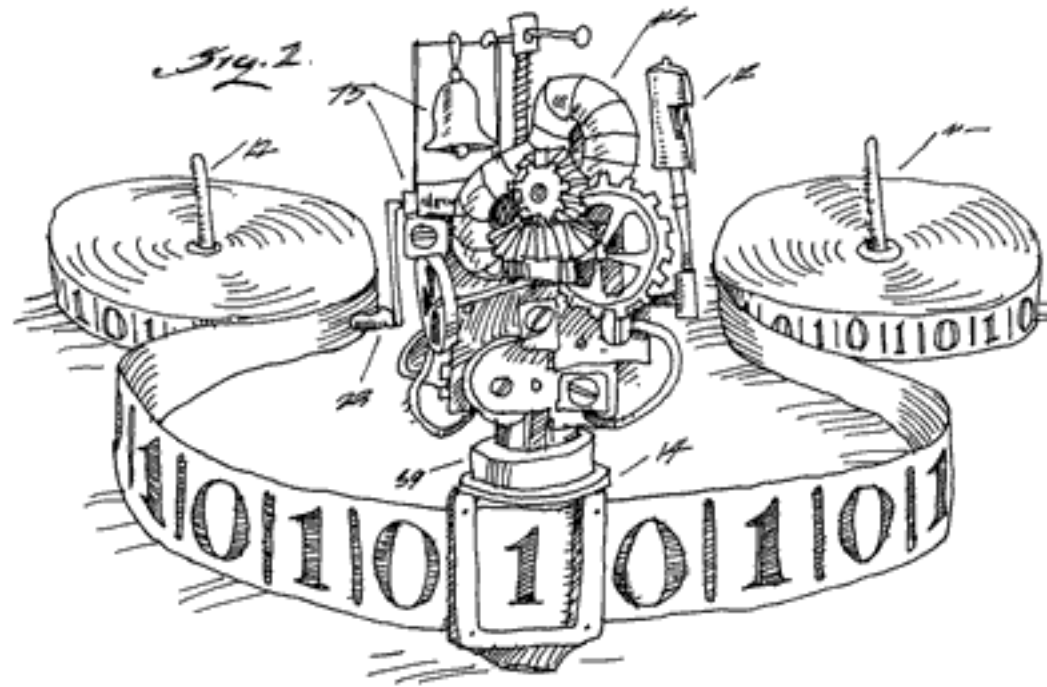


CSC4010: Introduction to Theoretical Computer Science



Lecture 5

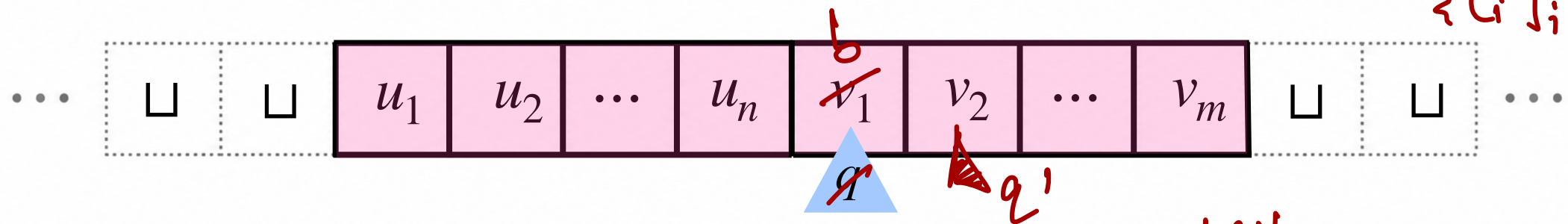
Administrative Things

- Quiz in Tutorial on 29/9/2026.
- First 20 minutes of class.
- 16 points
- True/False, MCQ, short answer type questions.
- Based on lectures 1-6

Recap

- Turing Machine is a 7-tuple $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$.
 Q → finite set of states.
 q_0 → initial state.
 δ → transition function.
 $\sqcup$ → tape alphabet.
 $q_{\text{accept}}, q_{\text{reject}}$ → halting states.
- A configuration uqv (where $u = u_1u_2 \cdots u_n \in \Sigma^*$, $v = v_1v_2 \cdots v_m \in \Sigma^*$ and $q \in Q$) is the current status of the Turing Machine.

Computational history: $C_1, C_2, \dots, C_i, \dots; \{C_i\}_{i \in \mathbb{N}}$.



$$\delta(q, v_1) = (q', b, R)$$

- Given C_i , $C_{i+1} = \text{NEXT}(C_i)$ is determined by δ .
- Computational history of a Turing machine on an input $w \in \{0,1\}^*$ is a series of configurations $C_1, C_2, \dots$ where $C_{i+1} = \text{NEXT}(C_i)$.
- Given a decidable language L , the language $\bar{L} = \{x \mid x \in \{0,1\}^* \setminus L\}$ is also decidable.

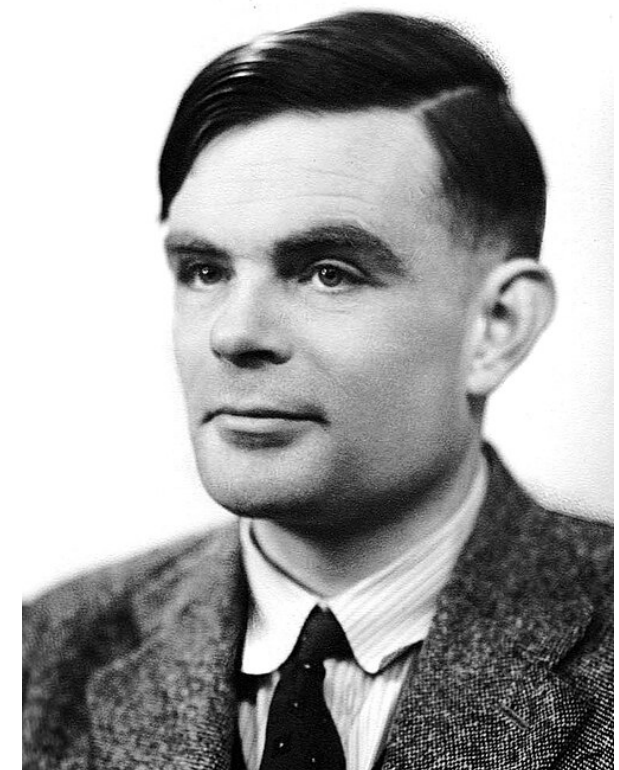
Church-Turing Thesis

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

Church-Turing Thesis: There exists an “algorithm” or a “procedure” for figuring out whether a $x \in Y$ **if and only** if there is a Turing Machine that decides Y .



Alonzo Church



Alan Turing

Church-Turing Thesis: What it says?

- The Church-Turing Thesis says:
 - The Turing Machine model is the “correct” way of modeling arbitrary computation.
 - The informal concept of an “algorithm” is successfully modeled by the rigorous definition of a Turing Machine.

Church-Turing Thesis: What it says?

- The Church-Turing Thesis says:
 - The Turing Machine model is the “correct” way of modeling arbitrary computation.
 - The informal concept of an “algorithm” is successfully modeled by the rigorous definition of a Turing Machine.

This is not a theorem - it is a falsifiable scientific hypothesis. And it has been thoroughly tested!

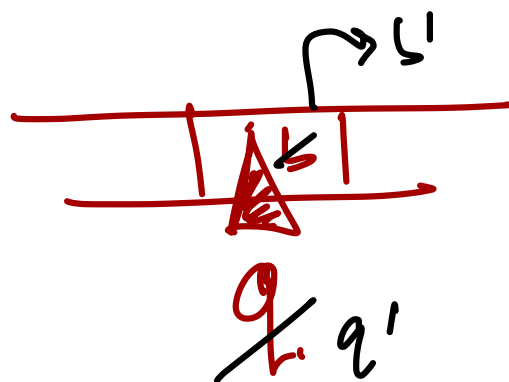
Can we verify the Church-Turing Thesis?

Left-Right-Stationary Turing Machine

Mathematically, a Left-Right-Stationary Turing Machine is a 7-tuple $M = (\cancel{Q}, \underline{q_{\text{accept}}}, \underline{q_{\text{reject}}}, q_0, \delta, \sqcup, \Sigma)$ such that:

$\hookrightarrow$ initial state.

- Q is a finite set (the set of “states”).
 - Σ is a finite set of symbols (“the tape alphabet”).
 - δ is a function $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R, S\}$. (The “transition function”).
 - S means the head will not move in this step.
- (can go left/right/stay put)



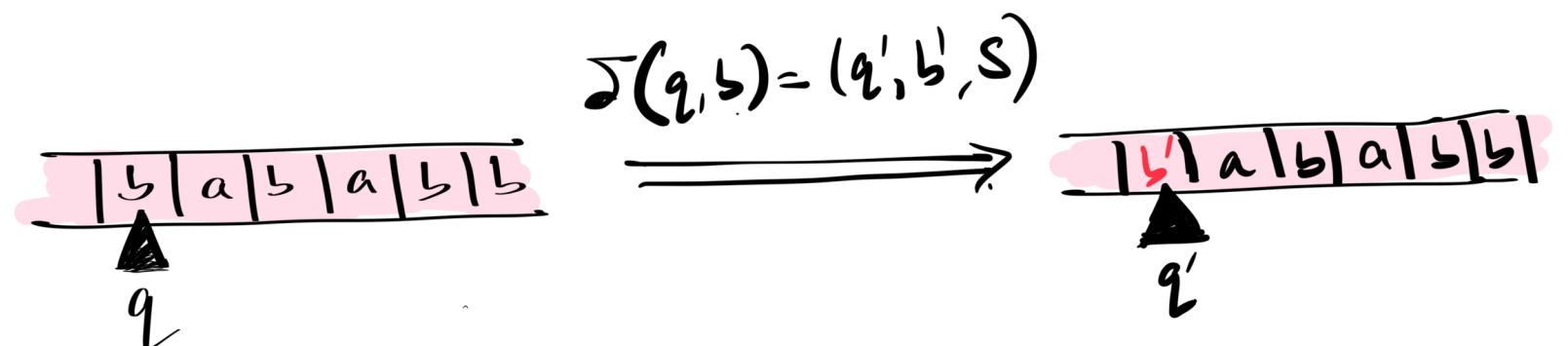
$$\delta(q, \sqcup) = (q', \sqcup, S)$$

$$\delta(q', \sqcup) = \underline{\hspace{2cm}}$$

Left-Right-Stationary Turing Machine

Mathematically, a **Left-Right-Stationary Turing Machine** is a 7-tuple $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$ such that:

- Q is a finite set (the set of “states”).
- Σ is a finite set of symbols (“the tape alphabet”).
- δ is a function $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R, S\}$. (The “transition function”).
- S means the head will not move in this step.



