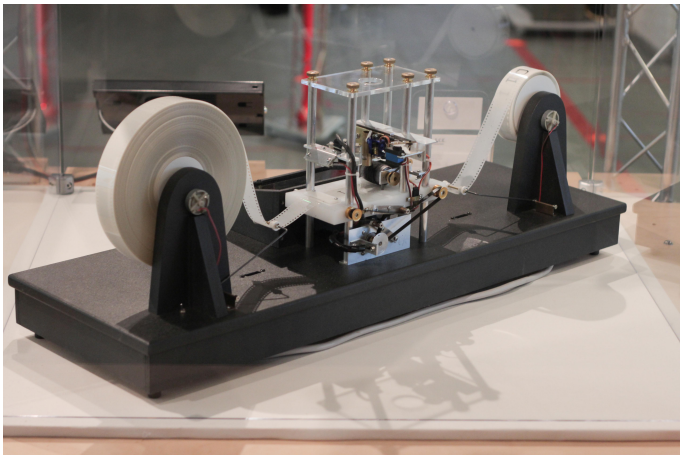
Chapter 2. Turing Machines

Note. A physical model of a Turing machine involves a tape of symbols '0' and '1', a head to read the tape, the ability to write on the tape, a way to advance the tape left or right, and a set of instructions to move the machine internally between different states.



This image is from the Wikipedia webpage on Turing Machines (accessed 6/15/2022)

2.1. Turing Machine Basics

Note. In this section, we give a brief informal description of a Turing machine and then state a very formal definition. We introduce some terminology and give several examples.

Note. A Turing machine is actually very elementary. It consists of an input which is a string of symbols. The machine can (1) move a "cursor" left and right on the string, (2) write on the current position of the cursor, and (3) "branch" depending

on the value of the current symbol. We will see that every a Turing machine can express every algorithm and simulate any programming language.

Note. In the popular level book by William Cook, *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*, Princeton University Press (2012), a Turing machine is described as follows (see page 169):

> Turing's machine has a tape for holding symbols, a head that moves along the tape reading and writing symbols in individual cells, and a controller to guide the read/write head. The machine also has a finite set of states, with two special states being *initial* and *halt*. The controller is actually a table that indicates what the machine should do if it is in a particular state s and it reads a particular symbol x. The "what it should do" is to print a new symbol x' on the cell of the tape, move the head either left or right one cell, and enter a new state s'. To solve a problem, the machine starts in its *initial* state, with the input to the problem written on the tape; it terminates when it reaches the *halt* state.

As an example, Cook considers the case of a string of 0's and 1's and addresses the question of whether the numbers of 1's is even or odd. The set of states is $\{initial, odd, even, halt\}$. The symbols are '0' and '1,' along with a blank which we denote as '_' here (the blank is used to halt the machine). The controller is given by the transition table in the figure below (Figure 9.1 on page 170 from Cook's book). There is a row of the table for each of the symbols and a column for each state other than the *halt* state. The table entries are triples which give the symbol to write, the direction to move the head on the tape, and the next state.

	initial	*odd*	*even*
0	_, right, *even*	_, right, *odd*	_, right, *even*
1	_, right, *odd*	_, right, *even*	_, right, *odd*
_	0, _, *halt*	1, _, *halt*	0, _, *halt*

Figure 9.1. Transition table for a parity-checking Turing machine.

The machine starts in the *initial* state and the head reads the symbol in the first cell. If the symbol is a blank then the machine prints a '0,' the head does not move, and then the machine goes into the *halt* state. If, instead, the head reads a '0' in the first cell then the machine prints a blank, the head advances to the right, and the machine enters the state *even* (since there are no 1's read at this stage). If the head reads a '1' in the first cell then the machine prints a blank, the head advances to the right, and the machine enters the state *odd* (since there is one 1 at this stage. Similarly, if the machine is in a state of *odd*, respectively *even*, and the head reads a '0' in the current cell, then the machine prints a blank, the head advances to the right, and the machine enters the state *odd*, respectively *even* (since a '1' was not read then the parity of the number of 1's remains unchanged). If the machine is in a state of *odd*, respectively *even*, and the head reads a '1' in the current cell, then the machine prints a blank, the head advances to the right, and the machine enters the state *even*, respectively *odd* (since a '1' was read then the parity of the number of 1's changes). Finally, if the machine is in a state of *odd*, respectively *even*, and the head reads a blank in the current cell, then the machine prints a '1', respectively '0', the head does not move, and the machine goes into the *halt* state. In this way, the original sequence of 0's and 1's is appended with a '0' if there are an even number of 1's in the original sequence, and is appended with a '1' if there is an odd number of 1's in the original sequence.

