

## Solutions to Problem Set 3

**3.2** This concerns the Turing machine  $M_1$  whose description and state diagram appears in Example 3.5 on the textbook. In each of the parts, give the sequence of configurations that  $M_2$  enters when started on the indicated input string.

a. 11.

$$q_1 11 \Rightarrow \sqcup q_3 1 \Rightarrow \sqcup 1 q_3 \sqcup \Rightarrow \sqcup 1 \sqcup q_{rej} \sqcup.$$

b. 1#1.

$$q_1 1\#1 \Rightarrow \sqcup q_3 \#1 \Rightarrow \sqcup \#q_5 1 \Rightarrow \sqcup \#1 q_5 \sqcup \Rightarrow \sqcup \#q_7 1 \Rightarrow \sqcup q_7 \#1 \Rightarrow q_7 \sqcup \#1 \Rightarrow \sqcup q_9 \#1 \Rightarrow \sqcup \#q_{11} 1 \Rightarrow \sqcup q_{12} \#x \Rightarrow q_{12} \sqcup \#x \Rightarrow \sqcup q_{13} \#x \Rightarrow \sqcup \#q_{14} x \Rightarrow \sqcup \#x q_{14} \sqcup \Rightarrow \sqcup \#x \sqcup q_{acc} \sqcup.$$

c. 1#1.

$$q_1 1\#1 \Rightarrow \sqcup q_3 \#1 \Rightarrow \sqcup \#q_5 \#1 \Rightarrow \sqcup \# \# q_{rej} 1.$$

d. 10#11.

$$q_1 10\#11 \Rightarrow \sqcup q_3 0\#11 \Rightarrow \sqcup 0 q_3 \#11 \Rightarrow \sqcup 0 \#q_5 11 \Rightarrow \sqcup 0 \#1 q_5 1 \Rightarrow \sqcup 0 \#11 q_5 \sqcup \Rightarrow \sqcup 0 \#1 q_7 1 \Rightarrow \sqcup 0 \#q_7 11 \Rightarrow \sqcup 0 q_7 \#11 \Rightarrow \sqcup q_7 0\#11 \Rightarrow q_7 \sqcup 0\#11 \Rightarrow \sqcup q_9 0\#11 \Rightarrow \sqcup 0 q_9 \#11 \Rightarrow \sqcup 0 \#q_{11} 11 \Rightarrow \sqcup 0 q_{12} \#x1 \Rightarrow \sqcup q_{12} 0\#x1 \Rightarrow q_{12} \sqcup 0\#x1 \Rightarrow \sqcup q_{13} 0\#x1 \Rightarrow \sqcup x q_8 \#x1 \Rightarrow \sqcup x \#q_{10} x1 \Rightarrow \sqcup x \#x q_{10} 1 \Rightarrow \sqcup x \#x1 q_{rej}.$$

e. 10#10.

$$q_1 10\#10 \Rightarrow \sqcup q_3 0\#10 \Rightarrow \sqcup 0 q_3 \#10 \Rightarrow \sqcup 0 \#q_5 10 \Rightarrow \sqcup 0 \#1 q_5 0 \Rightarrow \sqcup 0 \#10 q_5 \sqcup \Rightarrow \sqcup 0 \#1 q_7 0 \Rightarrow \sqcup 0 \#q_7 10 \Rightarrow \sqcup 0 q_7 \#10 \Rightarrow \sqcup q_7 0\#10 \Rightarrow q_7 \sqcup 0\#10 \Rightarrow \sqcup q_9 0\#10 \Rightarrow \sqcup 0 q_9 \#10 \Rightarrow \sqcup 0 \#q_{11} 10 \Rightarrow \sqcup 0 q_{12} \#x0 \Rightarrow \sqcup q_{12} 0\#x0 \Rightarrow q_{12} \sqcup 0\#x0 \Rightarrow \sqcup q_{13} 0\#x0 \Rightarrow \sqcup x q_8 \#x0 \Rightarrow \sqcup x \#q_{10} x0 \Rightarrow \sqcup x \#x q_{10} 0 \Rightarrow \sqcup x \#q_{12} xx \Rightarrow \sqcup x q_{12} \#xx \Rightarrow \sqcup q_{12} x \#xx \Rightarrow q_{12} \sqcup x \#xx \Rightarrow \sqcup q_{13} x \#xx \Rightarrow \sqcup x q_{13} \#xx \Rightarrow \sqcup x \#q_{14} xx \Rightarrow \sqcup x \#x q_{14} x \Rightarrow \sqcup x \#xx q_{14} \Rightarrow \sqcup x \#xx \sqcup q_{acc} \sqcup.$$