Left-Right-Stationary TM

- The model is **still realistic**, even though we added an extra feature.
- Is it a counterexample to the Church-Turing thesis?
- No!
- Let's prove the Left-Right-Stationary Turing Machine Model is equivalent to the original Turing Machine model.

Proof strategy: Let L be an arbitrary language for which there is a L-R-S TM that decides it, M_L . Then, $\exists$ a TM M'_L that decides L .

Left-Right-Stationary Turing Machine

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

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L .

1) Have a TM M'_2 , can we make M_2 ?
2) M_2 to M'_2

Left-Right-Stationary Turing Machine

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

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L .

ordinary me. $\uparrow$ modify δ' to get δ for M_L as follows

$\delta: Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R, S\}$
 $\delta(a, b) = \delta'(a, b)$

Proof: ($\Leftarrow$) M'_L is an L-R-S Turing Machine that does not use the “stationary function”.

Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L .

Left-Right-Stationery Turing Machine

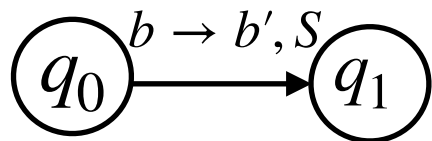
Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L .

Proof: ($\Rightarrow$) A proof idea: Simulate S by L followed by an R.

Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L .

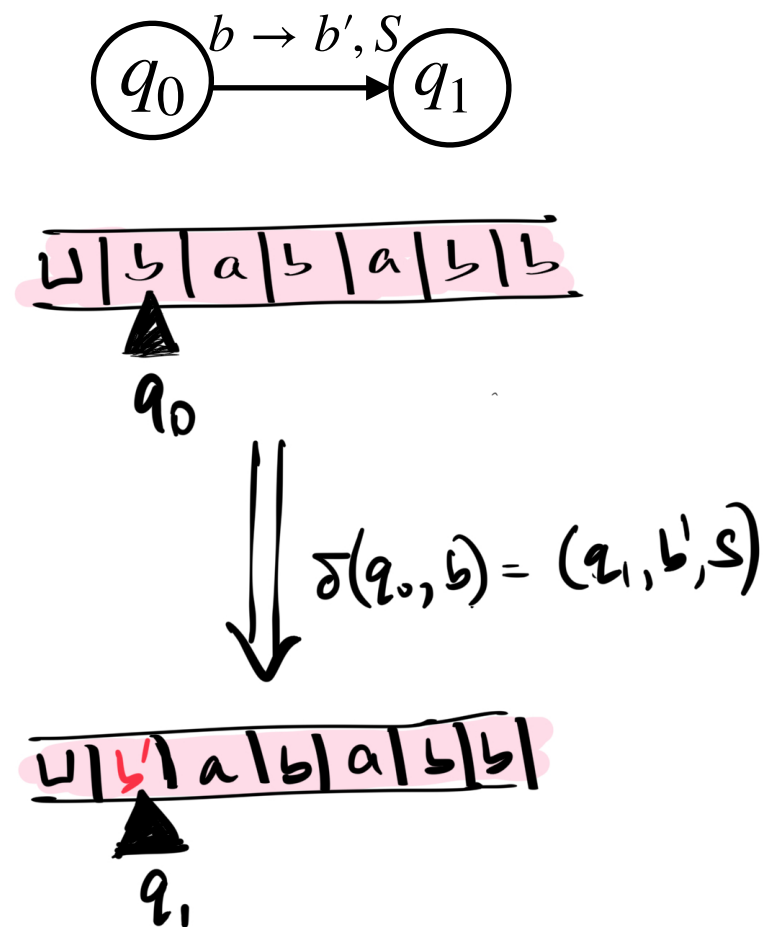
Proof: ($\Rightarrow$) A proof idea: Simulate S by L followed by an R.



Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L .

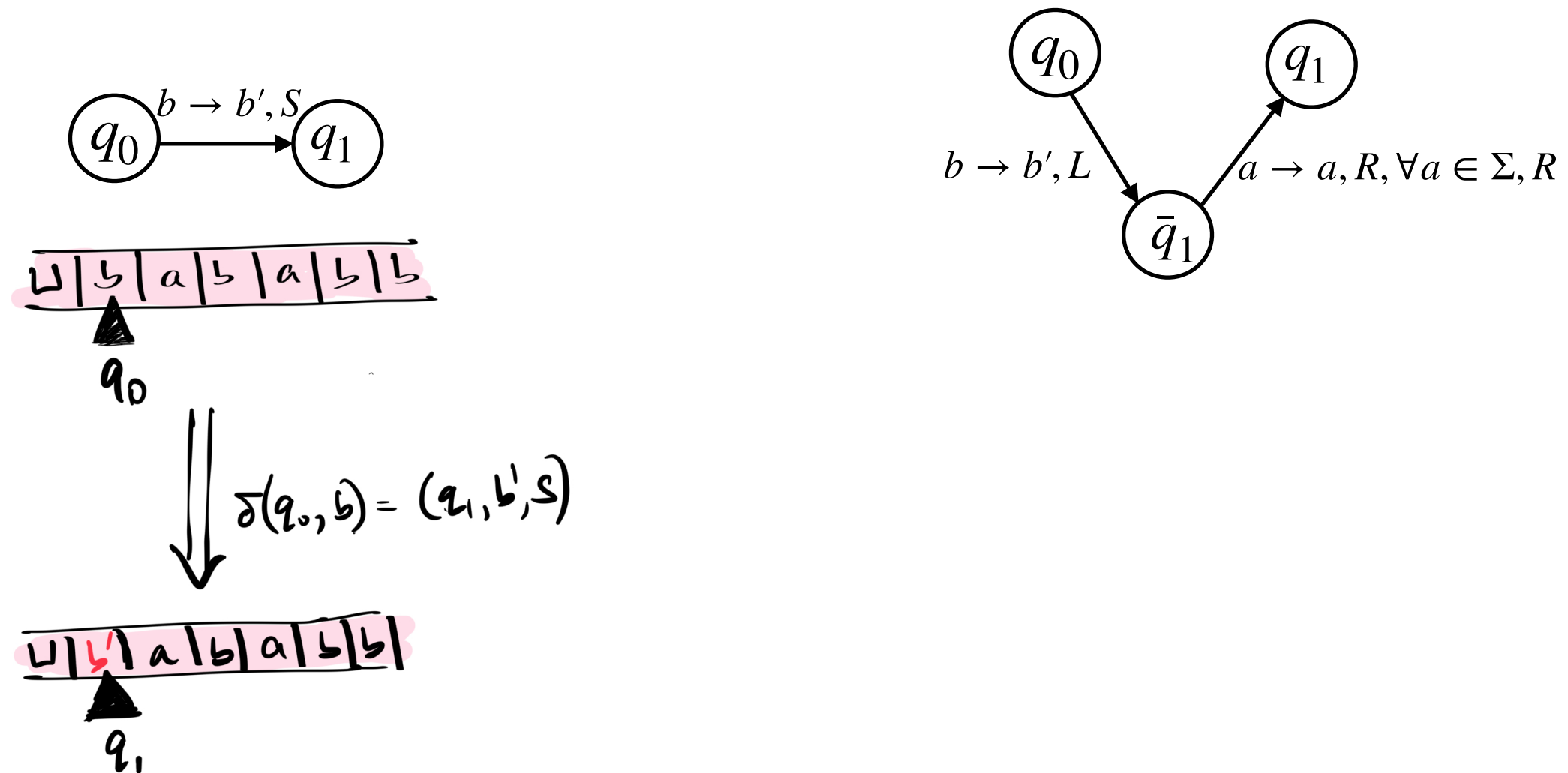
Proof: ($\Rightarrow$) A proof idea: Simulate S by L followed by an R.



Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L .

