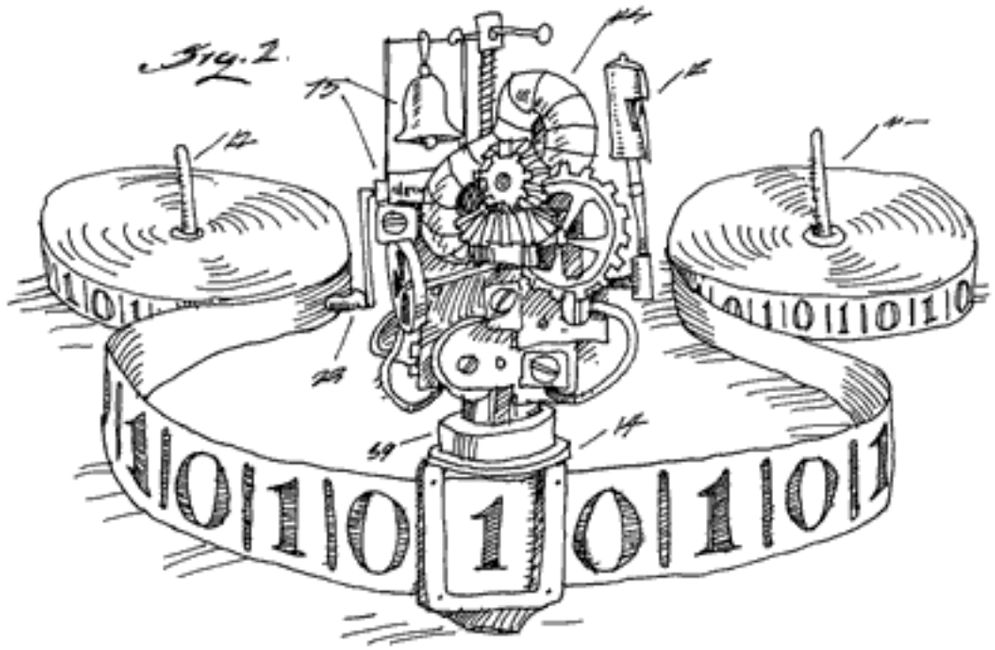
CSC4010: Introduction to Theoretical Computer Science



Lecture 6

Multi-Tape Turing Machine

2-Tape Turing Machine

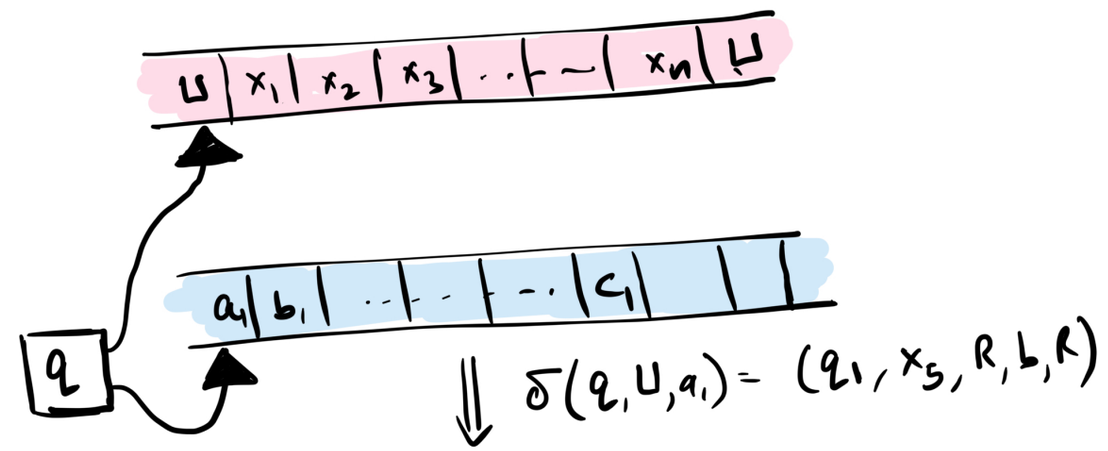
- A two-tape Turing Machine is like an ordinary Turing Machine with two tapes.

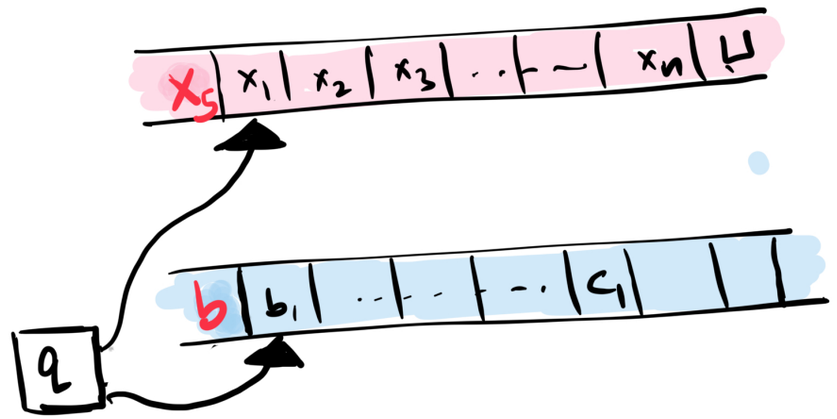
- Each tape has its own head for reading and writing.

- Initially, the input appears on one tape, and the second tape is blank.

- The transition function allows for reading, writing, and moving the head on all the tapes simultaneously:

$$\delta : Q \times \Sigma^2 \to Q \times \Sigma \times \{L, R\} \times \Sigma \times \{L, R\}$$





2-Tape Turing Machine

Let $L \subseteq \{0,1\}^*$ be a language.

Theorem: There exists a 2 Tape Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

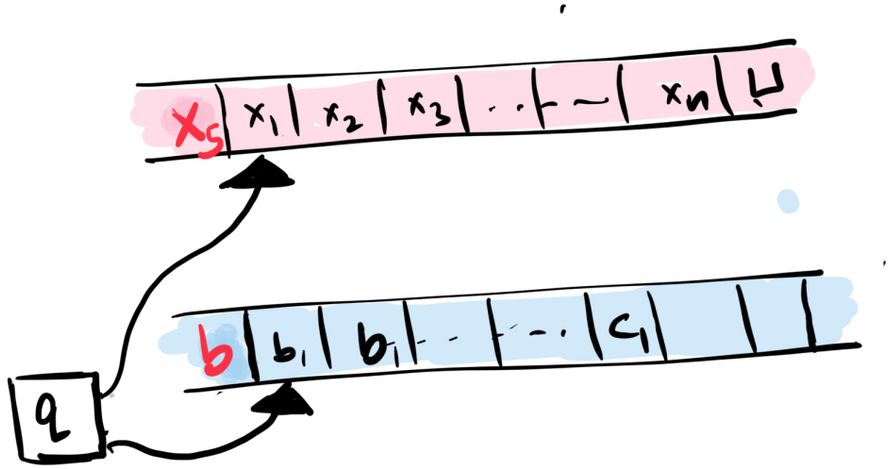
Proof: ($\Leftarrow$) M'_L is a 2 Tape Turing Machine that does not use the second tape.

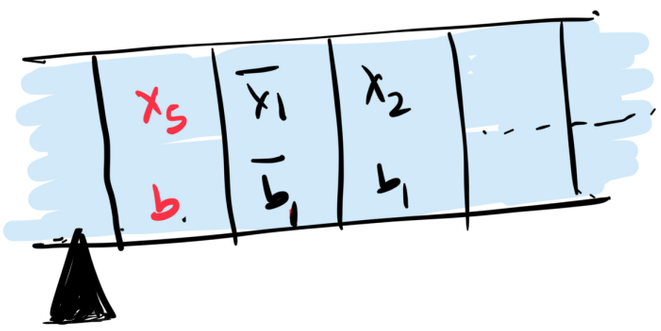
2 Tape Turing Machine

Proof: ($\Rightarrow$) Suppose $M_L = (Q, \Sigma, \delta, q_{\text{acc}}, q_{\text{rej}}, q_0, \sqcup)$ is a 2 Tape Turing Machine deciding L. Then, $M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

Idea:

- Divide the single tape into two tracks by adding two symbols in a single cell. Achieved by expanding the alphabet.
- To simulate two heads, use different symbols to denote the presence of "virtual heads".
- The real head goes back and forth, updating the position of the virtual heads.





$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

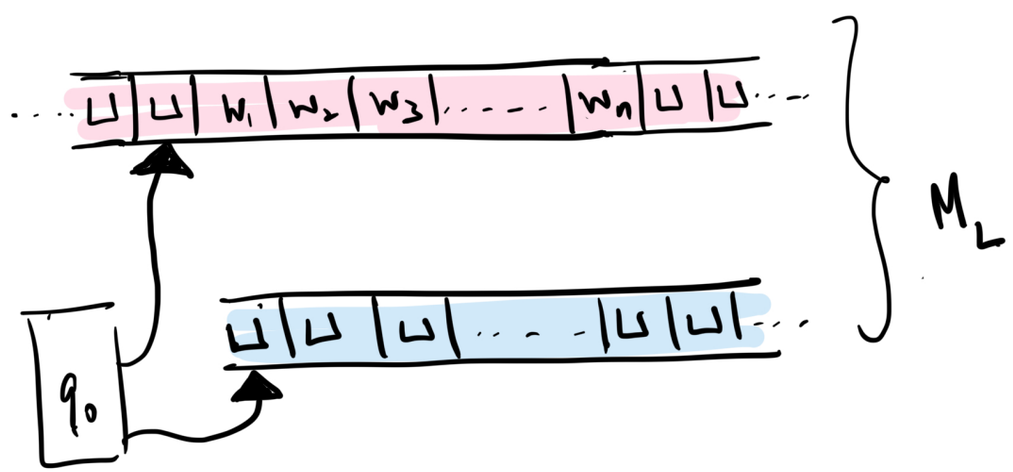
Simulating 2-Tape TM: Tape Alphabet

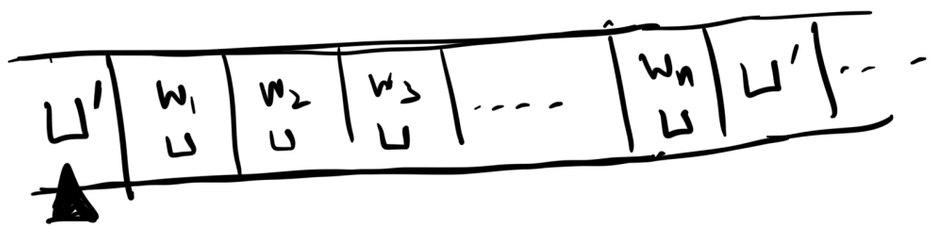
- To indicate presence of simulated head, define $\underline{\Sigma} = \{\overline{b} \mid b \in \Sigma\}$. A letter $\overline{b}$ denotes the presence of virtual head on the symbol.

