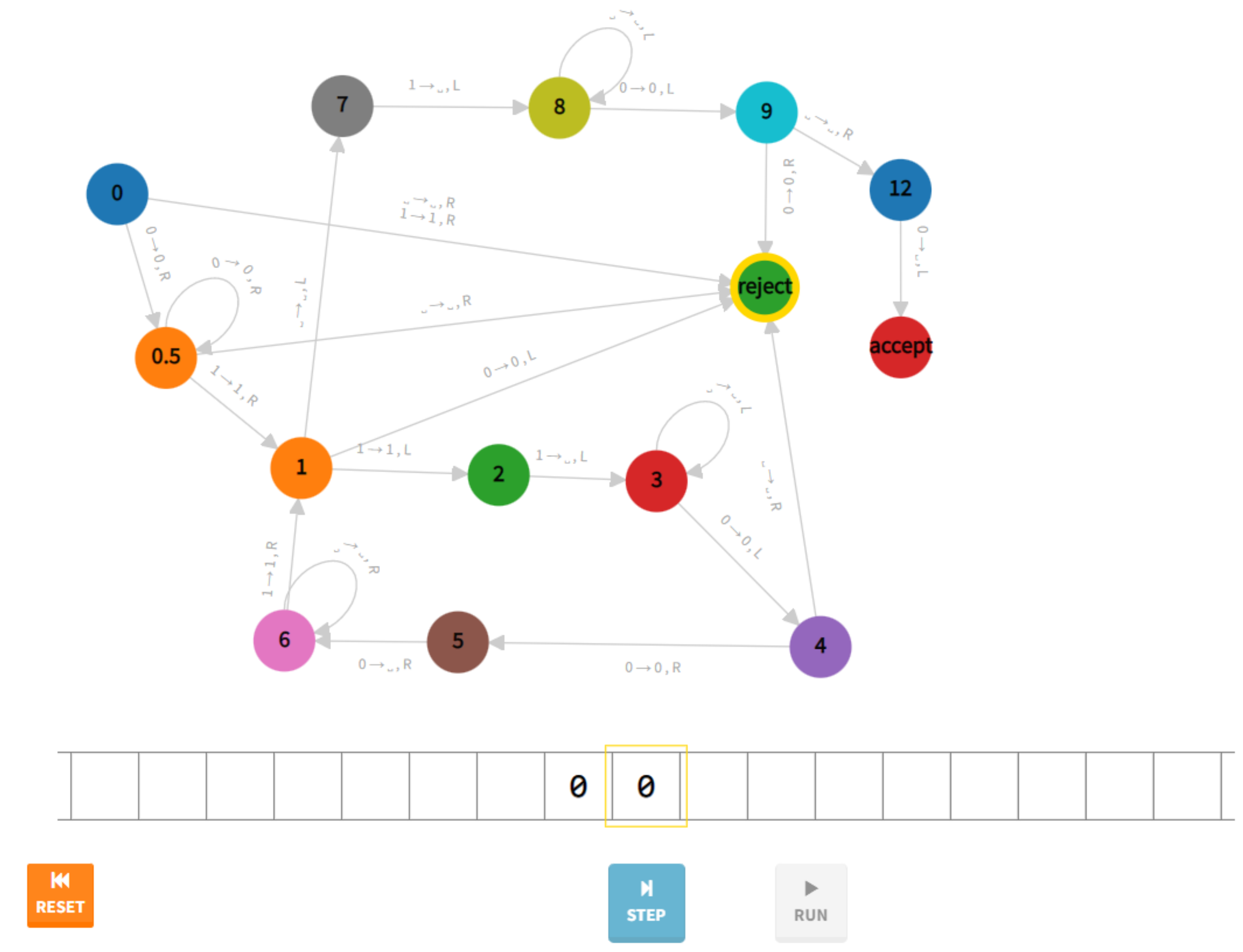

Figure 5: 0000111 End

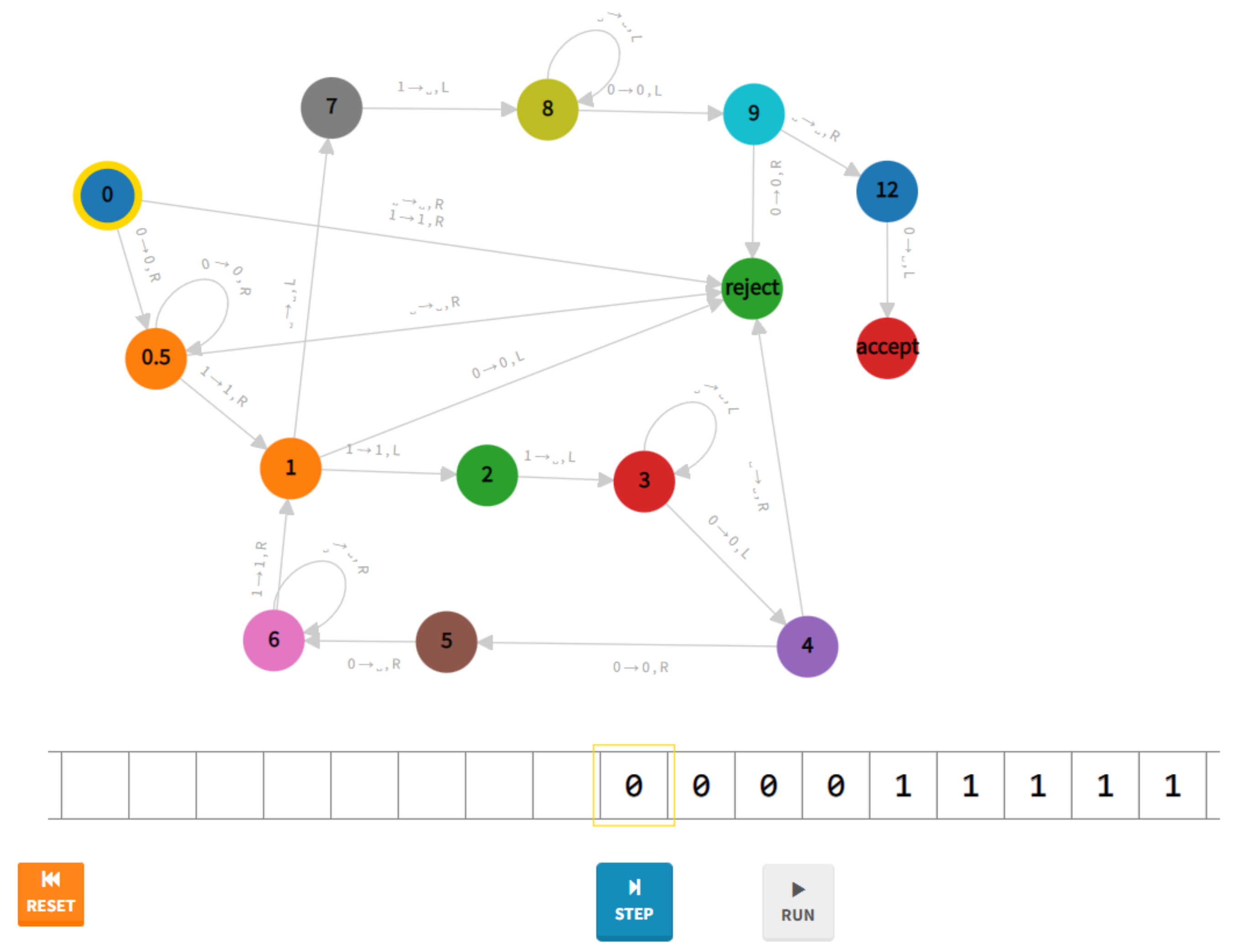

Figure 6: 000011111 Start

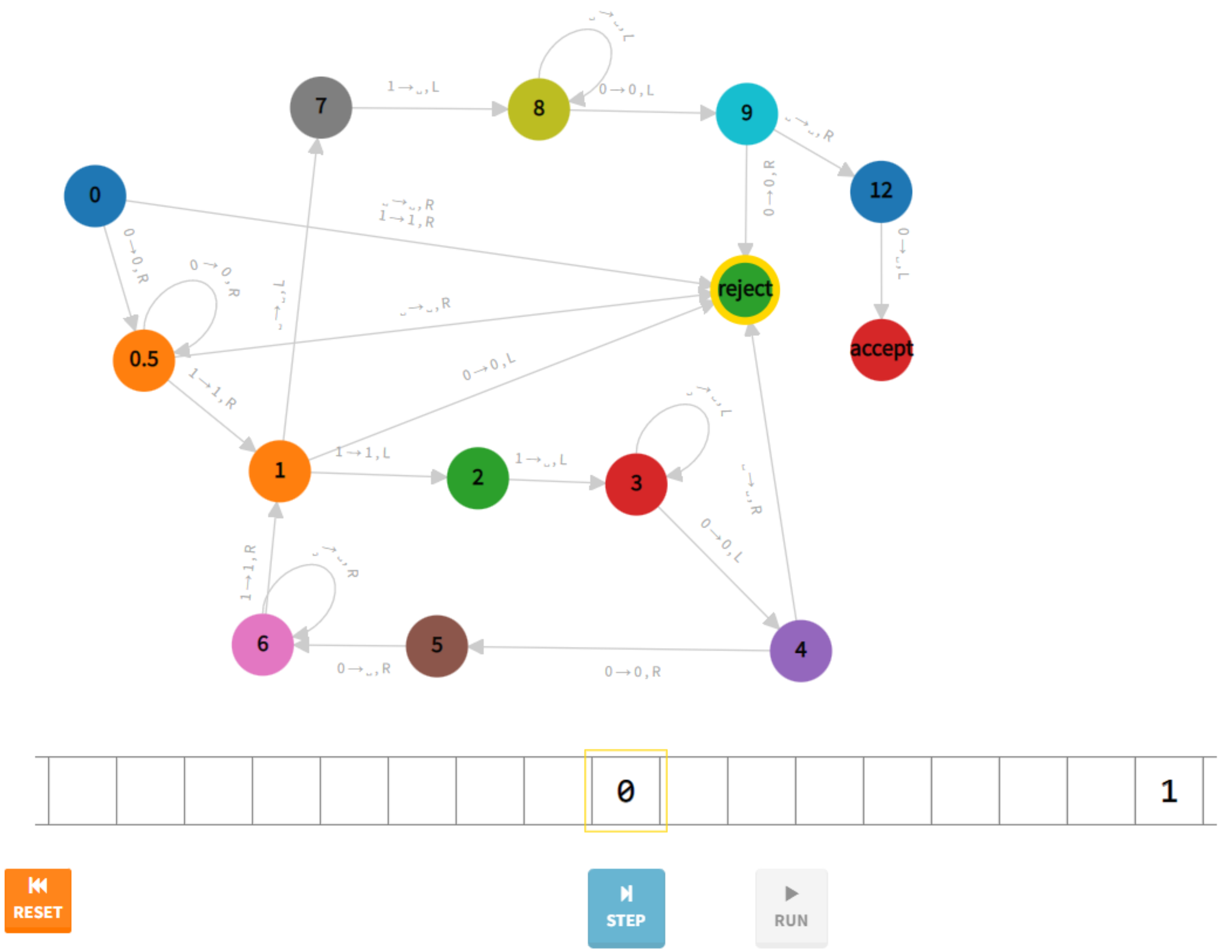

Figure 7: 000011111 End

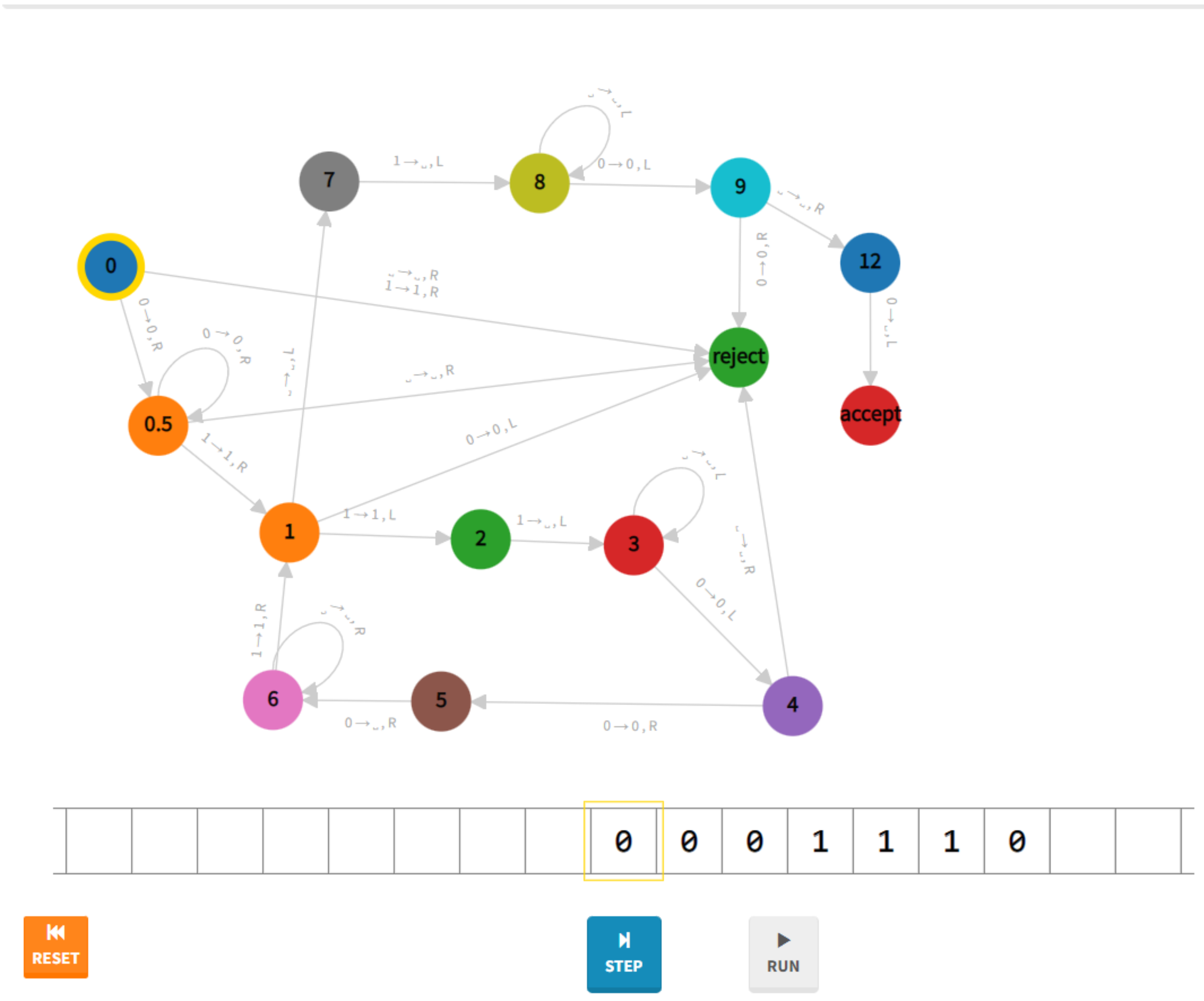

Figure 8: 0001110 Start

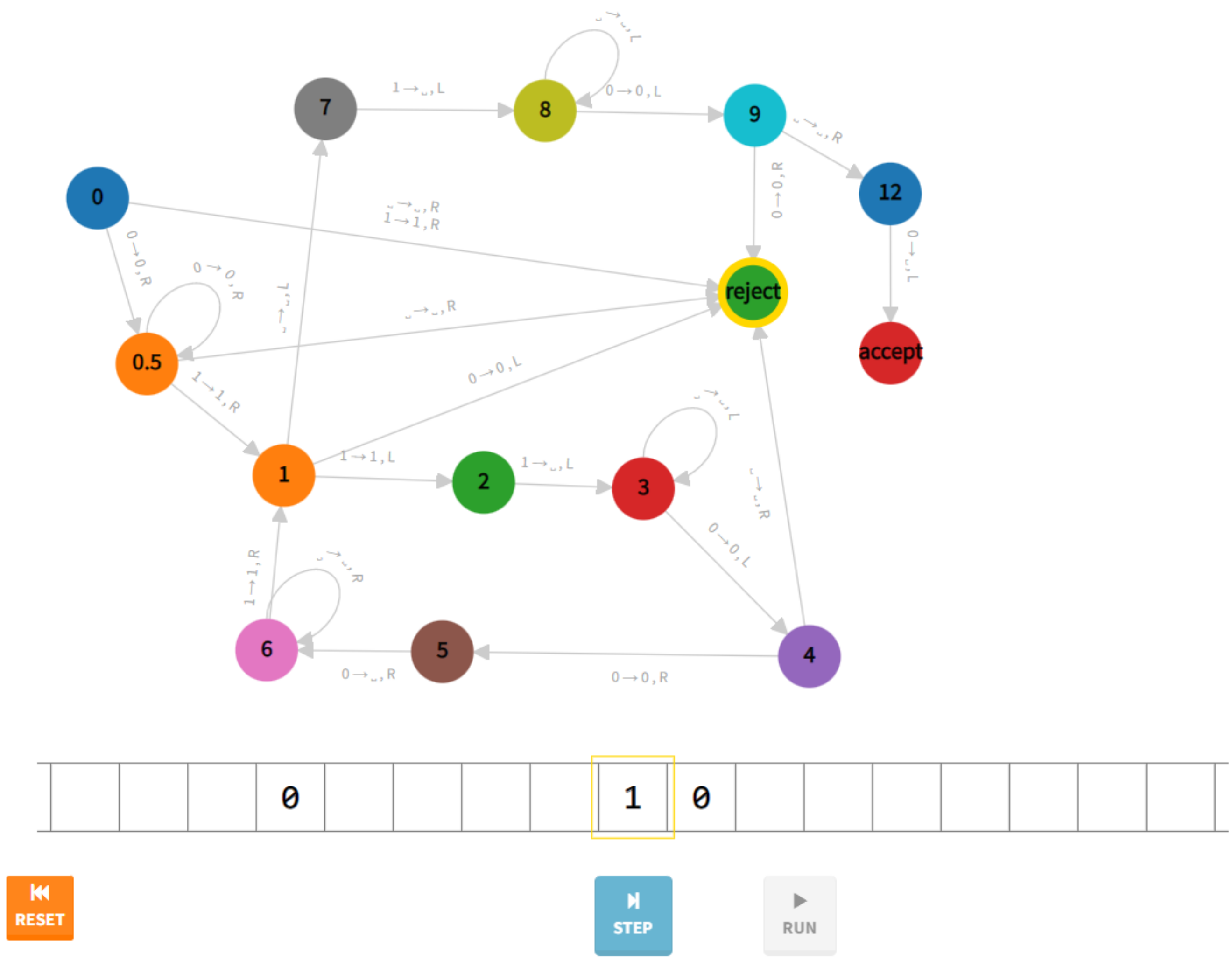

Figure 9: 0001110 End

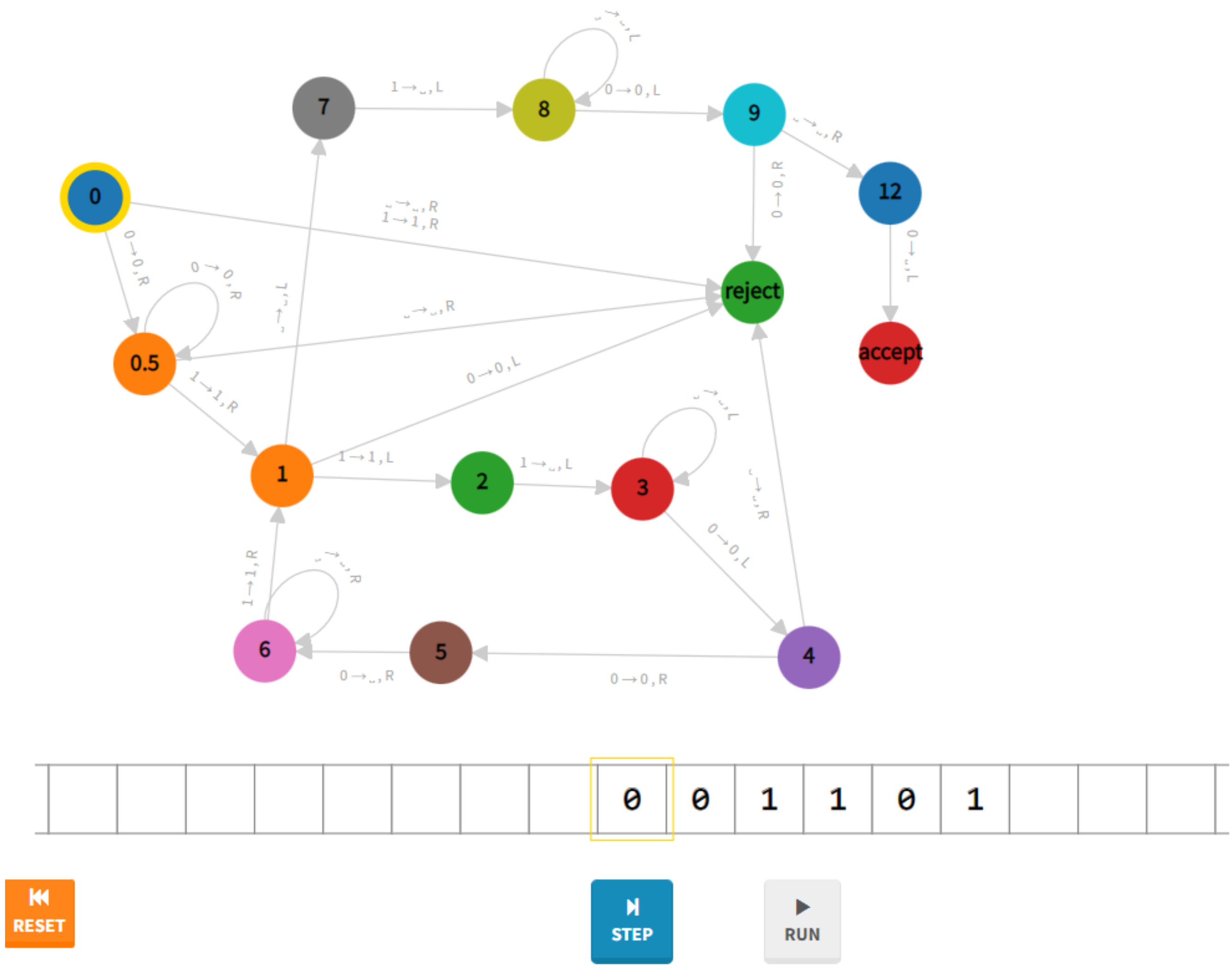

Figure 10: 001101 Start

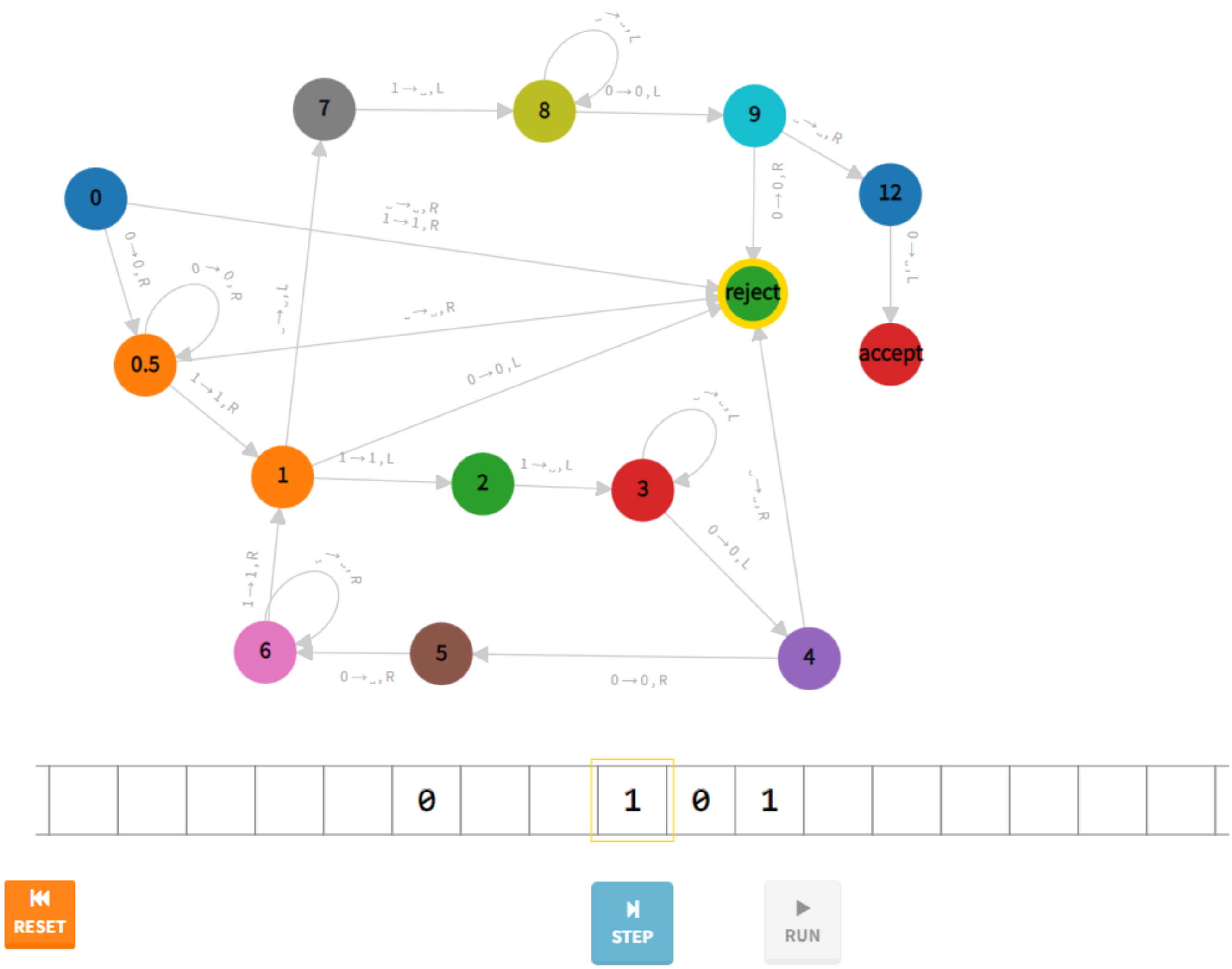

Figure 11: 001101 End

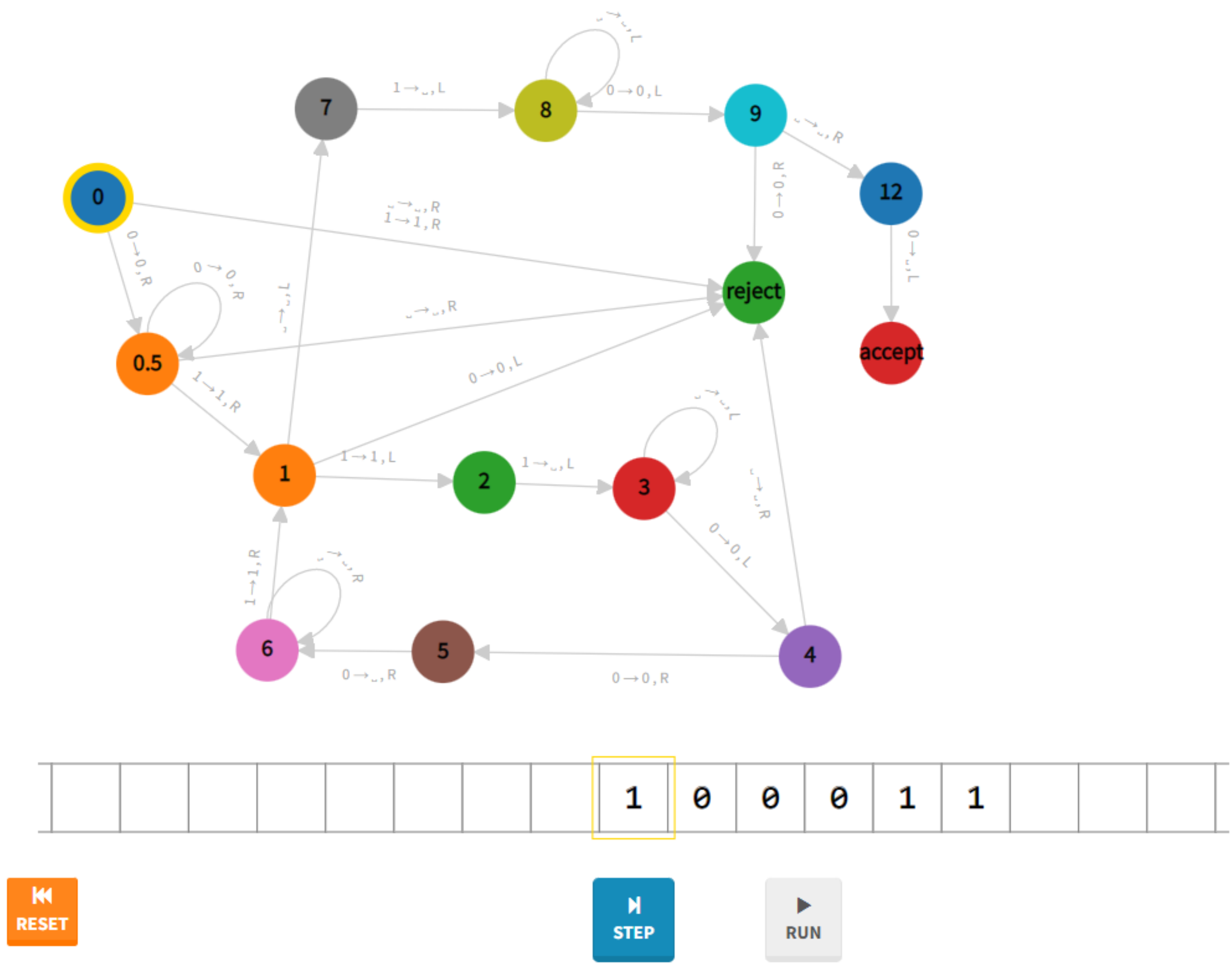