Note. Now we turn to a very formal definition of a Turing machine.

Definition 2.1. A *Turing machine* is a quadruple $M = (K, \Sigma, \delta, s)$. Here K is a finite set of *states*; $s \in K$ is the *initial state*. Σ is a finite set of *symbols* (we say Σ is the *alphabet* of M). We assume that K and Σ are disjoint sets. Σ always contains the special symbols $\sqcup$ (the *blank*) and $\triangleright$ (the *first symbol*). Finally, δ is a *transition function*, which maps $K \times \Sigma$ to $(K \cup \{h, \text{“yes”}, \text{“no”}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$. We assume that h (the *halting state*), “yes” (the *accepting state*), “no” (the *rejecting state*), and the *cursor directions* $\leftarrow$ for “left,” $\rightarrow$ for “right,” and $-$ for “stay,” are not in $K \cup \Sigma$.

Note. The controller or transition table of Cook's example corresponds to the δ of Definition 2.1. Notice that δ maps $K \times \Sigma$ (that is, ordered pairs consisting of a state and a symbol, just as in columns and rows of Cook's transition table) to triples of $(K \cup \{h, \text{“yes”}, \text{“no”}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$ (that is, ordered triples consisting of a state [along with the new states h, “yes,” and “no”], a symbol to be printed, and movement of the head; though the order of these triples does not agree with the order used by Cook).

Note 2.1.A. More precisely, the function δ takes the current state $q \in K$ and the current symbol $\sigma \in \Sigma$ and produces $\delta(q, \sigma) = (p, \rho, D)$ (technically, we should have $\delta((q, \sigma))$, but we suppress the unnecessary parentheses). In this triple, p is the next state, ρ is the symbol overwritten on σ, and $D \in \{\leftarrow, \rightarrow, -\}$ is the direction in

which the cursor will move. When symbol $\sigma = \triangleright$, we will require that for any state $q \in K$ that

$$\delta(q, \sigma) = \delta(q, \triangleright) = (p, \triangleright, \rightarrow) = (p, \rho, D)$$

for some state $p \in K$. In other words, symbol $\triangleright$ always directs the cursor to the right and is not erased (well, it is overwritten with itself since we require parameter $\rho = \triangleright$). The machine starts in initial state s, so the the machine processes the step $\delta(s, \triangleright) = (p, \triangleright, \rightarrow)$ which results in the cursor moving to the right and entering state p (which is determined from the definition of δ). Notice that cursor therefore can never move to the left of $\triangleright$. By convention, we start with a finite string of symbols, denoted x, that has $\triangleright$ as the first symbol and no where else in x, and that does not contain the blank symbol $\sqcup$ (though we may think of the last symbol as being followed by blanks). However, the cursor may write $\sqcup$ at any position other than the first position. The cursor can move to the right of the last symbol in the string and can write a symbol there. The machine stops when it reaches one of the halting states h, "yes," or "no."

Definition. The initial string x, preceded by $\triangleright$ (not part of x), containing $\triangleright$ nowhere else, and not containing $\sqcup$, is the *input* of the Turing machine M. When M enters reaches one of the halting states, the M has *halted*. If it reaches state "yes" the machine *accepts* its input x. If it reaches the state "no" the machine *rejects* its input x. If M halts on input x (as opposed to not halting) by reaching state "yes" or "no" then we define the *output* $M(x)$ as "yes" or "no," respectively. If M halts on input x by reaching state h then we define the *output* $M(x)$ as the string y of M at the time of halting (called the *output of the computation*), where

by convention the first symbol $\triangleright$ is dropped and the last symbol of y is not $\sqcup$ (so that blanks are truncated from the right end, if necessary, and we allow for the possibility that y is empty). We denote the case that M does not halt on input x and $M(x) = \nearrow$.