- New blank symbol $\sqcup'$.

- New alphabet $\Sigma' = \{ \sqcup' \} \cup (\Sigma \cup \underline{\Sigma})^2$.

- For example: If the input to M_L is $w_1 w_2 \cdots w_n$. Then, the input to M'_L is $(w_1, \sqcup), (w_2, \sqcup), \cdots, (w_n, \sqcup)$.



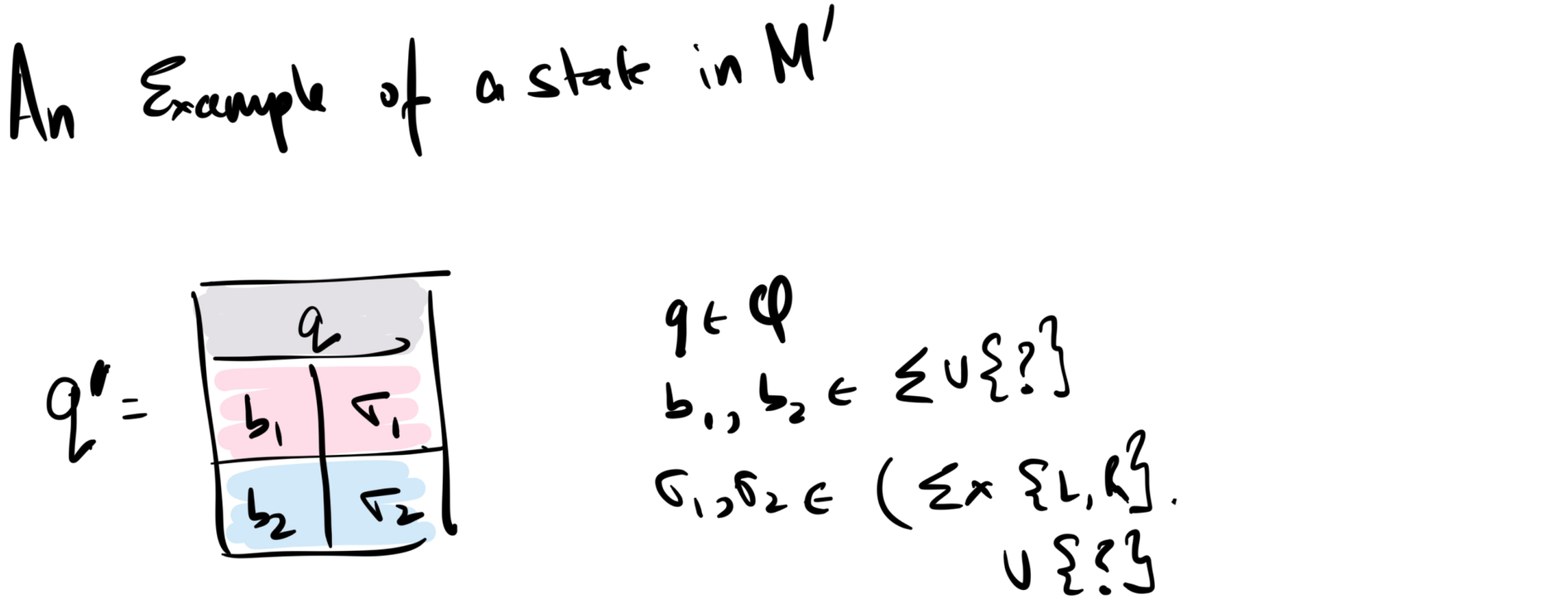


$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the overlined symbols and remember these symbols that are overlined.

- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)



$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the overlined symbols and remember these symbols that are overlined.

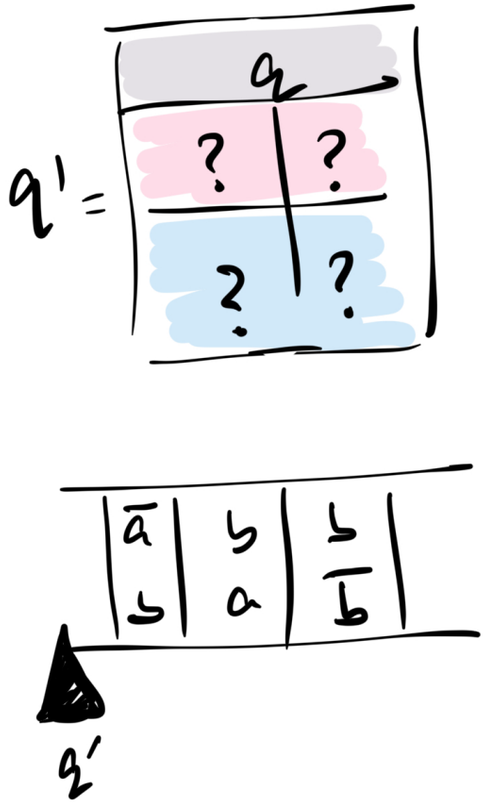
- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)

$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the overlined symbols and remember these symbols that are overlined.

- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)

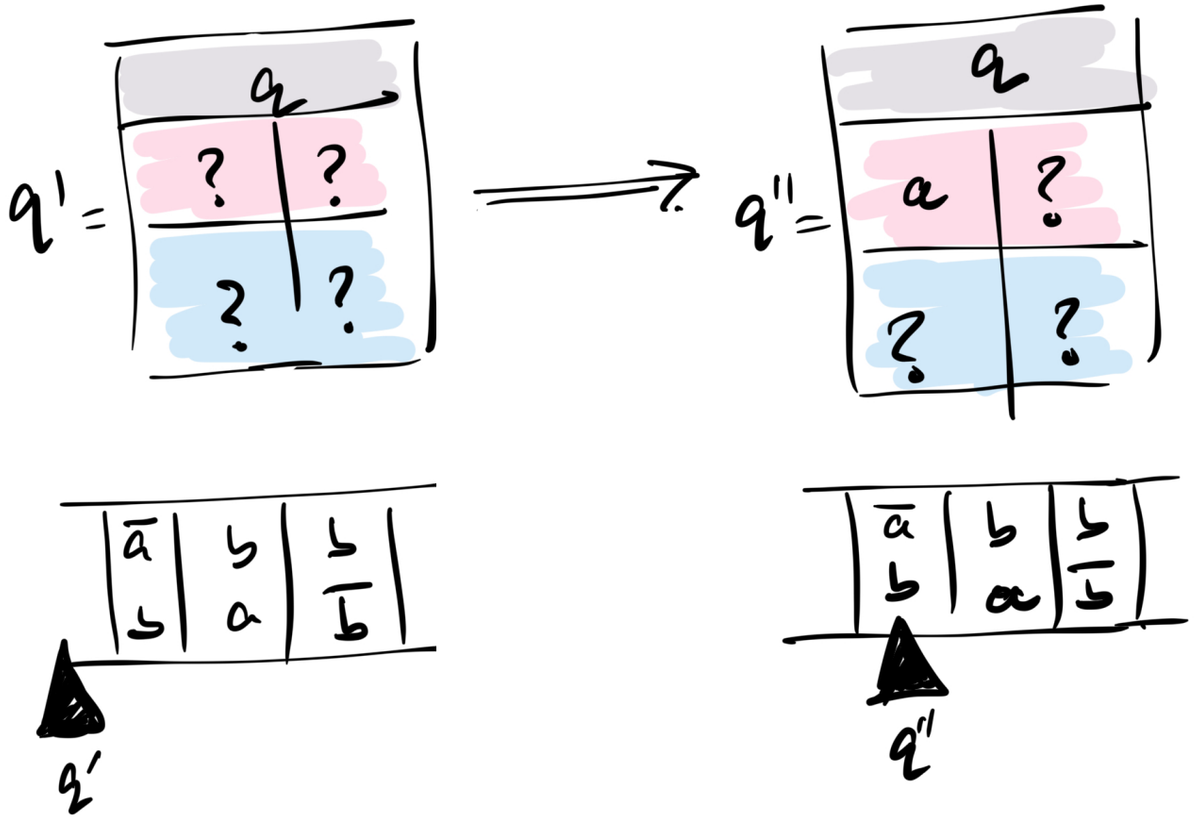


$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the overlined symbols and remember these symbols that are overlined.

- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)

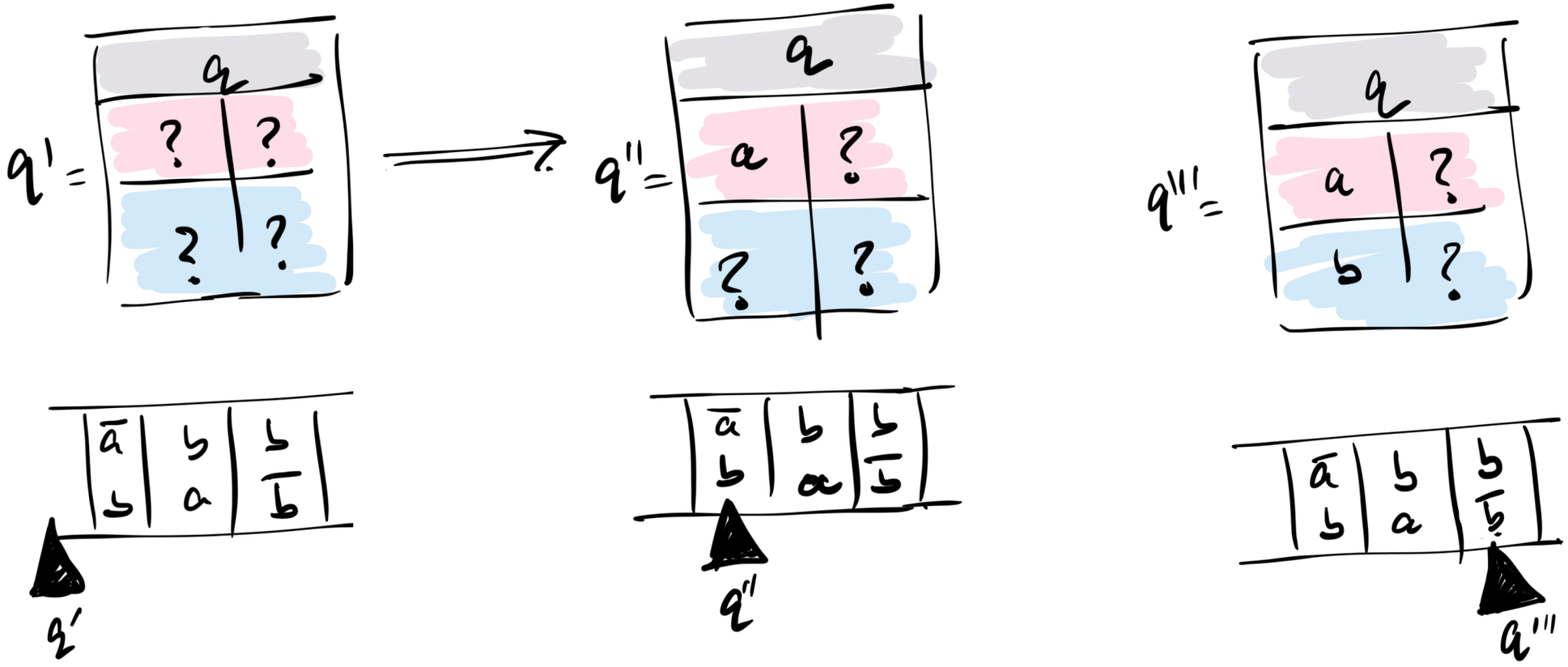


$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the overlined symbols and remember these symbols that are overlined.

