Exercise (8.2.4 from textbook)

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

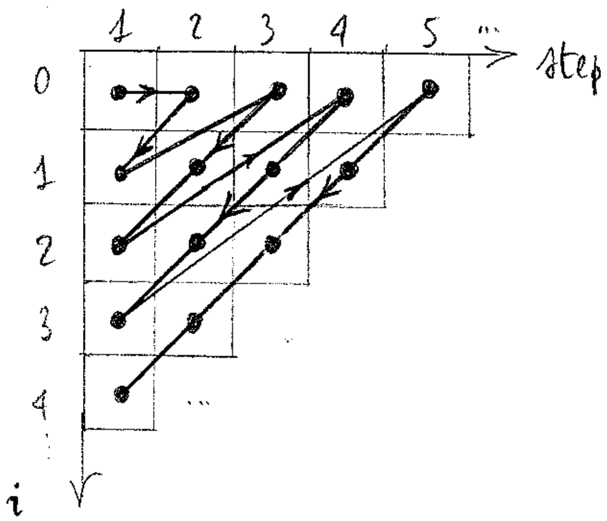
(b) Show that, given a TM accepting the graph of a function f, we can construct a TM that computes f.

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 $[f(x), y] = [y, y]$.

(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~~ tape, 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.

Alternative solution:

We make use of nondeterminism, and for each i, we guess whether to run M_G on $[x,i]$, or whether to switch directly to $i+1$.

E 3.3

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 #.

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

<u>Exercise</u> (8.4.9 from textbook)

We consider a <u>k-head Turing machine</u> 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.

<u>solution</u> 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.

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

$$q\, H_3\, H_7\, H_{11}\, 0\, 1\, 1\, b\!\!\!/\, H_6\, 1\, 1$$

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).

Exercise: (Section 9.3.2 from textbook)

Consider the following languages over $\Sigma = \{0,1\}$

$$L_e = \{ \mathcal{E}(M) \mid \mathcal{L}(M) = \emptyset \}$$

$$L_{ne} = \{ \mathcal{E}(M) \mid \mathcal{L}(M) \neq \emptyset \}$$

Hence: L_e ... set of all strings that encode T.M.s that accept the empty language

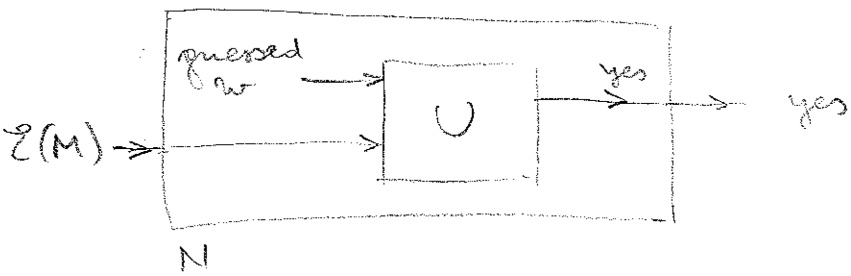
L_{ne} ... complement of L_e

Claim 1: L_{ne} is R.E.

Proof: construct NTM N for L_{ne} (and then convert N to an ordinary T.M.)

N works as follows: on input $\mathcal{E}(M)$

1) guess a string $w \in \Sigma^*$
2) simulate M on w (like a UTM)
3) accept $\mathcal{E}(M)$ if M accepts w



We have $\mathcal{E}(M) \in \mathcal{L}(N) \iff \exists w \text{ s.t. } \langle M, w \rangle \in \mathcal{L}(U)$

$\iff \exists w \text{ s.t. } w \in \mathcal{L}(M)$

$\iff \mathcal{E}(M) \in L_{ne}$

Claim 2 : L_{ne} is non-recursive

Proof : by reduction from L_u to L_{ne}

Reduction R is a function computable by a halting T.M.

with input : instance $\langle M, w \rangle$ of L_u

output : instance $\mathcal{E}(M')$ of L_{ne}

and s.t. $\langle M, w \rangle \in L_u \iff \mathcal{E}(M') \in L_{ne}$

Description of M':

- M' ignores completely its own input string x
- instead it replaces its input by the string $\langle M, w \rangle$ and simulates M on w using UTM.
- if M accepts w, then M' accepts x

  if M never halts on w or rejects w, then M' also never halts or rejects x

Note : if $w \in \mathcal{L}(M) \Rightarrow \mathcal{L}(M') = \Sigma^*$

if $w \notin \mathcal{L}(M) \Rightarrow \mathcal{L}(M') = \emptyset$

hence $\langle M, w \rangle \in L_u \iff \mathcal{E}(M') \in L_{ne}$

We can construct a halting T.M. M_R that, given $\langle M, w \rangle$ as input, constructs $\mathcal{E}(M')$ for an M' that behaves as above.

q.e.d.

To sum up, we have that L_{ne} is R.E. but non-recursive.

