Announcements

- already out: challenge problem 2
  (no credit, only instant gratification: chocolate + stickers)

- imminent! (will announce):
  - checkup 4: first attempt due Mon night
  - HW 2 solutions
  - soon: grades + feedback (+ don't cheat!)

- Next Thur (Oct 22) in class: Quiz 2
  - 35 minutes
  - closed book, notes, all, come on time to your section
  - covers: from NFA → end of lecture 8 / regular languages (no TM)
  - questions on quiz:
    - Here's a language, give an NFA
    - Here's a language, give a regular expression
    - Here's an NFA, describe language
    - Here's a regular expression, describe language

    (no justification required (but allowed for partial credit))
    - prove closure of regular languages under some op using DFA/NFA/RE or known closure properties,
    - a couple other true/false/no justification problems

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

Last time:

- regular languages $\subsetneq$ TM-decidable languages
- defined input-output TM and computable functions
- some examples of TM (low level algorithms) and computable functions

can be used as helper function / subroutine

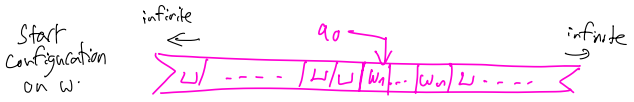
- defined 3 variants of standard TM we claimed equivalent:

proof sketched in class, more details in notes ← review

(1) TM where head can move right, left, or stay put

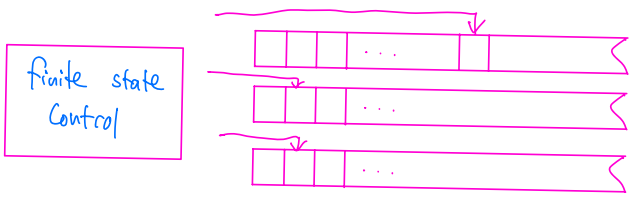
$\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R, C\}$

(2) A TM with a doubly-infinite tape

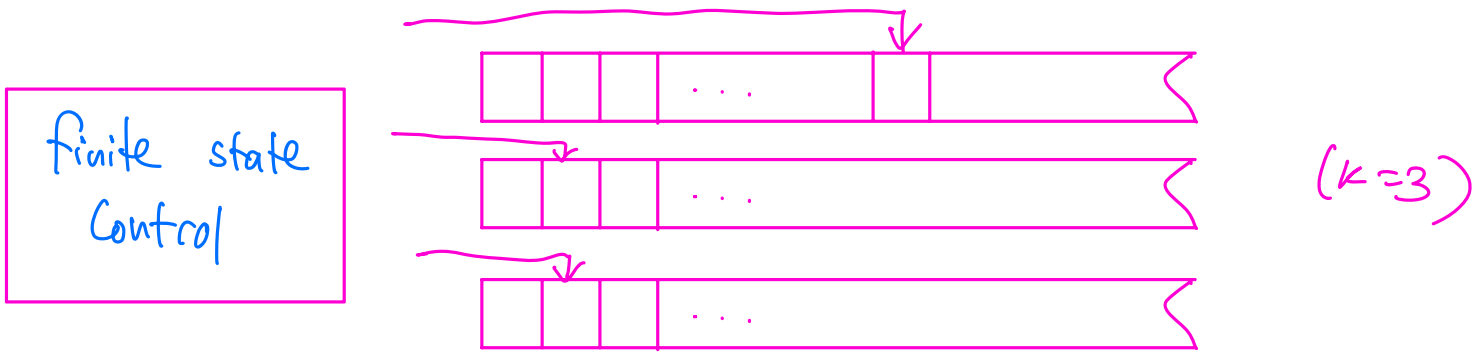


will prove next.

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



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



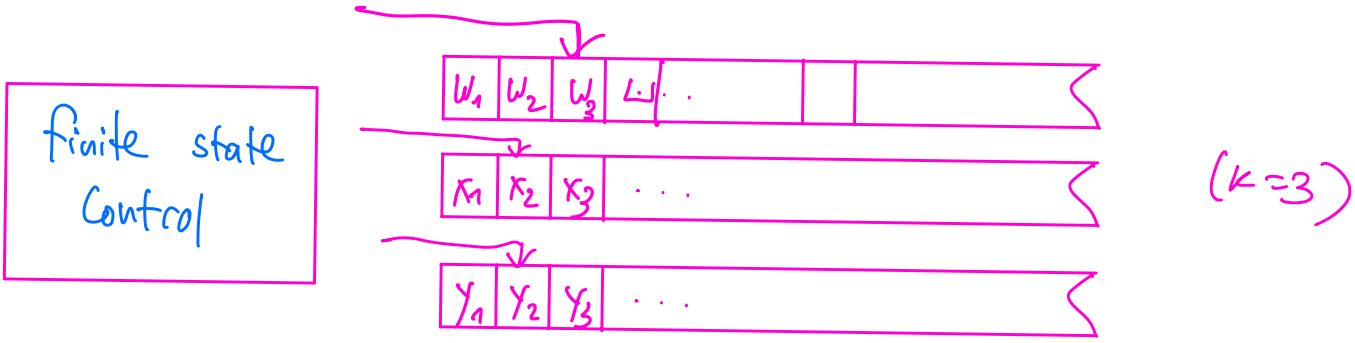
$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k$$