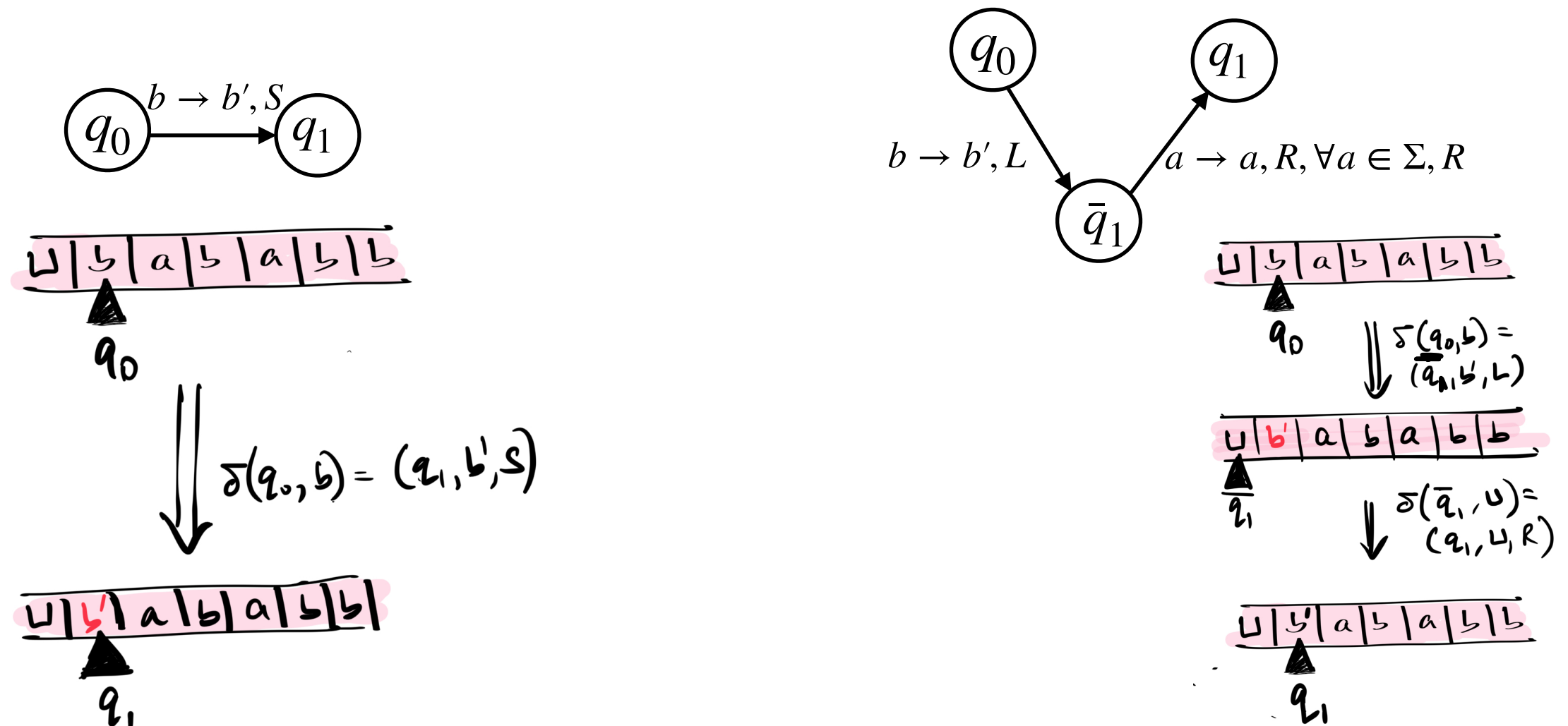
Proof: ($\Rightarrow$) A proof idea: Simulate S by L followed by an R.



Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L .

Proof: ($\Rightarrow$) A proof idea: Simulate S by L followed by an R.



Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L .

Proof: ($\Rightarrow$) Suppose $M_L = (Q, \Sigma, \delta, q_{acc}, q_{rej}, q_0, \sqcup)$ is an L-R-S Turing Machine deciding L . Then, $M'_L = (Q', \Sigma', \delta', q_{acc}, q_{rej}, q_0, \sqcup)$?

$$a) Q' = Q \cup \{\bar{q} \mid q \in Q\}$$

$$b) \Sigma' = \Sigma$$

$$c) \delta' :$$

$$\delta(q, b) = (q', b', D) \quad D \in \{L, R\}.$$

$$\delta'(q, b) = (q', b', D) \text{ as well.}$$

$$= \begin{array}{l} \delta(q, b) = (q', b', S) \\ \delta'(q, b) = (\bar{q}', b', L) \end{array} \quad \left| \quad \begin{array}{l} \forall \bar{q} \in Q', \text{ and } b \in \Sigma, \\ \delta'(\bar{q}, b) = (q, b, R) \end{array} \right.$$

Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L .

Proof: ($\Rightarrow$) Suppose $M_L = (Q, \Sigma, \delta, q_{\text{acc}}, q_{\text{rej}}, q_0, \sqcup)$ is an L-R-S Turing Machine deciding L . Then, $M'_L = (Q', \Sigma, \delta', q_{\text{acc}}, q_{\text{rej}}, q_0, \sqcup)$?

Check rigorously that M'_L decides L .

Turing Machine with Reset

Turing Machine with Reset

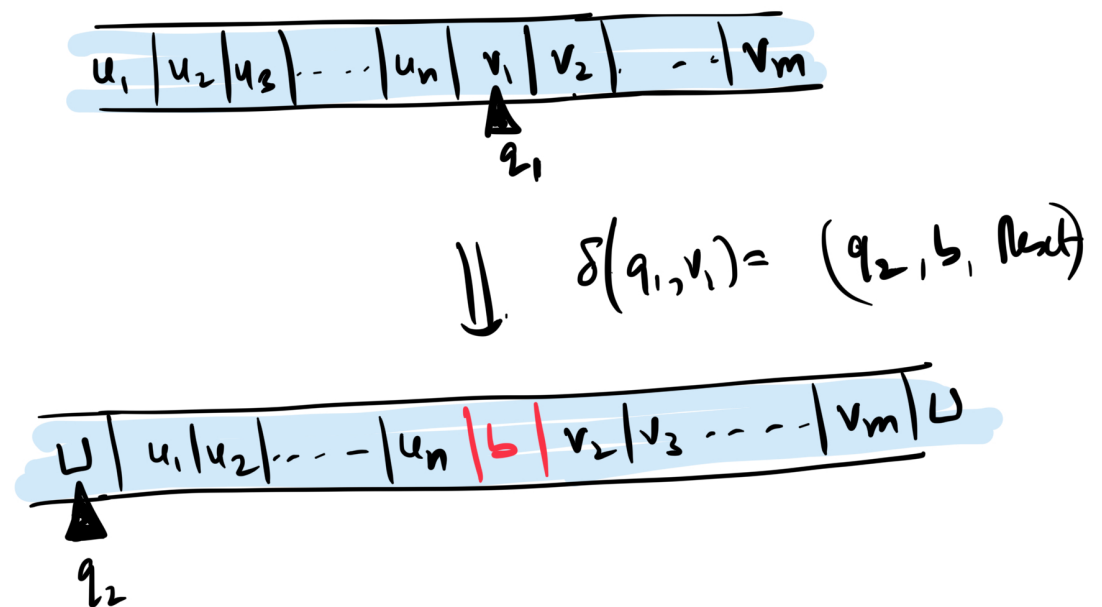
Mathematically, a Turing Machine with Reset is a 7-tuple $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$ such that:

- Q is a finite set (the set of “states”).)
- Σ is a finite set of symbols (“the tape alphabet”).
- δ is a function
 $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R, \text{Reset}\}$. (The “transition function”).)
- This implies: if $C_i = uq_1v$ and $\delta(q_1, v_1) = (b, q_2, \text{Reset})$, then $\text{NEXT}(C_i) = q_2 \sqcup ubv_2v_3 \cdots v_m$.

Turing Machine with Reset

Mathematically, a Turing Machine with Reset is a 7-tuple $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$ such that:

- Q is a finite set (the set of “states”).
- Σ is a finite set of symbols (“the tape alphabet”).
- δ is a function
 $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R, \text{Reset}\}$. (The “transition function”).
- This implies: if $C_i = uq_1v$ and $\delta(q_1, v_1) = (b, q_2, \text{Reset})$, then $\text{NEXT}(C_i) = q_2 \sqcup ubv_2v_3 \cdots v_m$.



Turing Machine with Reset

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

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

Turing Machine with Reset

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

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

Exercise: define the transition function, so that one can formally prove that it can be simulated by a TM w/out reset.

Proof: ($\Leftarrow$) M'_L is a Turing Machine with Reset that does not use the “Reset function”.

Turing Machine with Reset

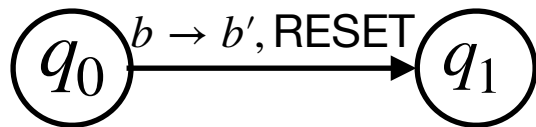
Theorem: There exists a Turing Machine with Reset M_L deciding L if and only if there exists a Turing Machine M'_L deciding L .

Proof: ($\Leftrightarrow$) A proof idea: Simulate RESET by going to a separate state and rewinding.

Turing Machine with Reset

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

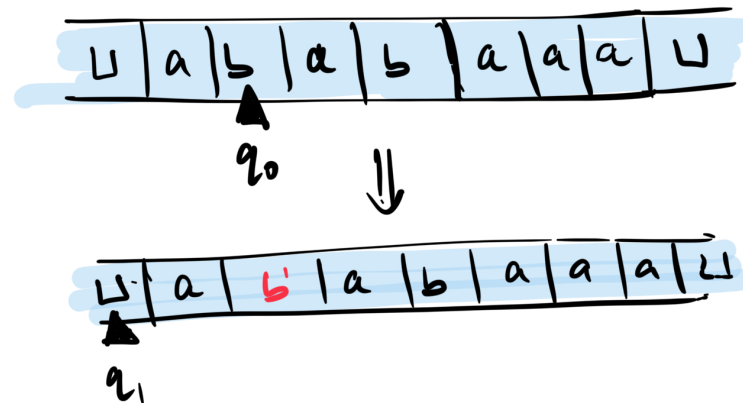
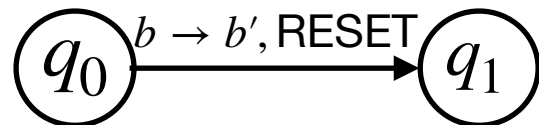
Proof: ($\Leftarrow$) A proof idea: Simulate RESET by going to a separate state and rewinding.



