

$L = \{ w \in \{a, b\}^* : |w| \text{ is even} \}$

Design a TM to decide  $L$ .

A TM  $M$  **decides** a language  $L$  if

$(s, \# w \#) \vdash^* (h, \# Y \#)$  for  $w \in L$  and

$(s, \# w \#) \vdash^* (h, \# N \#)$  for  $w \notin L$ .

$L = \{ u u^R \text{ where } u \in \{a, b\}^* \}$

We designed a TM which accepts this language (that is, it halts if the input is in  $L$  and hangs or computes forever when it is not).

How could you design a TM to decide  $L$  instead?

A TM  $M$  **decides** a language  $L$  if

$(s, \# w [\#]) \vdash^* (h, \# Y [\#])$  for  $w \in L$  and

$(s, \# w [\#]) \not\vdash^* (h, \# N [\#])$  for  $w \notin L$ .

What algorithm will you use?

# Announcements

Assignment #4 has been posted: due Wed. July 18.

Final exam tutorial:

Monday Aug. 6, 10am, ECS 116.

If the building is locked, I will prop open the back door to ECS (the one that opens on to the campus).

An artist's rendition of a steam-powered Turing machine. There is a mural of this between the second and third floors in Sieg Hall at UW Seattle.



# Machine Schema

We introduce machine schema- a powerful notation for drawing a picture of a TM.

This is a very concise way to represent a TM.

Using machine schema facilitates a procedural approach to TM design.

## Basic Building blocks:

Machine L: Move head one square left and halt.

Machine R: Move head one square right and halt.

Machine  $\sigma$ : Writes  $\sigma$  and halt.

$\xrightarrow{\sigma}$  take on  $\sigma$   $\longrightarrow$  take on any symbol

Halt if no arc exits with current symbol.

## Technical note:

$M_1 M_2$  (juxtaposition of two TM names)

means the same thing as:

$M_1 \longrightarrow M_2$  (take transition on any symbol)

## Example 1:

Machine schema for a

TM which on a input  $w \in \{0, 1\}^*$ ,

shifts  $w$  over one position to the right.

That is:  $(s, \# w [\#]) \vdash^* (h, \# \# w [\#])$ .

## Example 2: TM from last class.

$L = \{ w \in \{a, b\}^* : w \text{ has an even number of } a\text{'s} \}$

A TM  $M$  **decides** a language  $L$  if

$(s, \# w \#) \vdash^* (h, \# Y \#)$  for  $w \in L$  and

$(s, \# w \#) \vdash^* (h, \# N \#)$  for  $w \notin L$ .

