Lecture 12:

* Quiz 1 regrades
* Checkup 4 is coming (due Monday?)
* Quiz 2: next week
  - 35 minutes
  - closed book, notes, come to your section, be on time!
  - Covers NFA → end of lecture 8 / regular languages (no TM)
  - Questions:

no justification requiered (but allowed for partial credit)

- Here's L, give NFA
- Here's L, give regexp
- Here's an NFA describe language
- Here's a regexp, describe language

just like Q4

- Prove closure of regular languages under some operation using DFA/NFA/regexp or known closure properties
- a couple other T/F/no justification problems

(you won't be asked to do subset construction)

- HW 2: grades + feedback soon (+ don't cheat)

Last time:

On an input w, a TM can:

=

=

=

- Regular languages $\subsetneq$ TM-decidable languages
- Input/output TMs and computable-functions

↳ practice it !!! It's tedious but important
(Tal has solutions for some, in lecture 11)

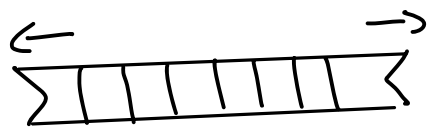
We looked at equivalent models of computation for TMs:

(1) Standard def (1 tape, ∞ to the right, go L/R)

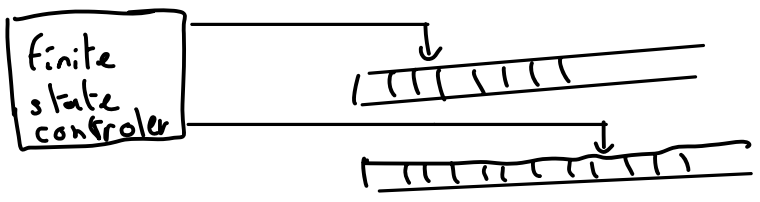
(2) Stay put TMs => go L/R / stay in same cell

$\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R, C\}$

(3) Doubly ∞ TMs => the tape is ∞ on both sides



(4) Multitape TMs: the TM has multiple tapes each with its own head

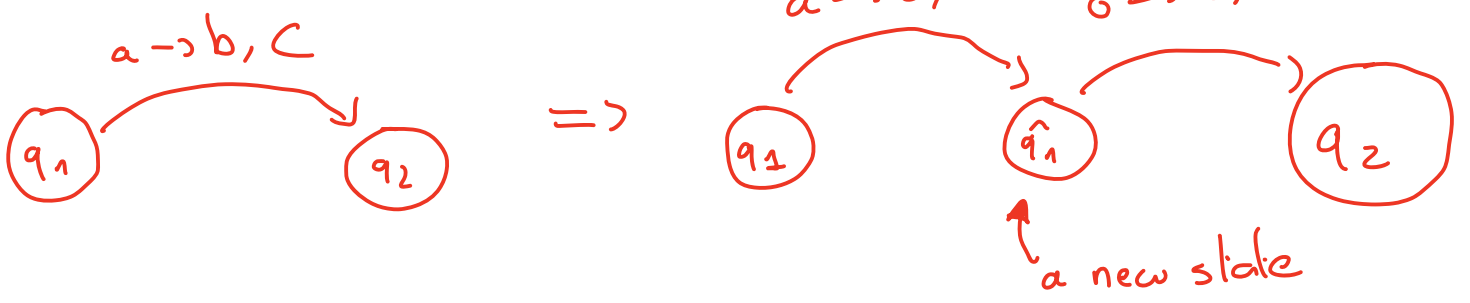


How to prove equivalence?

Show a standard TM can simulate a TM in the other model & vice versa.

↳ given a stay put TM M we explained how to build an equivalent standard TM M'
(i.e. accept/reject/loops on the same inputs)

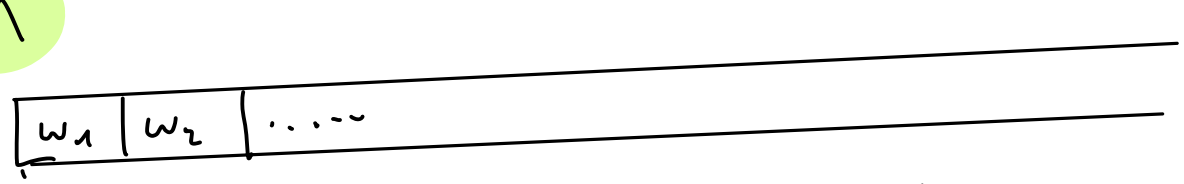
Idea: when M wants to stay put, we go $\xrightarrow{R}$ then $\xleftarrow{L}$ to simulate staying in the same cell.