Turing Machine with Reset

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

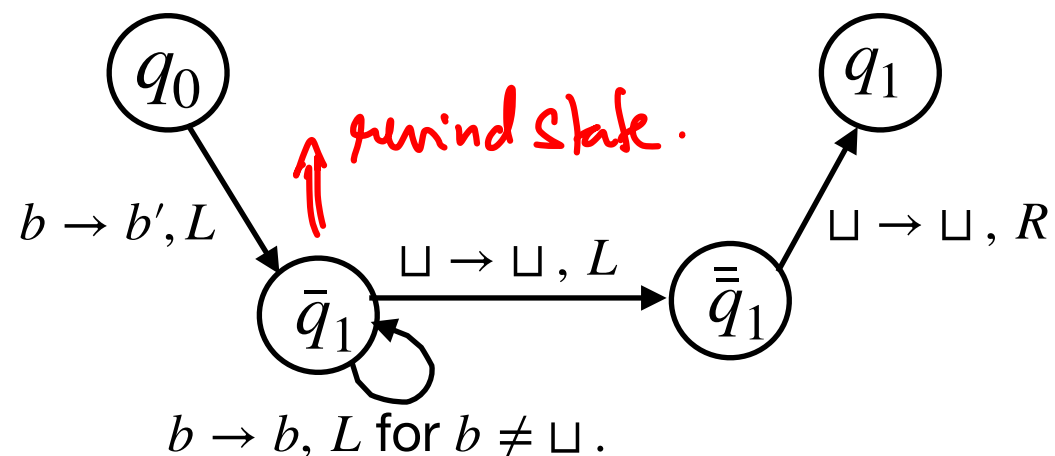
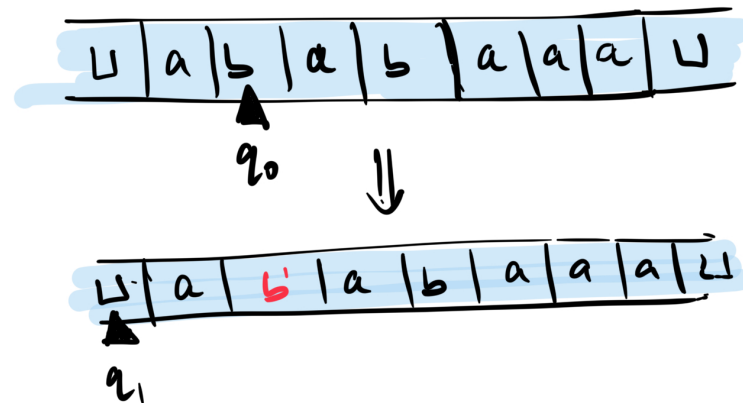
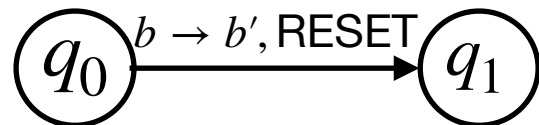
Proof: ($\Leftarrow$) A proof idea: Simulate RESET by going to a separate state and rewinding.



Turing Machine with Reset

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

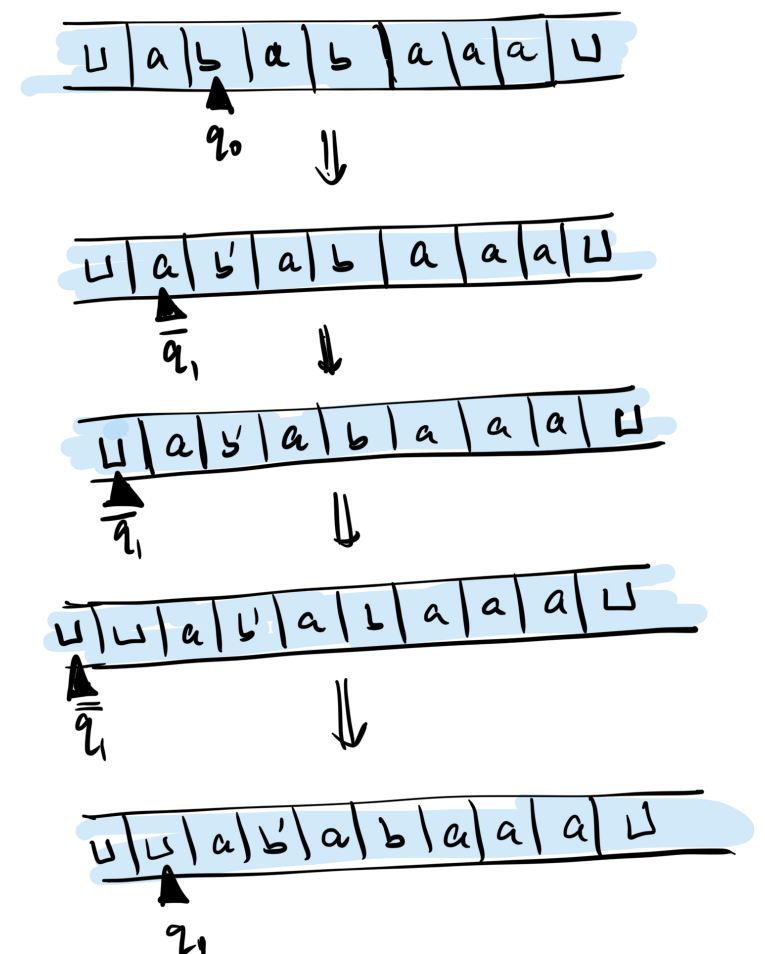
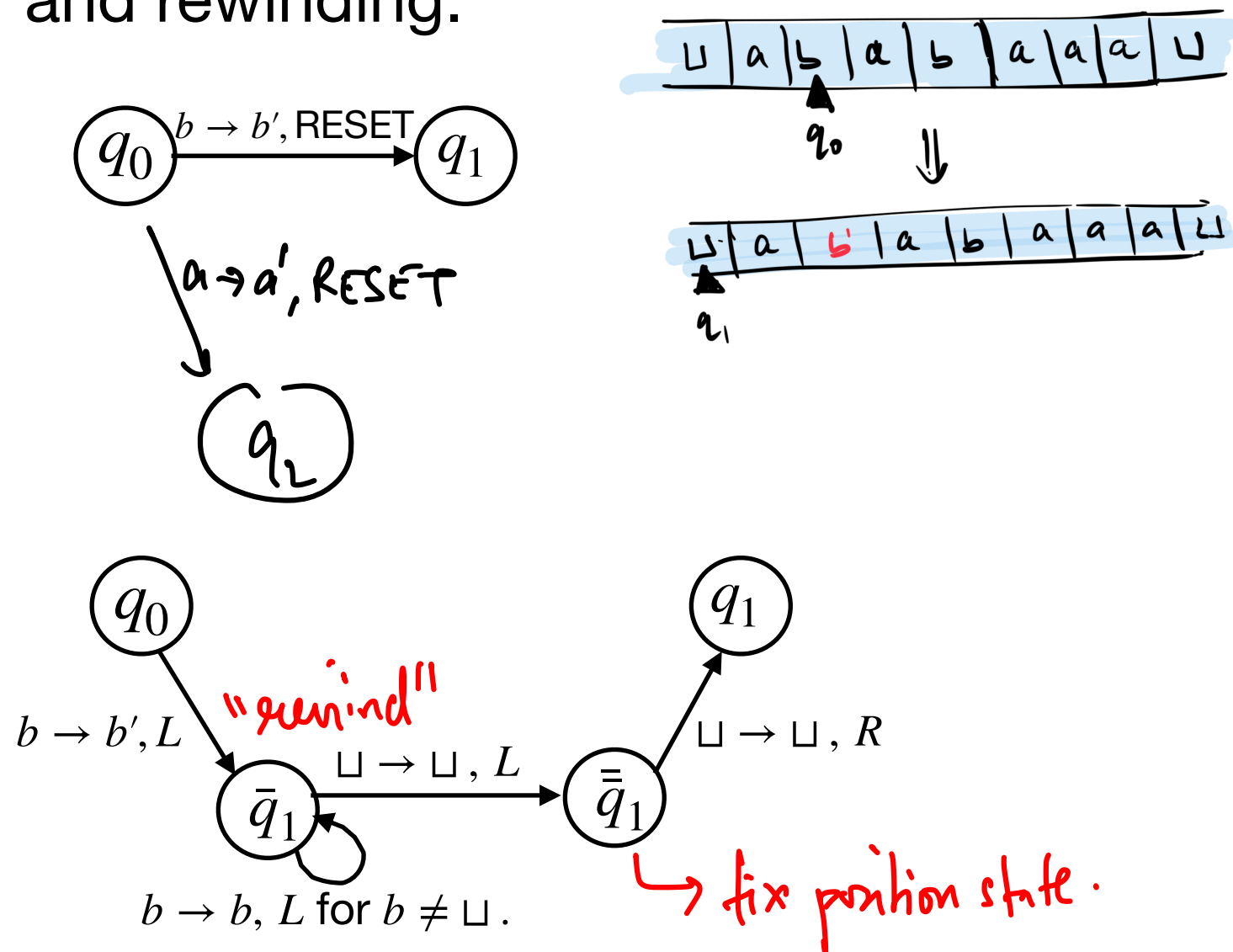
Proof: ($\Leftarrow$) A proof idea: Simulate RESET by going to a separate state and rewinding.



Turing Machine with Reset

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

Proof: ($\Leftarrow$) A proof idea: Simulate RESET by going to a separate state and rewinding.



Turing Machine with Reset

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

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

$$(1) Q' = Q \cup \underbrace{\{\bar{q} \mid q \in Q\}}_{\text{rewind states}} \cup \underbrace{\{\bar{\bar{q}} \mid q \in Q\}}_{\text{fix position.}}$$

(2) δ' :

$$a) \delta(q, b) = (q', b', D) \text{ where } D \in \{L, R\} \\ \delta'(q, b) = (q', b', D')$$

$$(b) \delta(q, b) = (q', b', \text{reset}) \\ \delta'(q, b) = (\bar{q}', b', L)$$

$$(c) \forall \bar{q} \in Q, \forall b \neq \sqcup \quad \delta'(\bar{q}, b) = \delta(\bar{q}, b, L) \quad \left. \vphantom{\delta'(\bar{q}, b) = \delta(\bar{q}, b, L)} \right\} \text{rewind transitions}$$

$$(d) \forall \bar{q} \in Q, \delta'(\bar{q}, \sqcup) = (\bar{\bar{q}}, \sqcup, L)$$

$$(e) \forall \bar{\bar{q}} \in Q, \delta'(\bar{\bar{q}}, \sqcup) = (q, \sqcup, R)$$

Turing Machine with Reset

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

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

Check rigorously that M'_L decides L .

