21/12/2005 E 10.1

Exercise (8.2.3 from textbook):

Design a Turing Machine that takes as input a number N in binary and turns it into $N+1$ (in binary); the number N is preceded by the symbol \$, which may be destroyed during the computation. For example, \$111 is turned into 1000; \$1001 is turned into \$1010.

solution

The idea is to toggle the rightmost digit, and, from right to left, all consecutive 1's until we get to the first 0 (which is also toggled). If there is no 0 to be toggled, a 1 is added on the left of the first digit (i.e., in place of the \$).

We need three states, where only q_2 is the final state; we briefly describe what the TM does in the different states.

q_0: the TM goes right until it reaches b, after the rightmost digit. When b is reached, the TM goes into q_1.

q_1: goes left toggling all 1's and the first 0 (from right); when 0 or \$ is reached, the symbol is turned into 1.

q_2: final state; the TM does nothing

	\$	0	1	b
q_0	$(q_0, \$, R)$	$(q_0, 0, R)$	$(q_0, 1, R)$	(q_1, b, L)
q_1	$(q_2, 1, L)$	$(q_2, 1, L)$	$(q_1, 0, L)$	—
q_2	—	—	—	—

Exercise (8.2.4 from textbook) 21/12/2005 E 10.2

We explore equivalence between function computation and language recognition for Turing machines.

DEFINITION

The graph of a function is the set of all strings $[x, f(x)]$, where x is a non-negative integer in binary, and $f(x)$ is the value of f evaluated on x, again in binary.

DEFINITION

A Turing machine computes function f if, starting with a string x on the tape, halts (in any state) with $f(x)$ on the tape. x and $f(x)$ are in binary.

Do the following:

(a) Show that, given a TM computing f, we can construct a TM accepting the graph of f

(b) Show that, given a TM accepting the graph of a function f, we can construct a TM that computes f.

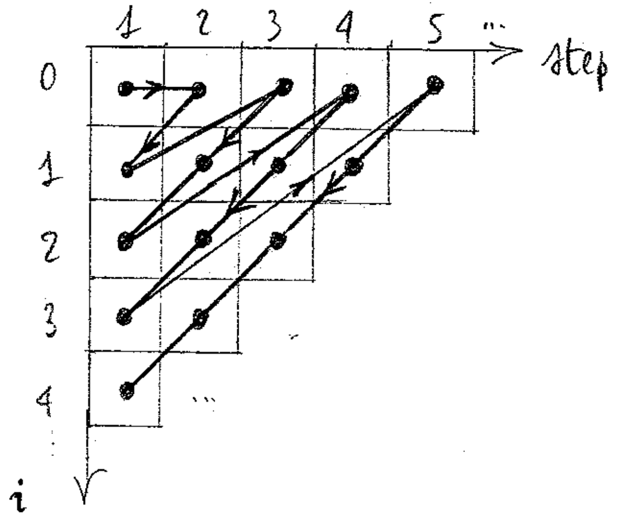
solution

(a) Our TM uses two tracks: in the first track it stores the input $[x, y]$. The goal is to check whether $y = f(x)$ so that $[x, y]$ belongs to the graph of f. On the second track, the TM emulates the TM that computes f, using x (first part of the first track) as working tape. When the emulated TM halts, there is $[f(x), y]$ on the first track, and our TM just checks whether $[f(x), y] = [y, y]$.

E 10.3

(b) In this case we have a TM M_G that accepts the graph of f, and we want to construct a TM M that computes f. We cannot try all (infinite) $[x,i]$ to see whether $[x,i]$ belongs to the graph of f: in fact, M_G may not terminate on some inputs.

Instead, we try all values of i in a different fashion: we emulate M_G with a second track, executing step 1 with $i=0$, then step 2 with $i=1$, and after that step 1 with $i=1$, as shown in the figure, where we move diagonally on the grid. i is denoted in decimal.



The visit of the grid in a diagonal fashion guarantees that, if $f(x)=y$ (and therefore M_G terminates), at a certain point we reach the value of i such that $i=y$ and M can check (by emulating M_G) that $[x,y]$ belongs to the graph of f, therefore having the correct value for $f(x)$. Note that M does always terminate if f is defined for all inputs x.

Exercise (8.4.2 from textbook) 21/12/2005 E 10.4

Consider the following NTM :

$$M = (\{q_0, q_1, q_2\}, \{0,1\}, \{0,1,\not{b}\}, \delta, q_0, \not{b}, \{q_2\})$$

with δ defined as follows

	0	1	$\not{b}$
q_0	$\{(q_0,1,R)\}$	$\{(q_1,0,R)\}$	$\emptyset$
q_1	$\{(q_1,0,R),(q_0,0,L)\}$	$\{(q_1,1,R),(q_0,1,L)\}$	$\{(q_2,\not{b},R)\}$
q_2	$\emptyset$	$\emptyset$	$\emptyset$

Show the ID's reached by M when the input is (a) 01 .
(b) 011

Solution

(a)
$$\begin{tikzcd}
q_0 01 \arrow[d, dash] \\
1 q_0 1 \arrow[d, dash] \\
10 q_1 \not{b} \arrow[d, dash] \\
10\not{b} q_2 \not{b}
\end{tikzcd}$$

Note that here we do not branch.

