Turing Machines Solutions

a. Consider the Turing Machine

$$(\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_{acc}, q_{rej}\}, \{a, b\}, \{a, b, X, \sqcup\}, \delta, q_0, q_{acc}, q_{rej})$$

where δ is defined as:

(q, s)	$\delta((q, s))$	(q, s)	$\delta((q, s))$	(q, s)	$\delta((q, s))$	(q, s)	$\delta((q, s))$
(q_0, a)	$(q_1, \sqcup, R)$	(q_2, a)	$(q_{rej}, \sqcup, R)$	(q_4, a)	(q_4, a, L)	(q_6, a)	$(q_{rej}, \sqcup, R)$
(q_0, b)	$(q_{rej}, \sqcup, R)$	(q_2, b)	(q_3, b, L)	(q_4, b)	$(q_{rej}, \sqcup, R)$	(q_6, b)	$(q_{rej}, \sqcup, R)$
(q_0, X)	$(q_{rej}, \sqcup, R)$	(q_2, X)	$(q_{rej}, \sqcup, R)$	(q_4, X)	(q_5, X, R)	(q_6, X)	(q_6, X, L)
$(q_0, \sqcup)$	$(q_{rej}, \sqcup, R)$	$(q_2, \sqcup)$	$(q_6, \sqcup, L)$	$(q_4, \sqcup)$	$(q_5, \sqcup, R)$	$(q_6, \sqcup)$	$(q_{acc}, \sqcup, R)$
(q_1, a)	(q_1, a, R)	(q_3, a)	(q_4, a, L)	(q_5, a)	(q_1, X, R)		
(q_1, b)	(q_2, X, R)	(q_3, b)	$(q_{rej}, \sqcup, R)$	(q_5, b)	$(q_{rej}, \sqcup, R)$		
(q_1, X)	(q_1, X, R)	(q_3, X)	(q_3, X, L)	(q_5, X)	$(q_{rej}, \sqcup, R)$		
$(q_1, \sqcup)$	$(q_{rej}, \sqcup, R)$	$(q_3, \sqcup)$	$(q_{rej}, \sqcup, R)$	$(q_5, \sqcup)$	$(q_{rej}, \sqcup, R)$		

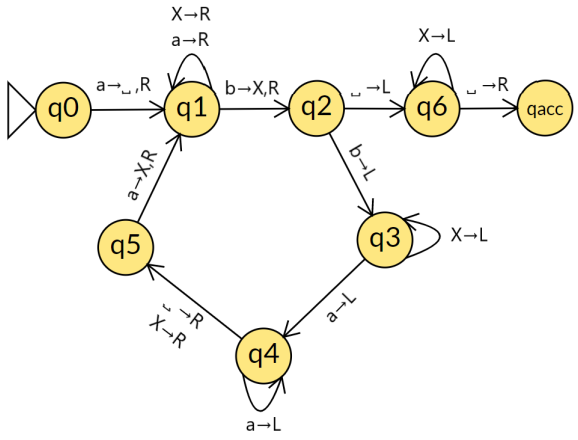
and

$$\delta((q_{acc}, s)) = (q_{acc}, \sqcup, L) \quad and \quad \delta((q_{rej}, s)) = (q_{rej}, \sqcup, L) \quad \text{for each } s \in \{a, b, X, \sqcup\}$$

i Draw the state diagram of this Turing machine. For clarity, in your diagram you may leave out all outgoing transitions from q_{acc}, and you can leave out q_{rej} entirely (including all transitions in and out).

(Hint: q_1, q_2, q_3, q_4, q_5 form a pentagon so draw that first.)

Solution:



We use the convention the book establishes such that the label $a \to \sqcup, R$ signifies that the head reads a, writes $\sqcup$, and moves the head to the right. For clarity we use the shorthand $X \to R$ to mean that the machine moves the tape head to the right when reading X but doesn't alter the tape.

ii Trace through the computation of this machine on input $aabb$ by listing each configuration of the machine in turn.

Solution:

Note: We get the definition of a configuration and how to represent one from the textbook: "As a Turing machine computes, changes occur in the current state, the current tape contents, and the current head location. A setting of these three items is called a **configuration** of the Turing machine. Configurations often are represented in a special way. For a state q and two strings u and v over the tape alphabet Γ, we write uqv for the configuration where the current state is q, the current tape contents is uv, and the current head location is the first symbol of v. The tape contains only blanks following the last symbol of v. For example, $1011q_701111$ represents the configuration when the tape is 101101111, the current state is $q7$, and the head is currently on the second 0. Figure 3.4 depicts a Turing machine with that configuration"

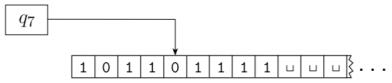


FIGURE **3.4**
A Turing machine with configuration $1011q_701111$

With this in mind, the Turing machine's configurations on input $aabb$ are:
$q_0aabb \to \llcorner q_1abb \to \llcorner aq_1bb \to \llcorner aXq_2b \to \llcorner aq_3Xb \to \llcorner q_3aXb \to q_4\llcorner aXb \to \llcorner q_5aXb \to \llcorner Xq_1Xb \to \llcorner XXq_1b \to \llcorner XXXq_2 \to \llcorner XXq_6X \to \llcorner Xq_6XX \to \llcorner q_6XXX \to q_6\llcorner XXX \to \llcorner q_{acc}XXX$

iii Will this machine always halt? Why or why not?