Multi-Tape Turing Machine

2-Tape Turing Machine

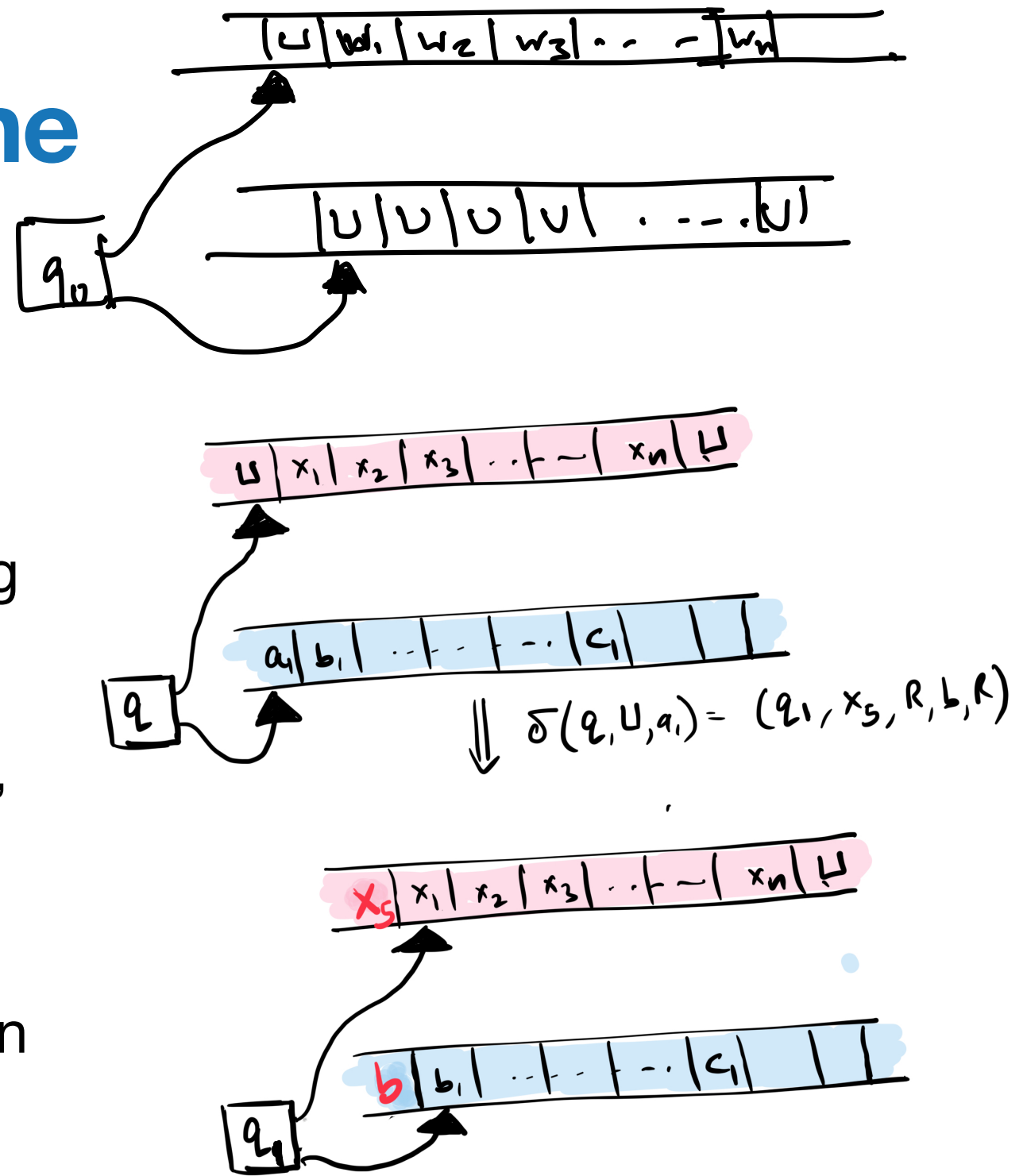
- A two-tape Turing Machine is like an ordinary Turing Machine with two tapes.
- Each tape has its own head for reading and writing.
- Initially, the input appears on one tape, and the second tape is blank.
- The transition function allows for reading, writing, and moving the head on all the tapes simultaneously:

symbol being read by the two heads.

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

current state

$$\delta : Q \times \Sigma^k \rightarrow Q \times \Sigma^k \times \{L, R\}^k$$



k-tape TM.

(i) k tapes, k heads which can read/write their own tapes.

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 .

Exercise: modify δ' to make it look like a two-tape TM.



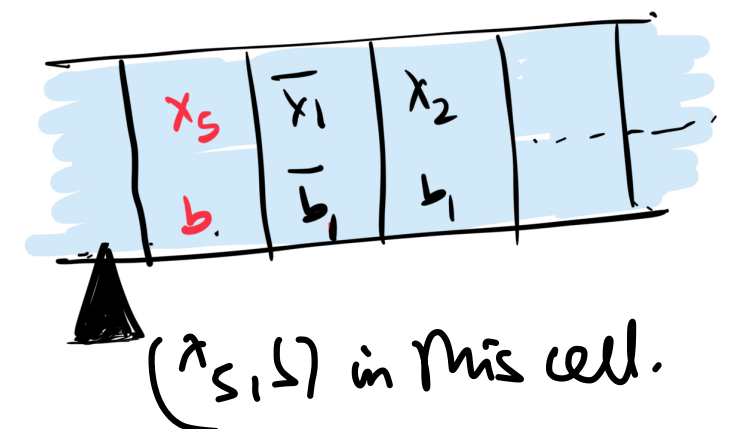
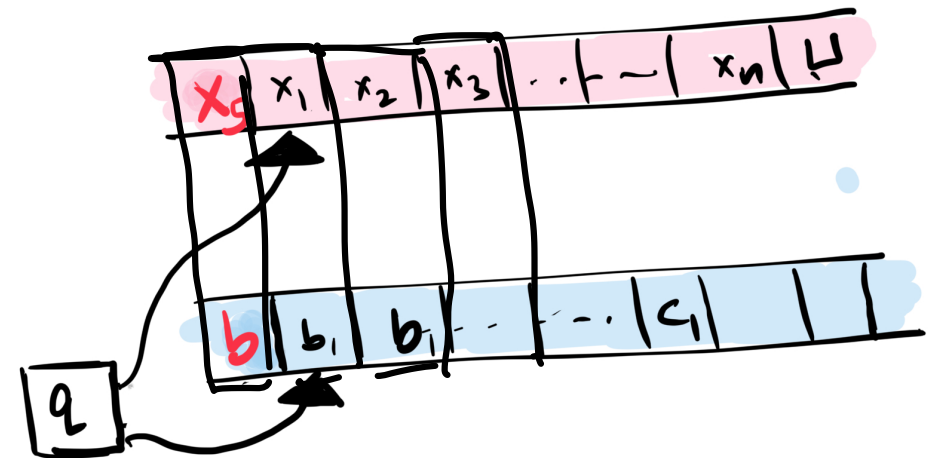
Proof: ($\Leftarrow$) M'_L is a 2 Tape Turing Machine that does not use the second tape.

2 Tape Turing Machine .

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

Idea:

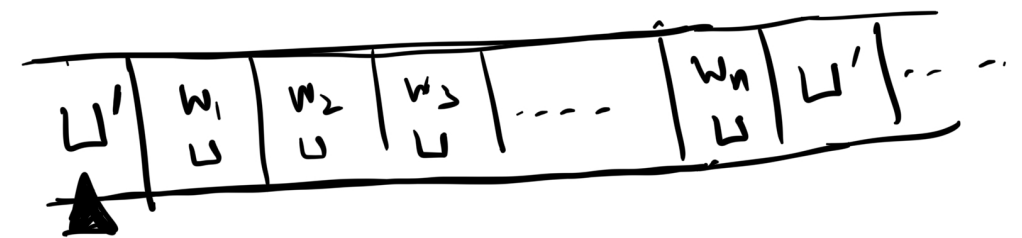
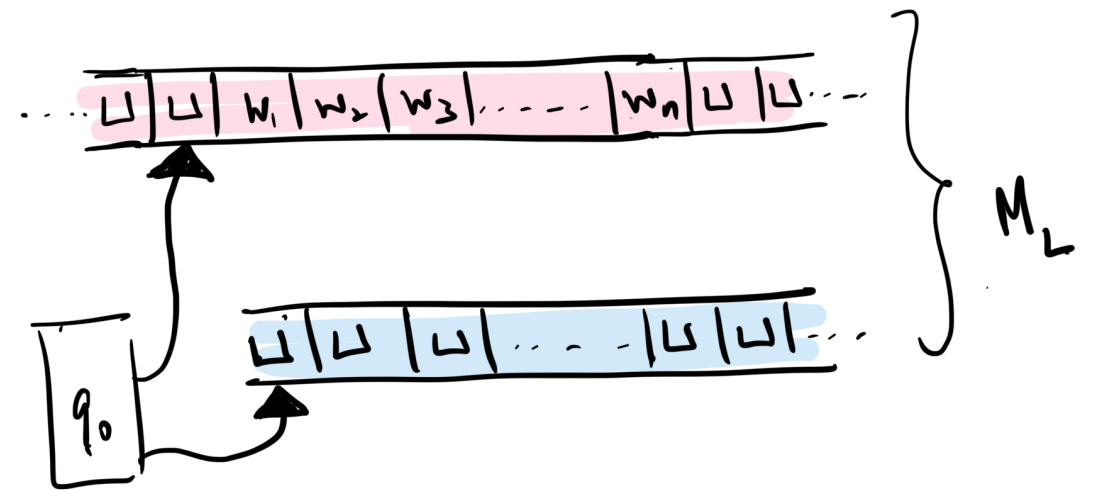
- Divide the single tape into **two tracks** by adding **two symbols** in a single cell. Achieved by **expanding the alphabet**. $\Leftarrow^2$
- To simulate two heads, use different symbols to denote the presence of “virtual heads”. $\Sigma = \{\bar{a} \mid a \in \Sigma\}$
- The real head goes back and forth, updating the position of the virtual heads.



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

Simulating 2-Tape TM: Tape Alphabet

- To indicate presence of simulated head, define $\underline{\Sigma} = \{\underline{b} \mid b \in \Sigma\}$. A letter $\underline{b}$ 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 $(w_1, \sqcup), (w_2, \sqcup), \cdots, (w_n, \sqcup)$.



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

Simulating 2-Tape TM: Transition and States

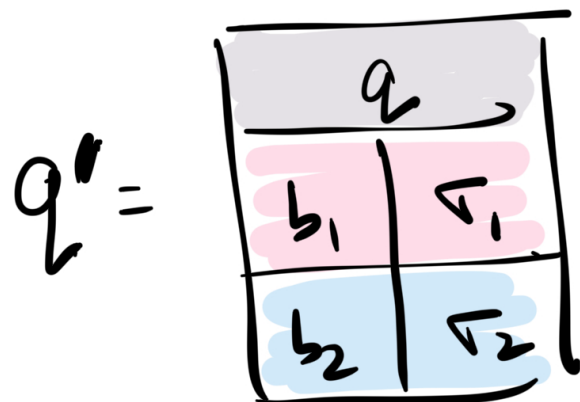
- Scan the tape using the **real head** to find the **underlined symbols** and **remember these symbols that are underlined**.
- (Because there are only a finite number of possibilities, the machine can ‘remember’ these symbols via a finite number of states, without writing anything to the tape.)

