1. Design 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 [#]).

2. Show the computation of your TM on the input 010:

(s, # 0 1 0 [#]) $\vdash$ …

3. Show what your TM does on input ε:

(s, # [#]) $\vdash$ …

There is a tutorial today.

The assignment has been revised (without changing the meanings of the questions) to clarify what is required:

4(a). L= { w in {a, b, c}* : w either has the same number of a's and b's or w has twice as many c's as a's (or satisfies both)}

10. $\{a^r b^n a^{n-r}$: $n \geq r$ (before it was $n \geq 0$)$\}$

L= { u u^R 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).

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.

What algorithm could you use to decide L?

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 σ: Writes σ and halt.

 take on σ  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:

$M_1 \rightarrow M_2$ (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:

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

A TM M decides a language L if

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

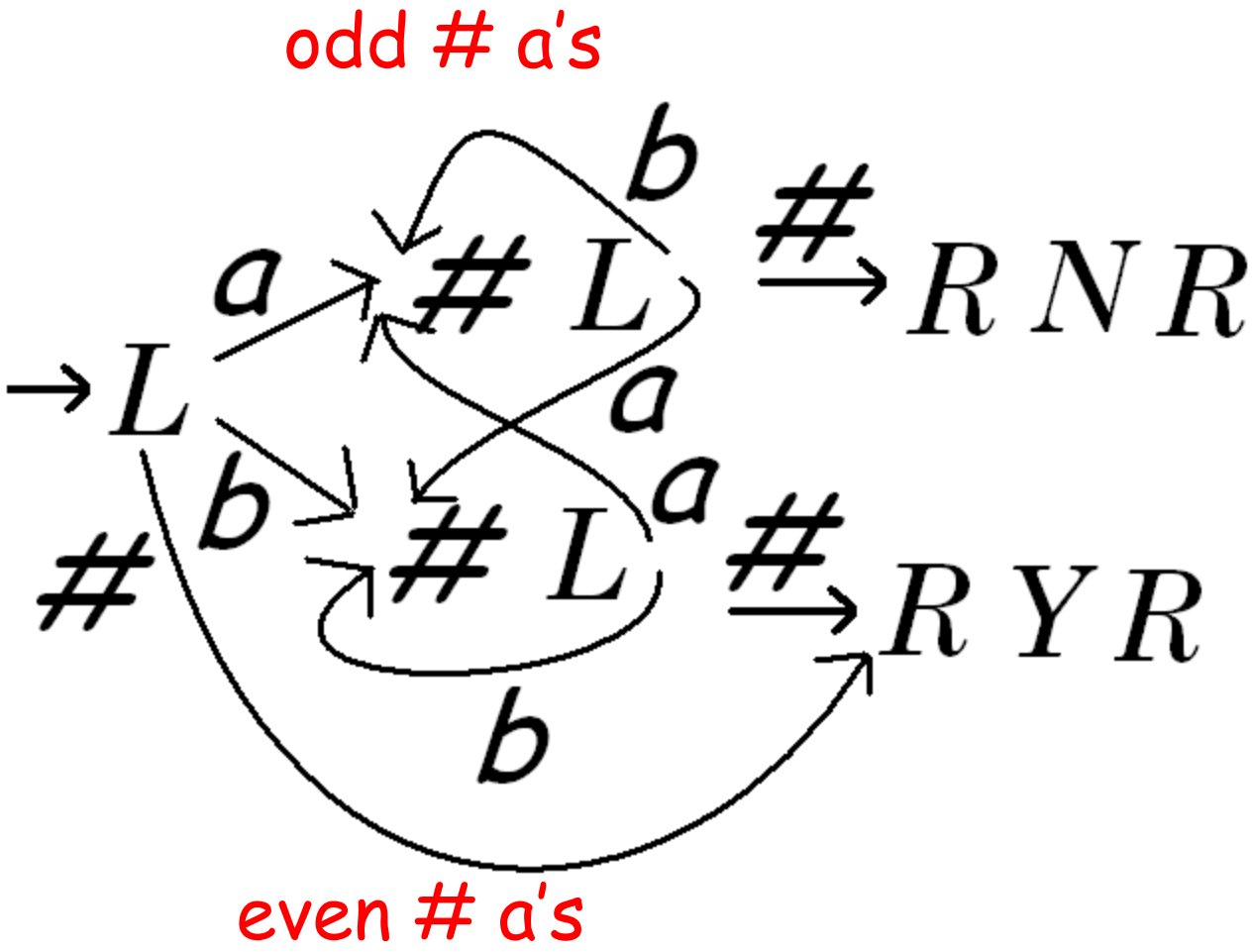
(s, # w [#]) $\vdash^*$ (h, # N [#]) for w $\notin$ L.

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 ∈ {a, b}* : w has an even number of a'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
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.
```

start state: middle

// Go right to the middle

```
RtoM_a  a  RtoM_a  R        RtoM_b  a  RtoM_b  R
RtoM_a  b  RtoM_a  R        RtoM_b  b  RtoM_b  R
RtoM_a  #  RtoR_a  R        RtoM_b  #  RtoR_b  R
```

// Go right to the right hand end

```
RtoR_a  a  RtoR_a  R        RtoR_b  a  RtoR_b  R
RtoR_a  b  RtoR_a  R        RtoR_b  b  RtoR_b  R
RtoR_a  #  left1   a        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	#