- (Because there are only a finite number of possibilities, the machine can ‘remember’ these symbols via a finite number of states, without writing anything to the tape.)

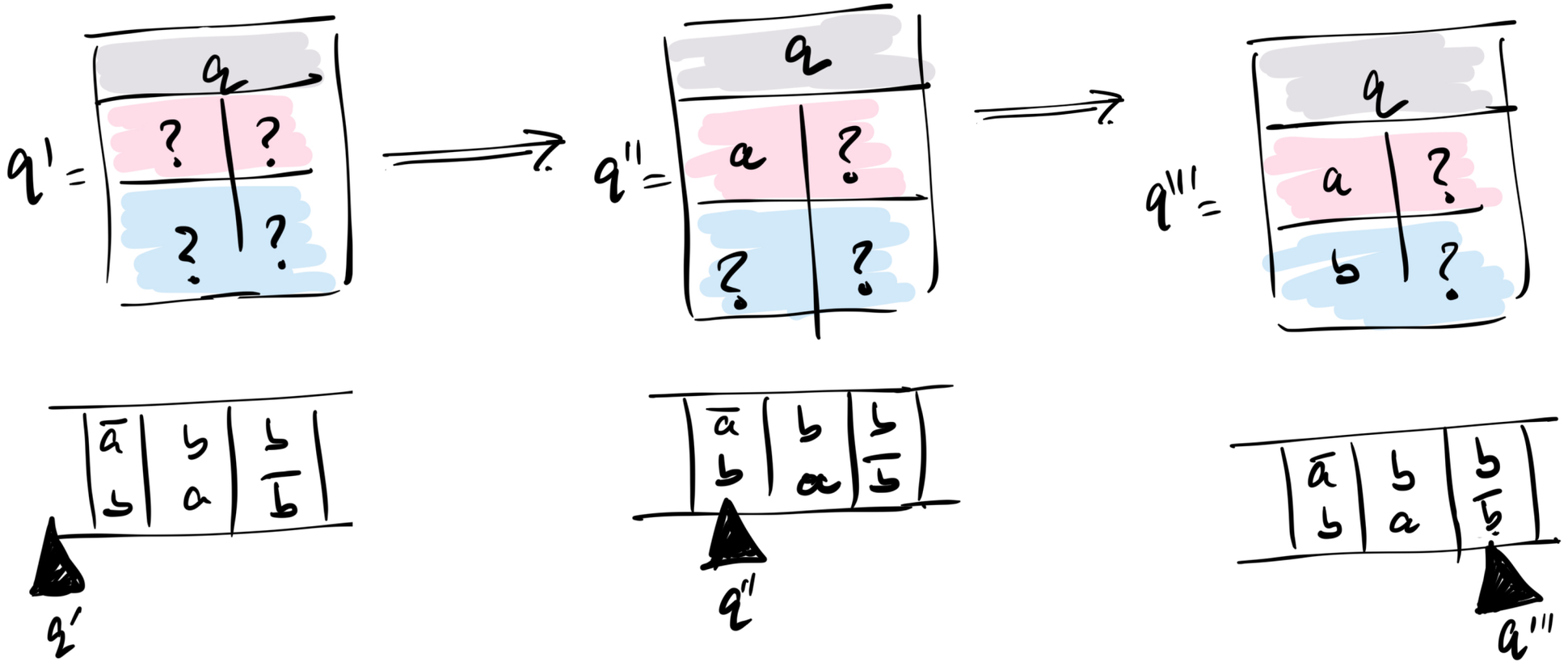


$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the overlined symbols and remember these symbols that are overlined.

- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)



$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M .

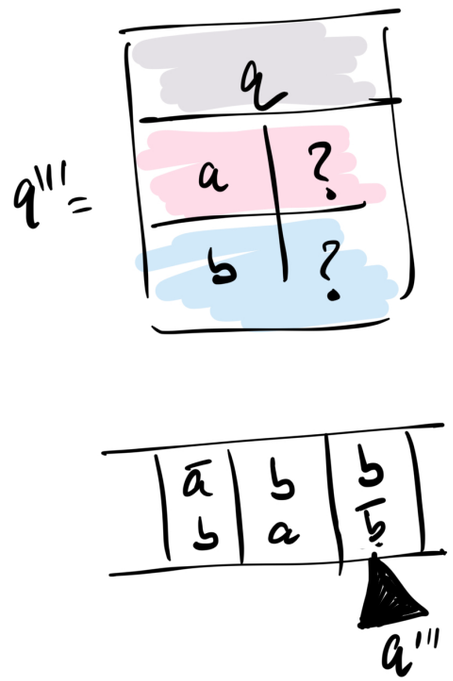
$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M.



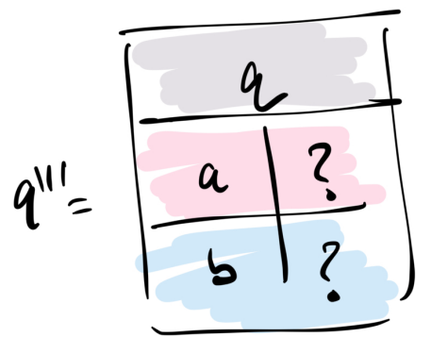
$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

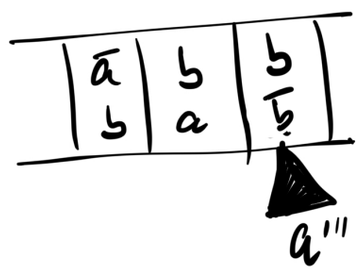
Simulating 2-Tape TM: Transition and States

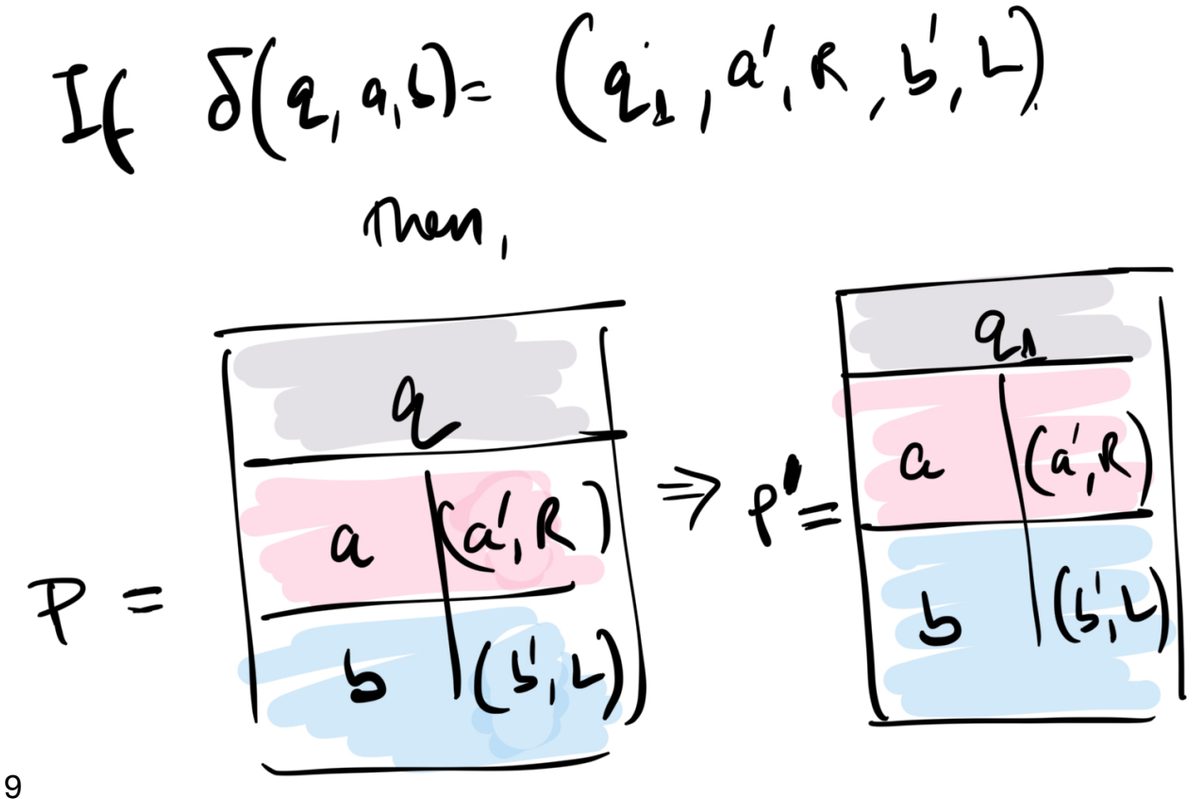
- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M.







$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

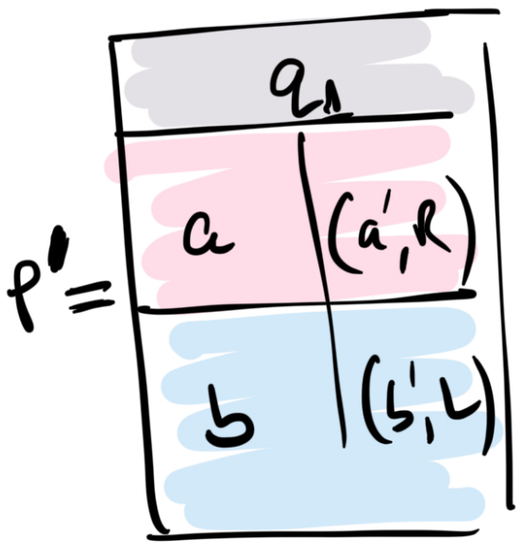
- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M.