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

Simulating 2-Tape TM: Transition and States

- Scan the tape using the **real head** to find the **underlined symbols** and **remember these symbols that are underlined**.
- (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.)

An Example of a state in M'



$$\begin{aligned}
 q &\in Q \\
 b_1, b_2 &\in \Sigma \cup \{?\} \\
 \sigma_1, \sigma_2 &\in (\Sigma \times \{L, R, S\}) \cup \{?\}
 \end{aligned}$$

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

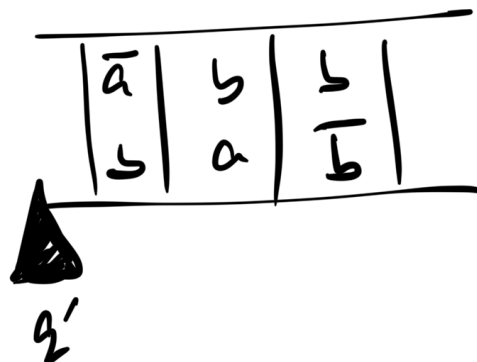
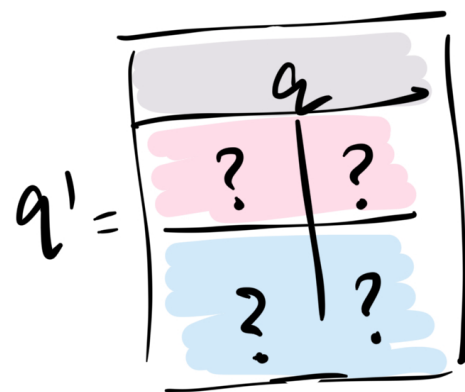
Simulating 2-Tape TM: Transition and States

- Scan the tape using the **real head** to find the **underlined symbols** and **remember these symbols that are underlined**.
- (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.)

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

Simulating 2-Tape TM: Transition and States

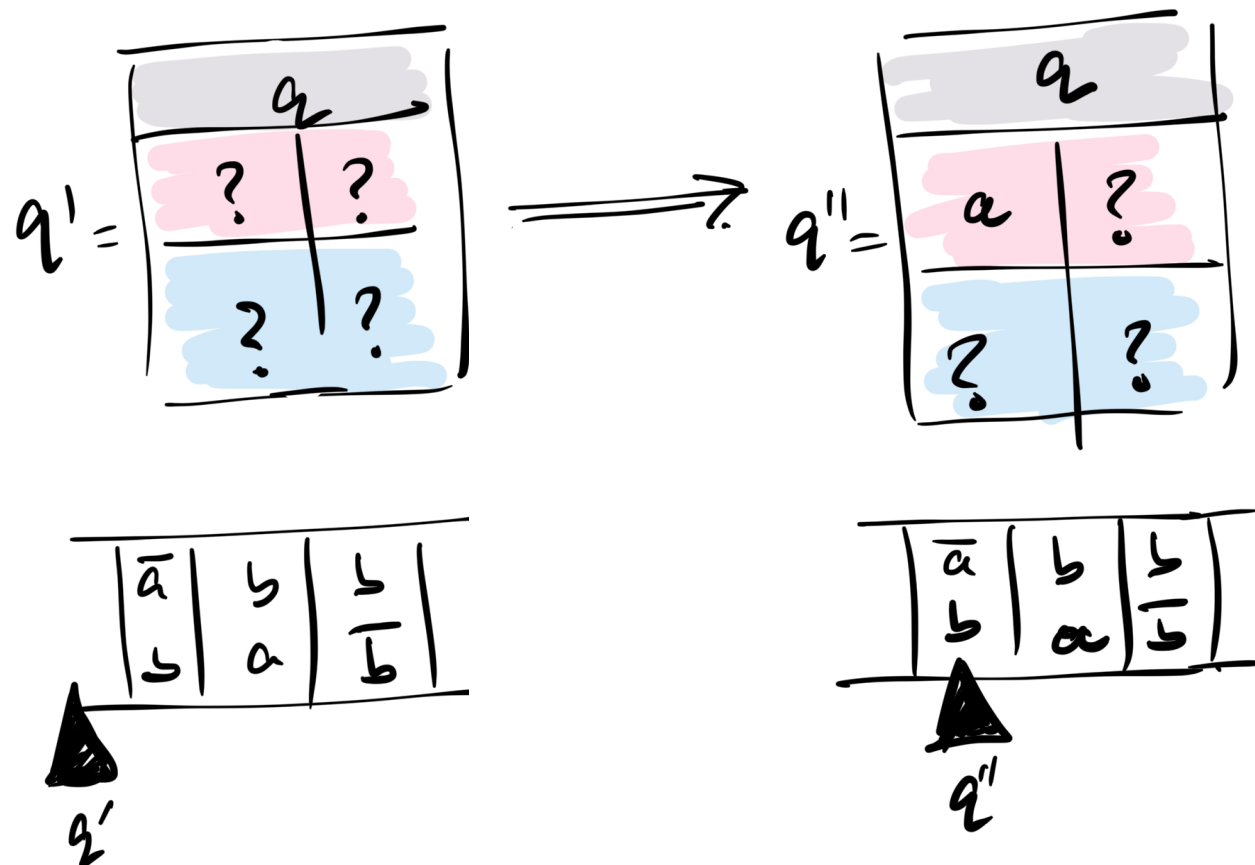
- Scan the tape using the **real head** to find the **underlined symbols** and **remember these symbols that are underlined**.
- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)



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

Simulating 2-Tape TM: Transition and States

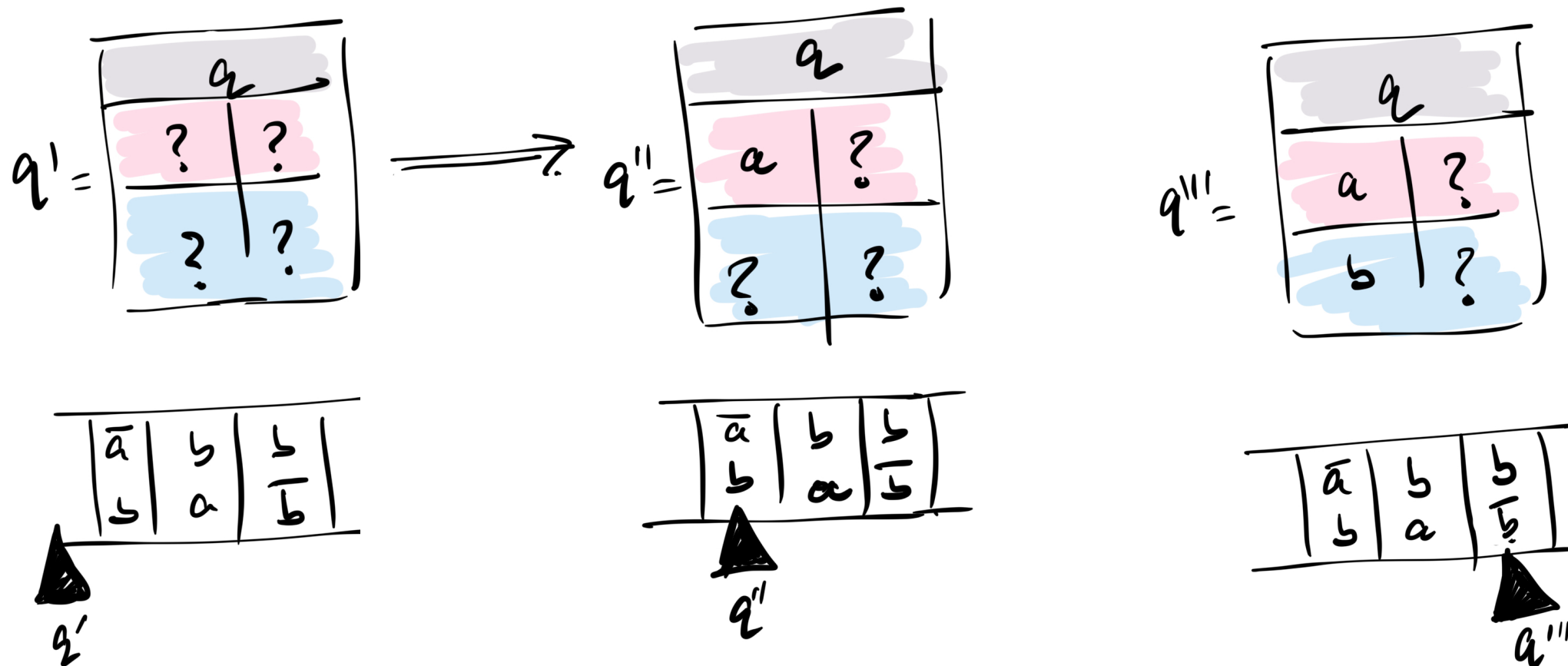
- Scan the tape using the **real head** to find the **underlined symbols** and **remember these symbols that are underlined**.
- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)



