

CS 334 – Fall 2004: Answers for Assignment 9

1. $[q_0]00 \vdash X[q_1]0 \vdash X0[q_1]$

The TM halts at this ID.

2, Here is the transition table for the TM:

state	0	1	B	X	Y
q0	(q2,X,R)	(q1,X,R)	(qf,B,R)	-	(q0,Y,R)
q1	(q3,Y,L)	(q1,1,R)	-	-	(q1,Y,R)
q2	(q2,0,R)	(q3,Y,L)	-	-	(q2,Y,R)
q3	(q3,0,L)	(q3,1,L)	-	(q0,X,R)	(q3,Y,L)
qf	-	-	-	-	-

In explanation, the TM makes repeated excursions back and forth along the tape. The symbols X and Y are used to replace 0's and 1's that have been cancelled one against another. The difference is that an X guarantees that there are no unmatched 0's and 1's to its left (so the head never moves left of an X), while a Y may have 0's or 1's to its left.

Initially in state q_0 , the TM picks up a 0 or 1, remembering it in its state (q_1 = found a 1; q_2 = found a 0), and cancels what it found with an X . As an exception, if the TM sees the blank in state q_0 , then all 0's and 1's have matched, so the input is accepted by going to state q_f .

In state q_1 , the TM moves right, looking for a 0. If it finds it, the 0 is replaced by Y , and the TM enters state q_3 to move left and look for an X . Similarly, state q_2 looks for a 1 to match against a 0.

In state q_3 , the TM moves left until it finds the rightmost X . At that point, it enters state q_0 again, moving right over Y 's until it finds a 0, 1, or blank, and the cycle begins again.

3. These constructions, while they can be carried out using the basic model of a TM are much clearer if we use some of the tricks of Sect. 8.3.

For part (a), given an input $[x,y]$ use a second track to simulate the TM for f on the input x . When the TM halts, compare what it has written with y , to see if y is indeed $f(x)$.

Accept if so.

For part (b), given x on the tape, we need to simulate the TM M that recognizes the graph of f . However, since this TM may not halt on some inputs, we cannot simply try all $[x,i]$ to see which value of i leads to acceptance by M . The reason is that, should we work on some value of i for which M does not halt, we'll never advance to the correct value of $f(x)$. Rather, we consider, for various combinations of i and j , whether M accepts $[x,i]$ in j

steps. If we consider (i,j) pairs in order of their sum (i.e., $(0,1)$, $(1,0)$, $(0,2)$, $(1,1)$, $(2,0)$, $(0,3)$,...) then eventually we shall simulate M on $[x,f(x)]$ for a sufficient number of steps that M reaches acceptance. We need only wait until we consider pairs whose sum is $f(x)$ plus however many steps it takes M to accept $[x,f(x)]$. In this manner, we can discover what $f(x)$ is, write it on the tape of the TM that we have designed to compute $f(x)$, and halt.

Now let us consider what happens if f is not defined for some arguments. Part (b) does not change, although the constructed TM will fail to discover $f(x)$ and thus will continue searching forever. For part (a), if we are given $[x,y]$, and f is not defined on x , then the TM for f will never halt on x . However, there is nothing wrong with that. Since $f(x)$ is undefined, surely y is not $f(x)$. Thus, we do not want the TM for the graph of f to accept $[x,y]$ anyway.

4. This TM only moves right on its input. Moreover, it can only move right if it sees alternating 010101... on the input tape. Further, it alternates between states q_0 and q_1 and only accepts if it sees a blank in state q_1 . That in turn occurs if it has just seen 0 and moved right, so the input must end in a 0. That is, the language is that of regular expression $(01)^*0$.

5. Here is the subroutine. Note that because of the technical requirements of the subroutine, and the fact that a TM is not allowed to keep its head stationary, when we see a non-0, we must enter state q_3 , move right, and then come back left in state q_4 , which is the ending state for the subroutine.

state	0	1	B
q1	(q2,0,R)	-	-
q2	(q2,0,R)	(q3,1,R)	(q3,B,R)
q3	(q4,0,L)	(q4,1,L)	(q4,B,L)

Now, we can use this subroutine in a TM that starts in state q_0 . If this TM ever sees the blank, it accepts in state q_f . However, whenever it is in state q_0 , it knows only that it has not seen a 1 immediately to its right. If it is scanning a 0, it must check (in state q_5) that it does not have a blank immediately to its right; if it does, it accepts. If it sees 0 in state q_5 , it comes back to the previous 0 and calls the subroutine to skip to the next non-0. If it sees 1 in state q_5 , then it has seen 01, and uses state q_6 to check that it doesn't have another 1 to the right.

In addition, the TM in state q_4 (the final state of the subroutine), accepts if it has reached a blank, and if it has reached a 1 enters state q_6 to make sure there is a 0 or blank following. Note that states q_4 and q_5 are really the same, except that in q_4 we are certain we are not scanning a 0. They could be combined into one state. Notice also that the subroutine is not a perfect match for what is needed, and there is some unnecessary jumping back and forth on the tape. Here is the remainder of the transition table.

state	0	1	B
q0	(q5,0,R)	(q5,1,R)	(qf,B,R)
q5	(q1,0,L)	(q6,1,R)	(qf,B,R)
q6	(q0,0,R)	-	(qf,B,R)
q4	-	(q6,1,R)	(qf,B,R)

6. $[q_0]01 \mid 1[q_0]1 \mid 10[q_1] \mid 10B[q_2]$

7. We'll use a second tape, on which the guess x is stored. Scan the input from left to right, and at each cell, guess whether to stay in the initial state (which does the scanning) or go to a new state that copies the next 100 symbols onto the second tape. The copying is done by a sequence of 100 state, so exactly 100 symbols can be placed on tape 2.

Once the copying is done, retract the head of tape 2 to the left end of the 100 symbols. Then, continue moving right on tape 1, and at each cell guess either to continue moving right or to guess that the second copy of x begins. In the latter case, compare the next 100 symbols on tape 1 with the 100 symbols on tape 2. If they all match, then move right on tape 1 and accept as soon as a blank is seen.