initial configuration:
- heads all point to leftmost cell on their tape
- input written on first tape
- $\sqcup$ everywhere else

Claim: k-tape TM is equivalent to 1-tape TM

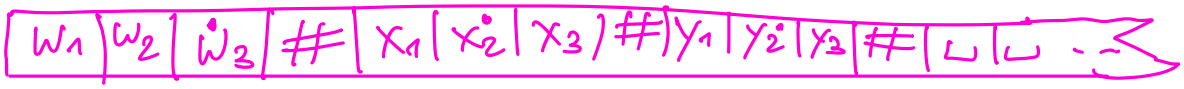
proof sketch:
→ given standard TM, can simulate it with multitape TM: obvious (just ignore all but first)

$\Leftarrow$: given k-tape TM, simulate with standard one as follows



Idea:

keep the content of each tape (upto ∞ many ⊔) on your one tape, separated by new symbol #



use alphabet $\Gamma \cup \dot{\Gamma} \cup \{\#\}$

- within each block, one symbol is dotted - marks location of read/write head in that block
- initially bring tape to right form:

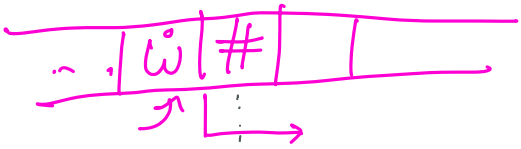
$[w_1 | w_2 | \dots | w_n | \sqcup | \sqcup \dots] \Rightarrow \# \dot{w}_1 w_2 \dots w_n \# \dot{\sqcup} \# \sqcup \# \dots$

$\delta(q_2, w_3, x_2, y_2) = (q_7, a, a, b, R, L, L)$ (e.g.)

At each step (i.e. single application of a transition in multi tape TM):

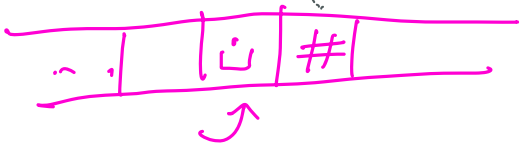
- scan tape from left to right, along the way collecting (via state) what each reading head points to.
- go back scanning to left, updating tape around each of the k dotted symbols

- take care of edge case when it happens:

say were here 

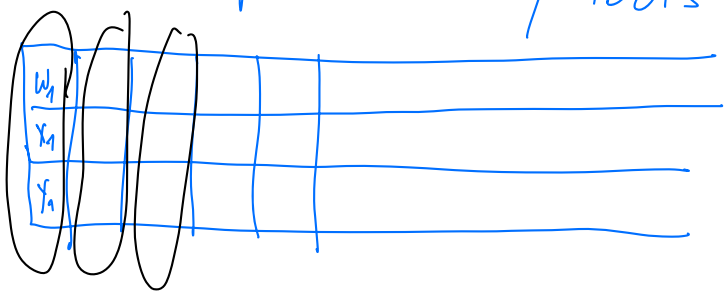
and need to move right (but no space/end of block)

→ shift content to right by one cell and insert ␣



(see end of lecture for a fully worked example).

Question in class: why not simulate k-tape TM by new alphabet symbols $\in \Gamma^k$, i.e.



(like we did for doubly infinite tape)?

Answer: this would work in a model where all k heads move R/L together. But since here they move independently, we can't capture the next triple read with a single R/L move

Informal note about complexity of this transformation:

- for each one step in k-tape TM, we scanned entire tape content (upto ⊔'s) back and forth
  ⇒ linear work per step
  (in tape content)

So total ⇒ quadratic blowup in runtime of standard (1-tape) TM vs k-tape TM ($k > 1$)

- Turns out this is inherent

  Eg for Palindromes $= \{w \in \{0,1\}^* \mid w = w^R\}$

  - can decide in linear time on 2-tape TM
    (do you see how?)

  - But it can be shown that any 1-tape TM for this language requires quadratic time (impossible in less)

We've seen a few equivalent models of Computing
(all equiv to TM).