(3) Doubly infinite TMs

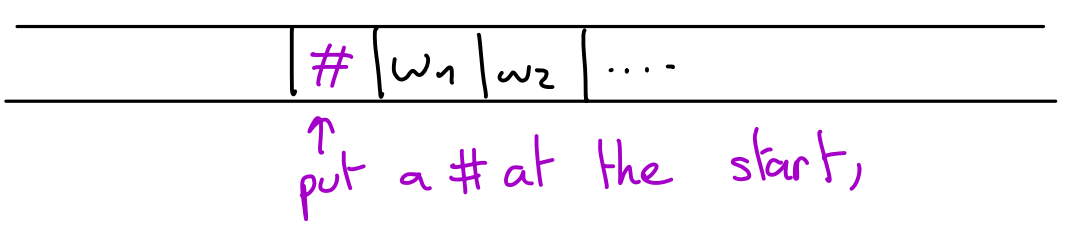
Last time: we can simulate a standard TM M using a doubly ∞ TM:

in M



if we try to go L here, we don't move

so in M'

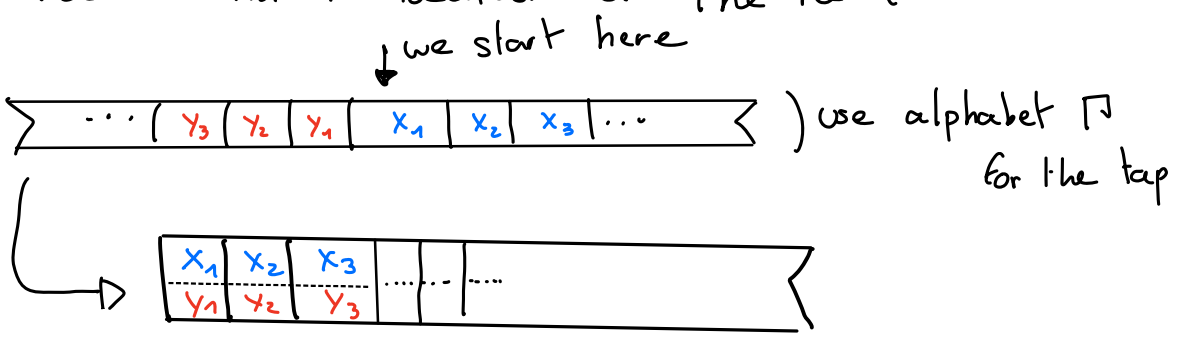


put a # at the start, when the head reads it immediately go right!

Every thing else is the same!

Other direction: Simulate ∞ infinite tape TM M with a standard TM M'

Imagine we "fold" the doubly ∞ tape around initial location of the head



$\begin{matrix} X_1 \\ Y_1 \end{matrix}$ is a new alphabet symbol

it records 2 symbols from the previous tape

=> the new tape alphabet will be $\Sigma \cup \Gamma^2 \cup \{\sqcup, \#\}$

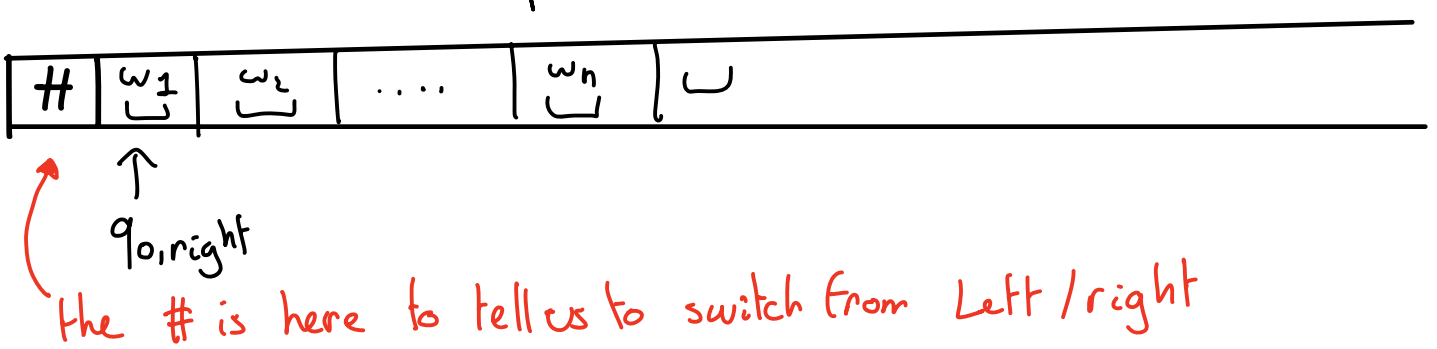
*) We create 2 version of each state q of M => q_{Left} q_{Right}

=> these are to record are we on the left/ right of the first symbol.