**3.5** This exercise tests your detail understanding of the formal definition of a Turing machine as given in Def. 3.1 on page 128-129 of the textbook. This was also covered in last week's discussion section.

a) Can a Turing machine ever write the blank symbol  $\sqcup$  on its tape?

Yes. The Turing machine can write any symbol from the tape alphabet  $\Gamma$  to the tape and the blank symbol  $\sqcup$  is mandated to be part of the tape alphabet  $\Gamma$ .

b) Can the tape alphabet  $\Gamma$  be the same as the input alphabet  $\Sigma$ ?

No. The tape alphabet  $\Gamma$  always contains the blank symbol  $\sqcup$ , while the input alphabet  $\Sigma$  cannot contain this symbol. If the blank symbol were part of the input alphabet, the Turing machine would never know when the input actually ends.

c) Can a Turing machine's head ever be in the same location in two successive steps?

Yes. But the only situation this can happen is when the Turing machine is on the first tape square and it tries to move left. It will stay in place instead of falling off the tape. However,

if it is not on the leftmost square, then in the next move the tape head cannot remain in the same location.

d) Can a Turing machine contain just a single state?

No. A Turing machine must contain at least two states: an accept state and a reject state. Because being in either of these states halts the computation, a different start state would be necessary in order for the TM to read any input whatsoever.

**3.8** Give implementation-level descriptions of Turing machines that decide the following languages over the alphabet  $\{0, 1\}$ :

a)  $\{ w : w \text{ contains an equal number of 0s and 1s} \}$

$M_0 =$  “On input  $w$ :

1. Repeat the following steps:

- (a) Place a mark on top of the leftmost tape symbol (say a dot). If that symbol is blank, accept.
- (b) Scan right until a symbol “different” than the symbol marked is found (that is, if the symbol marked in (a) was 1, scan for 0; if it was a 0, scan for 1). Cross off the found symbol. If no “different” symbol is found, reject.
- (c) Go back left until a marked symbol (with a dot) is found. Remove the mark and cross it off.
- (d) Scan right to the next non-crossed-off (or blank) symbol.”

b)  $\{ w : w \text{ contains twice as many 0s as 1s} \}$

$M_1 =$  “On input  $w$ :

1. If  $w = \epsilon$ , accept.
2. Repeat the following steps:
  - (a) Scan right to the next 1.
    - i. If there is one, cross it off.
    - ii. If no 1 is found, scan tape looking for a symbol 0. If a symbol 0 is found, reject. Otherwise (found a 0), accept.
  - (b) Scan right and left until a symbol 0 is found. If no 0 is found, reject. If there is one, cross it off.
  - (c) Again, scan right and left until another symbol 0 is found. If no 0 is found, reject. If there is one, cross it off.”

c)  $\{ w : w \text{ does not contain twice as many 0s as 1s} \}$

$M_2 =$  “On input  $w$ :

1. Run machine  $M_1$  on input  $w$ . Answer the opposite, that is, if  $M_1$  accepts, reject and if  $M_1$  rejects, accept.”

(Notice that machine  $M_2$  is the machine that decides the previous problem).

**3.10** This problem is a little different from the equivalence proofs we've seen in that the given new model - **write-once** TM seems to have less power instead of more power than the ordinary TM model. So we need to show that given an ordinary TM  $M$ , we can use an equivalent **write-once** TM  $S'$  to simulate  $M$ . Following the hint, we first solve the problem when we can write **twice** on the tape.