Example 2.1. Consider the Turing machine $M = (K, \Sigma, \delta, s)$ where $K = \{s, q, q_0, q_1\}$, $\Sigma = \{0, 1, \sqcup, \triangleright\}$, and δ is as follows (left):

$p \in K$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
s	0	$(s, 0, \rightarrow)$
s	1	$(s, 1, \rightarrow)$
s	$\sqcup$	$(q, \sqcup, \leftarrow)$
s	$\triangleright$	$(s, \triangleright, \rightarrow)$
q	0	$(q_0, \sqcup, \rightarrow)$
q	1	$(q_1, \sqcup, \rightarrow)$
q	$\sqcup$	$(q, \sqcup, -)$
q	$\triangleright$	$(q, \triangleright, \rightarrow)$
q_0	0	$(s, 0, \leftarrow)$
q_0	1	$(s, 0, \leftarrow)$
q_0	$\sqcup$	$(s, 0, \leftarrow)$
q_0	$\triangleright$	$(h, \triangleright, \rightarrow)$
q_1	0	$(s, 1, \leftarrow)$
q_1	1	$(s, 1, \leftarrow)$
q_1	$\sqcup$	$(s, 1, \leftarrow)$
q_1	$\triangleright$	$(h, \triangleright, \rightarrow)$

0. s $\underline{\triangleright}010$
1. s $\triangleright\underline{0}10$
2. s $\triangleright0\underline{1}0$
3. s $\triangleright01\underline{0}$
4. s $\triangleright010\underline{\sqcup}$
5. q $\triangleright01\underline{0}\sqcup$
6. q_0 $\triangleright01\sqcup\underline{\sqcup}$
7. s $\triangleright01\underline{\sqcup}0$
8. q $\triangleright0\underline{1}\sqcup 0$
9. q_1 $\triangleright0\sqcup\underline{\sqcup}0$
10. s $\triangleright0\underline{\sqcup}10$
11. q $\triangleright\underline{0}\sqcup 10$
12. q_0 $\triangleright\sqcup\underline{\sqcup}10$
13. s $\triangleright\underline{\sqcup}010$
14. q $\underline{\triangleright}\sqcup 010$
15. h $\triangleright\underline{\sqcup}010$

We take the input as 010. This produces the computation given above (right), called the "configuration" of M (to be formally defined below). The output is simply the input preceded by $\sqcup$. In fact, one can show inductively that for any input x, M gives output $\sqcup x$ (again, notice that the first symbol $\triangleright$ is omitted from both the input and output; any blanks on the right end of the output are deleted, but not blanks on the left of the output). Notice that if M ever has input $p = q$ and $\sigma = \sqcup$ then will not halt because $\delta(p, \sigma) = \delta(q, \sqcup) = (q, \sqcup, -)$; M "gets stuck" at this step! However, we start (by convention) with a string of the form $\triangleright x$ where x contains no $\sqcup$'s and this input will never lead to execution of $\delta(q, \sqcup)$. This implies that we could redefine the value of δ on $(q, \sqcup)$ to be anything else, and the new machine would always yield the same output (Papadimitriou says these to machines are "precisely equivalent"). It is to be shown by induction in Exercise 2.8.2 that if the input of M is x, then the output is $y = \sqcup x$.

Definition. A *configuration* of Turing machine M is a triple (q, w, u), where $q \in K$ is a state, and w, u are strings in Σ^*, where Σ^* denotes all possible finite strings of symbols.

Note. In configuration (q, w, u), q represents the current state, w is to be the string to the left of the cursor (including the symbol at the current location of the cursor), and u is the string to the right of the cursor (possibly empty). For example, the 16 steps in Example 2.1 can be expressed as the (where, like Papadimitriou, we

denote the empty set as $\epsilon = \varnothing$):

