Notes to accompany Section 9.2 – Combining Turing Machines for Complicated Tasks

“Building block” Turing machines – define “small” machines first, which carry out frequently used routines, and then concatenate them, passing the ending configuration of one along to the next…

Assume for all examples that $\Sigma = \{a, b, c\}$. We'll use lower case l/r to indicate movement left/right. Also, for each example, assume we have initial state q_0 and final state q_f, with other states added as needed.

Throughout the following, “TM” is an abbreviation for “Turing machine.”

Example: “R” = the TM that moves the read/write head one place to the right. (easy)
Transition function:

$$\begin{aligned} \delta(q_0, a) &= (q_f, a, r) \\ \delta(q_0, b) &= (q_f, b, r) \\ \delta(q_0, c) &= (q_f, c, r) \\ \delta(q_0, \Box) &= (q_f, \Box, r) \end{aligned}$$

(Note: The above can be abbreviated as simply $\delta(q_0, x) = (q_1, x, r), \forall\, x \in \{a, b, c, \Box\}$.)

Example: $\Box$ = the TM that erases the current cell, and halts reading the erased cell.
Transition function:

$$\begin{aligned} \delta(q_0, x) &= (q_1, \Box, r), \forall x \in \{a, b, c, \Box\} \\ \delta(q_1, x) &= (q_f, x, l), \forall x \in \{a, b, c, \Box\} \end{aligned}$$

Note that we need an additional state whose job is simply to move us back one place to the left, so that we halt on the erased cell. (If the text's definition allowed us to read/write without moving the head afterwards, this would be a bit simpler. It's not hard to see that an alternative definition, wherein each transition allowed us to move left, move right, or remain in place, would be effectively equivalent to the text's definition of “Turing machine.”)

Note: We could design a TM with the name “a,” “b,” or “c” in a similar way. In general, we'll define TM “x” to be the TM that erases the current cell, writes an x there (in place of what was there before), then halts on that cell.

Example: $R_\Box$ = the TM that moves to the right until it finds a blank, then halts on that blank. (This comes in handy often; e.g., finding the end of the original input string, which is typically the first blank cell to the right.) If the current cell is a blank, this machine will just halt on that cell, rather than moving to the right.
Transition function:

$$\begin{aligned} \delta(q_0, x) &= (q_0, x, r), \forall x \in \{a, b, c\} \\ \delta(q_0, \Box) &= (q_1, \Box, r) \\ \delta(q_1, x) &= (q_f, x, l), x \in \{a, b, c, \Box\} \end{aligned}$$

Note: We can easily define machines R_a, R_b, etc. similarly; i.e., R_x may be defined to be the machine which moves to the right until it finds the first instance of x, then halts on that cell. In a similarly straightforward way, we can design machine L_x, which moves left, rather than right, until finding x for the first time, and then halts on that cell. (Try designing this one yourself as an exercise.)

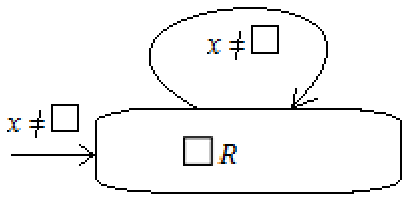
Another useful building block machine (design as an exercise):
$R_{\bar{x}}$ = the TM that moves to the right until it finds a cell which does *not* have an x in it, then halts on that cell. For instance, if we give Turing machine $R_{\bar{a}}$ an input tape with configuration $abab\mathrm{q}_0\boldsymbol{a}aabaab$, it should halt with $ababaaa\mathrm{q}_\mathrm{f}\boldsymbol{b}aab$. That is, it will halt in a favorable state, reading the first "b" to the right of where the read/write head started.

This allows us to more easily design TM's which carry out more complicated algorithms. In particular, using "building block" machines gets us around the need to deal with defining more and more (dozens/hundreds?/…) states and corresponding transition rules.

Example: "Eraser" – a machine which erases all input to the right of the read/write head (until the first blank is encountered) – that is, given an initial configuration $w_1 q_0 w_2$, where w_2 is assumed to be nonempty, the machine will halt on $w_1 q_f \square$ - that is, w_2 will be erased, and the machine will halt reading the first blank cell after w_1. (Or, if the initial configuration was $q_0 w$ – that is, if the tape is blank to the left of the read/write head - then the machine will just erase w and then halt on an empty tape.) If $w_2$is empty, this machine will immediately halt on $w_1 q_f \square$ (making no changes to the tape; just going to a final state).

Steps:
1. Read the current cell. If it's blank, halt on that cell. If not, erase the current cell.
2. Move one cell to the right.
3. Repeat from step 1.

So, as long as the current cell is nonblank, we apply the "building block" machines $\square$, then R, repeatedly. A diagram for this machine is shown to the right. The diagram indicates that, as long as the currently read symbol on the tape ("x") is not a blank, we will erase the current cell, then move one cell to the right. The loop will end when the current symbol is a blank; at this point, the machine will halt.



Note: with these diagrams, we're not dealing with states at all. We're only concering ourselves with the contents of the tape and the position of the read/write head when the machine halts. (We can always assume that each machine is halting in a "final" state, and design the machine accordingly.)