Solution:

Yes. We will show that for any input, there is no place the machine could loop forever (so the computation will halt/ terminate in either an accept or reject state).

- We see that we can't loop on state q_1 forever. This is because we start with a finite number of as and Xs on our tape, and never replace any $\llcorner$s (our only infinite resource) with as or Xs anywhere. So we will always have a finite number of as and Xs on our tape.
- Similarly, we can't loop on state q_3, q_4, or q_6 forever.
- We can't loop through states q_1, q_2, q_3, q_4, q_5 forever. This is because we start with a finite number of as and bs on our tape, in particular we can say $\#a\text{s} + \#b\text{s}$ is an integer. In order to loop around, we need to transition to state q_2 (decreasing the number of bs by one) and we also need to transition to state q_1 (decreasing the number of as by one). At no point in the loop do we increase the number of as or bs. So after one loop around, the $\#a\text{s} + \#b\text{s}$ has gone down by 2. Since we can't have a negative number of as and bs on the tape, we can't loop here forever.

iv What language does this machine recognize?

Solution:

This machine recognizes the language $\{a^kb^k \mid k \geq 1\}$.
We can come up with an implementation-level description of the TM by thinking of what each state is doing:

1. Read an a and replace it with $\llcorner$. If the first symbol of our string is not an a, reject. (We do this in state q_0)

2. Scan the tape (from left to right) until we find a b. Then cross off that b. If we do not find a b, reject. (We do this in state q_1)
3. If the next symbol (right after the b we just found) is an a or an X, reject. If it's $\sqcup$, go to step 7. If it's a b, continue to step 4. (We do this in state q_2)
4. Scan the tape (from right to left - starting at where we found the b) until we find an a. If we cannot find one, reject. (We do this in state q_3)
5. Starting from the a we found in the previous step, continue going left on the tape as long as we see as. As soon as we see an X or $\sqcup$, continue to step 6. (We do this in state q_4)
6. Cross off the last a we saw (which is the first a in the cluster of as we just traversed). Then repeat this process starting from step 2. (We do this in state q_5)
7. Starting from the symbol to the left of the $\sqcup$ we found in the previous step, continue going left on the tape as long as we see Xs. As soon as we see $\sqcup$, accept. If we see any as or bs, reject. (We do this in state q_6)

At a higher level, the TM is crossing off the first a, then crossing off the first b, then crossing of the second a, then crossing off the second b, and so on so forth. Note that it is rejecting if we ever encounter a number of incorrect looking inputs (for example having any as follow a b).

b. Give an implementation-level description of a TM which decides the language $\{w \mid w$ does not contain (exactly) twice as many 0s as 1s$\}$.
Solution:

1. Scan the tape (from left to right) until we find a 1 and cross it off. If we do not find a 1, go to step 5. Otherwise move the head back to the beginning of the tape.
2. Scan the tape (from left to right) until we find a 0, then cross off that 0. If we do not find a 0, accept.
3. Scan the tape again (from left to right starting from where we left off) until we find a 0, then cross off that 0. If we do not find a 0, accept.
4. Move the head back to the beginning of the tape and repeat this process starting from step 1.
5. Move the head back to the beginning of the tape and scan the tape (from left to right) until we find a 0. If we do not find a 0, reject. Otherwise, accept.

c. (Sipser Problem 3.22) Let A be the language containing only the single string s, where

$$s = \begin{cases} 0 \text{ if life never will be found on Mars.} \\ 1 \text{ if life will be found on Mars someday.} \end{cases}$$

Is A decidable? Why or why not? For the purposes of this problem, assume that the question of whether life will be found on Mars has an unambiguous yes or no answer.
Solution:
Yes, A is decidable. Since the question of whether life will be found on Mars has an unambiguous yes or no answer, we know that A is either equal to $\{0\}$, or it is equal to

{1}. In either case, we can easily build a decider (a TM that always halts) that recognizes A. So the language A is decidable.

d. Is the class of decidable languages closed under complement?
Solution:
Yes. Given a decidable language L, we know there exists a TM M that decides L (remember that means the TM accepts every string that is in L and rejects every string that is not in L - it does not loop forever in any input). We can construct the TM M' that decides $\overline{L}$ simply by swapping the accept and the reject states from M. This way, we know that:

- for any $s \in \overline{L}$, $s \notin L$, the computation of s on M will end q_{rej}, which means the computation of s on M' will end q_{acc}, so M' accepts s as desired.
- for any $s \notin \overline{L}$, $s \in L$, the computation of s on M will end q_{acc}, which means the computation of s on M' will end q_{rej}, so M' rejects s as desired.