q_{Left} focus on the bottom symbol y

q_{Right} focus on the right symbol x

*) At first we'll update the tape to be



*) We also change the transition function δ to δ' in M'
depending on wether we're in q_{Right} / q_{left}
we read / update the top / bottom symbol

For example if we had in M: $\delta(q_1, a) = (q_8, b, R)$

we add the rules:

$$\forall c \in \Sigma,\ \delta'\left(q_1, Right, \begin{pmatrix} a \\ c \end{pmatrix}\right) = \left(q_8, \begin{pmatrix} b \\ c \end{pmatrix}, R\right)$$

↑ top symbols

↑ update the top symbol

↓ bottom symbols

$$\delta'\left(q_1, left, \begin{pmatrix} c \\ a \end{pmatrix}\right) = \left(q_8, \begin{pmatrix} c \\ b \end{pmatrix}, L\right)$$

↑ we go in opposite direction

Use # to switch between left & right:

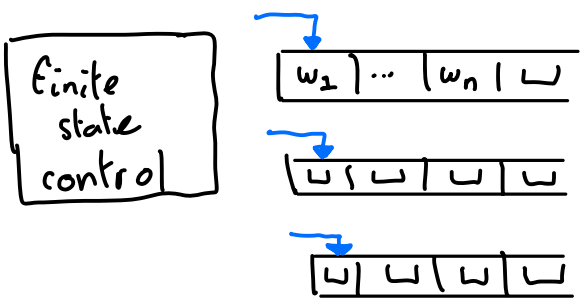
$$\delta'(q_i, right, \#) = (q_i, left, \#, R)$$

$$\delta'(q_i, left, \#) = (q_i, Right, \#, R)$$

(3) Multi-tape TM: has a fixed k number of tapes and read/write heads:

We have k tapes so $\delta: Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{L, R\}^k$

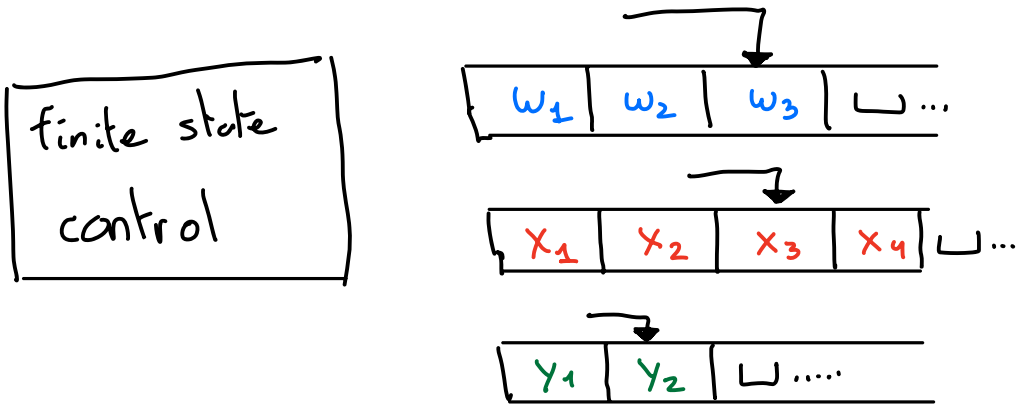
- initial configuration:
  - * each head points to the left most cell on their tape
  - * input written on first tape
  - * $\sqcup$ every where else



Claim: Multitape TM is equivalent to standard to single tape.

Proof sketch: given standard TM M, how to simulate it with a multitape TM?

- How to simulate k-tape TM using only 1 tape?



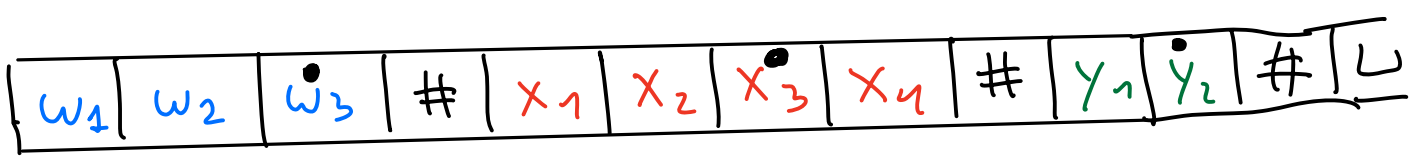
We'll track all the k tapes on a single tape

* only store up to when it's so many blanks
* we separate the tapes using #
* within each simulated tape => one symbol is dotted "$\dot{a}$" (new symbol)

=> it says where the head of the corresponding head should be'