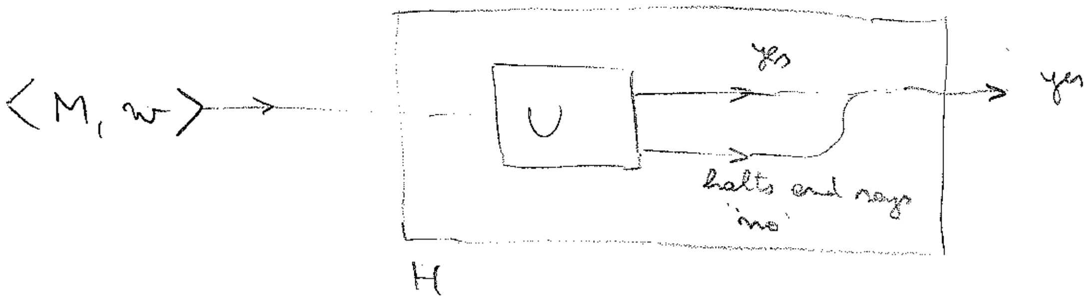
Hence L_e must be non-R.E.

Exercise: 9.2.1

The halting problem L_H, i.e., the set $\langle M, w\rangle$ s.t. M halts on w (with or without accepting) is R.E. but not recursive.

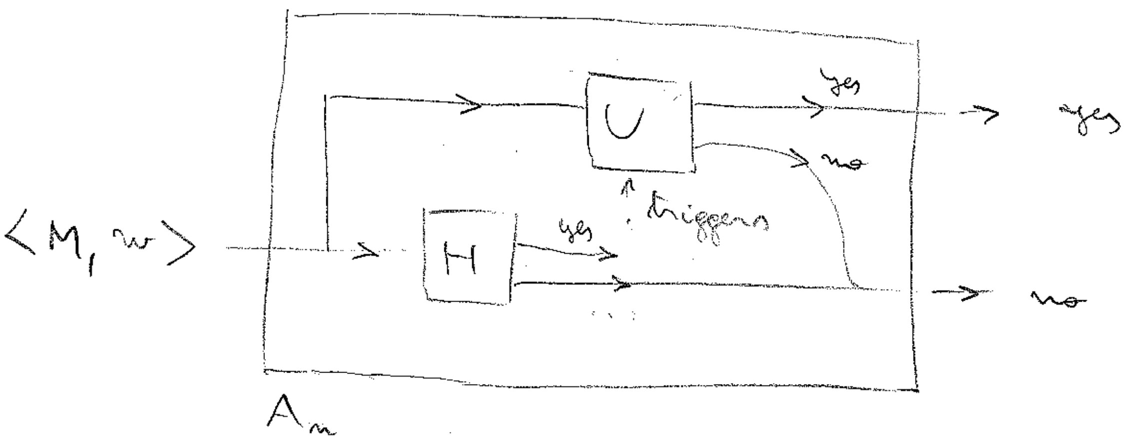
To show R.E., we construct a T.M. H s.t.

$\mathcal{L}(H) := L_H = \{\langle M, w\rangle \mid M \text{ halts on } w\}$



To show that L_H is not recursive, we assume by contradiction it is so, and derive that L_u is recursive.

By contradiction, let H be an algorithm for L_H and U a procedure for L_u



A_u would be an algorithm for L_u.

Contradiction

Exercise 3.2.5

Let L be R.E. and $\overline{L}$ be non-R.E.

Consider $L' = \{0w \mid w \in L\} \cup$
$\{1w \mid w \notin L\}$

What do we know about L' and $\overline{L'}$?

We show that L' is non-R.E.

Suppose by contradiction that we have a procedure $M_{L'}$ for L'.

Then we can construct a procedure $M_{\overline{L}}$ for $\overline{L}$ as follows

- on input w, $M_{\overline{L}}$ changes the input to $1w$ and simulates $M_{L'}$.
- if $M_{L'}$ accepts $1w$, then $w \in \overline{L}$, and $M_{\overline{L}}$ accepts
- if $M_{L'}$ does not terminate or terminates and answers no, then $w \notin \overline{L}$, and $M_{\overline{L}}$ does not terminate or terminates and answers no.

$\Longrightarrow$ $M_{\overline{L}}$ would accept exactly $\overline{L}$. Contradiction

$\overline{L'} = \{0w \mid w \notin L\} \cup \{1w \mid w \in L\} \cup \{\varepsilon\}$

Reasoning as for L', we get that $\overline{L'}$ is non-R.E.

Exercise 9.3.7 a)

$\overline{H}$, the complement of the halting problem, i.e., the set of pairs $\langle M, w \rangle$ such that M on input w does not halt, is non-R.E.

Proof: By reduction from $\overline{L}_u$, which is non-R.E.

Idea: we show how to convert any TM M into another TM M_H s.t. M_H halts on w iff M accepts w.

Construction:

1) Ensure that M_H does not halt unless M accepts.
   - add to the states of M a new loop state p, with $\delta(p, x) = (p, x, R)$ for all $x \in \Gamma$
   - for each $\delta(q, y)$ that is undefined and $q \notin F$, add $\delta(q, y) = (p, y, R)$

2) Ensure that, if M accepts, then M_H halts
   - make $\delta(q, x)$ undefined for all $q \in F$ and $x \in \Gamma$

3) The other moves of M_H are as those of M

q.e.d.