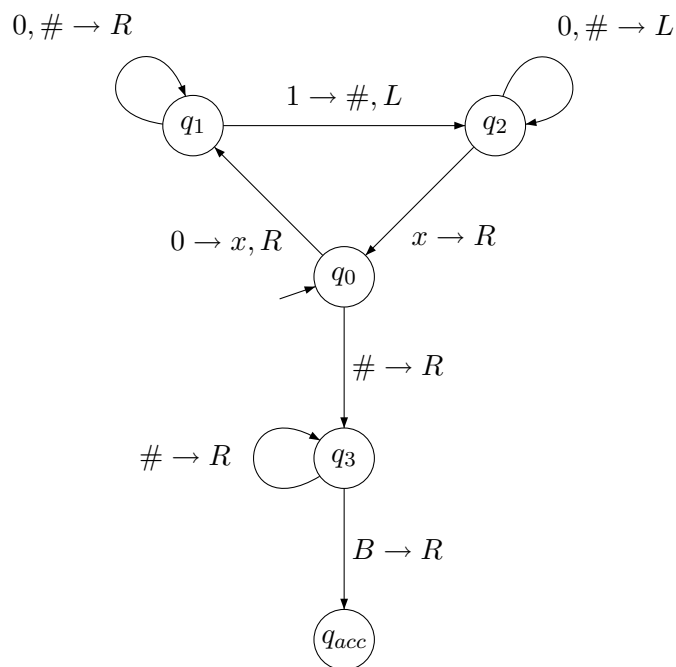


# CS 273: Intro to Theory of Computation, Fall 2007

## Problem Set 8 Solutions

### 1. TM tracing

Here is the state diagram of a simple TM, where  $\Sigma = \{0, 1\}$  and  $\Gamma = \{0, 1, x, \#, B\}$ . The accept state is  $q_{acc}$ . Any transition not shown in the diagram is assumed to go to the reject state,  $q_{rej}$ .



- What language  $L$  does this TM accept?
- Describe the pattern of how the TM moves its head while processing a string from  $L$ , and also state where it leaves its head when it halts.
- Trace the execution of the TM as it processes the string 00111. That is, show us the sequence of configurations that the TM goes through while processing this string.

**Solution:**

(a) The TM accepts the language  $L = \{0^n 1^n | n \geq 1\}$ . (Notice that B is one notation for the blank character.)

(b) Starting from the left end of the input, the TM repeatedly changes '0' to an 'x' moving to the right over any '0's or '#s until it sees a '1'. It then changes the '1' to a '#', and then moving left over any '#s and '0's until it see a 'x'. It then looks for a '0' to the immediate right, changes that to an 'x' and repeats the process again.

Once all the '0's and '1's are matched and changed to 'x's and '#s, the TM moves right until it finds a blank at the end of the input and accepts. The head is left on the second blank position at the end of the tape.

(c) The following are he configurations of the TM processing the string 00111. It starts at  $q_0 0111$ .

```

q000111
xq10111
x0q1111
xq20#11
q2x0#11
xq00#11
xxq1#11
xx#q111
xxq2##1
xq2x##1
xxq0##1
xx#q3#1
xx##q31

```

At this point the TM fails to have a next move, and rejects.

**2. Encodings**

In this problem we demonstrate a possible encoding of a TM using the alphabet  $\{0, 1, ;, | \}$  where  $|$  is the newline character. We encode  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$  as a string  $n|i|j|t|r|s|w$  where  $n, i, j, t, r, s$  are integers in binary representing  $|Q|, |\Sigma|, |\Gamma|, q_0, q_{accept}, q_{reject}$  and  $w$  represents  $\delta$  as described below. We adopt the convention that states are numbered from 0 to  $n - 1$ , the input alphabet symbols are numbered from 0 to  $i - 1$ , and the tape alphabet symbols are numbered from 0 to  $j - 1$  with  $j - 1$  representing the special blank symbol (therefore  $j > i$ ).