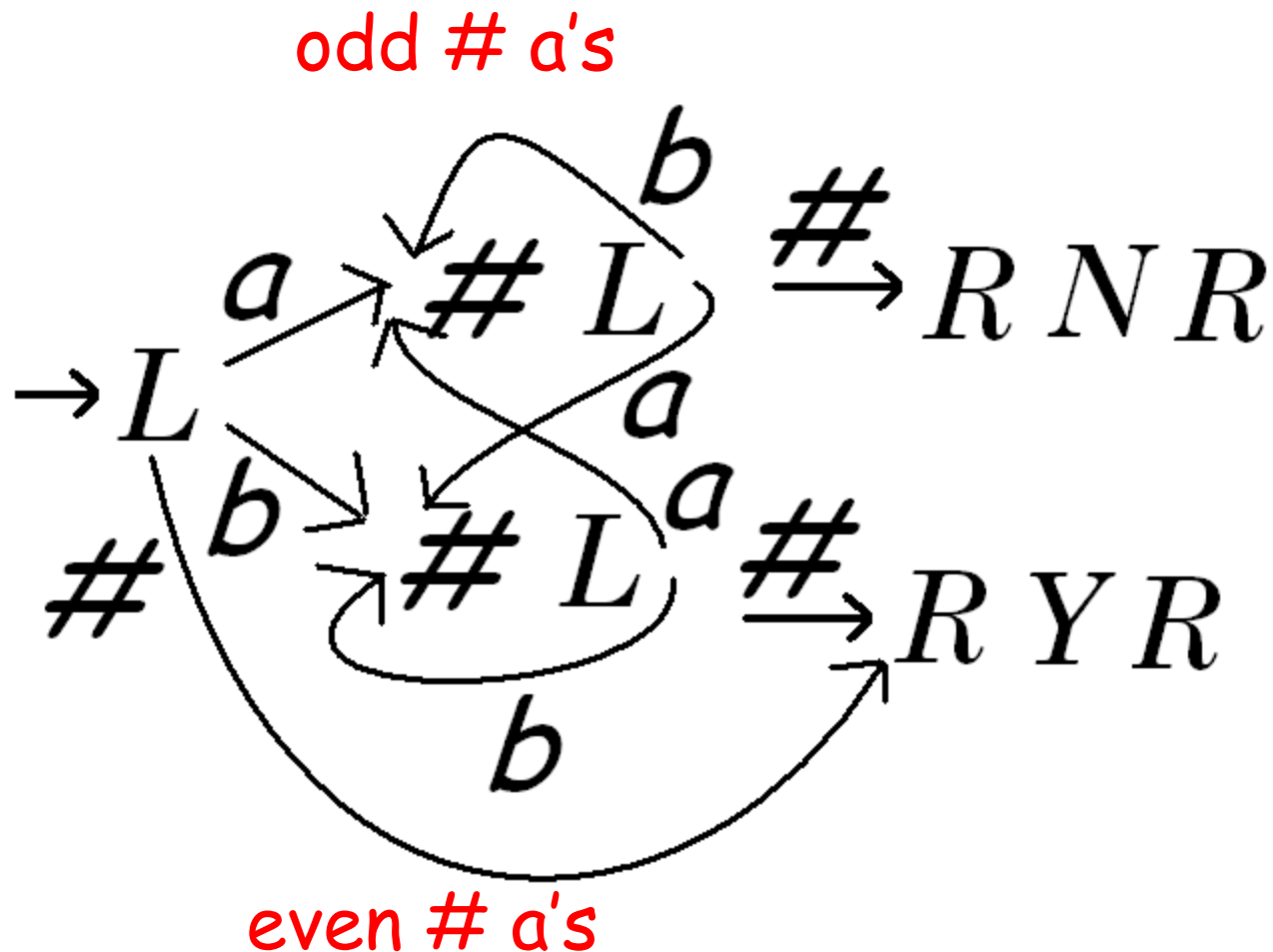
To decide  $L$ :

Move left erasing symbols as we go and keeping track of the number of  $a$ 's modulo 2 until reaching the blank at the end and then write the answer on the tape.



TM which decides

$L = \{ w \in \{a, b\}^* : w \text{ has an even number of } a\text{'s} \}$



### Ex. 3: A COPY TM.

On input  $w \in \{a, b\}^*$ , this TM halts with  $w$  followed by  $\#$  followed by a copy of  $w$ .

That is:

$$(s, \# w [\#]) \vdash^* (h, \# w \# w [\#]).$$

The program for this TM is available from the page which gives the TM simulator.

The algorithm changes each  $a$  to  $A$  and each  $b$  to  $B$  in the first copy of  $w$  to mark that it has been copied over already.

// Find leftmost symbol of w not copied yet.

middle # goleft L

start state: middle

goleft a goleft L

goleft b goleft L

// Found either #, A, B from part being copied.

goleft A next\_s R

goleft B next\_s R

goleft # next\_s R

// Go to # between w and copy of w

//remembering symbol to copy.

next\_s a next\_s A

next\_s b next\_s B

next\_s A RtoM\_a R

next\_s B RtoM\_b R

next\_s # clean L

// Done- clean up.

// Go right to the middle

RtoM\_a a RtoM\_a R  
RtoM\_a b RtoM\_a R  
RtoM\_a # RtoR\_a R

RtoM\_b a RtoM\_b R  
RtoM\_b b RtoM\_b R  
RtoM\_b # RtoR\_b R

// Go right to the right hand end

RtoR\_a a RtoR\_a R  
RtoR\_a b RtoR\_a R  
RtoR\_a # left1 a

RtoR\_b a RtoR\_b R  
RtoR\_b b RtoR\_b R  
RtoR\_b # left1 b

// Go left to blank in middle.

left1 a left1 L

left1 b left1 L

left1 # middle #

// Clean up the tape-

//change A back to a and B back to b.

|       |   |        |   |
|-------|---|--------|---|
| clean | A | clean  | a |
| clean | B | clean  | b |
| clean | a | clean  | L |
| clean | b | clean  | L |
| clean | # | right1 | R |

// Position head to right of copy of w.

|        |   |        |   |
|--------|---|--------|---|
| right1 | a | right1 | R |
| right1 | b | right1 | R |
| right1 | # | right2 | R |
| right2 | a | right2 | R |
| right2 | b | right2 | R |
| right2 | # | h      | # |