$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).



- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M.

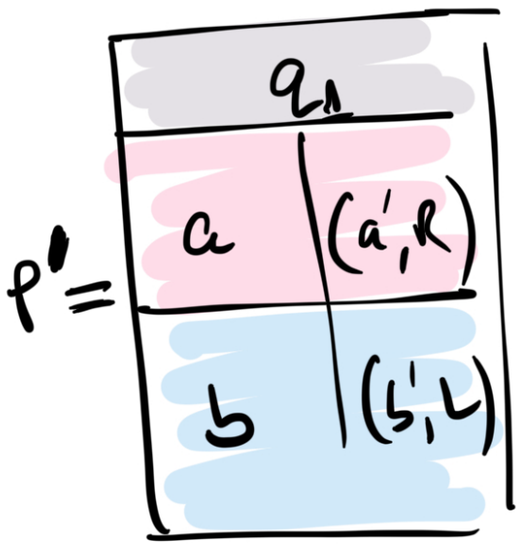
$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

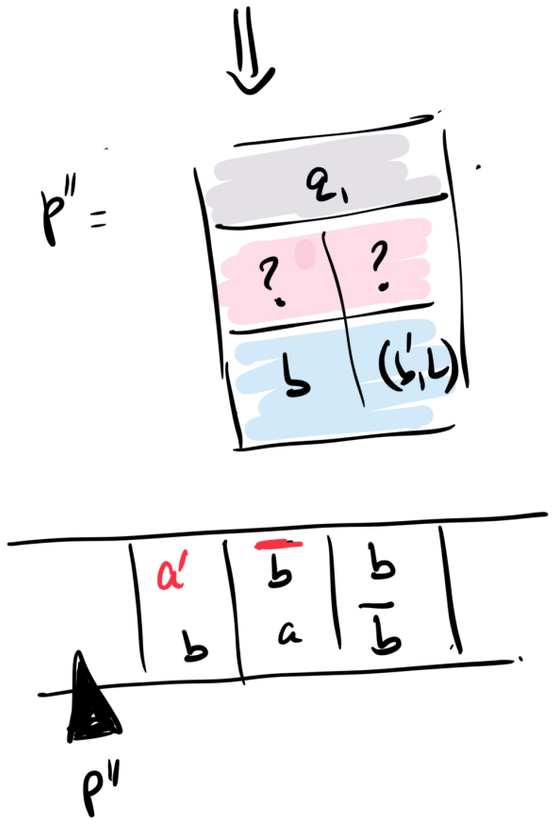
Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M.





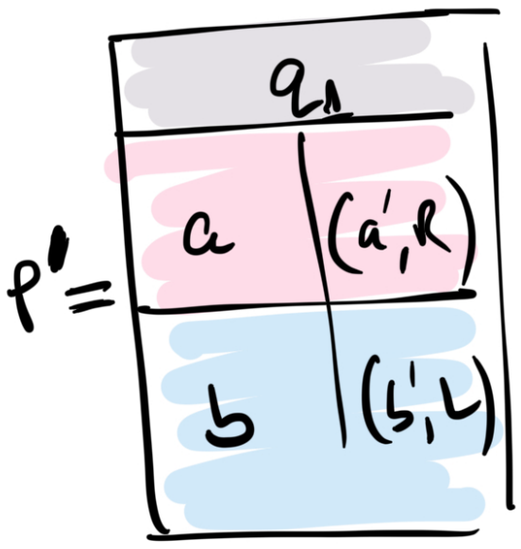
$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

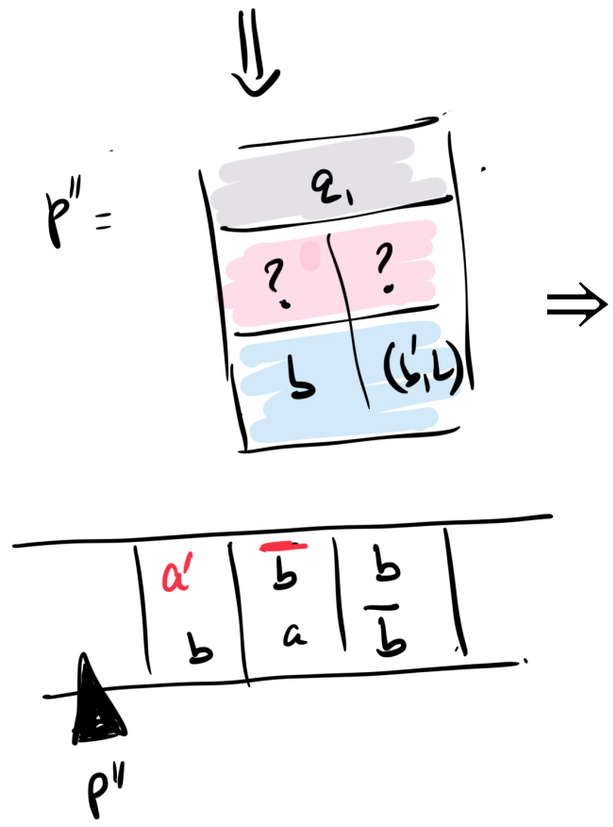
Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M.





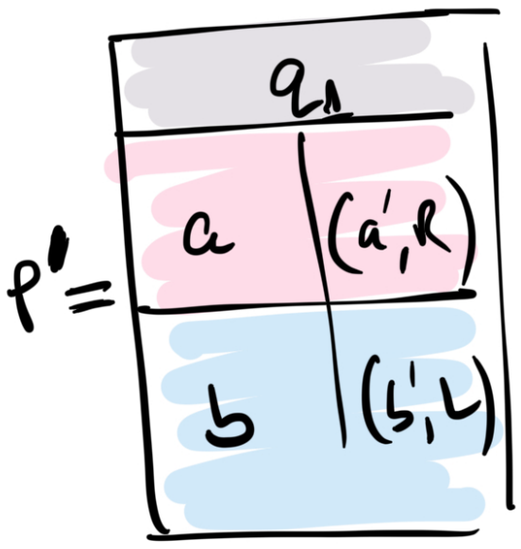
$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

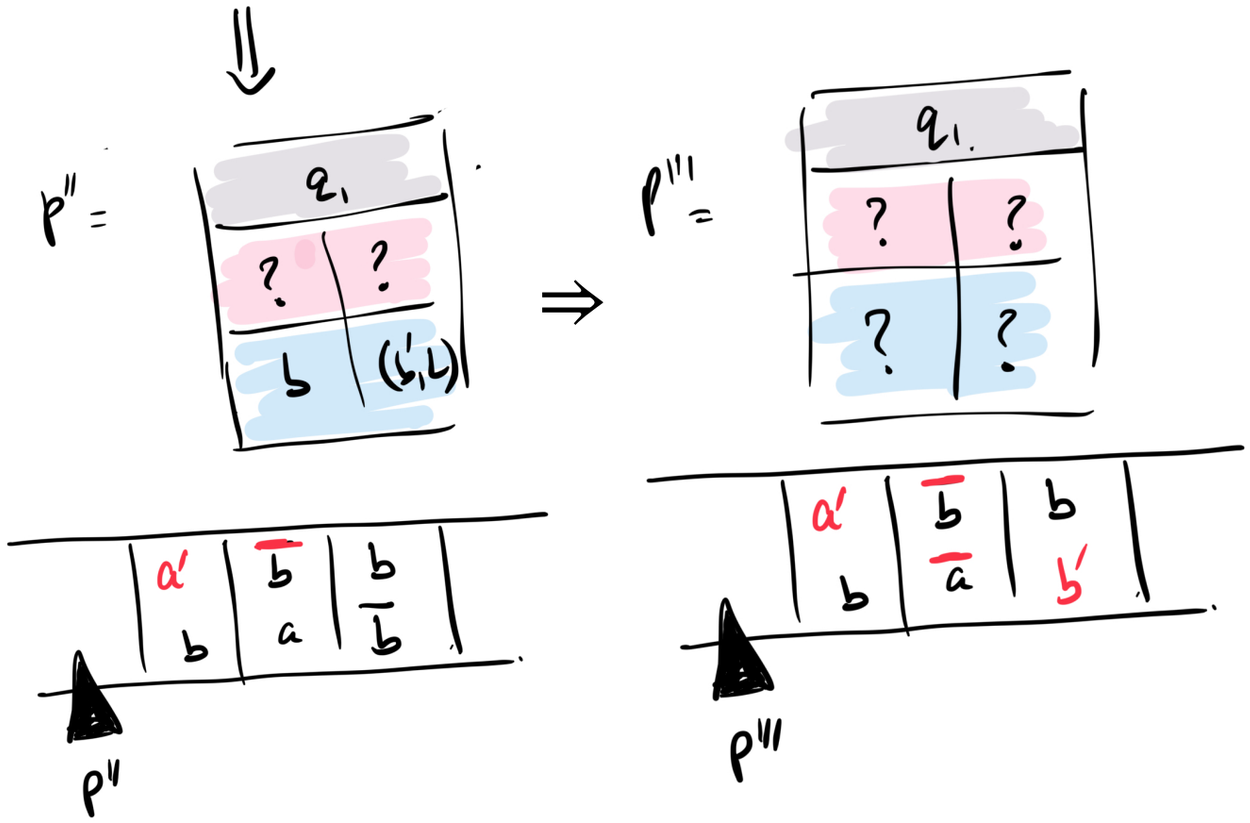
Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

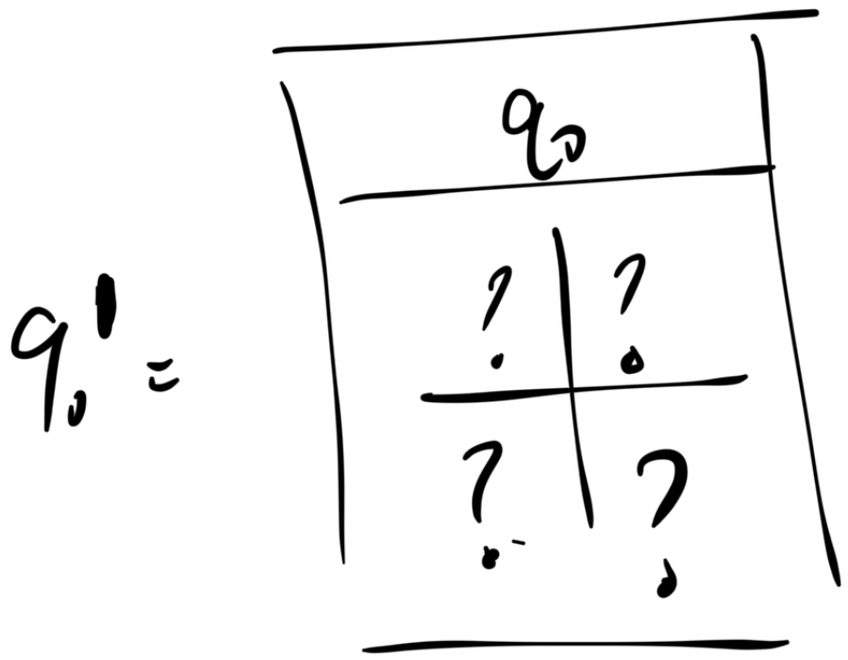
- Move the virtual heads to the right locations as specified by the transition function of M.