The string  $w$  represents  $\delta$  as follows. Each transition  $(q, a) \rightarrow (q', b, D)$  is represented as a 5-tuple  $q; a; q'; b; D$  where  $q, a, q', b$  are integers in binary and  $D$  is either 0 (move left) or 1 (move right). We adopt the convention that only useful transitions are represented and any transition not represented leads to the reject state. The string  $w$  consists of 5-tuples separated by  $|$ . Thus  $w = w_1|w_2| \dots |w_p$  where  $p$  is the number of useful transitions.

Here is the representation of a mystery Turing machine  $M$ , using this encoding. For ease of reading, we've shown | as an actual line break and given the integers in decimal rather than binary.

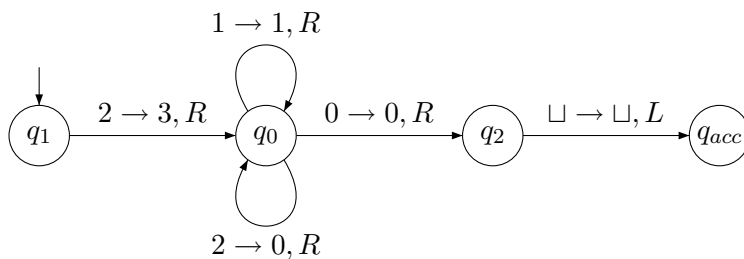
5  
 3  
 5  
 1  
 4  
 3  
 1;2;0;3;1  
 0;2;0;0;1  
 0;1;0;1;1  
 0;0;2;0;1  
 2;4;4;4;0

- (a) Draw a state diagram for the Turing machine  $M$ . Your diagram should omit the reject state and all transitions into it.
- (b) Give a string representation as above for a new TM which behaves like  $M$  but just before accepting, moves its head from the current position to the first position on the tape, then halts and accepts.

This problem is designed to help you gain a concrete understanding of encoding and how one can modify code of one TM to produce code for another TM (using a TM!).

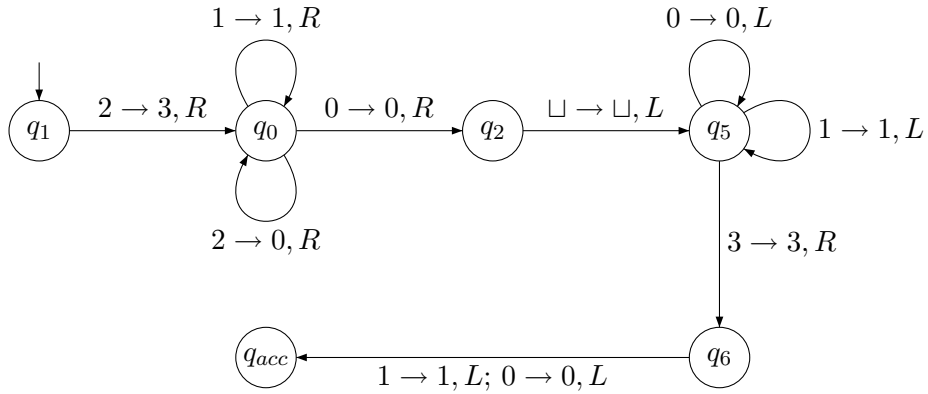
**Solution:**

(a) Here is the state diagram for  $M$ . I have translated the final tape symbol (4) into  $\sqcup$ , since it's supposed to be the blank character. I have also labelled state 4 as  $q_{acc}$ , since it's supposed to be the accept state. It's ok to have done slightly more or less translation of tape symbols into normal symbols, so long as we can easily tell what you meant.



(b) As  $M$  makes its initial rightward pass through the input, it changes the symbol at the first tape position to a 3 and writes a 0 or 1 over each subsequent position. So, to return to the start of the tape, we need to read past any 0's or 1's, until we find a 3. The first pass also ensured that there were at least two non-blank symbols on the tape: the initial 2 that turns into a 3 and a zero at the end of the input.