E 10.5

(b)

$$\begin{tikzcd}
 & q_0\,011 \arrow[d, no head] & \\
 & 1\,q_0\,11 \arrow[d, no head] & \\
 & 10\,q_1\,1 \arrow[dl, no head] \arrow[dr, no head] & \\
101\,q_1\,\not{b} \arrow[d, no head] & & 1\,q_0\,01 \arrow[d, no head] \\
101\not{b}\,q_2\,\not{b} & & 11\,q_0\,1 \arrow[d, no head] \\
 & & 110\,q_1\,\not{b} \arrow[d, no head] \\
 & & 110\not{b}\,q_2\,\not{b}
\end{tikzcd}$$

21/12/2005

Exercise (8.4.5 from textbook) $\boxed{E\ 10.6}$

Suppose you have a tape with all b's except a single \$, with the head in some (unknow) position (a) Write a NTM able to enter into a final state (starting from initial state) by scanning \$.

(b) Then, write a deterministic TM doing the same job.

solution

(a) The TM just needs to guess whether \$ is on the left or on the right. We call q, q_f the two states (q_f is final).

$$\delta(q, b) = \{(q, b, L), (q, b, R)\}$$
$$\delta(q, \$) = \{(q_f, \$, R)\}$$

(b) The deterministic TM goes back and forth examining one new position on the tape on the left, and then one on the right; marked symbols are turned from b to #.

	b	#	\$
q_0	$(q_1, \#, L)$	$(q_0, \#, R)$	$(q_2, \$, R)$
q_1	$(q_0, \#, R)$	$(q_1, \#, L)$	$(q_2, \$, R)$
q_2	—	—	—

In q_0, the TM looks for the next b on the right, while in q_1 it looks for the next one on the left. When a b is reached, it is turned into # and the search starts over in the opposite direction.

11/1/2006

$\boxed{E\ 10.7}$

Exercise (8.4.6 from textbook)

Design a 2-tape TM accepting strings over $\{0,1\}$ having an equal number of 0's and 1's. The first tape reads the input from left to right; the second tape is a working tape, storing the excess of 0's over 1's or vice-versa.

solution

The idea is that our multitape Turing machine M writes initially a symbol # on the working tape; the input tape is scanned sequentially, while the head on the working tape moves right (resp. left) if the input symbol read is 1 (resp. 0). If at the end of the input the head of the working tape is on #, then the string is accepted.

The transition function is defined as follows:

$$(q_0,[0,b]) \mapsto (q_1,[0,\#],[S,S])$$
$$(q_0,[1,b]) \mapsto (q_1,[1,\#],[S,S])$$

Intuitively, the above rules make M write # (in place of b, since the working tape is all blank) on the working tape, whatever symbol (0 or 1) is read on the input symbol; the heads do not move.

Then, the input is scanned in state q_1

$$(q_1,[0,\#]) \mapsto (q_1,[0,\#],[R,L])$$
$$(q_1,[1,\#]) \mapsto (q_1,[1,\#],[R,R])$$
$$(q_1,[0,b]) \mapsto (q_1,[0,b]),[R,L])$$
$$(q_1,[1,b]) \mapsto (q_1,[1,b]),[R,R])$$

At the end of the input tape, we go in state q_2 (only final state) only if the head of the working tape reads #.

E 10.8

$$(q_1, [b, \#]) \mapsto (q_2, [b, \#], [S, S])$$

Exercise (8.4.9 from textbook)

We consider a k-head Turing machine having a single tape and k heads; more than one head can be on the same symbol. At each move, the TM can change state, write a symbol on each cell under a head, and move each cell, or keep it stationary. We number the heads with numbers $\{1, \dots, k\}$: when there is more than one head on a single cell, the written symbol will be the one written by the head with highest number.

Prove that the languages accepted by k-head Turing machines are the same languages accepted by ordinary TM's.

solution We prove the assertion by showing that it is possible to emulate a k-head TM with a single-head (ordinary) TM.

Our TM will use additional symbols $H_1, \dots, H_k$ to mark the positions of the k heads on the tape, while the head will move back and forth from the leftmost head symbol to the rightmost one. At each head symbol H_i, our TM emulates the move of the i-th head, writing the correct symbol and making the corresponding move.

E 10.9

The problem is to deal with multiple heads on the same cell. In this case, we choose to put the head symbols one adjacent to the other, ordered so that the head symbol with the smallest number is the leftmost one. For example, we can have a configuration like

$$q H_3 H_7 H_{11} 011 \cancel{b} H_6 11$$

denoting that heads number 3,7 and 11 are on the same cell.

This complicates the things, but the emulation is still possible. No symbol is written after some H_i if the symbol on the right is another head symbol. When a head symbol is to be moved, it has to skip all other head symbols; moreover, if it is arriving on a cell "pointed" by other head symbols, it needs to be put in the correct place so as to respect the order (also this is easily feasible).