$S =$  "On input  $w = w_1...w_n$ :

1. Place a dot mark on top of the symbol in the cell the head points to. (This dot will represent the next symbol to read by the head).
2. For each move that require reading a symbol (say  $a$ ), simulate the move as it would require reading the symbol with a dot (say  $\dot{a}$ ). That is, for each transition of the form  $\delta(q, a) = (r, b, D)$ , where  $D \in \{R, L\}$ , simulate it as it were  $\delta(q, \dot{a}) = (r, b, D)$ .
3. To simulate a single move where  $M$  attempts to write to a tape cell (say write  $b$  after reading  $\dot{a}$  on the cell), go all the way right till blank cells are reached. Write down a special symbol (say  $\#$ ) and copy the input to the right of symbol  $\#$  as follows:
  - (a) When copying, cross each symbol off except the dotted symbol (which is not overwritten).
  - (b) When we are about to copy the cell previous to the one with the dotted symbol, there are two cases, depending on whether the move was leftwards or rightwards:
    - i. If the move was leftwards, when copying the symbol immediate to the left of the dotted symbol, write down the same symbol with a dot. Then, move right and write down the new symbol the machine attempted to write in step 3 (say  $b$ ) (notice there is no dot). Then finish copying the rest of the input.
    - ii. If the move was rightwards, write down the symbol to the left of the dotted symbol unmodified, then move right and write down the new symbol the machine attempted to write in step 3 (say  $b$ ). Then finish copying the rest of the input.
4. Locate the head over the dotted symbol and continue the simulation.
5. If during the simulation, the machine attempts to move past a symbol  $\#$ , do not allow it, and move the head to the right of it. (That is, symbol  $\#$  represent the leftmost end of the tape).
6. Repeat the simulation above every time  $M$  attempts to write a cell.

Notice that the above machine writes each tape cell at most two times (the first time is when a symbol is copied "into" a cell and second, when it's crossed off).

But this is not enough. We need to design a Turing machine  $S'$  that writes at most **once** each cell. In order to do that, we use the above machine  $S$  to build a new one  $S'$  in which we represent each tape cell of  $S$  as *two* cells in  $S'$ . To write in a (double) cell twice, we proceed as follows: the "left" cell is used for the first time  $S$  writes its the tape cell and the "right" cell is used the second time  $S$  writes on the same cell. When  $S'$  reads a (double) cell, if the "right" cell is blank the symbol read is taken to be whatever is on the "left" cell. If the "right" cell is a symbol  $s$  not blank, the symbol read is taken to be  $s$ . (This transformation can be seen as defining  $S'$  the same as  $S$  but replacing the "low-level" reading/writing functions with these new ones.)

It is possible to see the machine  $S'$  can simulate  $M$  using solely a **write-once** tape. Therefore, since a (regular) Turing machine can always simulate a **write-once** Turing machine (see why?), **write-once** Turing machines are equivalent to (regular) Turing machines.

**3.14** Show that the collection of decidable languages is closed under the following operations. Recall that a language is decidable if there exists a decider (Turing machine) that for any input  $w$  either accepts or rejects it (never loops).

a) union.

We want to prove that, for any two decidable languages  $L_1$  and  $L_2$ , there is a decider  $M$  for the language given by the union of them  $L = L_1 \cup L_2$ . Let  $M_1$  be the decider for language  $L_1$  and let  $M_2$  be the decider for language  $L_2$ . The following machine is a decider for language  $L$ :

$M =$  “On input  $w$ :

1. First run  $M_1(w)$ . Then run  $M_2(w)$ .
2. If any of the machines accept, accept. Otherwise, reject.”

b) concatenation.

We want to prove that, for any two decidable languages  $L_1$  and  $L_2$ , there is a decider  $M$  for the language given by the concatenation of them  $L = L_1 \circ L_2 = \{ w : w = w_1w_2 \text{ and } w_1 \in L_1, w_2 \in L_2 \}$ . Let  $M_1$  be the decider for language  $L_1$  and let  $M_2$  be the decider for language  $L_2$ . The following machine  $M$  is a decider for language  $L$ :

$M =$  “On input  $w$ :

1. Nondeterministically “split”  $w$  into two strings  $w_1$  and  $w_2$  such that  $w = w_1w_2$ .
2. Run  $M_1(w)$ . If rejects, reject. (Otherwise, continue).
3. Run  $M_2(w)$ . If accepts, accept. Otherwise, reject.”