There are many more! (even models that seem very different)

Examples:

equiv. to TM

- NFA with a queue
- NFA with 2 stacks





(note: context free lang ≡ NFA + 1 stack)
(weaker than TM)

- lambda calculus (due to Church)
- random access machine

- your favorite programming language

can be simulated by TM
(i.e. equiv or weaker)

⋮

TM can simulate all known models of computation that we use!

(yes, also including Quantum computers, AI models, ...)

Church-Turing thesis

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

Thus, to show a lang is TM-recognizable / TM-decidable you can use a high level algorithm description (as long as every step is implementable)

3 levels of describing a TM/algorithm:

- formal level ← state diagram or 7-tuple — (full specification)
- implementation level ← low-level alg description. — (refer to tape, states, transitions...)
- high level ← our default — pseudo-code / anything you can implement in your favorite model of computation.

Recall: To prove L is TM-recognizable:

- Construct algorithm M
- prove $x \in L \Rightarrow M$ accepts x

  $x \notin L \Rightarrow M$ rejects x

  or M runs forever on x

To prove L is TM-decidable

- Construct algorithm M
- prove $x \in L \Rightarrow M$ accepts x

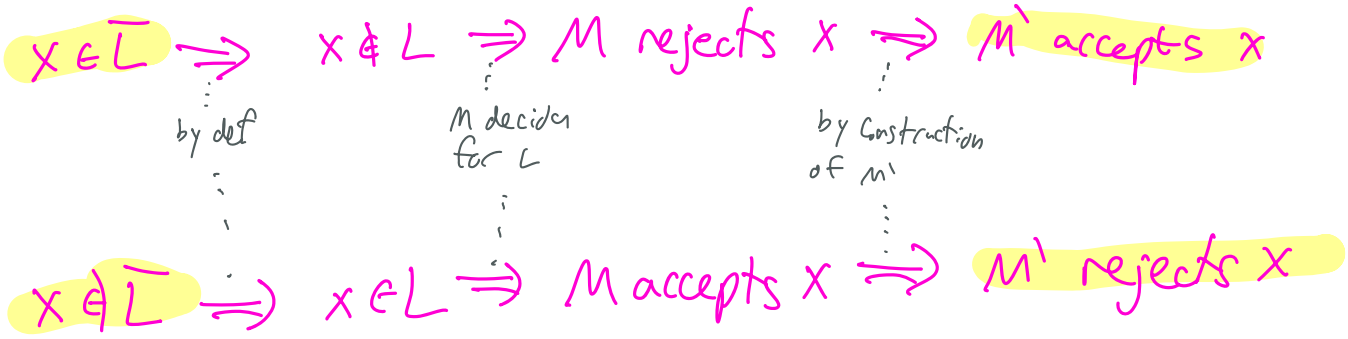
  $x \notin L \Rightarrow M$ rejects x

  ~~or M runs forever on x~~

Thm: The class of TM-decidable languages is closed under Complement

proof: Let L be a decidable language, and let M be a TM decides s.t. $L(M)=L$.

Construct M' that is the same as M, except with accept/reject states flipped)



Thus, we proved M' is a decider for $\overline{L}$. ☒

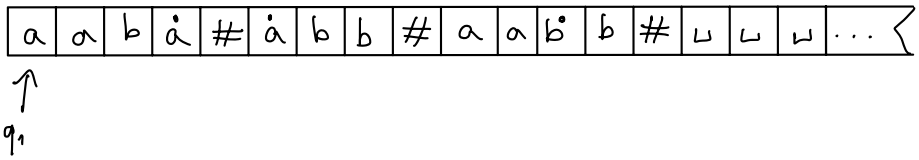
——— END OF CLASS.

Below is an example, added after class, of how to simulate one step in 3-tape TM on a 1-tape TM

You do not need to know these details, but included here in case it helps you.

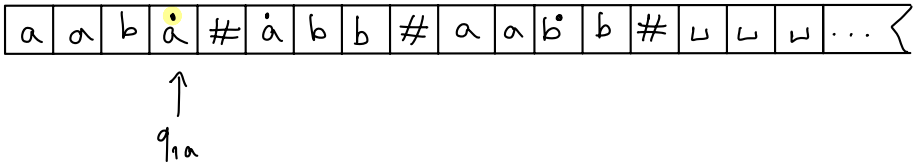
Example:

Suppose we are simulating a 3-tape TM with a standard one, and at some point in the computation we have the following configuration:

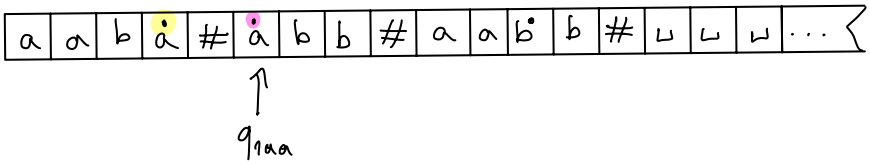


we start scanning tape content from left to right.