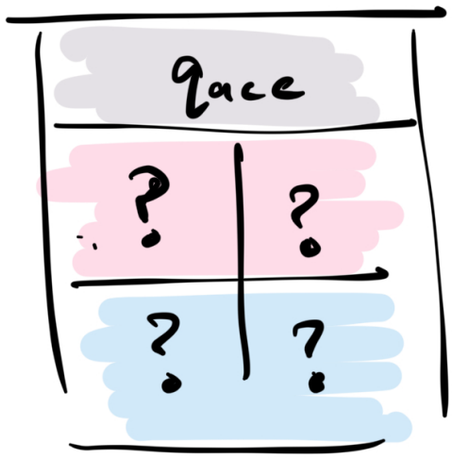
$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

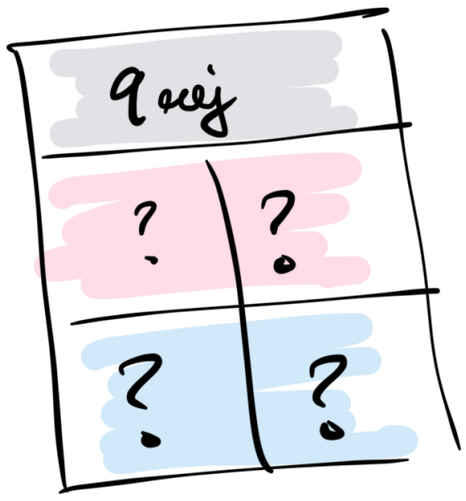
Simulating 2-Tape TM: Initial State

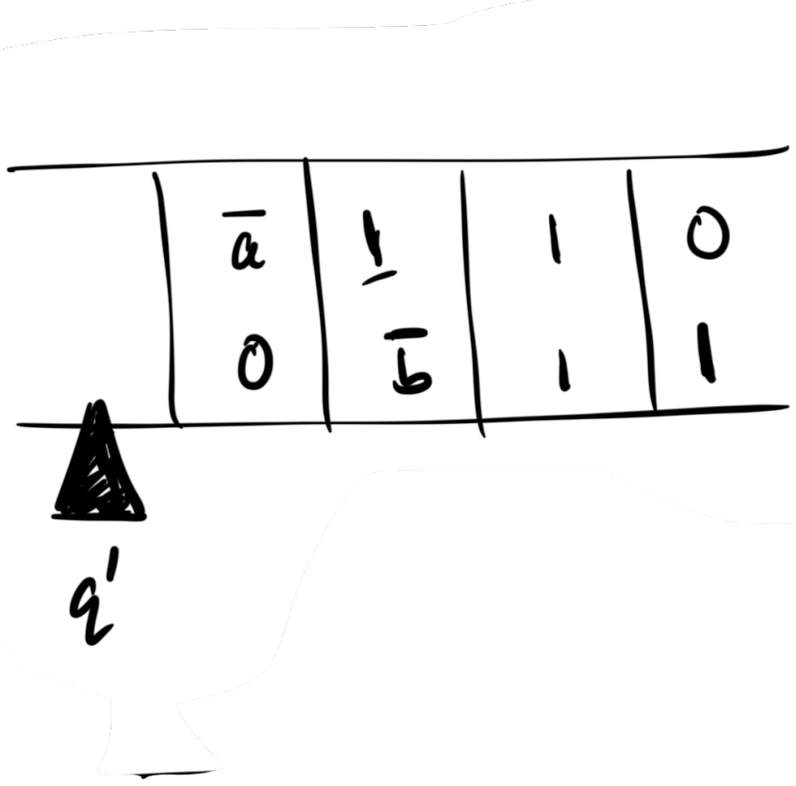


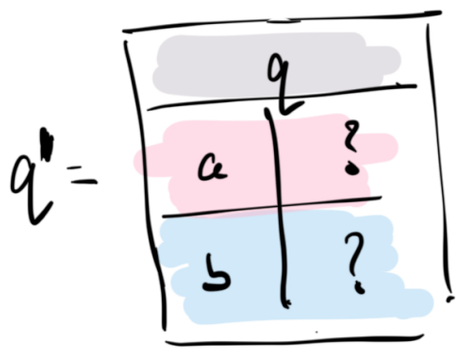
$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Halting States





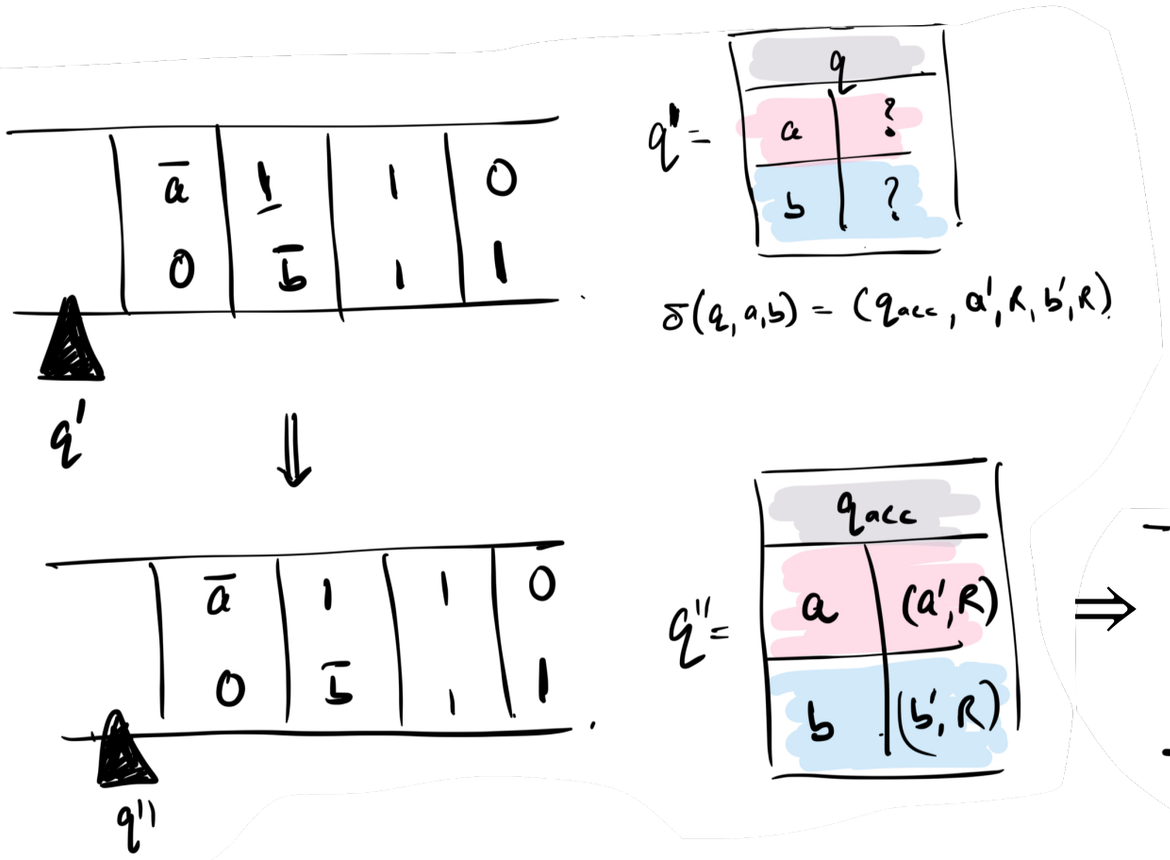




$\delta(q, a, b) = (q_{acc}, a', R, b', R)$

$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Halting States



Some Thoughts on the Proof Sketch

- Constructing this one-tape machine from the definition of the two-tape machine is a tedious, but completely mechanical, process.

- The important point is that the one-tape machine perfectly simulates the operation of the original two-tape machine, even though it typically takes many steps to carry out what the original machine does in a single step.

- If the two-tape machine accepts an input, then the one-tape simulation will do so as well, and similarly if the original machine rejects or loops.

LaTeX

Universal Turing Machine

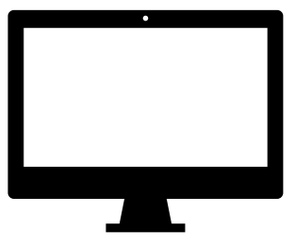
Turing Machine vs a Laptop

- Each Turing Machine can solve one problem.
- A laptop is a single device that can run arbitrary computations.
- Therefore, a Turing Machine cannot model a laptop.

Zoom Laptop

Email Laptop

Powerpoint Laptop



Word Processing Laptop

A Turing Machine M has a finite description.

A computer program can be treated as data.

Universal Turing Machine

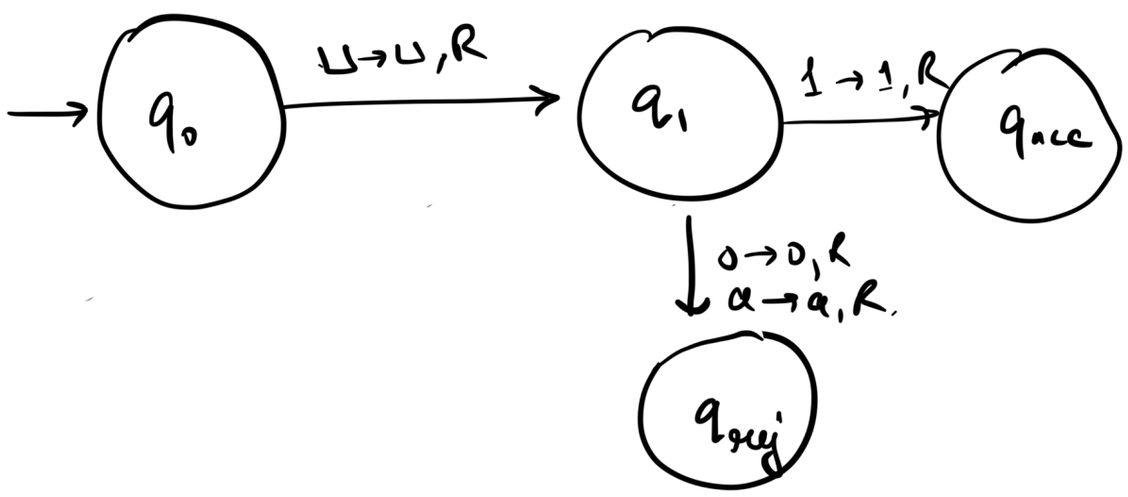
Theorem: There exists a Turing Machine U such that for every Turing Machine M and every input $w \in \{0,1\}^*$:

- If M accepts w, then U accepts $\langle M, w \rangle$.
- If M rejects w, then U rejects $\langle M, w \rangle$.
- If M loops on w, then U loops on $\langle M, w \rangle$.