Note: the fact that machine  $M$  is nondeterministic is irrelevant since it can always be transformed into an equivalent deterministic Turing machine.

c) star.

We want to prove that, for any decidable language  $L$ , there is a decider  $S$  for the language given by the star operation  $L^*$ . Let  $M$  be the decider for language  $L$ . The following machine  $S$  is a decider for language  $L^*$ :

$S =$  “On input  $w$ :

1. If  $w = \epsilon$  accept.
2. Nondeterministically “split”  $w$  into several strings  $w_1$  and  $w_2, \dots, w_k$  such that  $w = w_1w_2 \dots w_k$ .
3. For each  $i$  from  $i = 1$  up to  $i = k$  do the following:
  - (a) Run  $M(w_i)$ .
4. If all runs accept, accept. Otherwise, reject.”

Note: machine  $S$  is nondeterministic. As before, it can be transformed into an equivalent deterministic Turing machine.

d) complementation.

We want to prove that, for any decidable language  $L$ , there is a decider  $N$  for the language given by the complement operation  $A = \overline{L}$ . Let  $M$  be the decider for language  $L$ . The following machine  $N$  is a decider for language  $A$ :

$N =$  "On input  $w$ :

1. Run  $M(w)$ . Answer the opposite."

e) intersection.

We want to prove that, for any two decidable languages  $L_1$  and  $L_2$ , there is a decider  $M$  for the language given by the intersection of them  $L = L_1 \cap L_2$ . Let  $M_1$  be the decider for language  $L_1$  and let  $M_2$  be the decider for language  $L_2$ . The following machine is a decider for language  $L$ :

$M =$  "On input  $w$ :

1. First run  $M_1(w)$ . Then run  $M_2(w)$ .
2. If both machines accept, accept. Otherwise, reject."

**3.15** A language  $A$  is Turing-recognizable means that there is a TM  $M$  which recognizes  $A$ , i.e. for all strings  $w \in \Sigma^*$

- If  $w \in A$  then  $M(w)$  accepts
- If  $w \notin A$  then  $M(w)$  rejects or loops

a) union.

We need to show that if  $A_1, A_2$  are both recognizable, then  $A = A_1 \cup A_2$  is also recognizable. Given TM  $M_1$  that recognizes  $A_1$  and TM  $M_2$  that recognizes  $A_2$ , we give the following TM  $M$  to recognize  $A = A_1 \cup A_2$ :

$M =$  "On input string  $w$ :

1. Run  $M_1(w)$  and  $M_2(w)$  in parallel
2. If either one accepts, accept
3. If both reject, reject

Correctness: If  $w \in A_1 \cup A_2$ , either  $M_1$  or  $M_2$  will halt and accept at some point, and  $w \in L(M)$ . If  $w \notin A_1 \cup A_2$ , neither  $M_1$  or  $M_2$  will accept. They reject or loop forever. In either case,  $M$  will reject or loop forever.

b) concatenation.

We need to show that if  $A_1, A_2$  are both recognizable, then  $A = A_1 \circ A_2$  is also recognizable. Given TM  $M_1$  that recognizes  $A_1$  and TM  $M_2$  that recognizes  $A_2$ , we give the following TM  $M$  to recognize  $A = A_1 \circ A_2$ :

$M =$  "On input string  $w$ :

1. Run in parallel for all possible string sequences  $x_1, x_2 \in \Sigma^*$  satisfying  $w = x_1x_2$  (only a finite number of them), synchronized to run one step at a time for all of them

- run  $M_1(x_1)$  and  $M_2(x_2)$  simultaneously
  - If both accept, accept.
2. End while
  3. Reject

Correctness: If  $w \in A_1 \circ A_2$ , then there is some  $x_1, x_2 \in \Sigma^*$  s.t.  $w = x_1x_2$  and that  $x_1 \in A_1$ ,  $x_2 \in A_2$ . When we run these two computations for every possible division  $w = x_1x_2$  in parallel, the two computations for some division will halt at some point with  $M_1$  accepting  $x_1$  and  $M_2$  accepting  $x_2$ , and so  $w \in L(M)$ . If  $w \notin A_1 \cdot A_2$ ,  $M$  will reject or loop forever.

c) star.