Example: “Leftshift” – a machine which shifts an entire input string one cell to the left. More precisely: given an initial configuration $w_1 q_0 w_2$, where w_2 is assumed to be nonempty (and include no blanks), “leftshift” will shift w_2 one cell to the left. (If w_1 is nonempty, its rightmost cell will end up being overwritten by the leftmost cell of w_2.) This machine will halt on the first cell of w_2 after it’s been shifted – i.e., in configuration $w_1' q_f w_2$, where w_1' is w_1 with its last cell removed (assuming w_1 was nonempty in the first place).

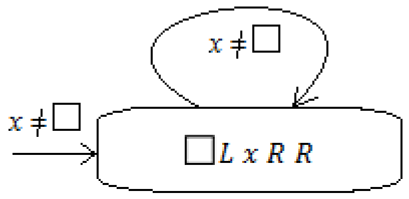
For example: if the input tape accac**b**bac is given to “Leftshift” as input (where the read/write head is currently reading the highlighted “b”), then the machine will halt on acca**b**bac. (Note that the string before the current cell had been “accac;” now it’s just “acca” because the last “c” got overwritten.)

Another example: if the input is $w\square w$, where the read/write cell is currently on the first character of the second instance of “w,” then the machine will halt on ww, reading the first character of the second instance of w.

Steps:
1. Read the current cell. If it’s blank, stop. Otherwise, erase it (temporarily, to mark its location) and memorize it (with an internal state).
2. Move one cell to the left, then write the memorized character in that cell.
3. Move two cells to the right. (Note – one cell to the right is the cell you just erased, so move one more place.) Then repeat from step 1.

So, we’re using the following “building blocks,” if the current cell , x, is nonblank:
$\square$, then L, then x, then R twice. That is: $\square\ L\ x\ R\ R$. If the current cell is blank, then we halt.

Diagram:



Comment: This machine does “leftshift” the string to the right of the read/write head’s initial location, but we did miss a detail here. Do you see it? (One of the exercises at the end of this handout invites you to fix the “mistake!”)

Example: Turing machine C is to be a machine that will make an extra copy the input string. Specifically, given input tape with configuration $q_0 w$, the machine will halt on $w\square w$. (Note: it's easier to design this machine with a blank between copies, though it could be removed with a few extra steps.)

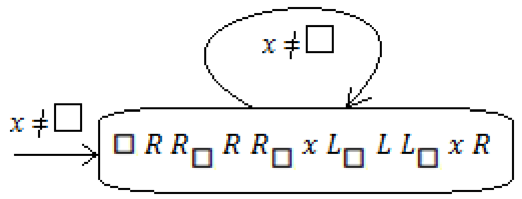
First, describe the algorithm, step by step, in words…
1. "Memorize" the current character, x, being read by the read/write head. (This can be done with states; e.g., go to state q_x to indicate that you've read x…
2. Erase this cell, temporarily. (This is how we'll memorize the position of the character we just read.)
3. Move to the right until we reach a blank cell – this is the blank between our copies of w.
4. Move to the right again until we reach the next blank cell – this is the end of the "copy" to this point.
5. Copy the "memorized" character.
6. Move left to the next blank cell (between copies), then again to the next blank cell (marking the position of the cell we just copied).
7. Rewrite the memorized character in this cell.
8. Move one cell to the right. If this cell is blank, move left to the first cell of the original input and halt. If this cell is not blank, repeat from step 1.

Example: (Here we'll just underline the current cell, since we're not naming states)
Start with $\underline{a}$bba. Copy/erase the first a, move to the second blank, and write it; this gives us □bba□$\underline{a.}$
Next go back to the left, replace the memorized character: $\underline{a}$bba□a
Next move one place to the right. Read a b (not blank), so repeat…
abba□a … a□ba□ab… abba□ab… etc.
Eventually end up with abba□abba.

Diagram (rather than transition function) for this in terms of "building blocks"…
Idea: as long as next character in the input string, x, is not a blank, then we do all of the following:

$$\square \quad R\, R_{\square}\; R\, R_{\square}\; x\, L_{\square}\; L\, L_{\square}\; x\, R$$

A diagram for this machine is shown to the right:


$x \neq \square$
$x \neq \square$
$\square\, R\, R_{\square}\, R\, R_{\square}\, x\, L_{\square}\, L\, L_{\square}\, x\, R$

(Note: we didn't specify which cell the head would be reading when this machine halts. Where will it end up, according to these instructions? If we wanted to change that – e.g., move it to the first nonblank cell on the tape – how would we do that?)

Practice Exercises: Write out transition functions (add additional states, tape alphabet characters, etc. as necessary) for each of the following “building block” machines, described above. (Assume $\Sigma = \{a, b, c\}$ for each).

We’ll discuss these in class, and/or solutions will be provided, in the near future…

1. R_a

2. $R_{\overline{\square}}$ (move right until the first non-blank cell is reached)

3. A version of “leftshift” which actually does halt on the first cell of the shifted string. (Our example didn’t halt in the right place – can you fix it? Note – this may be trickier than it first appears…)

4. “Rightshift” (reverse of “leftshift” – that is, given input tape $w_1 q_0 w_2$, halt on $w_1 \square q_f w_2$, which halts on the first cell of w_2 after it’s been shifted to the right).

5. Modify “C” (the “copier” machine) so that it will halt while reading the first (leftmost) nonblank cell on the tape. (Hint: Use the $L_{\square}$ building block machine…)

6. Design a machine similar to machine “C,” but which will halt on $w \square w^R$ rather than on $w \square w$. That is, given input string w, insert a blank and then write the reverse of w, rather than w itself…