Figure 12: 100011 Start

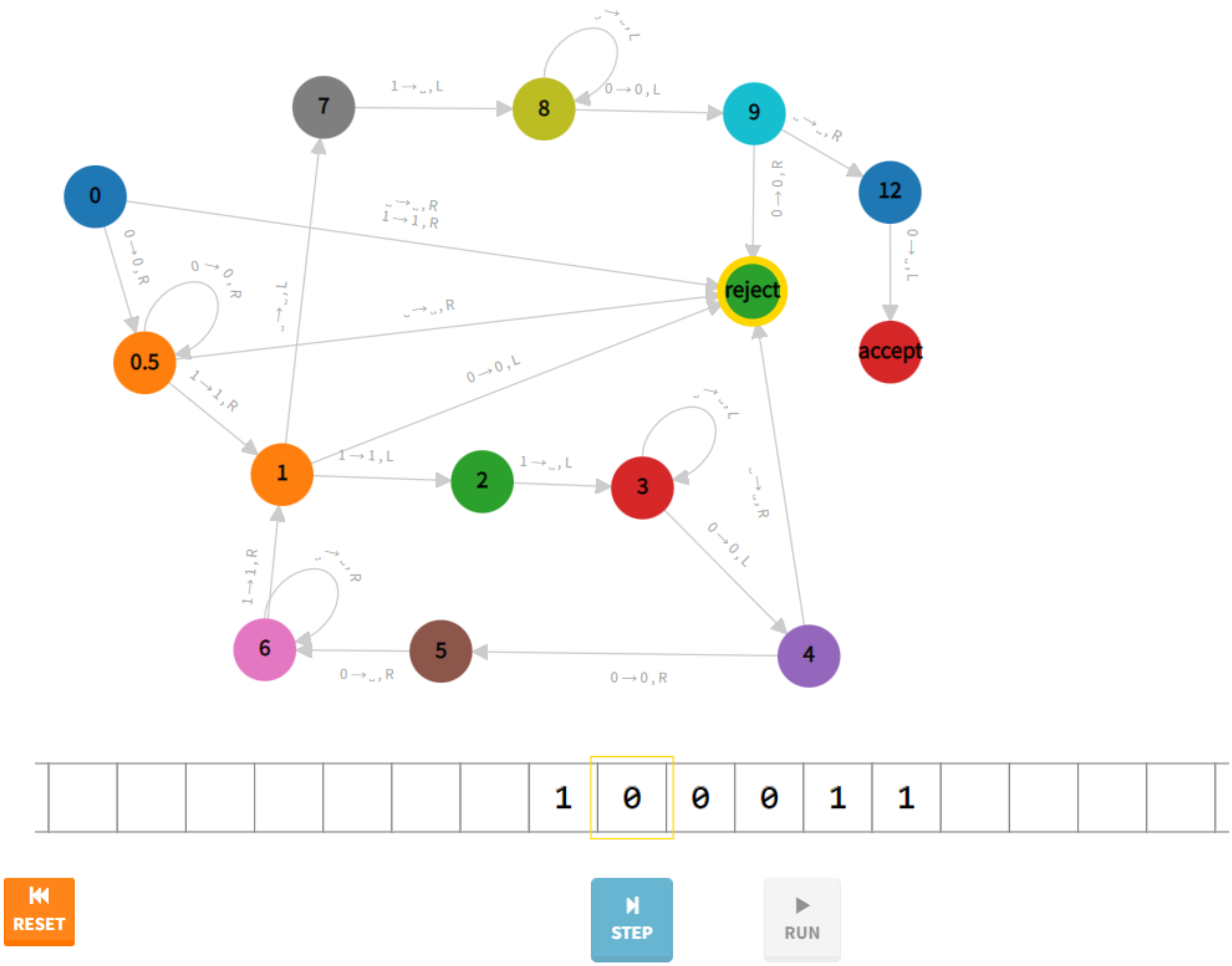

Figure 13: 100011 End

2

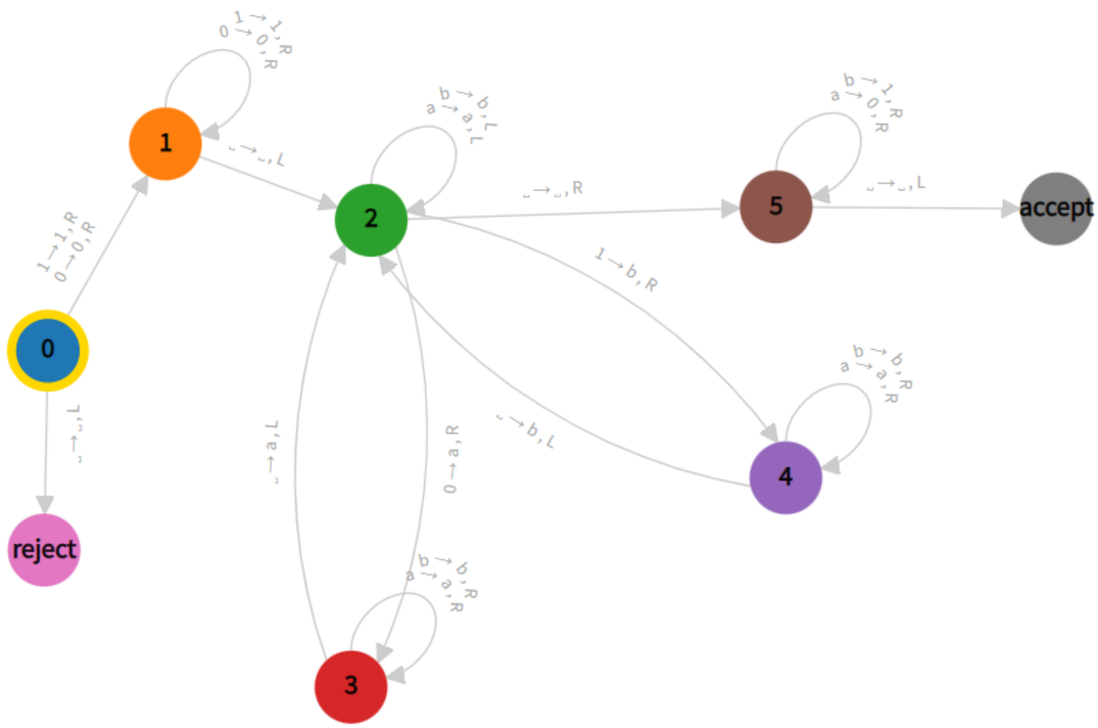

Figure 14: $f(w) = ww^R$

0 : **Check word length. If word is empty reject. If there is at least one character go to state** 1**.**

1 : **Exhaust the word until reaching the last character. Then go to state** 2**.**

2 : **When there are a's and b's go to left until finding anything else.**
if 0 **is found write a at its place and go to state** 3**.**
if 1 **is found write b at its place and go to state** 4**.**
if blank is found it means we processed the whole string and the only thing that remains is to convert a**'s and** b**'s to** 0**'s and** 1**'s. So we go to state** 5**.**

3 : **Go to right until finding a blank space and write a in there while going to state** 2

4 : **Go to right until finding a blank space and write b in there while going to state** 2

5 : **We convert** a**'s and** b**'s to** 0**'s and** 1**'s and produce our new word.**

Accept: Meaning whole process is done.