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

Simulating 2-Tape TM: Transition and States

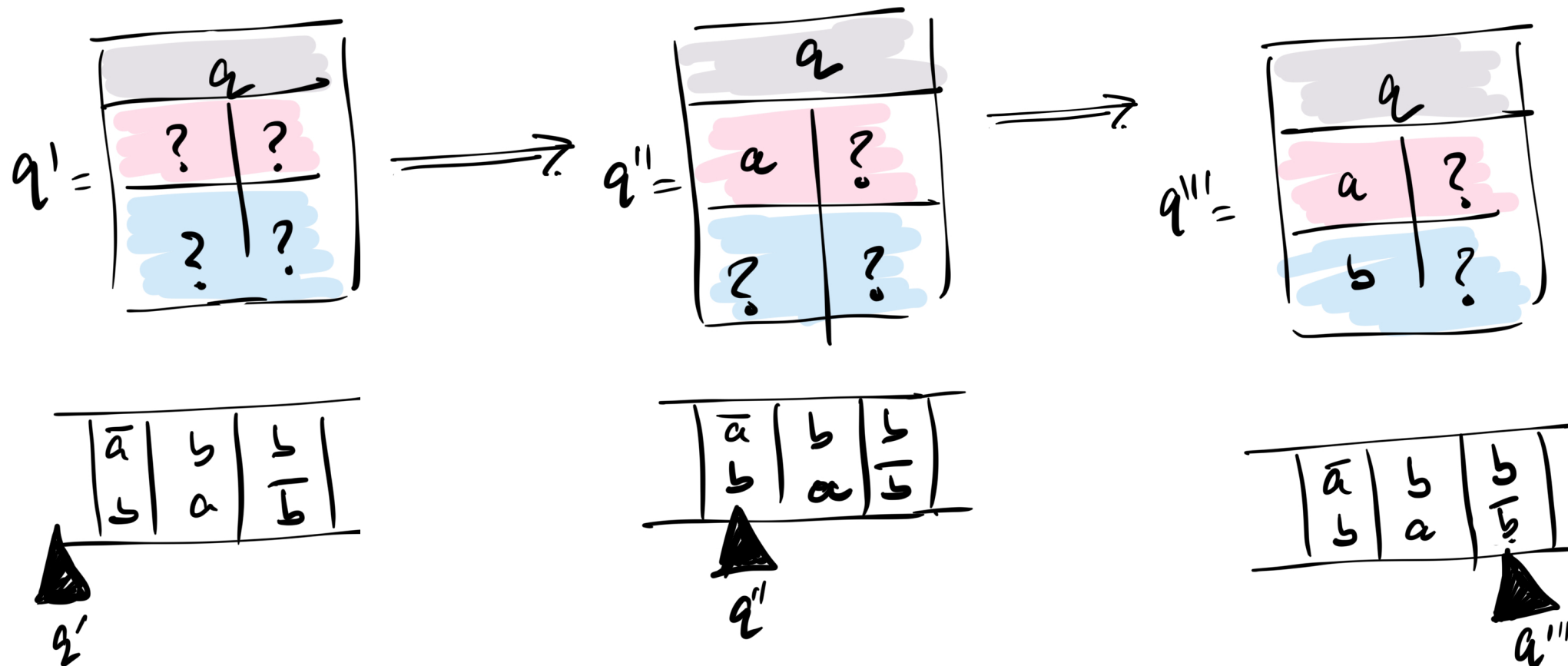
- Scan the tape using the **real head** to find the **underlined symbols** and **remember these symbols that are underlined**.
- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)



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

Simulating 2-Tape TM: Transition and States

- Scan the tape using the **real head** to find the **underlined symbols** and **remember these symbols that are underlined**.
- (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.)



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

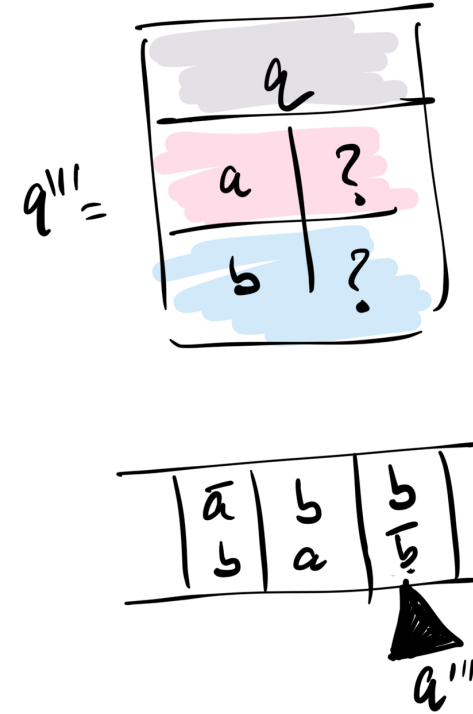
Simulating 2-Tape TM: Transition and States

- 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 .

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

Simulating 2-Tape TM: Transition and States

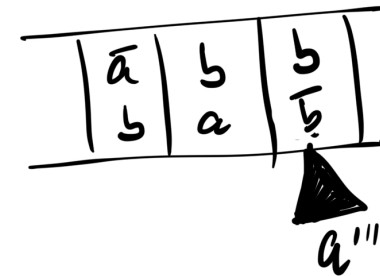
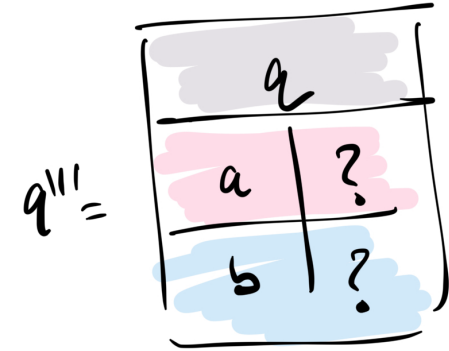
- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).
- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ .
- Move the virtual heads to the right locations as specified by the transition function of M .



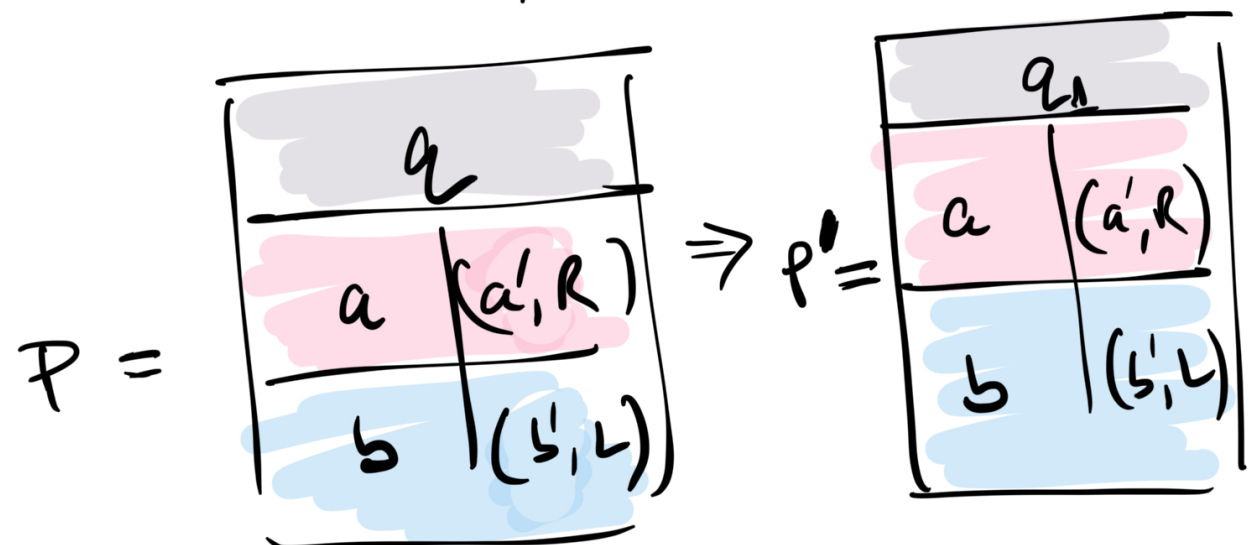
$$M'_L = (Q', \check{\Sigma}', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \check{\sqcup}')?$$

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 .



If $\delta(q, a, b) = (q_1, a', r, b', L)$
 then,



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

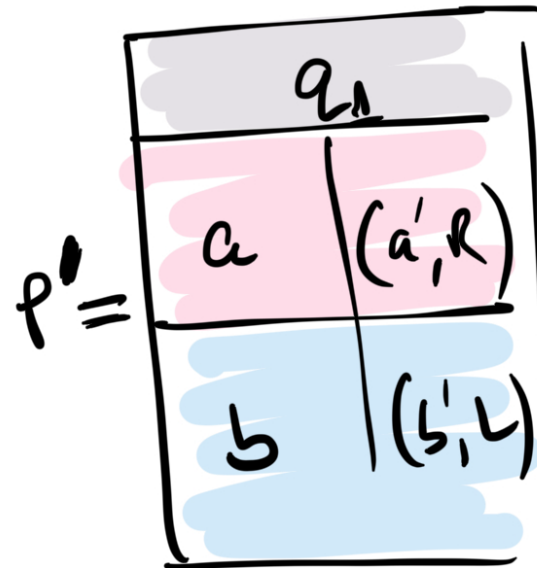
Simulating 2-Tape TM: Transition and States

- 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 .

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

Simulating 2-Tape TM: Transition and States

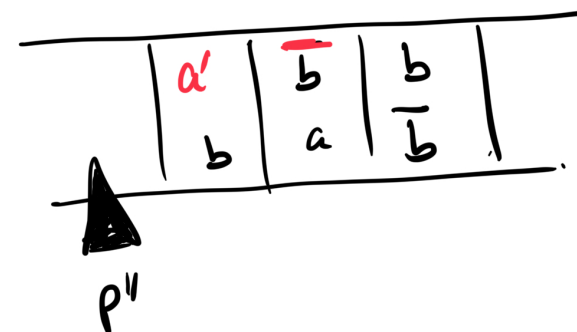
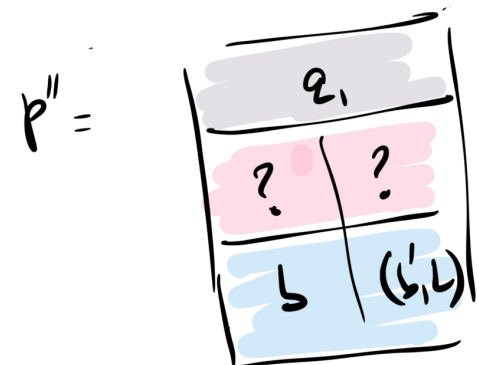
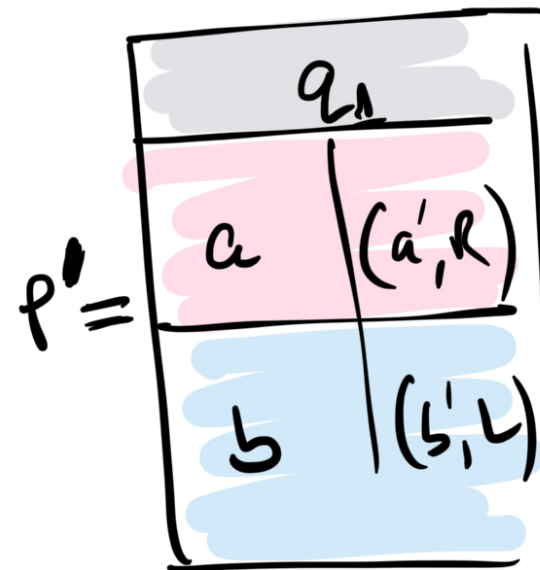
- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).
- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ .
- Move the virtual heads to the right locations as specified by the transition function of M .