We need to show that if  $A_1$  is recognizable, then  $A = A_1^*$  is also recognizable. Given TM  $M_1$  that recognizes  $A_1$ , we give the following TM  $M$  to recognize  $A = A_1^*$ :

$M =$  "On input string  $w$ :

1. If  $w = \epsilon$ , then accept and return
2. Run in parallel for all possible string sequences  $x_1, \dots, x_k \in \Sigma^* - \{\epsilon\}$  satisfying  $w = x_1 \dots x_k$  (only a finite number of them), synchronized to run one step at a time for all of them
  - run  $M(x_1), \dots, M(x_k)$  simultaneously
  - when all  $k$  of them accept, accept
3. EndFor
4. Reject

Correctness: If  $w \in A_1^*$ , then there is some  $x_1, \dots, x_k \in \Sigma^*$  s.t.  $w = x_1x_2 \dots x_k$  where  $1 \leq k \leq |w|$  and  $x_i \in A_1$  for all  $i$ . When we run these  $k$  computations for every possible division  $w = x_1 \dots x_k$  in parallel, the computations for some division will halt at some point with  $M_1$  accepting  $x_i$  for all  $i$ , and so  $w \in L(M)$ . If  $w \notin A_1^*$ , no such division exists, and  $M$  will reject or loop forever.

d) intersection.

We need to show that if  $A_1, A_2$  are both recognizable, then  $A = A_1 \cap A_2$  is also recognizable. Given TM  $M_1$  that recognizes  $A_1$  and TM  $M_2$  that recognizes  $A_2$ , we give the following TM  $M$  to recognize  $A = A_1 \cap A_2$ :

$M =$  "On input string  $w$ :

1. Run  $M_1(w)$  and  $M_2(w)$  simultaneously
2. If both accept, accept.
3. Otherwise, reject.

Correctness: If  $w \in A_1 \cap A_2$ , both  $M_1$  and  $M_2$  will halt and accept at some point, and so  $w \in L(M)$ . If  $w \notin A_1 \cap A_2$ , either  $M_1$  or  $M_2$  will reject or loop. In either case,  $M$  will reject or loop forever.

**4.1** The following questions are related to the DFA  $M$  depicted in problem 4.1 (page 168) in the textbook.

1. Is  $\langle M, 0100 \rangle \in A_{DFA}$  ?  
Yes, because  $M(0100)$  accepts.
2. Is  $\langle M, 011 \rangle \in A_{DFA}$  ?  
No, because  $M(011)$  rejects.
3. Is  $\langle M \rangle \in A_{DFA}$  ?  
No, because  $\langle M \rangle$  is not a correct encoding of a Turing machine and a string.
4. Is  $\langle M, 0100 \rangle \in A_{REG}$  ?  
No, because  $\langle M, 0100 \rangle$  is NOT a correct encoding of a *regular expression* and a string.
5. Is  $\langle M \rangle \in E_{DFA}$  ?  
No, because there exists an input string  $w = 0100$  such that  $M(0100)$  accepts (and  $L(M)$  is not empty).
6. Is  $\langle M, M \rangle \in EQ_{DFA}$  ?  
Yes. Notice that  $EQ_{DFA}$  contains only strings  $\langle M_1, M_2 \rangle$  such that  $M_1$  and  $M_2$  are Turing machines that generate the same language, that is,  $L(M_1) = L(M_2)$ . In our case  $M_1 = M_2 = M$ , so the answer is yes.

**4.2** Formulate the problem as the following language:

$$L = \{ \langle A, R \rangle \mid \text{DFA } A \text{ is equivalent to the regular expression } R \}$$

Let  $T$  be the following TM to decide  $L$ :

$T =$  "On input  $\langle A, R \rangle$ :

1. Convert  $R$  into an equivalent DFA  $B$
2. Run TM  $F$  from Theorem 4.5 (which decides  $EQ_{DFA}$ ) on input  $\langle A, B \rangle$
3. If  $F$  accepts, accept; if  $F$  rejects, reject

Therefore  $L$  is decidable.