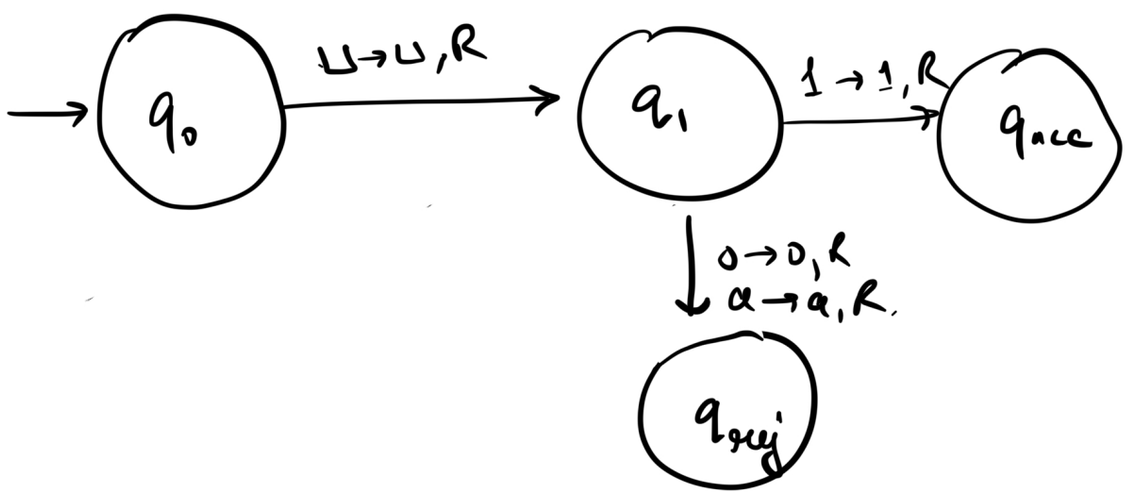
Encoding a Turing Machine as a String

- To encode a Turing Machine $M = (Q, \Sigma, q_{acc}, q_{rej}, q_0, \sqcup, \delta)$:
  - Assume that $|Q| = 2^k$, and, $|\Sigma| = 2^m$ for some $k, m \in \mathbb{N}$.
  - We can assume $Q = \{0,1\}^k$, $q_0 = 0^k$, $q_{acc} = 1^{k-1}0$, $q_{rej} = 1^k$.
  - Encode $b \in \Sigma$ as $\langle b \rangle \in \{0,1\}^m$. Let $\langle 0 \rangle = 0^m$, $\langle 1 \rangle = 10^{m-1}$, and $\langle \sqcup \rangle = 1^m$.
  - Encode $(q, b, D) \in Q \times \Sigma \times \{L, R\}$ as $\langle q, b, D \rangle = q\langle b \rangle \langle D \rangle \in \{0,1\}^{k+m+1}$.
  - Then, $\langle M \rangle = 1^k 01^m 0 \langle \delta \rangle$. Here, $\langle \delta \rangle$ is a list of $\langle \delta(q, b) \rangle$ for all $(q, b) \in Q \times \Sigma$.

Example



Example



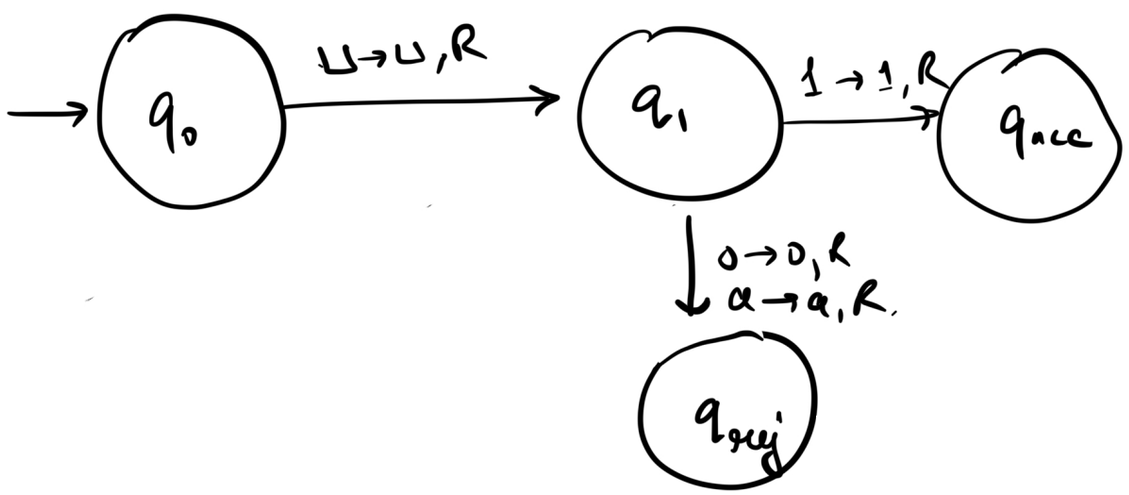
$Q = \{0,1\}^2 \quad q_0 = 00, q_{acc} = 10, q_{rej} = 11$

$q_1 = 01$

$\Sigma = \{00, 01, 10, 11\}$

$\langle 0 \rangle \quad \langle a \rangle \quad \langle 1 \rangle \quad \langle \sqcup \rangle$

Example



$Q = \{0,1\}^2 \quad q_0 = 00, q_{acc} = 10, q_{rej} = 11$

$q_1 = 01$

$\Sigma = \{00, 01, 10, 11\}$

$\langle 0 \rangle \quad \langle a \rangle \quad \langle 1 \rangle \quad \langle \sqcup \rangle$

Order $(q,b) \in Q \times \Sigma$ in lexicographic order.

And list out $\langle \delta(q,b) \rangle$ in that order.

Given an encoding $1^2 \cdot 0 1^2 0 1 0 1 1 1 0 1 1 1 0 \cdots$

$|Q| \quad |\Sigma| \quad \delta(q_0,b_0) \quad \delta(q_0,b_1) \cdots$

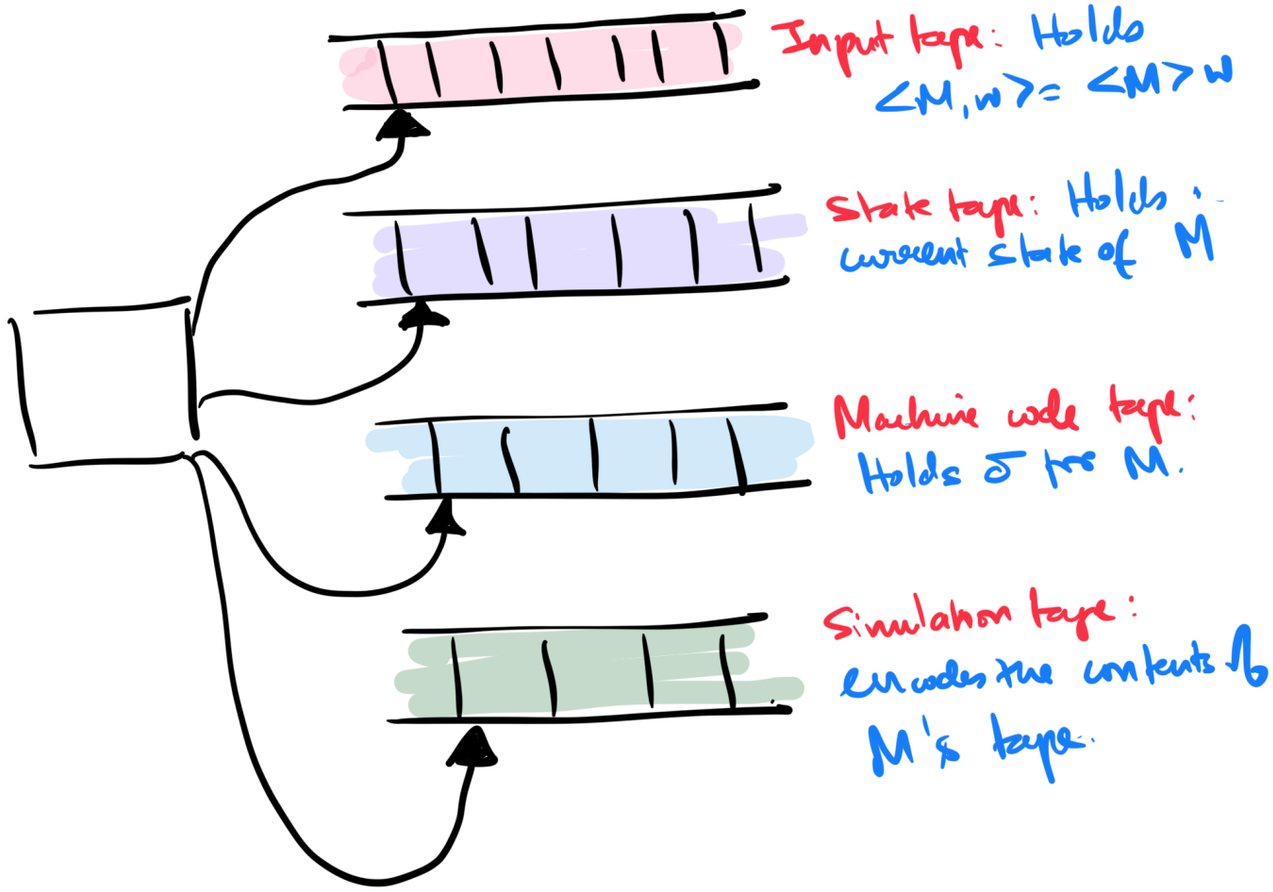
$|\langle q,b,D \rangle| = |q\langle b \rangle\langle D \rangle|$

$= 5$

Designing The Universal TM

- **Want:** U accepts
  $\langle M, w \rangle \iff M$ accepts w .