Here is the state diagram (not required by the problem statement) for one version of the modified TM:



The encoding of this new machine would be:

7  
3  
5  
1  
4  
3  
1;2;0;3;1  
0;2;0;0;1  
0;1;0;1;1  
0;0;2;0;1  
2;4;5;4;0  
5;0;5;0;0  
5;1;5;1;0  
5;3;6;3;1  
6;0;4;0;0  
6;1;4;1;0

### 3. Low-level TM design

Let  $\Sigma = \{a, b, c\}$ . Define  $L$  to be

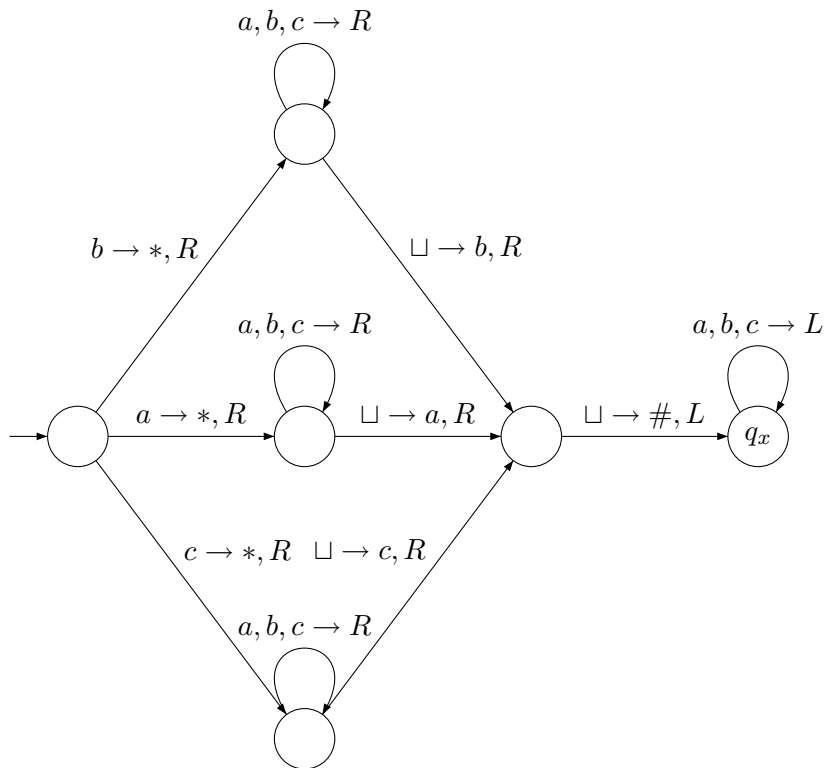
$$L = \{w \mid w \text{ has twice as many } b\text{'s as } a\text{'s and three times as many } c\text{'s as } a\text{'s}\}$$

Give the state diagram for a Turing machine that recognizes  $L$ . (Your diagram can omit the reject state and all transitions into it.)

**Solution:**

We assume that the input is marked on the left by  $*$  and on the right by  $\#$ . Indeed, if the input is not in this form, we can mark the first character by  $*$  and write the letter that we have overwritten at the end of the tape and then append a  $\#$ . This does not change whether the string is accepted or rejected since it keeps the number of  $a, b, c$  the same.

Here is a state diagram that accomplishes this patch to the input:



The main computation then works by first finding an  $a$ . For every  $a$  that it finds, it marks off (changes to  $x$ ) 2  $b$ 's and 3  $c$ 's. It repeats this procedure for every  $a$  that it finds. This guarantees for every  $a$ , we have 2  $b$ 's and 3  $c$ 's. If the input string is part of the language, then the TM will change every letter between  $*$  and  $\#$  into  $x$ . If this is the case, the TM will accept.



#### 4. High-level TM design

Design a Turing Machine which, when given input of the form  $0^a\$0^b$  will leave only  $\$0^c$  on the tape where  $c \in \mathbb{N}_0$  is the smallest number such that  $c^2 \equiv a \pmod{b}$ , and accept if such a number exists. If no such number exists, the machine should reject and leave nothing on the tape. For example, if the machine was given  $0^3\$0^5$ , it would reject and leave a blank tape. If the machine was given  $0^2\$0^7$ , it would leave  $\$0^3$  on the tape and accept (since  $3^2 = 9$  and  $9 \equiv 2 \pmod{7}$ ). Write your solution in pseudo-code, describing what the machine is doing to the tape at each stage.