**4.12** Show that the language  $L = \{ \langle G \rangle : G \text{ is a CFG over } \{0, 1\} \text{ and } 1^* \cap L(G) \neq \emptyset \}$  is decidable.

The solution follows the same idea of Theorem 4.7 (page 157) from the textbook. The following machine decides  $L$ .

$M =$  "On input  $\langle G \rangle$ :

1. Mark the terminal symbol 1.
2. Repeat until no new variables get marked:
  - (a) Mark any variable where  $G$  has a rule  $A \rightarrow U_1 U_2 \cdots U_k$  and *each* symbol  $U_1, \dots, U_k$  has already been marked.
3. If the start symbol is marked, accept. Otherwise, reject.

(It's possible to prove it works by induction on the length of a derivation of  $w$ , for any  $w \in L(G)$ ).

**4.19** Let  $T$  be the following TM to decide  $S$ :

$T =$  "On input  $\langle M \rangle$ :

1. Convert  $M$  into another DFA  $M'$  which accepts the language  $L(M)^R$
2. Run TM  $F$  from Theorem 4.5 (which decides  $EQ_{DFA}$ ) on input  $\langle M, M' \rangle$
3. If  $F$  accepts, accept; if  $F$  rejects, reject

Therefore  $S$  is decidable. The construction of  $M'$  can be done by reversing the arrows in  $M$ . The old start state now becomes the only new accept state. A new start state needs to be added with  $\epsilon$  transitions to each old accept state. Then we need to convert this NFA into an equivalent DFA  $M'$ .

- 5.1** We need to show is that  $EQ_{CFG}$  is undecidable. Recall that  $EQ_{CFG} = \{ \langle G, H \rangle \mid G, H \text{ are CFGs and } L(G) = L(H) \}$ . We prove it by reducing  $ALL_{CFG}$  to this language. This is easy if we remember that a very simple grammar can generate  $\Sigma^*$ .

Thus, for contradiction's sake, let  $D$  be a decider for  $EQ_{CFG}$ . Then, the following machine decides  $ALL_{CFG}$ :

$M =$  "On input  $\langle G \rangle$ :

1. Let  $H$  be the grammar that generates  $\Sigma^*$ , that is,  $(\{ S \}, \Sigma, R, S)$  where  $R$  is the set of rules  $\{ S \rightarrow 0S \mid 1S \mid \epsilon \}$  (assume  $\Sigma = \{0, 1\}$ ).
2. Run  $D(\langle G, H \rangle)$ .
3. If accept, accept. Otherwise, reject.

- 5.2** What we need to show is that  $\overline{EQ_{CFG}} = NEQ_{CFG}$  is Turing recognizable.  $\overline{EQ_{CFG}}$  accepts any  $\langle M_1, M_2 \rangle$  when  $L(M_1) \neq L(M_2)$ . Recall that  $A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \}$  is decidable. Let  $R_A$  be the TM that decides  $A_{CFG}$ . Let  $R$  be the following TM to recognize  $\overline{EQ_{CFG}}$ :

$R =$  "On input  $\langle M_1, M_2 \rangle$ :

1. enumerate  $x \in \Sigma^*$  one by one
2. if  $R_A(\langle M_1, x \rangle)$  accepts and  $R_A(\langle M_2, x \rangle)$  rejects, then accept and return

Since  $R$  recognizes  $\overline{EQ_{CFG}}$ ,  $EQ_{CFG}$  is co-Turing-recognizable.

- 5.7** Show that if  $A$  is Turing-recognizable and  $A \leq_m \overline{A}$ , then  $A$  is decidable.

**Note:** This problem requires knowledge of "mapping reductions" which has not been covered so far; please disregard this problem when preparing for the Quiz.

We use the fact that a language  $A$  is decidable if and only if  $A$  is Turing-recognizable and its complement ( $\overline{A}$ ) is Turing-recognizable.