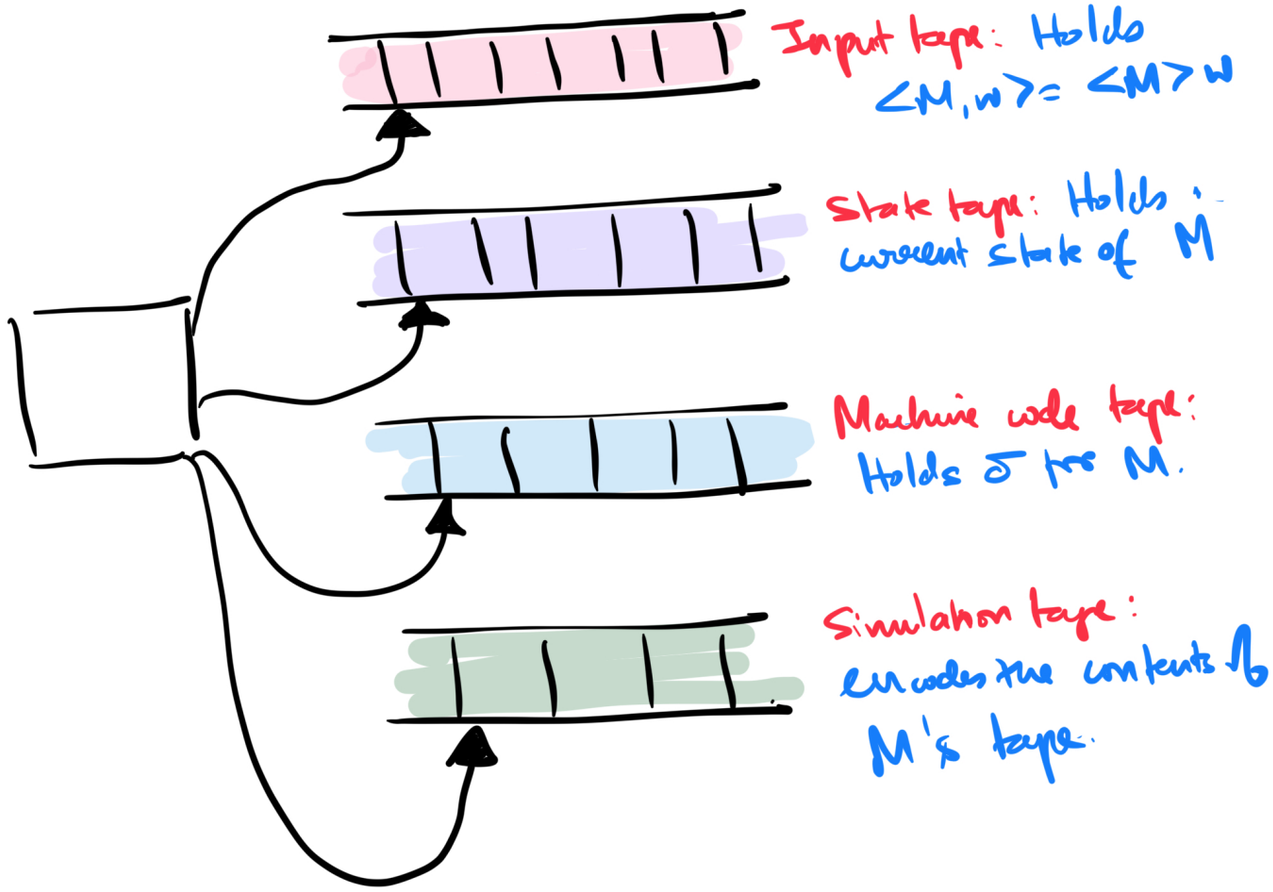
- Construct a multi-tape Turing Machine with four tapes:
  - Input Tape: Receives $\langle M, w \rangle = \langle M \rangle w$.
  - State Tape: Holds current state of M.
  - Machine Code Tape: Holds δ for M .
  - Simulation Tape: Encodes the contents of M's tape.



Designing The Universal TM

- **Want:** U accepts
  $\langle M, w \rangle \iff M$ accepts w .

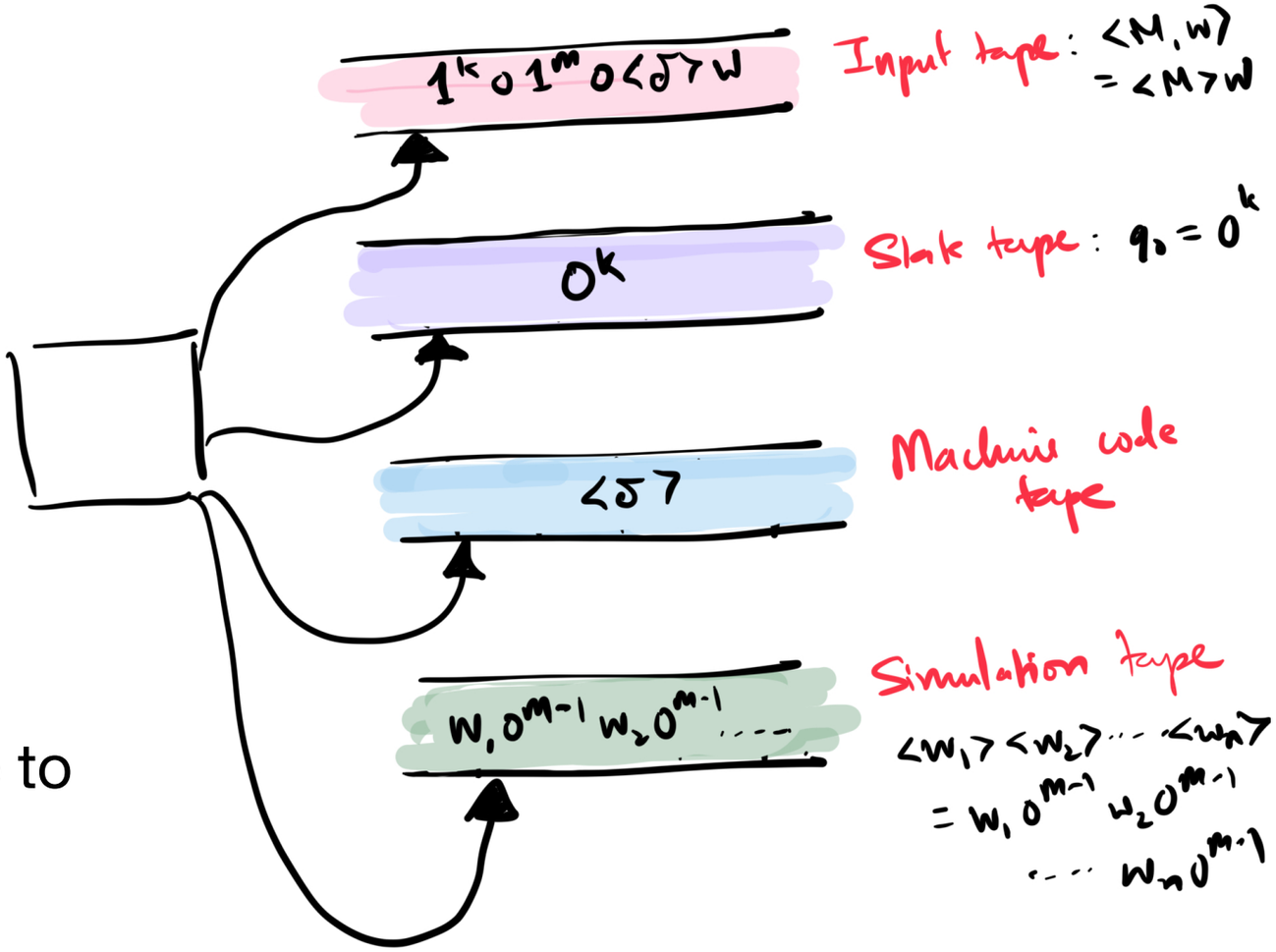
- Construct a multi-tape Turing Machine with four tapes:
  - Input Tape: Receives $\langle M, w \rangle = \langle M \rangle w$.
  - State Tape: Holds current state of M.
  - Machine Code Tape: Holds δ for M .
  - Simulation Tape: Encodes the contents of M's tape.



To simulate a step M, U looks up the matching transition in Machine Code Tape, and updates the Input Tape and Simulation Tape.