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

Simulating 2-Tape TM: Transition and States

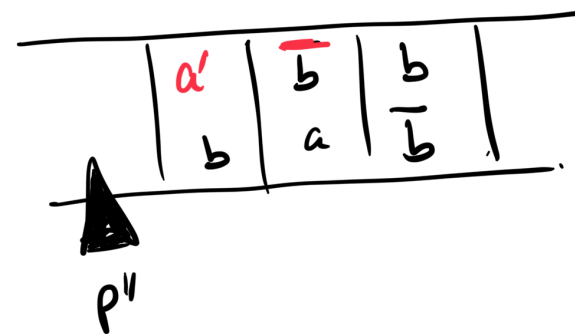
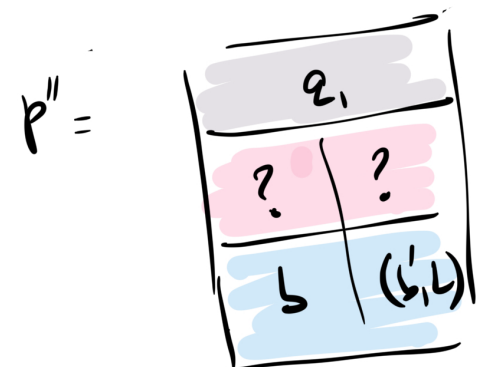
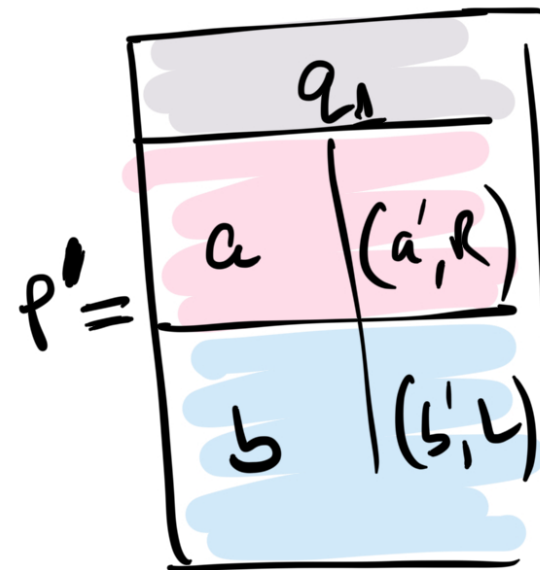
- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).
- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ .
- Move the virtual heads to the right locations as specified by the transition function of M .



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

Simulating 2-Tape TM: Transition and States

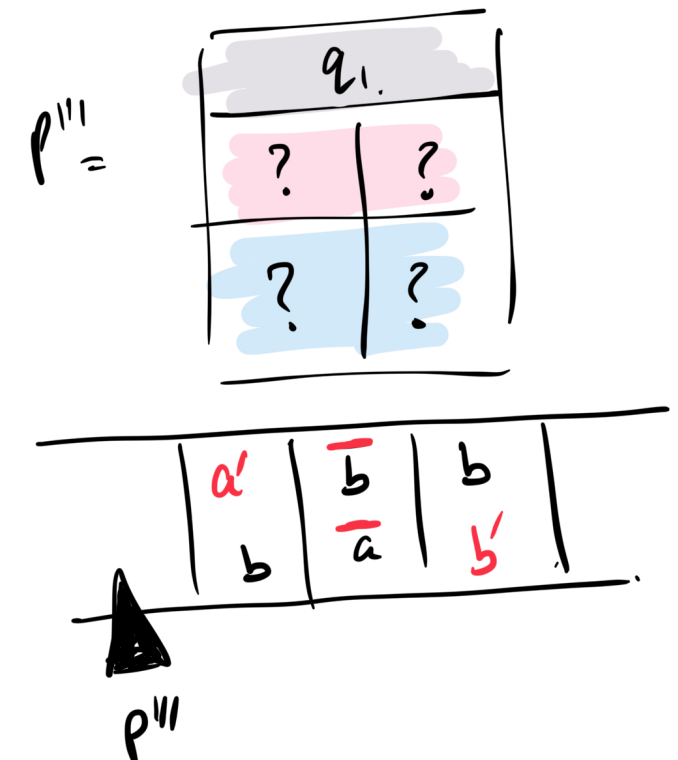
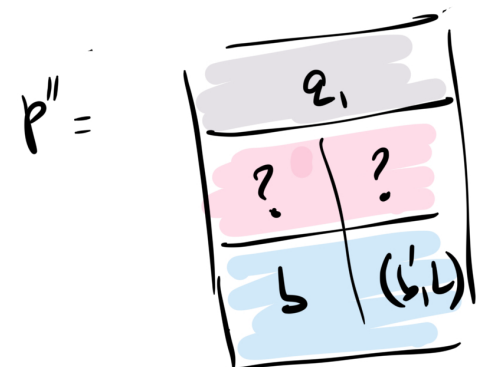
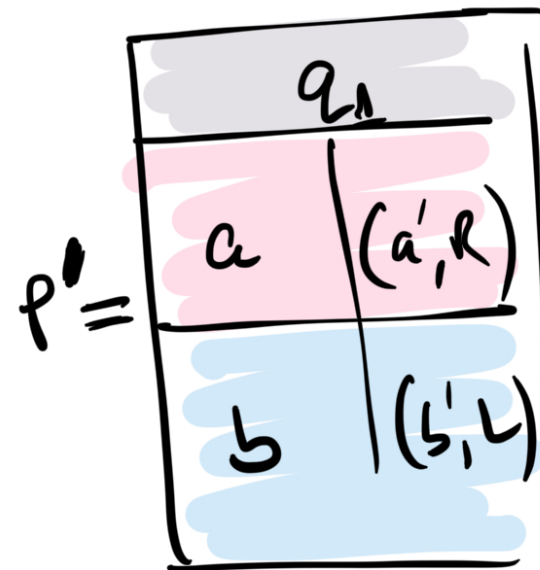
- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).
- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ .
- Move the virtual heads to the right locations as specified by the transition function of M .



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

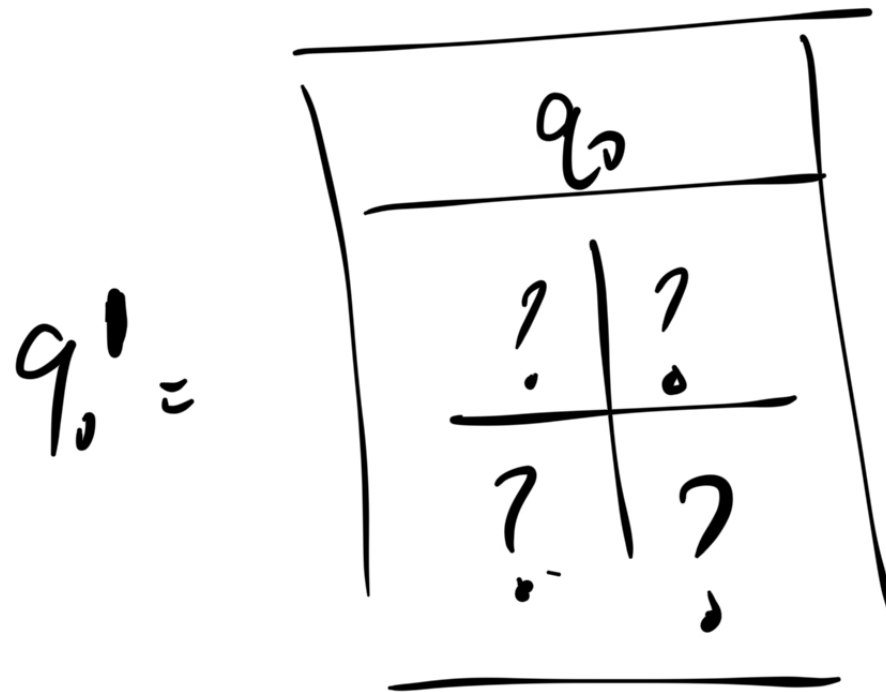
Simulating 2-Tape TM: Transition and States

- 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 .



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

Simulating 2-Tape TM: Initial State



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

Simulating 2-Tape TM: Halting States

q_{acc}	
?	?
?	?

q_{rej}	
?	?
?	?

	$\bar{a}$	$\bar{b}$	1	0
	0	1	1	1

q'

$q' =$

q	
a	?
b	?

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

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

Simulating 2-Tape TM: Halting States

	$\bar{a}$	$\bar{1}$	1	0
q'	0	$\bar{b}$	1	1

q	?
a	?
b	?

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

	$\bar{a}$	1	1	0
q''	0	$\bar{b}$	1	1

q_{acc}	
a	(a', R)
b	(b', R)

$\Rightarrow$

	a'	$\bar{1}$	1	0
	0	$\bar{b}$	1	1
q'''				

q_{acc}	
?	?
b	(b', R)

$\Downarrow$

	a'	$\bar{1}$	1	0
	0	b'	$\bar{1}$	1
q''''				

q_{acc}	
?	?
?	?

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.



L^AT_EX

Summary: TMs can simulate all “reasonable” machines

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,

$$\delta : Q \times \Sigma \times \Sigma \rightarrow Q \times \Sigma \times \{L, R\} \times \Sigma \times \{L, R\} .$$

None of these changes have any affect on the model.