**Solution:** This problem depends on two facts about modular arithmetic. The first is that we need only search through values  $c = 0, 1, 2, \dots, b - 1$ . To see this, note that each number  $n \in \mathbb{N}_0$  has a unique representation  $n = c + bk$ , where  $k \in \mathbb{N}_0$ , and  $0 \leq c < b$ . Thus  $n^2 = (c + bk)^2 = c^2 + 2cbk + b^2k^2 \equiv c^2 \pmod{b}$ , since  $b$  divides  $2cbk + b^2k^2 = b(2ck + bk^2)$ . The second fact is that if we want to know if two numbers  $p = c + mb$  and  $q = d + nb$ , where  $0 \leq c, d < b$ , then all we need to check is that  $c = d$ . To implement this as an algorithm, we can repeatedly subtract  $b$  from  $p$  and  $q$  until both are between 0 and  $b - 1$ , and then compare the remainders.

Now on to the problem. Many people wrote that it is sufficient to check all values  $c \leq \lfloor \sqrt{a+b} \rfloor$ .  $a = 2$ ,  $b = 31$  is a small counterexample. The smallest  $c$  such that  $c^2 \equiv 2 \pmod{31}$  is 8, but  $\lfloor \sqrt{2+31} \rfloor = 5 < 8$ . In fact, it suffices to check up all  $c \leq b/2$ , since  $(b-c)^2 = b^2 - 2cb + c^2 \equiv c^2 \pmod{b}$ , but the machine is simplified if we use  $b - 1$  as our bound. There are two ways to approach this problem. The first is to note that for all  $c \geq 1$ ,  $c^2 = 1 + 3 + \dots + (2c - 1)$ . This allows us to avoid writing a multiplication subroutine, and use only addition. The other is to just square each number  $0 \leq c < b$ , reduce modulo  $b$ , and then check if the result is equal to  $a$ . You could adapt the solution below to produce the latter through use of the multiplication algorithm found in Sipser. We will use five tapes in our Turing machine, with the output on the first tape. If  $b = 0$ , we adopt the convention that  $c^2 \equiv a \pmod{b}$  has no solutions (this case was not considered in the grading, and should have been disallowed by the problem statement).

*Step 1* Move everything on the tape 1 forward by 1 space, and add a  $\$$  to the beginning of the tape. Put the head at the beginning of the tape. Mark the beginning of each tape with a  $\$$ . For each 0 on the first tape between the first two  $\$$ 's, place a 0 at the end of the second tape. For each 0 after the second  $\$$ , put a 0 at the end of the third tape. At this point, the second tape contains  $0^a$ , and the second contains  $0^b$ . If  $b \neq 0$ , reduce the value on the second tape ( $a$ ) modulo the value on the third tape ( $b$ ). From this point on, we will call the value on tape two  $a$ .

*Step 2* (Special cases) If there are no zeros on the second tape ( $a = 0$ ), erase everything but the first  $\$$  on the first tape, and ACCEPT. (If  $a \equiv 0 \pmod{b}$ , then  $c = 0$  is a solution). If there are no zeros on the third tape, delete everything on all tapes and REJECT. (If  $b = 0$ , then we REJECT).

*Step 3* Add a single 0 to the end of the tape 4, and copy the contents of the tape 3 to tape 5. For each 0 on tape 2, remove one 0 from the end of tape 5. At this point, we have  $\$0^{b-a}$  on the fifth tape, which corresponds to subtracting  $a \pmod{b}$ . From this point on, we will call the contents of tape four  $c$ .

*Step 4* For each 0 on tape 4, add two 0's to the end of tape 5. Then remove one 0 from the end of tape 5. (This corresponds to adding  $2c - 1$  to the fifth tape, which by the above formula for calculating squares will bring us to the next square number). Reduce the contents of tape 5 modulo  $b$ . We now have  $c^2 - a \pmod b$  zeros on tape 5. If there are now no zeros on tape five, then we have a solution to the equation. (In this case, delete the contents of tape 1, and copy the contents of tape 4 to tape 1. Delete everything on all other tapes, and ACCEPT). Otherwise, go on to step 5.

*Step 5* Add a zero to the end of tape 4. If there are now the same number of zeros on tapes 3 and 4 (i.e. if  $c = b$ ), then we have checked all possibilities, so delete everything on all tapes, and REJECT. Otherwise, go to step 4.