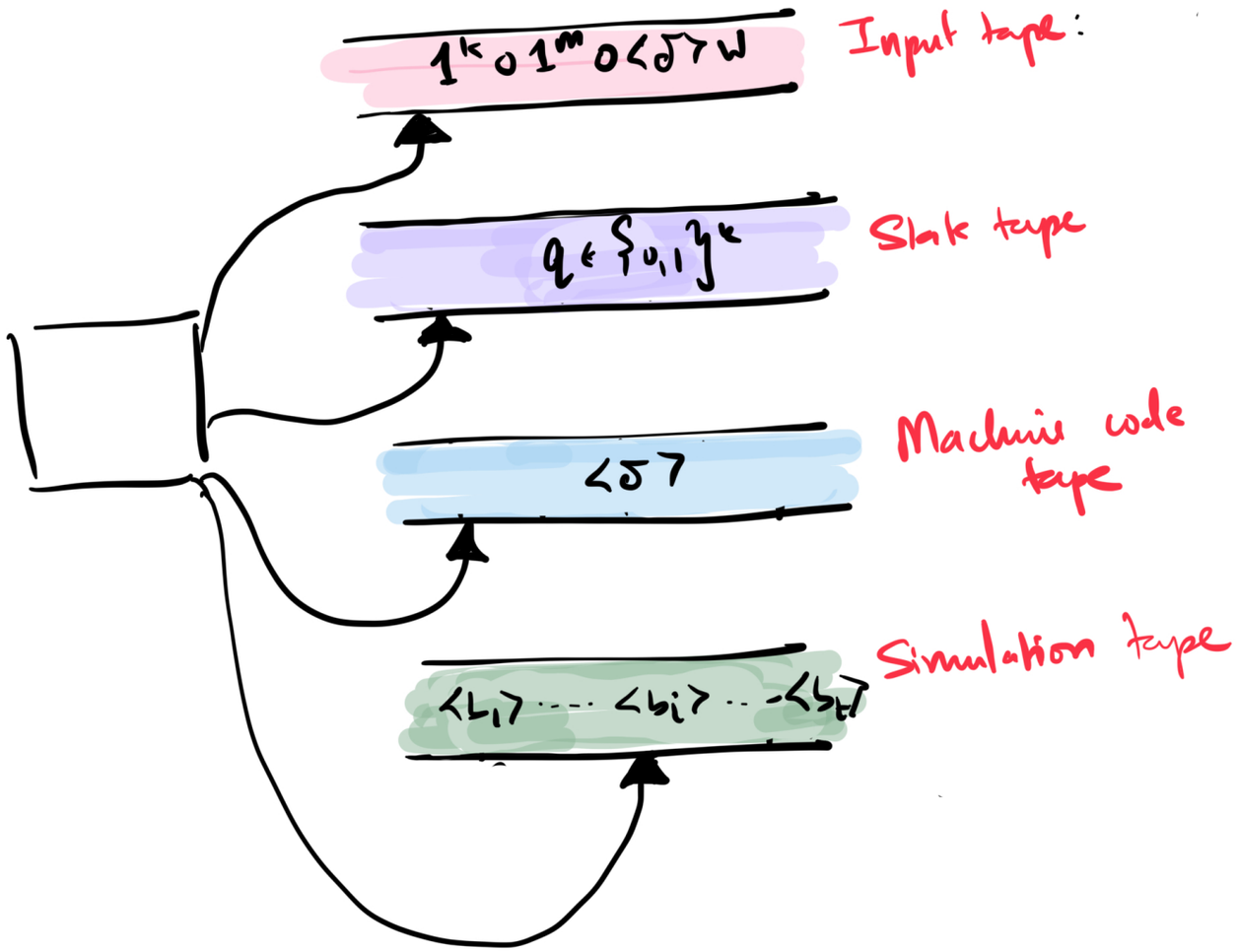
Initializing the Tape

- **Want:** U accepts $\langle M, w \rangle \iff M$ accepts w .
- Initialize the tape as follows:
  - Input Tape: Receives $1^k 0 1^m 0 \langle \delta \rangle w$.
  - State Tape: Initialize to $q_0 = 0^k$
  - Machine Code Tape: Read input tape to figure out $\langle \delta \rangle$.
  - Simulation Tape: Initially contains $\langle w_1 \rangle \langle w_2 \rangle \cdots \langle w_n \rangle$.



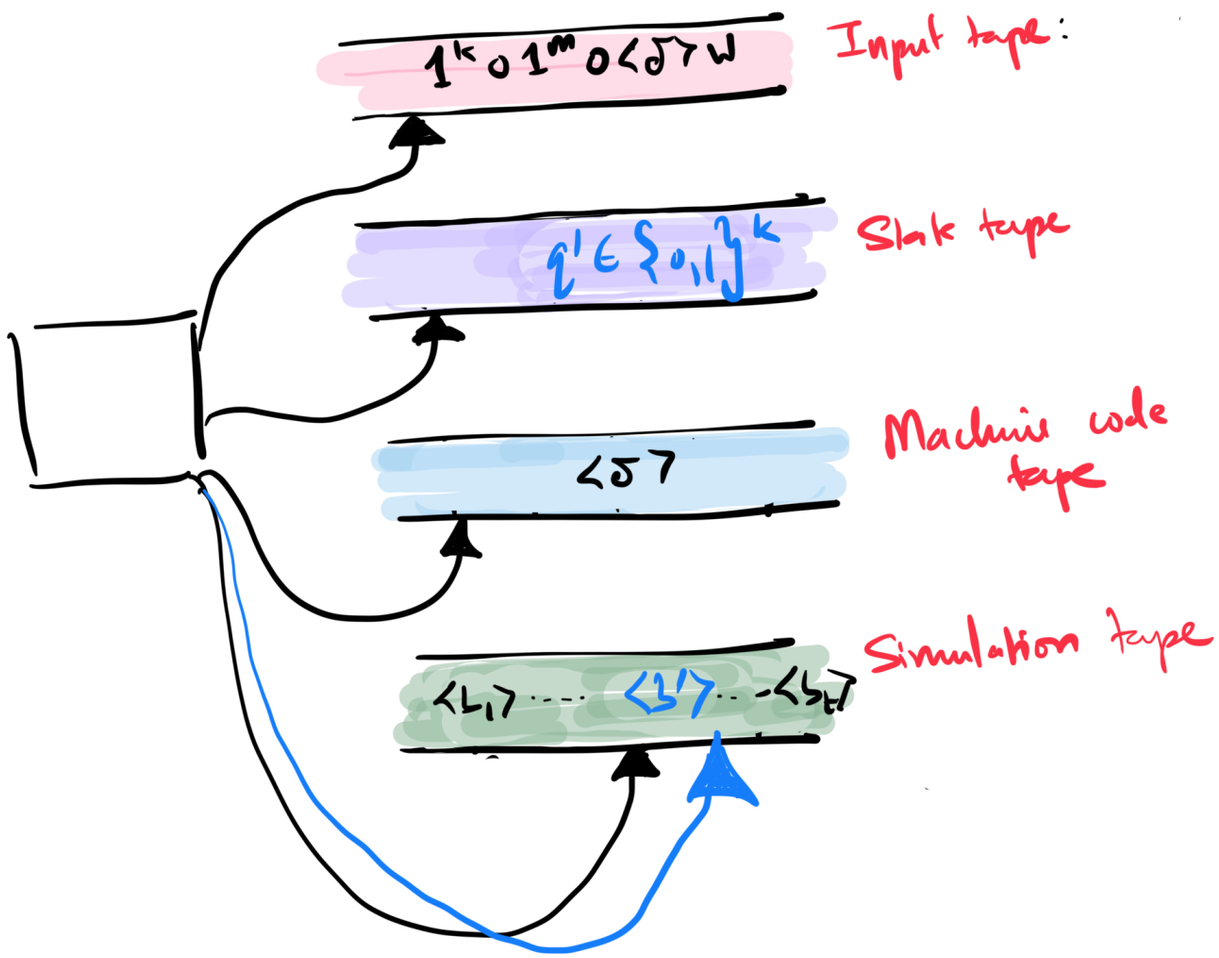
Running a Simulation

- Currently the status of U:
  - Input Tape: Contains $1^k 0 1^m 0 \langle \delta \rangle w$.
  - State Tape: Says M's state is $q \in \{0,1\}^k$
  - Machine Code Tape: Contains $\langle \delta \rangle$.
  - Simulation Tape: M's tape contains $\langle b_1 \rangle \langle b_2 \rangle \cdots \langle b_t \rangle$ and M's head is on $\langle b_i \rangle$



Running a Simulation

- Currently the status of U:
  - Input Tape: Contains $1^k 0 1^m 0 \langle \delta \rangle w$.
  - State Tape: Says M's state is $q \in \{0,1\}^k$
  - Machine Code Tape: Contains $\langle \delta \rangle$.
  - Simulation Tape: M's tape contains $\langle b_1 \rangle \langle b_2 \rangle \cdots \langle b_t \rangle$ and M's head is on $\langle b_i \rangle$.



- Find $\langle \delta(q,b) \rangle = \langle q', b, D \rangle$ in $\langle \delta \rangle$.
- Replace q by q' and $\langle b_i \rangle$ by $\langle b' \rangle$.
- Move the head of simulation tape m cells in D direction.

Putting Things Together

- What we just showed:
  - The Universal Turing Machine given $\langle M, w \rangle$ as input can simulate the initial configuration of M on w .
  - If it is simulating a configuration C_i of M on w, then it can derive $C_{i+1} = \text{NEXT}(C_i)$.
  - Proceeding inductively, we can derive the following theorem.

Theorem: There exists a Turing Machine U such that for every Turing Machine M and every input $w \in \{0,1\}^*$:

- If M accepts w, then U accepts $\langle M, w \rangle$.
- If M rejects w, then U rejects $\langle M, w \rangle$.
- If M loops on w, then U loops on $\langle M, w \rangle$.

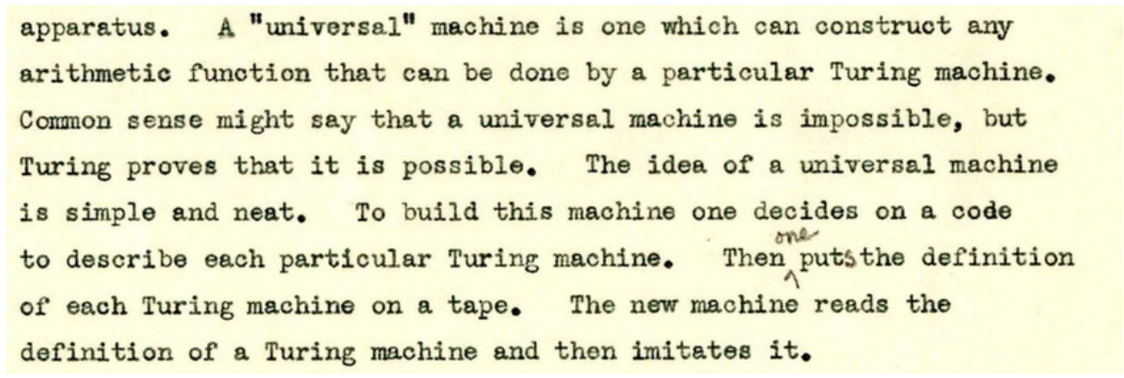
Interpretation of Universal Turing Machine

- This is a fundamental property of Turing Machines: There is a Turing Machine which can run arbitrary Turing Machine code.

- A Universal Turing Machine can be “programmed” to do any computationally possible task.

- The important idea is that the program is represented as data.

- Instead of constructing a new machine for every task, one keeps the hardware fixed and changes the instructions stored in memory. That is the logical ancestor of the general-purpose stored-program computer (such as EDVAC).


apparatus. A "universal" machine is one which can construct any arithmetic function that can be done by a particular Turing machine. Common sense might say that a universal machine is impossible, but Turing proves that it is possible. The idea of a universal machine is simple and neat. To build this machine one decides on a code to describe each particular Turing machine. Then puts one the definition of each Turing machine on a tape. The new machine reads the definition of a Turing machine and then imitates it.

Von Neumann’s Lectures on High Speed Computing (1945)

Summary

We can add other “features” to the basic Turing Machine model:

- The ability to observe two neighboring cells.
- **Multiple-head TM:** there are multiple heads operating independently on a single tape. Here,
  $\delta : Q \times \Sigma \times \Sigma \to Q \times \Sigma \times \{L, R\} \times \Sigma \times \{L, R\}$.

None of these changes have any affect on the model.

We can treat “program as data” and feed it to a Universal Turing Machine to simulate any Turing Machine.