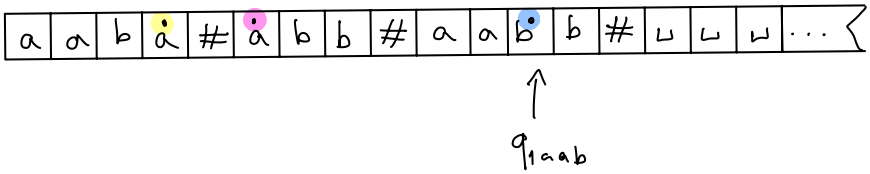
After we get to first dotted symbol we record it via state:



And again when we reach dotted symbol in second block (use new state):



And when we reach dotted symbol in last block:



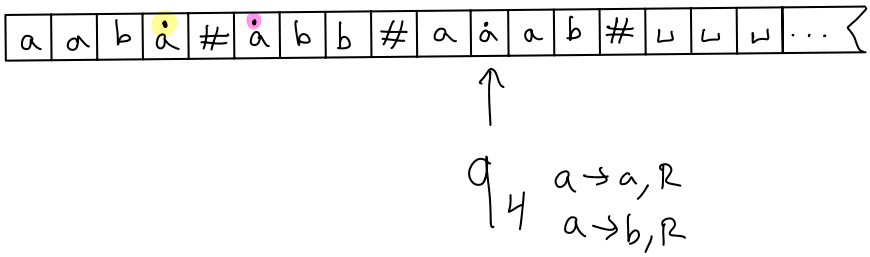
After we finish scanning to right we know the dotted symbols are (a, a, b).
Suppose we had $\delta(q_1, \dot{a}, \dot{a}, \dot{b}) = (q_4, a, b, a, R, R, L)$ (in the 3-tape TM)
We change to some state that remembers this, let's call it

$(q_4, \begin{matrix} a \to a, R \\ a \to b, R \\ b \to a, L \end{matrix})$ ↖ (name of state)

We start scanning left, when we reach the $\dot{b}$ we:

- rewrite $\dot{b}$ with a
- move to left
- rewrite the symbol we see there (a) with a dotted version $\dot{a}$
- switch to another state $(q_4 \begin{matrix} a \to a, R \\ a \to b, R \end{matrix})$

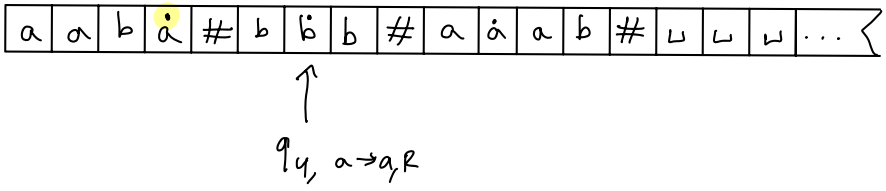
We now have:



We keep scanning left, when we reach $\dot{a}$:

- rewrite $\dot{a}$ with b
- move to right
- rewrite the symbol we see there (b) with $\dot{b}$
- switch to another state $(q_4, a \to a, R)$

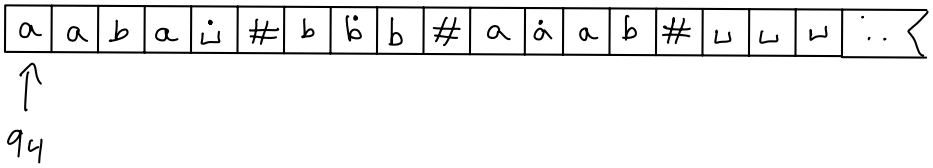
We now have:



We scan to the left a third time, when we reach $\dot{a}$:

- rewrite $\dot{a}$ with a
- move to right
- since you see the symbol #: (edge case to deal with)
  - insert an extra $\sqcup$ before the #
    (executing a subroutine that moves the rest of the string one spot to the right, and inserts a blank)
- replace the $\sqcup$ with dotted version $\dot{\sqcup}$
- switch to q_4 and Complete left scan.

Now we have:



and we start over with right scan, getting to end of third block in state $q_{4\sqcup ba}$ then scan left and update tape (depending on the transition $\delta(q_4, \sqcup, b, a)$) and so on.