We know that  $A \leq_m \overline{A}$  for some language  $A$ . By definition, this means that there exists a computable function  $f$  such that for all  $w$ ,  $w \in A$  if and only if  $f(w) \in \overline{A}$ . We can rewrite the latter as " $w \in \overline{A}$  if and only if  $f(w) \in A$ ". Therefore,  $\overline{A} \leq_m A$ . Since  $A$  is Turing-recognizable, by Theorem 5.22 (page 193 in the textbook),  $\overline{A}$  is also Turing-recognizable. By Theorem 4.16 (page 167 in the textbook),  $A$  is decidable since it is Turing-recognizable and co-Turing-recognizable. (Recall that  $A$  is co-Turing-recognizable if  $\overline{A}$  is Turing-recognizable).

**5.14** Formulate the language to be:

$L = \{\langle M, w \rangle \mid M \text{ on input } w \text{ ever attempts to move its head left when its head is on the leftmost tape cell}\}$

To show that  $L$  is undecidable, the proof is by contradiction (as usual). Assume  $L$  is decidable. Let  $R$  be a Turing machine that decides  $L$ . We can then construct a Turing machine  $S$  that decides  $A_{TM}$  as follows:

$S =$  "On input  $\langle M, w \rangle$ , where  $M$  is a TM and  $w$  is a string:

1. Construct the following TM  $M'$   
 $M' =$  "On input  $x$ :
  - run  $M$  on  $x$  and move its head to the left from the leftmost cell ONLY when  $M$  accepts"
2. Run  $R$  on input  $\langle M', w \rangle$
3. If  $R$  accepts, accept; if  $R$  rejects, reject."

But we know that  $A_{TM}$  is undecidable  $\Rightarrow$  contradiction! Therefore  $L$  is also undecidable.

The construction of  $M'$  needs a little more clarification. To ensure that during its computation  $M'$  doesn't move the head left from the leftmost position,  $M'$  first shifts the input  $w$  one position to the right, and place a special symbol on the leftmost tape cell. The computation of  $M'$  starts with the head on the second tape cell. If during its computation  $M'$  ever attempts to move its head to the leftmost tape cell,  $M'$  finds out that it did so by reading the special symbol and then puts the head back to the second cell, and continues its execution. If  $M$  would enter an accept state, the  $M'$  enters a loop that forces the head to move always to the left. By doing the above, we have made sure that  $M'$  moves its head to the left from the leftmost cell ONLY when  $M$  accepts its input.

**5.16** Formulate the language as:

$WB2T_{TM} = \{\langle M \rangle \mid M \text{ is a 2-tape Turing machine and there exists string } w \text{ such that } M \text{ on input } w \text{ (at some point during the computation) writes } \sqcup \text{ on the second tape }\}$ .

We need to prove  $WB2T_{TM}$  is undecidable by reducing  $A_{TM}$  to it. Assume there exists a decider  $D$  for  $WB2T_{TM}$ . The following machine  $S$  decides  $A_{TM}$ :

$S =$  "On input  $\langle M, w \rangle$ :

1. Create the following machine:  
 $M' =$  "On input  $x$ :
  1. Run  $M(w)$ .
  2. If it accepts, write  $\sqcup$  on the second tape and accept.
  3. Otherwise, (*do not write anything* and) reject.
2. Run  $D(\langle M' \rangle)$ .
3. If accept, accept. Otherwise, reject."

Let's do a sanity check: (HIGHLY recommended!) How do we know  $S$  is correct, ie. that  $S$  decides  $A_{TM}$ ? There are two cases:

1. If  $M \in A_{TM}$  then  $M$  on input  $w$  will accept, and therefore,  $M'$  (on any input) will write  $\sqcup$  on the second tape. Then,  $D(\langle M' \rangle)$  will accept, making  $S$  accept.
2. If  $M \notin A_{TM}$  then  $M$  on input  $w$  will loop or reject. If  $M(w)$  loops,  $M'$  never gets a chance to write anything to the second tape, and if  $M(w)$  rejects,  $M'$  does not write anything either. In both cases,  $M'$  does not write  $\sqcup$  to the second tape. Therefore,  $D(\langle M' \rangle)$  will reject, making  $S$  reject.