Step i	q	$\delta^i(\triangleright x)$	Configuration
0	s	$\underline{\triangleright}010$	$(s, \triangleright, 010)$
1	s	$\triangleright\underline{0}10$	$(s, \triangleright 0, 10)$
2	s	$\triangleright 0\underline{1}0$	$(s, \triangleright 01, 0)$
3	s	$\triangleright 01\underline{0}$	$(s, \triangleright 010, \epsilon)$
4	s	$\triangleright 010\underline{\sqcup}$	$(s, \triangleright 010\sqcup, \epsilon)$
5	q	$\triangleright 01\underline{0}\sqcup$	$(q, \triangleright 010, \sqcup)$
6	q_0	$\triangleright 01 \sqcup \underline{\sqcup}$	$(q, \triangleright 01 \sqcup \sqcup, \epsilon)$
7	s	$\triangleright 01\underline{\sqcup}0$	$(s, \triangleright 01\sqcup, 0)$
8	q	$\triangleright 0\underline{1} \sqcup 0$	$(q, \triangleright 01, \sqcup 0)$
9	q_1	$\triangleright 0 \sqcup \underline{\sqcup}0$	$(q_1, \triangleright 0 \sqcup \sqcup, 0)$
10	s	$\triangleright 0\underline{\sqcup}10$	$(s, \triangleright 0\sqcup, 10)$
11	q	$\triangleright\underline{0} \sqcup 10$	$(q, \triangleright 0, \sqcup 10)$
12	q_0	$\triangleright \sqcup \underline{\sqcup}10$	$(q_0, \triangleright \sqcup \sqcup, 10)$
13	s	$\triangleright\underline{\sqcup}010$	$(s, \triangleright\sqcup, 010)$
14	q	$\underline{\triangleright} \sqcup 010$	$(q, \triangleright, \sqcup 010)$
15	h	$\triangleright\underline{\sqcup}010$	$(h, \triangleright\sqcup, 010)$

Definition 2.2. Let M be a (fixed) Turing machine. Configuration (q, w, u) *yields in one step* configuration (q', w', u'), denoted $(q, w, u) \xrightarrow{M} (q', w', u')$, if the following holds. Let σ be the last symbol of w and let $\delta(q, \sigma) = (p, \rho, D)$. Then $q' = p$ holds.

(1) If $D = \rightarrow$ then w' is the string w but with its last symbol (which was σ) replaced

by ρ and the first symbol u added to the right hand end of ("appended to") w (or $\sqcup$ appended to w if $u = \epsilon$); u' is u with the first symbol removed (or if $u = \epsilon$, then $u' = \epsilon$).

(2) If $D = \leftarrow$ then w' is w with σ omitted from its right end, and u' is u with ρ attached to the left end.

(3) If $D = -$ then w' is w with the ending σ replaced by ρ, and $u' = u$.

Configuration (q, w, u) *yields in* k *steps* configuration (q', w', u'), denoted $(q, w, u) \overset{M^k}{\rightarrow} (w', q', u')$, where $k \geq 0$ is an integer, if there are configurations (q_i, w_i, u_i) for $i = 1, 2, \ldots, k+1$, such that $(q_i, w_i, u_i) \overset{M}{\rightarrow} (q_{i+1}, w_{i+1}, u_{i+1})$ for $i = 1, 2, \ldots, k$, $(q_1, w_1, u_1) = (q, w, u)$, and $(q_{k+1}, w_{k+1}, u_{k+1}) = (q', w', u')$. Configuration (q, w, u) *yields* configuration (q', w', u'), denoted $(q, w, i) \overset{M^*}{\rightarrow} (q', w', u')$ if there is an integer $k \geq 0$ such that $(q, w, u) \overset{M^K}{\rightarrow} (q', w', u')$.

Note. Informally, configuration (q, w, u) yields in one step configuration (q', w', u'), if applying M (based on transition function δ) to configuration results in configuration (q', w', u'). The idea of yielding in k steps is simply iteration of M (in terms of δ) k times. In the notation of Definition 2.2, with M as the Turing machine of Example 2.1 we have:

$$(s, \triangleright, 010) \overset{M}{\rightarrow} (s, \triangleright 0, 10),\ (s, \triangleright, 010) \overset{M^{15}}{\rightarrow} (h, \triangleright\sqcup, 010),\ \text{and so } (s, \triangleright\sqcup, 010).$$

We now consider two more examples of Turing machines, one which finds a binary successor of an input and one which determines whether an input is a palindrome or not.

Example 2.2. Consider the Turing machine:

$p \in K$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
s	0	$(s, 0, \rightarrow)$
s	1	$(s, 1, \rightarrow)$
s	$\sqcup$	$(q, \sqcup, \leftarrow)$
s	$\triangleright$	$(s, \triangleright, \rightarrow)$
q	0	$(h, 1, -)$
q	1	$(q, 0, \leftarrow)$
q	$\triangleright$	$(h, \triangleright, \rightarrow)$