Now we need to implement the helper functions that we used in the above procedure.

“ $< b$ ” - determine if the contents of tape  $k$  (written  $x$ ) are smaller than the contents of tape 3 (i.e.  $< b$ ): Place the heads of both tapes  $k$  and 3 at the beginning of their tapes. Advance the heads forward one step at a time until one of them reads a blank. If the head on tape 3 reads a blank before or at the same time as the head on tape  $k$ , then  $x \geq b$ . Otherwise,  $x < b$ .

“ $\pmod b$ ” - reduce the contents of tape  $k$  (written  $x$ ) modulo the contents of tape 3: If  $x < b$ , then we are done. Otherwise, for each 0 on tape 3, remove a 0 from the end of tape  $k$ . Now reduce the contents of tape  $k$  modulo the contents of tape 3 (i.e. recursively call this function). When this function is done, the contents of tape  $k$  will be  $y$ , where  $x = y + nb$  for some  $n$ , and  $0 \leq y < b$ .

## 5. TM simulation

Michigan's Computer Science department has invented a new kind of Turing machine called the MTM. An MTM can read not only the character at the head position, but also the character immediately to its left. Only the character at the head position can be rewritten. When the head is at the lefthand end of the tape, a blank is supplied in place of the non-existent character to the left.

- (a) What is the type signature (domain and range) for the transition function of an MTM?  
 (b) Show that an MTM accepts the same languages as a normal TM, by showing how to simulate an arbitrary MTM on a normal TM.

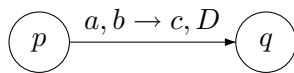
**Solution:**

(a)  $\delta : Q \times \Gamma \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

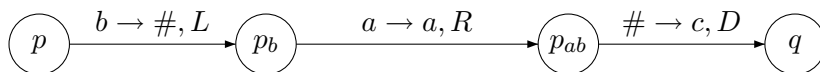
(b) Let  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$  be an MTM. We will construct a normal TM  $N = (Q', \Sigma, \Gamma', \delta', q_0, q_{accept}, q_{reject})$  that simulates  $M$ .

The idea is that each move of  $M$  will be simulated by a short sequence of moves: read character at current position, move left to read character at previous position, move back, then make the move stipulated by  $\delta$ . To handle the cases at the lefthand edge, we will overwrite the character at the current position with a  $\#$  (where  $\#$  is some character not in  $\Gamma$ ) before attempting to move left. If we still see a  $\#$  after the move, we know that we must have been at the left end of the tape.

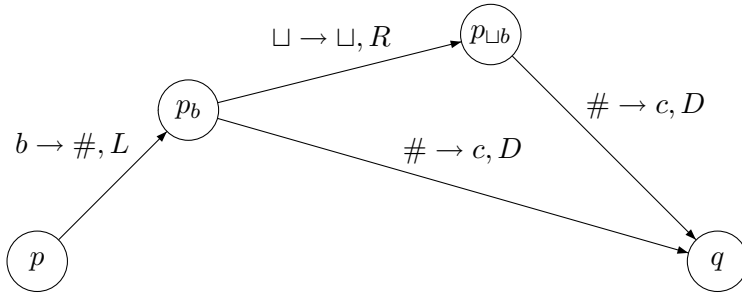
Suppose that part of  $M$ 's state diagram looked like this:



Then the corresponding part of  $N$ 's state diagram would look like this:



Except if  $a = \sqcup$ , in which case it looks like:



So, putting this into tuple notation:

$$\Gamma' = \Gamma \cup \{\#\}$$

$$Q' = Q \cup \{p_b \mid p \in Q\} \cup \{p_{ab} \mid p \in Q\}$$

and  $\delta'$  contains the following transitions (ignoring those into the reject state):

$$\delta'(p, b) = (p_b, \#, L) \text{ for every } b \in \Gamma$$

$$\delta'(p_b, a) = (p_{ab}, R) \text{ for every } a \in \Gamma$$

$$\delta'(p_b, \#) = (q, c, D) \text{ whenever } \delta(p, \sqcup, b) = (q, c, D)$$

$$\delta'(p_{ab}, \#) = (q, c, D) \text{ whenever } \delta(p, a, b) = (q, c, D)$$