2.1 Examples

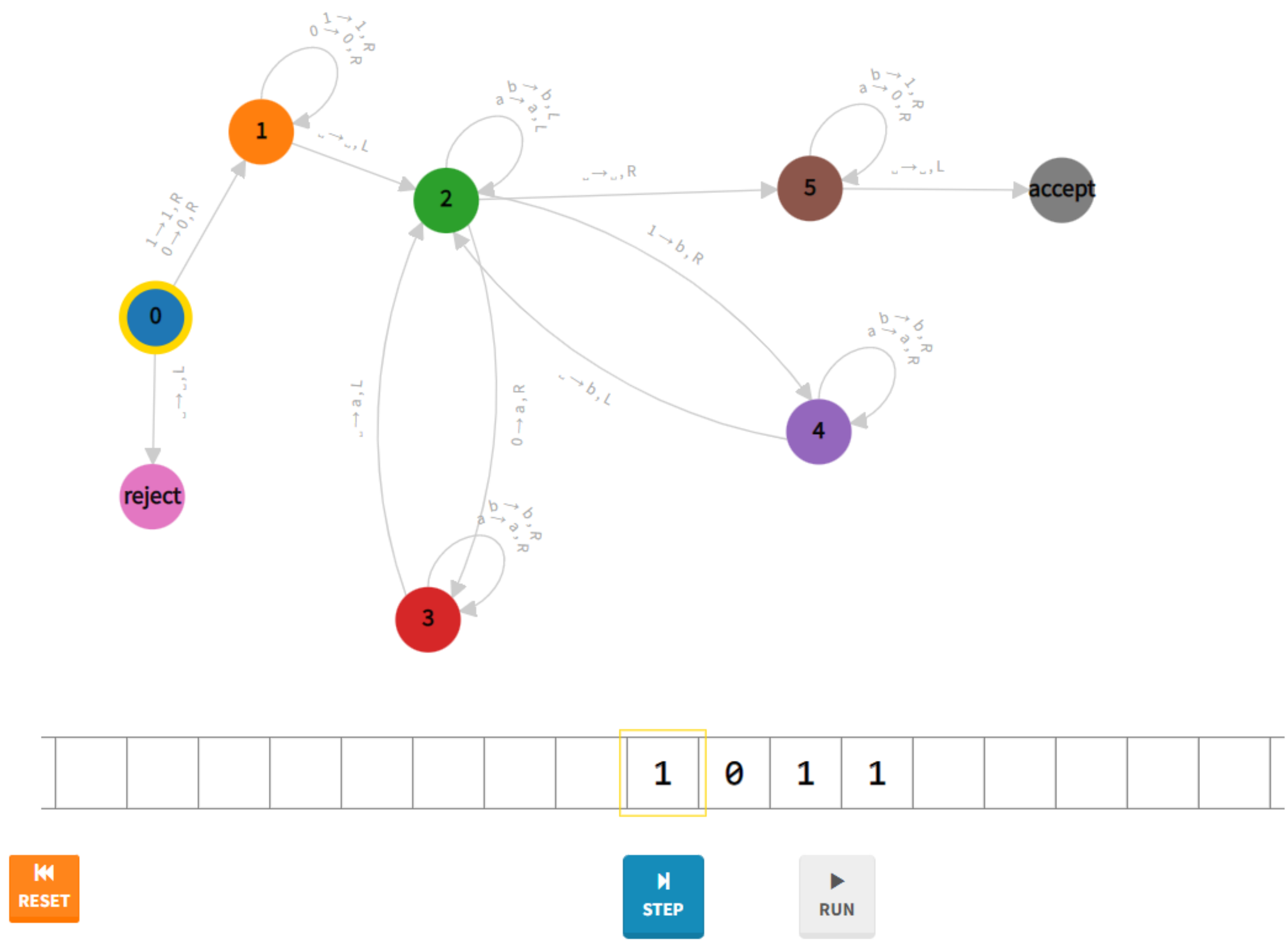

Figure 15: 1011 Start

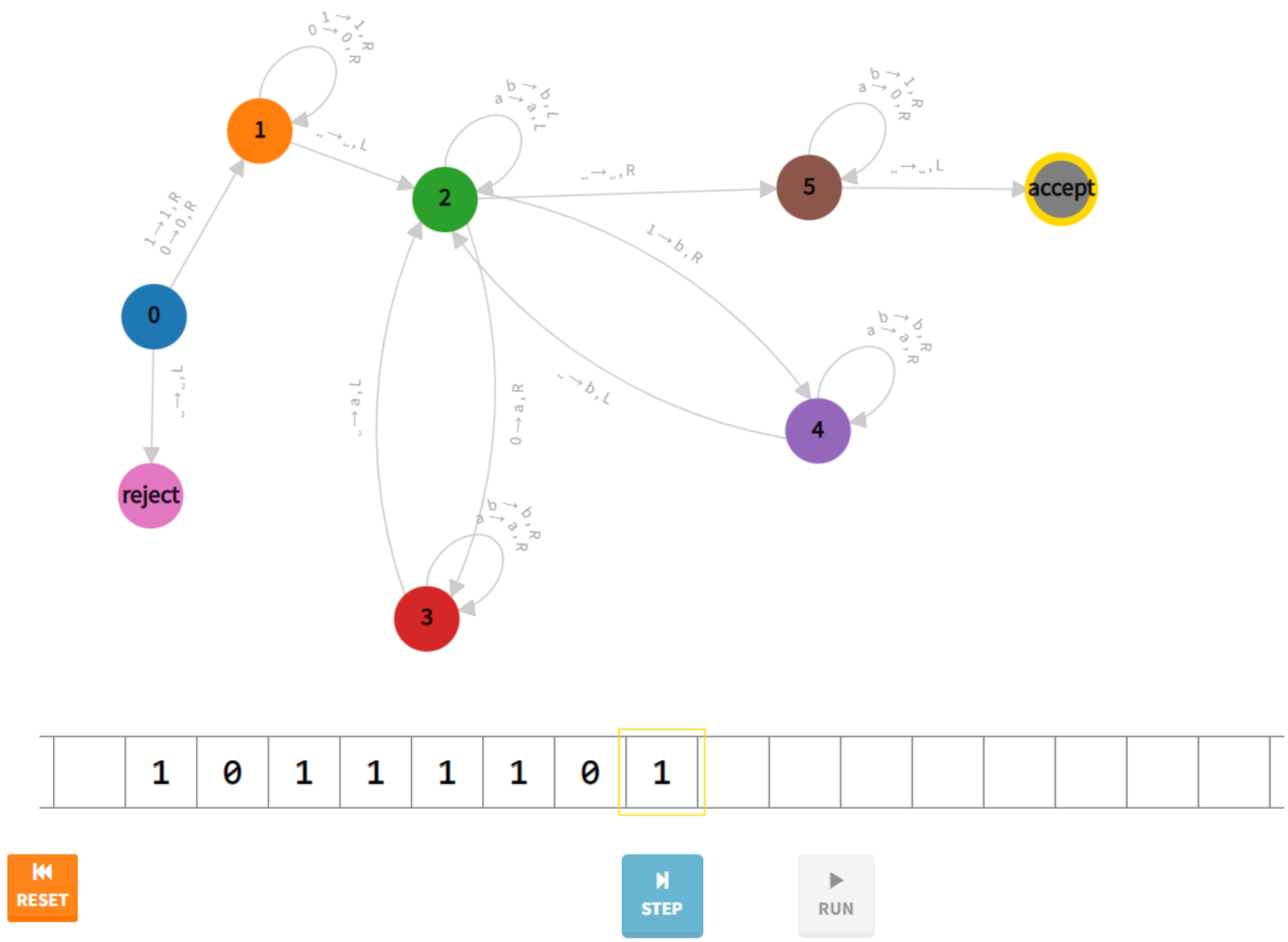

Figure 16: 1011 End

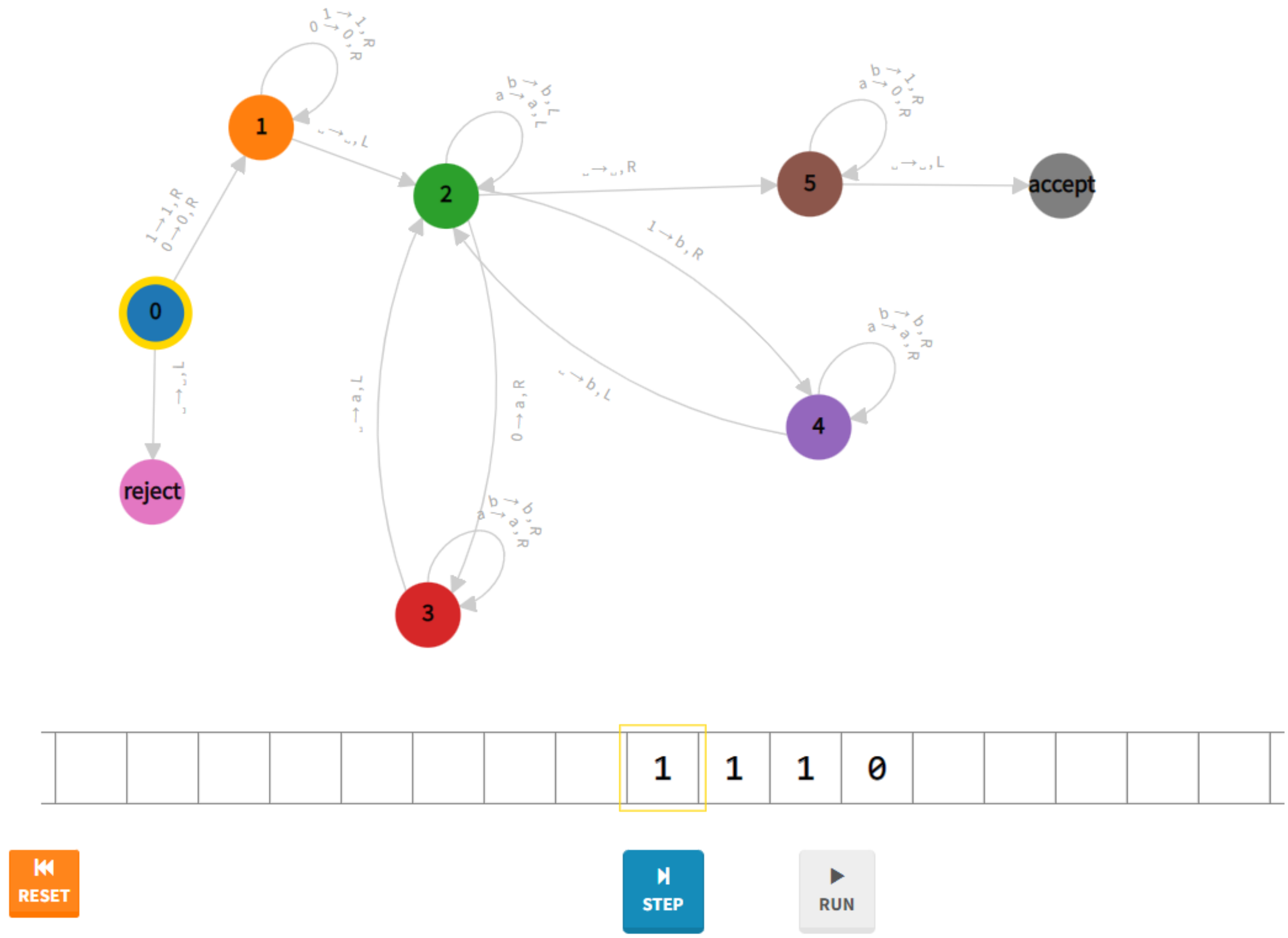

Figure 17: 1110 Start

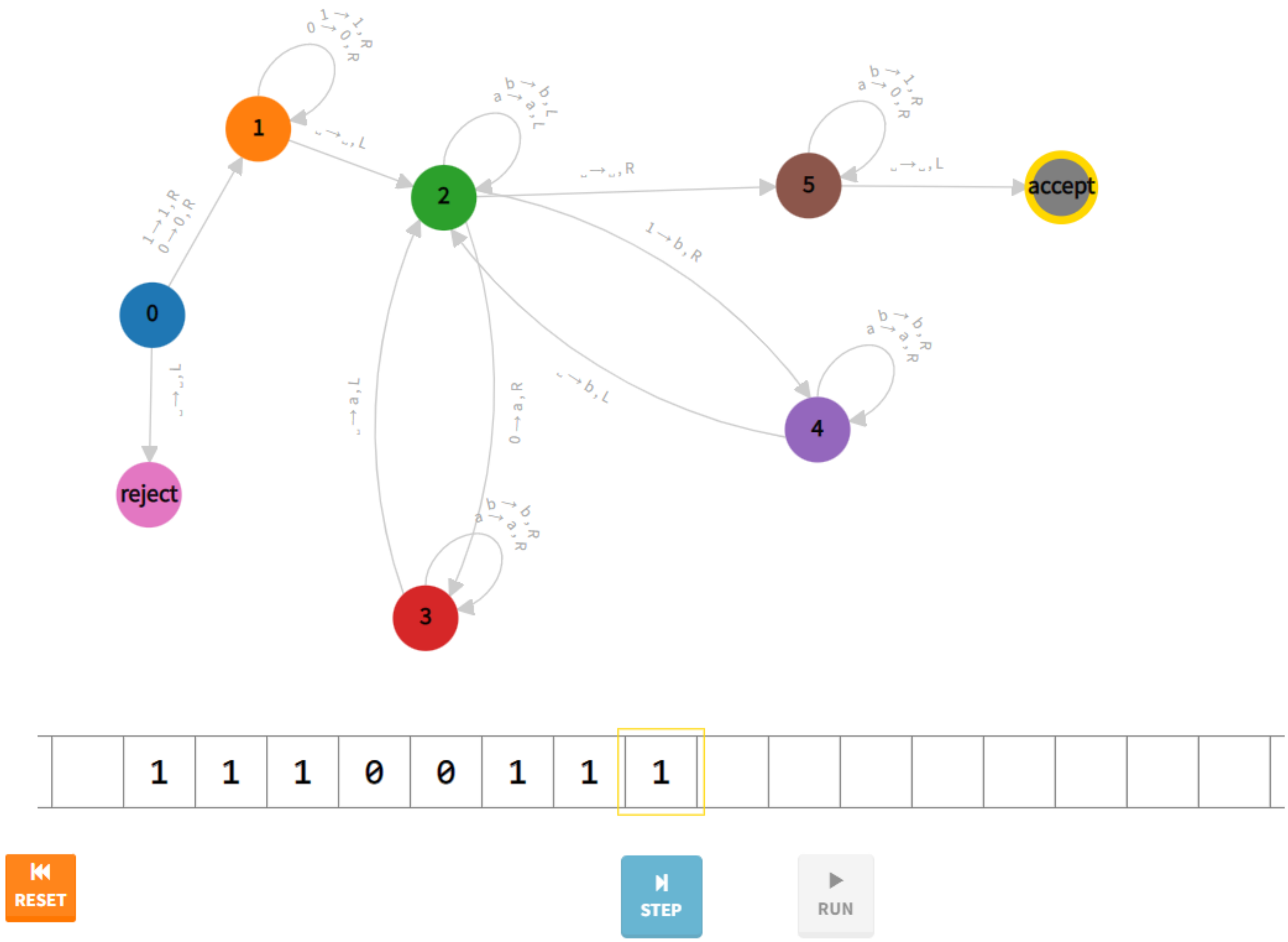

Figure 18: 1110 End

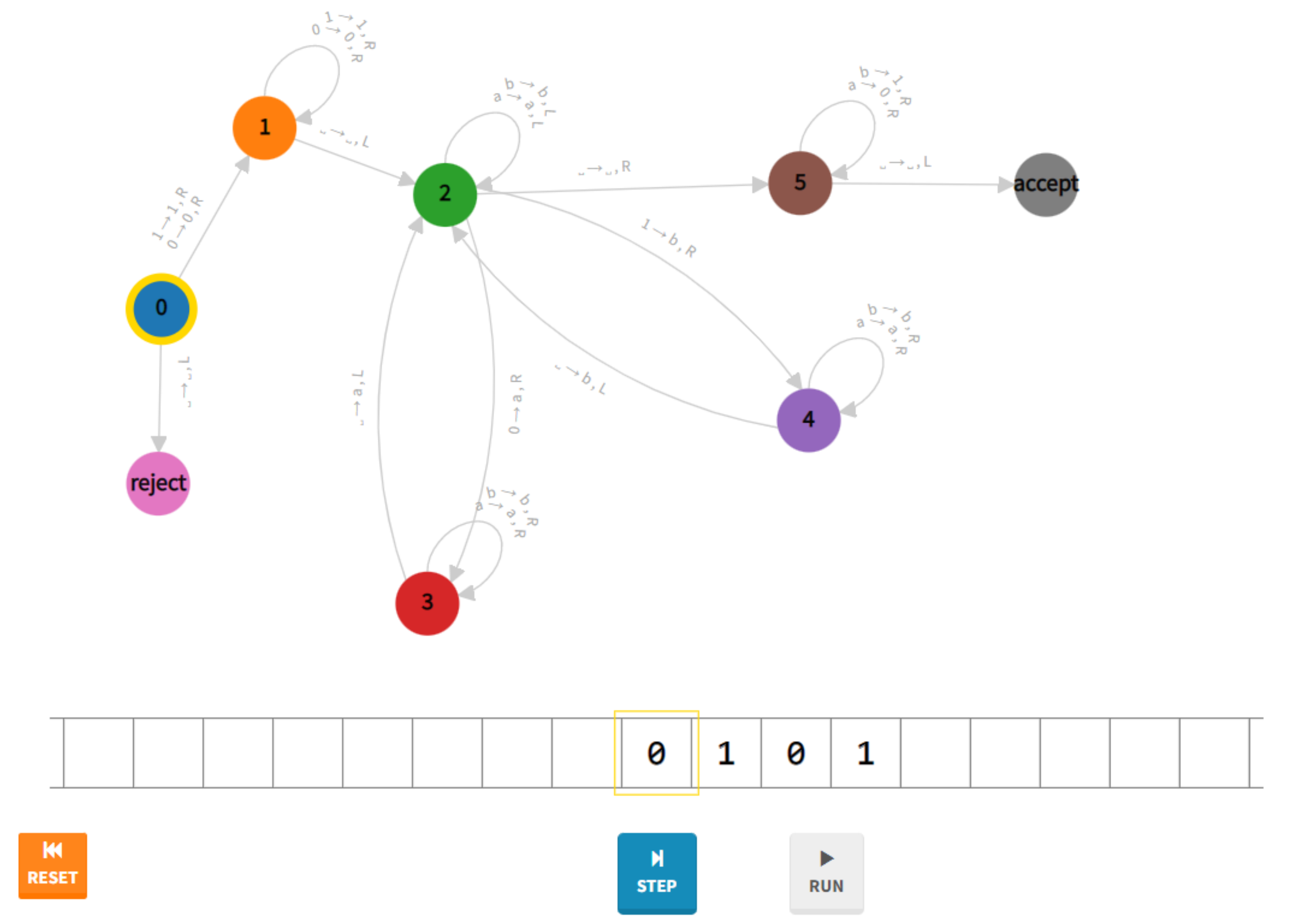

Figure 19: 0101 Start

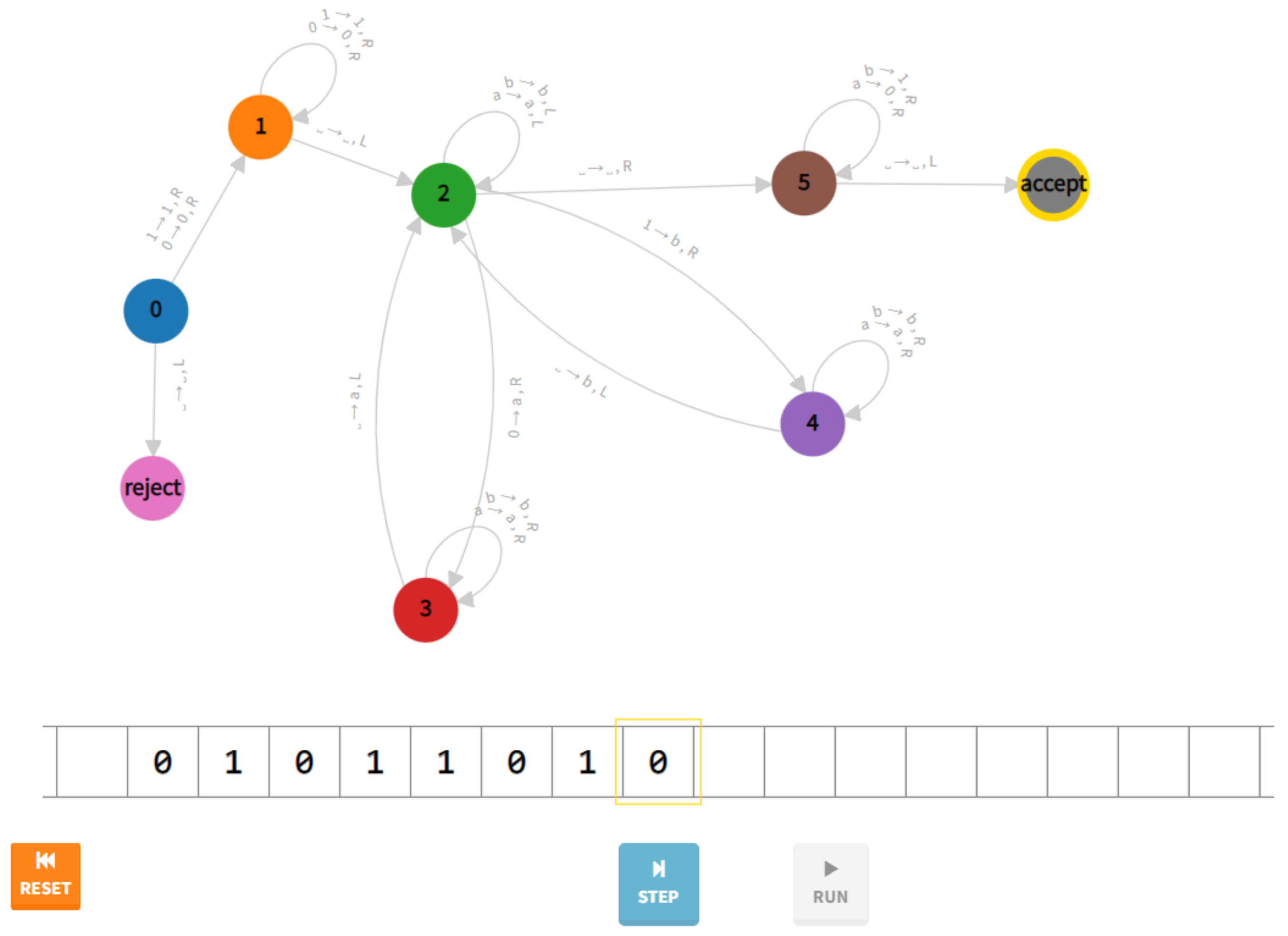

Figure 20: 0101 End

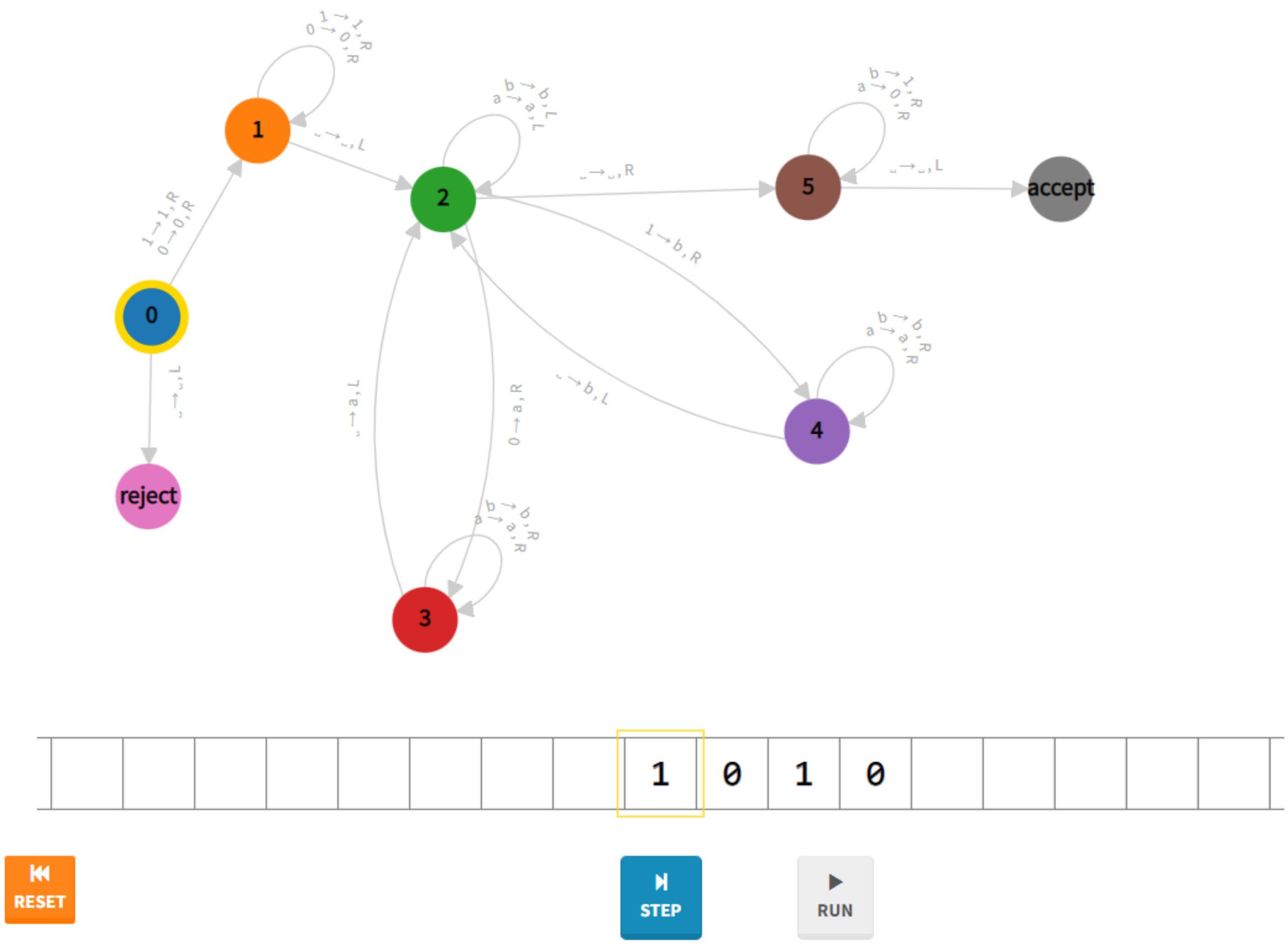

Figure 21: 1010 Start

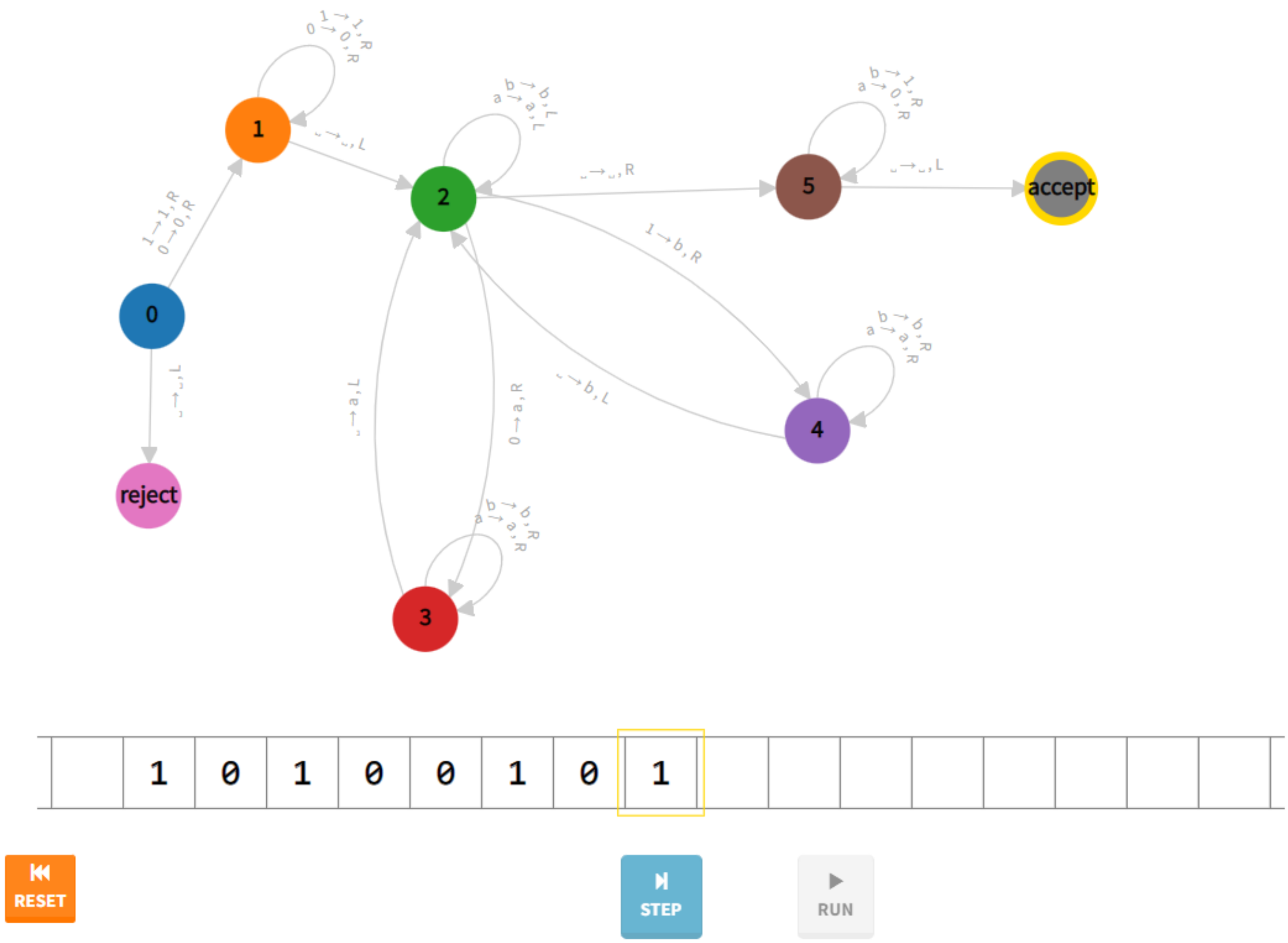

Figure 22: 1010 End

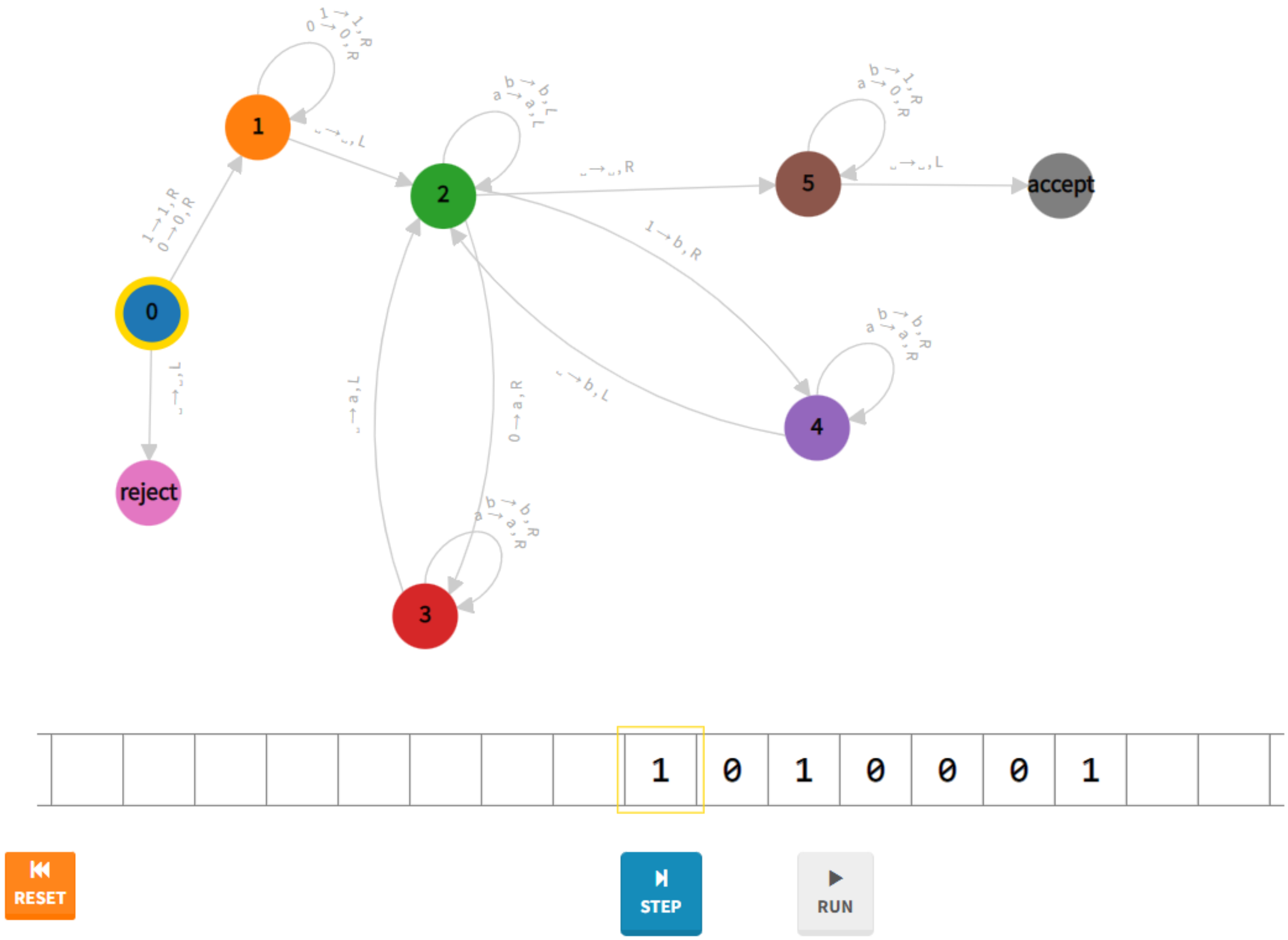

Figure 23: 1010001 Start

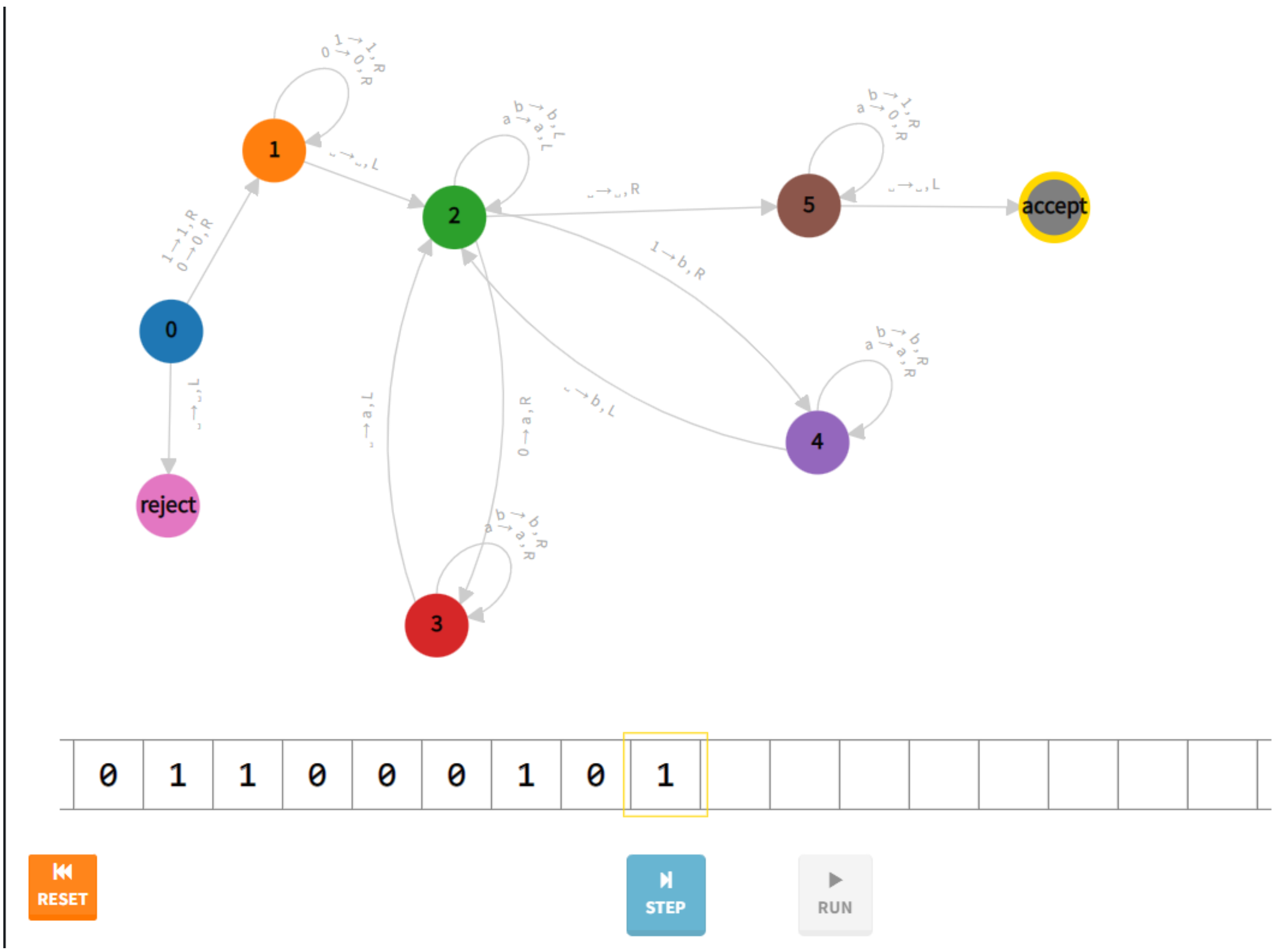

Figure 24: 1010001 End

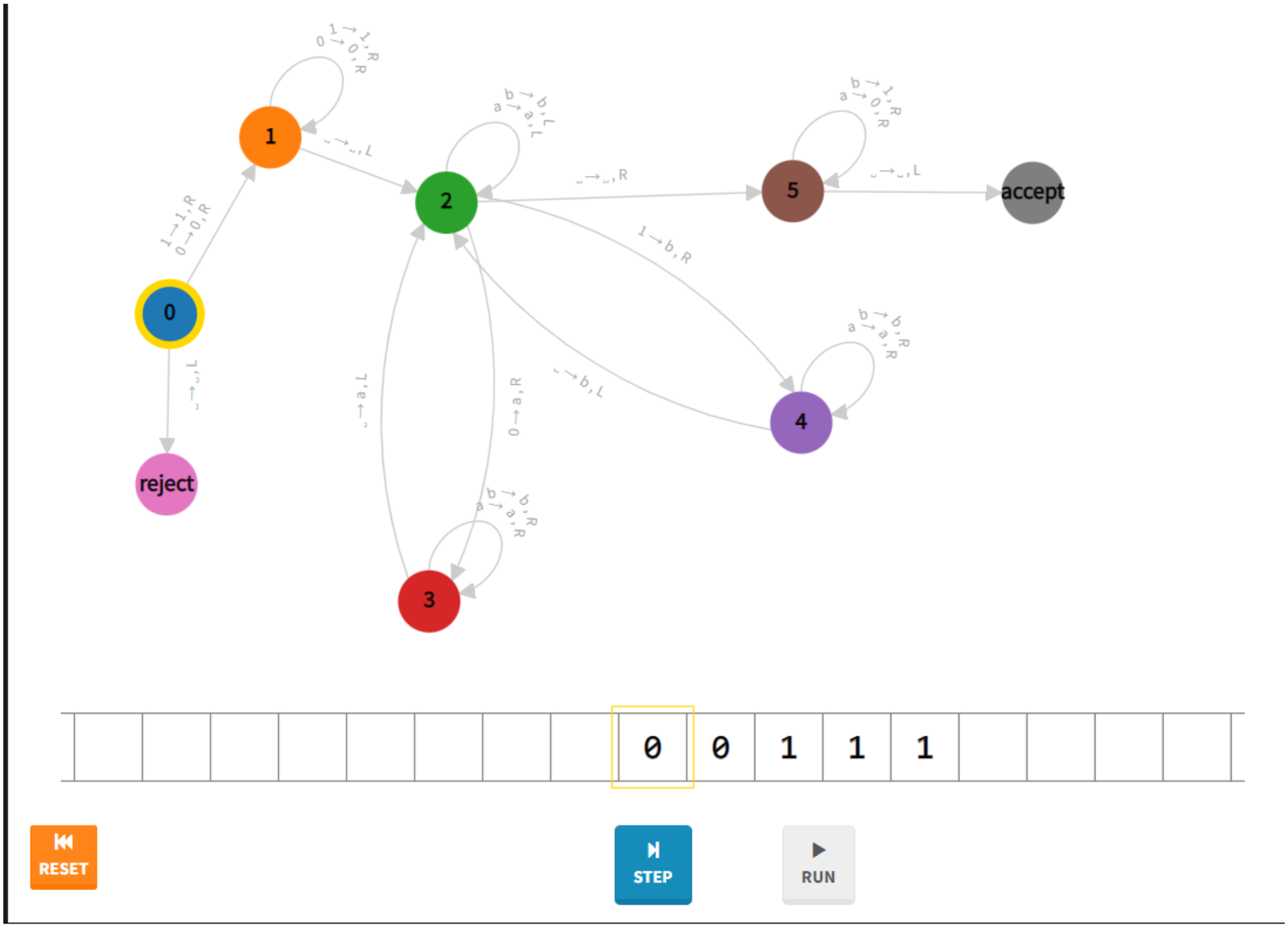

Figure 25: 00111 Start

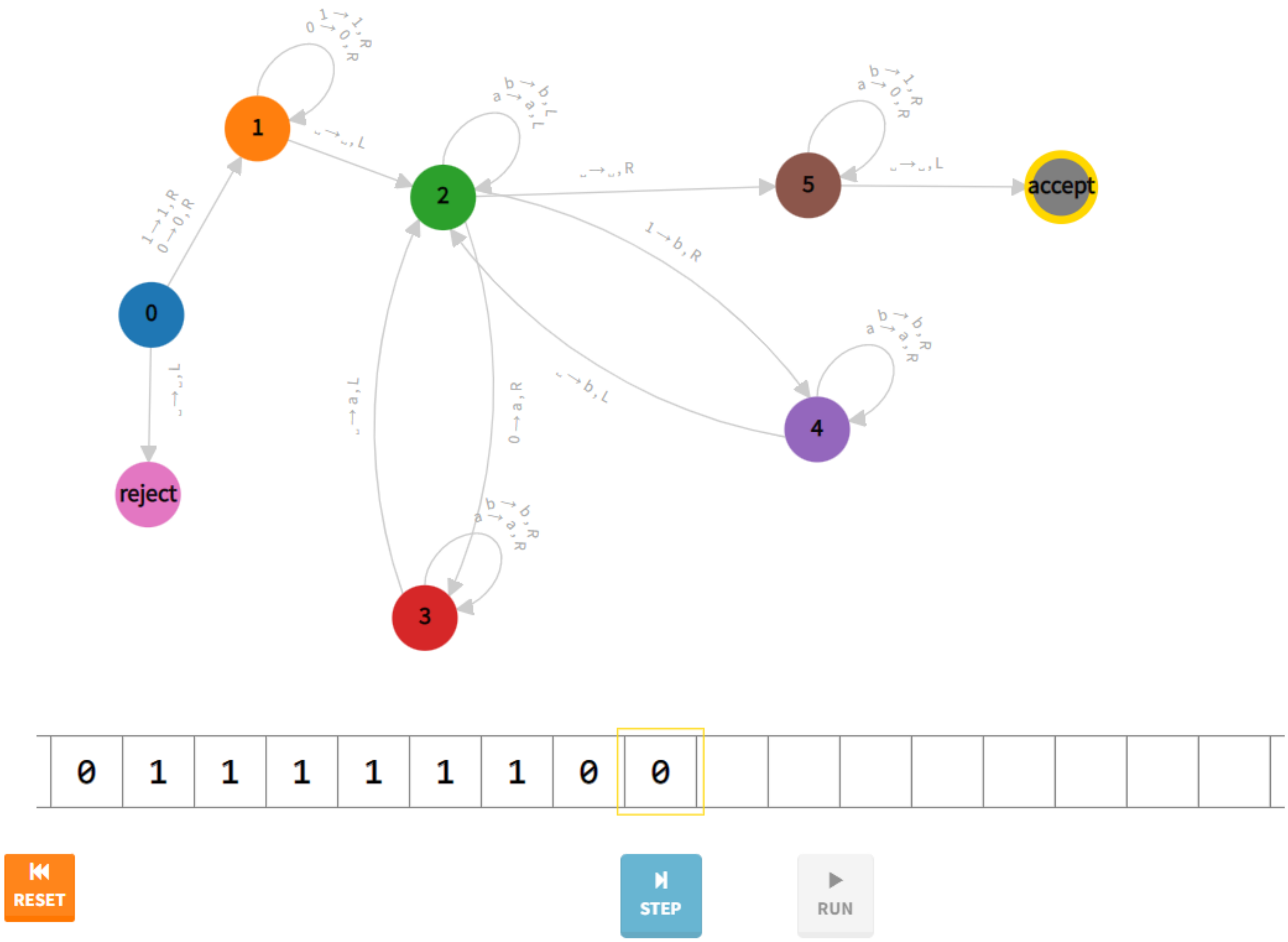

Figure 26: 00111 End

3

$M = \{\ K,\ \Sigma,\ \delta,\ s,\ F\ \}$ **K, s, F is same as normal Turing Machine, δ has the same definition as Turin Machine with the addition of tape bottom marker $\triangle$ and δ is K x Σ to K x $(\Sigma \cup \{\leftarrow, \rightarrow, \uparrow, \downarrow\})$ such that $\delta(q_1, \triangleright) = (q_2, \rightarrow)$ and $\delta(q_1, \triangle) = (q_2, \uparrow)$ for all q_1. The configuration for the 2-D Turing machine is**

$$K \text{ x } N \text{ x } N \text{ x } T$$

where S is set of functions from N x N to Σ that have $t(0, y) = \triangleright$ for all $y \in N$, $t(x, 0) = \triangle$ for all $x > 0$ and have $t(x, y) = 0$ for all but finite number of (x, y) pairs. So the configuration is represented by the current state, current head position and a list of non blank squares on the tape. We say $(q_1, x_1, y_1, t_1) \vdash (q_2, x_2, y_2, t_2)$ if $\delta(q_1, t_1(x_1, y_1)) = (q_2, \sigma)$ and one of the following

$x_1 = x_2, y_1 + 1 = y_2, t_1 = t_2$ **and** $\sigma = \rightarrow$
$x_1 = x_2, y_1 - 1 = y_2, t_1 = t_2$ **and** $\sigma = \leftarrow$
$x_1 + 1 = x_2, y_1 = y_2, t_1 = t_2$ **and** $\sigma = \uparrow$
$x_1 - 1 = x_2, y_1 = y_2, t_1 = t_2$ **and** $\sigma = \downarrow$
$x_1 = x_2, y_1 = y_2, t_2(x_1, y_1) = \sigma, t_2(x, y) = t_1(x, y)$ **for all other pairs** (x, y)**,**