e. Give an implementation-level description of how to duplicate a string on the tape, i.e. if the tape starts with w then it should end with ww. *(You are not describing a complete machine here - it does not accept or reject at the end. The point is to have a useful subroutine to call upon in later problems.)*
Solution:

1. If our input is ε, we are done. Otherwise, keep track of the first symbol on our tape and mark it (for example we can mark the symbol a with $\hat{a}$)
2. Scan the tape (from left to right) until we find $\sqcup$, then replace the $\sqcup$ with a boxed version of the symbol we just saw (for example we can mark the symbol a with $\boxed{\text{a}}$)
3. Scan the tape (from right to left from where we left it) until we find a 'marked' symbol, then replace the marked symbol with its corresponding not marked symbol and move the tape head right once. If we are on a boxed symbol, replace it with the unboxed version and we are done. Otherwise, continue.
4. Keep track of the symbol our tape head is on and mark it
5. Scan the tape (from left to right) until we find $\sqcup$, then replace the $\sqcup$ with the symbol we just saw
6. Scan the tape (from right to left from where we left it) until we find a 'marked' symbol, then replace the marked symbol with its corresponding not marked symbol and move the tape head right once. If we are on a boxed symbol, replace it with the unboxed version and we are done. Otherwise, repeat this process starting from step 4.

f. Is the class of decidable languages closed under union? intersection? Give implementation-level descriptions for any TMs you need for this problem, except that you can use the command "do whatever machine M would do" or minor variants thereof (which would usually be too high level to really count as "implementation-level").
Solution: Yes and yes.
Given two decidable languages L_1, L_2, we will construct a TM M which decides the language $L_1 \cup L_2$ and a TM N which decides the language $L_1 \cap L_2$. Since L_1 is a decidable

language, we know there exists a TM M_1 which decides L_1. Similarly, we can say the TM M_2 decides L_2.
Note that we can easily change the subroutine from part e to make the tape go from w to $w\#w$. (In step 2 replace the $\sqcup$ with # and just write the nonboxed symbol to the right of it. And in step 6, instead of checking if we're on a boxed symbol, we check if we're on a #.)
Also note that we can make slight modifications to a TM so that it never moves the tape head left past where the tape head starts off. We can also easily change make slight changes to a TM so that it never writes a $\sqcup$. We'll make these changes to M_1 and call the resulting TM M_1'.
Closure under union:
M:

1. Scan the tape (from left to right) until we find a #. Then move the tape head right once.
2. Do whatever M_1' would do, except whenever it rejects, continue to step 3 instead.
3. Move the tape head back to the beginning of the tape and scan the tape (from left to right) until we find a #. Then replace every non-blank symbol on the tape with $\sqcup$. Once we reach a blank symbol, move the tape head back to the beginning of the tape.
4. Do whatever M_2 would do.

Closure under intersection:
N:

1. Scan the tape (from left to right) until we find a #. Then move the tape head right once.
2. Do whatever M_1' would do, except whenever it accepts, continue to step 3 instead.
3. Move the tape head back to the beginning of the tape and scan the tape (from left to right) until we find a #. Then replace every non-blank symbol on the tape with $\sqcup$. Once we reach a blank symbol, move the tape head back to the beginning of the tape.
4. Do whatever M_2 would do.

g. There exist languages which are recognizable but not decidable. (We will prove this later.) Do you think the class of recognizable languages is closed under complement? (We haven't developed all the tools to fully justify an answer yet; just give your intuitive reasoning.)
Solution: No, the class of recognizable languages is not closed under complement.
Intuition: Note that a TM M that recognizes a language A has the following behavior:

- For any $s \in A$, the computation of s on M will end in q_{acc} and M accepts.
- For any $s \notin A$, the computation of s on M might end in q_{rej} and M rejects, or the computation of s on M might loop forever.

So we note that the trick we did with swapping the accept and reject states to get M' in part d doesn't work here. This is because strings that loop forever on M will also loop

forever in M', and so those strings will neither be in $L(M)$ or in $L(M')$ (which doesn't match the behavior we wanted from M'). This points us to the issue we have when trying to build a TM that recognizes $\overline{A}$: how do we deal with strings that loop forever?

A **proof** that you should be able to follow using the fact (which we will prove later) that there exist languages which are recognizable but not decidable:
Assume towards a contradiction that the class of recognizable languages are closed under complementation. Then for any recognizable language A, there exists a TM M which recognizes A, and a TM M' which recognizes $\overline{A}$. So we can build the TM N that decides the language A.
N: "On input x:

1. For $i = 1, 2, 3, \ldots$
    - Run M on input x for i steps. If M accepts, accept.
    - Run M' on input x for i steps. If M' accepts, reject."

We know that for any string $s \in A$, M accepts after some finite number of steps, so N will accept s at some point. Similarly, we know that for any string $s \notin A$, $s \in \overline{A}$, so M' accepts after some finite number of steps, so N will reject s at some point. So N decides A. However A is an arbitrary recognizable language that we picked. So if we can build a decider for every recognizable language, then there does not exist a recognizable language which is not decidable, but this is a contradiction with the fact given in the problem statement.