If the input is an integer n in binary form (possibly with leading 0's), the output is the binary representation of $n+1$. The third value of δ in the table above shows that M starts by moving the cursor to the right. Then, the first two values of δ show that M continues to move the cursor to the right until it reads $\sqcup$, at which time the third value of δoverwrites the $\sqcup$ with another $\sqcup$ and moves the cursor left (to what is the right-most bit of n). The 5th and 6th values of δ show that, when moving to the left, the first time the cursor encounters a 0 it overwrites it with a 1 and halts (since $\delta(q, 0) = (h, 1, -)$); when it encounters a 1 it overwrites it with a 0 and the cursor continues to move left (since $\delta(q, 1) = (q, 0, \leftarrow)$). These steps correspond to adding 1 to n because, in binary, adding one to the right-most bit, changing 0 to 1 OR changing 1 to 0 and "carrying the 1" to the next left bit; this process is then iterated until the right-most 0 is encountered. The last value of δ halts the machine when the cursor returns to $\triangleright$ in its leftward movement, after overwriting $\triangleright$ with another $\triangleright$ and then the cursor moves to the right (since $\delta(q, \triangleright) = (h, \triangleright, \rightarrow)$). "There is a 'bug' in this machine," as Papadimitriou sates on

page 23. In the event that the input is $n = 2^k - 1$ (in binary, a string of k 1's), then M will overwrite all of the 1's, encounter $\triangleright$, and halt with an output of 0 (well, a string of k 0's). For example, if $x = 111$ then the machine will follow the steps:

$$(s, \triangleright, 111) \stackrel{M}{\to} (s, \triangleright 1, 11) \stackrel{M}{\to} (s, \triangleright 11, 1) \stackrel{M}{\to} (s, \triangleright 111, \epsilon) \stackrel{M}{\to}$$

$$(q, \triangleright 111, \sqcup) \stackrel{M}{\to} (q, \triangleright 11, 0\sqcup) \stackrel{M}{\to} (q, \triangleright 1, 00\sqcup) \stackrel{M}{\to} (q, \triangleright, 000\sqcup) \stackrel{M}{\to} (h, 0, 00\sqcup).$$

To address this error, the steps in the machine from Example 2.1 could be ran first on input x (as a "subroutine," as stated on page 23), where the value of δ is modified to $\delta(q_1, \triangleright) = (q_2, \triangleright, \rightarrow)$ and δ is defined on the new configuration as $\delta(q_2, \sqcup) = (h, 0, \leftarrow)$ (the value of $\delta(p, \sigma)$ for $p = q_2$ and other values of σ is irrelevant, since when $p = q_2$ we must have $\sigma = \sqcup$). If we add these steps to the current machine, then we need to relabel the states to insure no ambiguity. Also, we need to take redefine $\delta(q_2, \sqcup)$ from $(h, 0, \leftarrow)$ to $(p, 0, \leftarrow)$ where p is the initial state of the Turing machine given in this example. Debugged!

Example 2.3. We now describe a Turing machine that halts in a state of "yes" or "no," unlike the previous examples (though the first example from Cook's *Traveling Salesman* book could be modified to be of this form). Machine M tests a binary string to see if it is a palindrome; that is, if it reads the same from left-to-right as it reads from right-to-left. The cursor reads a symbol at the left end of the input string, overwrites it with $\triangleright$, and enters a state which allows it to "remember" the symbol read. It then moves the cursor to the right end of the string and checks if it matches the first symbol; if it does not match then "no" is output, and it does match then the right-most symbol is overwritten with $\sqcup$. The cursor then moves back to the left end (now to the second symbol from the original string) and repeats

the process until the machine ends with "no" or completes the comparisons and ends with "yes." Notice that it alway reads the left of the string end and checks the right end of the string. This could be made more efficient by reading the right end symbol while the cursor is there and then comparing that to the left end symbol; but the point is to demonstrate a Turing machine that solves the problem with no regard for efficiency (but maybe a preference for simplicity).

$p \in K$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
s	0	$(q_0, \triangleright, \rightarrow)$
s	1	$(q_1, \triangleright, \rightarrow)$
s	$\triangleright$	$(q, \sqcup, \leftarrow)$
s	$\sqcup$	("yes", $\sqcup, -$)
q_0	0	$(q_0, 0, \rightarrow)$
q_0	1	$(q_0, 1, \rightarrow)$
q_0	$\sqcup$	$(q_0', \sqcup, \leftarrow)$
q_1	0	$(q_1, 0, \rightarrow)$
q_1	1	$(q_1, 1, \rightarrow)$
q_1	$\sqcup$	$(q_1', \sqcup, \leftarrow)$