and $\sigma \notin \{\rightarrow, \leftarrow, \uparrow, \leftarrow\}$

Given a string w, let $t_w \in T$ be the function that has $t(i+1, 1) = w(i)$ for $0 < i \leq |w|, t(0, y) = \triangleright$ for $y \in N, t(x, 0) = \triangle$ for all $x > 0$, and $t(x, y) = \sqcup$ otherwise. If we have 2-D dimensional tape Turing machine M with halting state y(yes) and n(no) such that for any string w either $(s, 1, 1, t_w) \vdash_M^* (y, i, j, t^{'}\ )$ or $(s, 1, 1, t_w) \vdash_M^* (n, i, j, t^{'}\ )$ for some $i, j \in N$ and $t^{'} \in T$, we let the language defined by M be the set of string that halts at the y state.

Note that from the definition of T, any $t \in T$ can be encoded as an finite set of ordered triples (x, y, σ) for which t differs from the function which has end-markers along thee left and bottom edges and remaining is blank. Since t is a function, there can be only one such σ for a given x and y, so if these values are encoded along the tape, a unique square can be assigned every possible x and y value. The non blank entries of t are filled with this tabular format, so that $t(x, y)$ is stored in $1/2[(i+j)^2 + 3i + j]th$ square (the first blank entry to the right of the $\triangleright$ being considered as 0).At any time during computation there are only finite number of square that will not be blank.

Therefore, we simulate M with a three-tape Turing machine $M^{'}$. $M^{'}$ uses one tape for calculation, one to hold the encoding of the tape M, and takes the input in the third. The calculations M performs are to increment and decrement the two registers i,j(to indicate the current location of the $M^{'}$ on its tape, then to convert this number to binary which tells $M^{'}$ where the symbol in cell is stored). $M^{'}$ starts by copying out the tape of M into its internal representation by scanning forward through the actual input and marking each symbol encoded location M would receive that symbol as input.

At every step $M^{'}$ looks at the current symbol on the encoding tape and at the current state of M. If the instructions are to write a symbol, $M^{'}$ writes the symbol and continues. If the instructions are to move, $M^{'}$ increment or decrements the i or j, then calculates the value with $1/2[(i+j)^2+3i+j]$. $M^{'}$ then moves the encoding tape head all to way to left, then to appropriate square.

At any given time, if i and j represent the largest x and y coordinates that M has been, the n i and j is certainly not larger than t. Thus, simulating one move takes the time to increment or decrement a binary register, compute a few binary sums and products, and move one head by a number of spaces that is a quadratic polynomial in i and j, and thus is bounded by a quadratic polynomial in t. Increments and arithmetic are fast and can be done in time polynomial in t. So the time to carry out t steps, after finishing up the initial setup is $\mathcal{O}(t^3)$. The initial set up can be regarded as an 2n-step computation of alternating writes and moves, so it can be finished in time $\mathcal{O}(n^3)$. Therefore the simulation can be carried out in time polynomial in t and n.