New alphabet is $\Gamma \cup \dot{\Gamma} \cup \{\#\}$

↳ $\dot{a}, \dot{b}, \dot{\sqcup}$ for instance

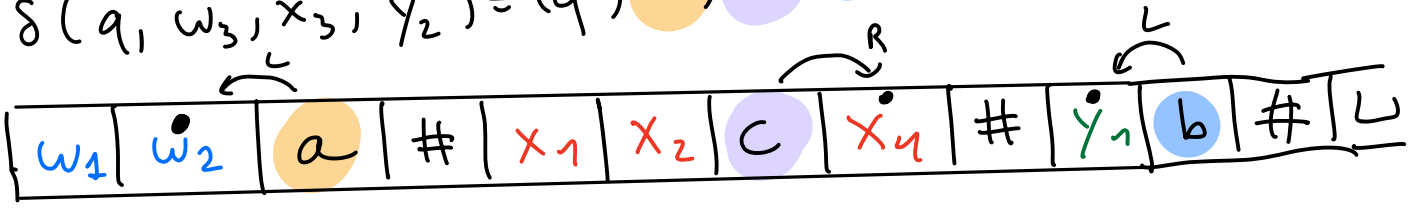


At each step we want to apply $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$

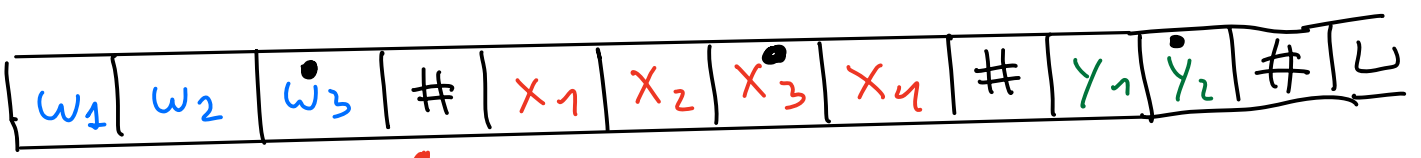
↳ we can't read what's under each head, we have only one head now!

- scan tape right to left, store (using states)
  the symbol under each head (so we know which transition to take)
- Scan from left to right, apply δ to each simulated tape 1 by 1:

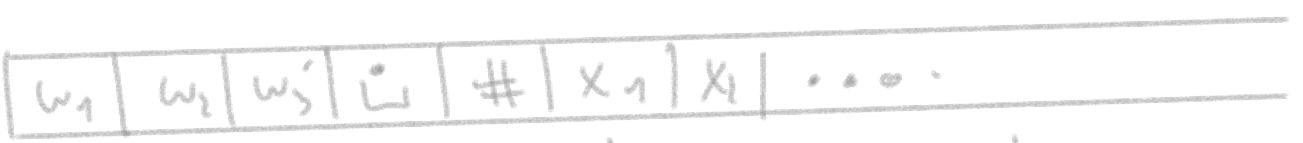
$\delta(q, w_3, x_3, y_2) = (q', a, c, b, L, R, L)$



- take care of edge cases:



what if we wanted the head to move right (on what should be blank)



we shifted everything!

Informal note about the complexity of this transformation.

for each step in the k-tape TM ($k > 1$)
we need to go back & forth to update the tapes + overhead

=> quadratic overhead in running time

This is inherent : ex $L = \{w \in \{0,1\}^* \mid w = w^R\}$

→ can decide in linear time using 2 tapes $O(|w|)$

→ need quadratic time on 1 tape $\Omega(|w|^2)$

We've seen a few equivalent models to TMs (standard ones)

- NFA + queue
- NFA + 2 stacks
- lambda calculus
- Random access memory
- Quantum computer.

TMs can simulate all known models of computation that we use

Church-Turing thesis

Every reasonable model of computation (or notion of "algorithm") can be simulated by a TM.

=> To show a language L is decidable / recognizable: we can use high-level program descriptions.

\ pseudo-code

(as long as every step is implementable)

3 levels of describing a TM algorithm
- formal level <= 7-tuple or diagram
- implementation level <= low level algorithms
- high level <= default from now on

To prove L is TM-recognizable:
- construct an algorithm M

- prove that: $x \in L \Rightarrow M$ accepts x

$x \notin L \Rightarrow M$ rejects x

or M runs forever on x

To prove L is TM-decidable:

- construct an algorithm M
- prove that: $x \in L \Rightarrow M$ accepts x

$x \notin L \Rightarrow M$ rejects x

Thm The class of TM-decidable languages is closed under complement.

Let L be a decidable language, let M decide L.

Then M' that is obtained by flipping accept/reject state of M is a decider for $\bar{L}$.

If $x \in \bar{L} \Rightarrow x \notin L \Rightarrow M$ rejects $x \Rightarrow M'$ accepts x

If $x \notin \bar{L} \Rightarrow x \in L \Rightarrow M$ accepts $x \Rightarrow M'$ rejects x