$p \in K$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
q_0'	0	$(q, \sqcup, \leftarrow)$
q_0'	1	("no", $1, -$)
q_0'	$\triangleright$	("yes", $\sqcup, \rightarrow$)
q_1'	0	("no", $1, -$)
q_1'	1	$(q, \sqcup, \leftarrow)$
q_1'	$\triangleright$	("yes", $\triangleright, \rightarrow$)
q	0	$(q, 0, \leftarrow)$
q	1	$(q, 1, \leftarrow)$
q	$\triangleright$	$(s, \triangleright, \rightarrow)$

When the state is s, the cursor is reading "turning around" to move right (when the symbol $\triangleright$ is read), reading the left-most symbol (when that symbol is '0', M goes into state q_0 and when that symbol is '1', M goes into state q_1; this is how it "remembers" the symbol), or it is done (when the symbol is $\sqcup$). When the state is q_0, the last read left-most (non-$\triangleright$) font is 0 and the cursor advances to the right; when the state is q_1, the last read left-most symbol is 1 and the cursor advances to the right. The cursor stops moving to the right when it reaches a $\sqcup$. It then

overwrites $\sqcup$ with another $\sqcup$, replaces state q_i with q_i' (where $i = 0, 1$). The cursor then reads the right-most (non-blank) symbol to see if it matches the subscript of the state the machine is in. If not it outputs "no," if so then the right-most symbol is overwritten with $\sqcup$, the machine enters state q and this leads the cursor to move back to the left until it reaches the left end (indicated by $\triangleright$). The process is iterated. The machine halts when a symbol check fails (that is, when $(p, \sigma) = (q_0', 1)$ or $(p, \sigma) = (q_1', 0)$), or when there are no more symbols to read. There are no more symbols to read when

(1) the cursor has moved left and $\triangleright$ is next to $\sqcup$ (when $(p, \sigma) = (s, \sqcup)$),

(2) the cursor has made the last successful comparison, overwritten the last symbol '0' with $\sqcup$ so that the cursor has moved to the left and reads $\triangleright$ (when $(p, \sigma) = (q_0', \triangleright)$), or

(3) the cursor has made the last successful comparison, overwritten the last symbol '1' with $\sqcup$ so that the cursor has moved to the left and reads $\triangleright$ (when $(p, \sigma) = (q_1', \triangleright)$).

To illustrate, consider the inputs 0010 and 101. These give the configurations:

$$(s, \triangleright, 0010) \overset{M^5}{\to} (q_0, \triangleright\triangleright 010\sqcup, \epsilon) \overset{M}{\to} (q_0', \triangleright\triangleright 010, \sqcup) \overset{M}{\to} (q, \triangleright\triangleright 01, \sqcup\sqcup)$$

$$\overset{M^2}{\to} (q, \triangleright\triangleright, 01\sqcup\sqcup) \overset{M}{\to} (s, \triangleright\triangleright 0, 1\sqcup\sqcup) \overset{M}{\to} (q_0, \triangleright\triangleright\triangleright, 1\sqcup\sqcup)$$

$$\overset{M^2}{\to} (q_0, \triangleright\triangleright\triangleright 1\sqcup, \sqcup) \overset{M}{\to} (q_0', \triangleright\triangleright\triangleright 1, \sqcup\sqcup) \overset{M}{\to} (\text{"no"}, \triangleright\triangleright\triangleright 1, \sqcup\sqcup), \text{ and}$$

$$(s, \triangleright, 101) \overset{M}{\to} (s, \triangleright 1, 01) \overset{M}{\to} (q_1, \triangleright\triangleright 0, 1) \overset{M^3}{\to} (q_1', \triangleright\triangleright 01, \sqcup) \overset{M}{\to} (q, \triangleright\triangleright 0, \sqcup\sqcup)$$

$$\overset{M}{\to} (1, \triangleright\triangleright, 0\sqcup\sqcup) \overset{M}{\to} (q_0, \triangleright\triangleright\triangleright, \sqcup\sqcup) \overset{M}{\to} (q_0, \triangleright\triangleright\triangleright\sqcup, \sqcup) \overset{M}{\to} (q_0', \triangleright\triangleright\triangleright, \sqcup\sqcup) \overset{M}{\to} (\text{"yes"}, \triangleright\triangleright\triangleright\sqcup, \sqcup).$$