Homework 2 Solutions

Shobhit Behl
(IMT2016024)

November 26, 2018

- *Discussions*

  I have discussed with the following people regarding the solutions given below. This might be a cause of similarity in our solutions.

  1. Aditya Gulati (IMT2016052)
  2. Aditya Hegde (IMT2016054)
  3. Biswesh Mohapatra (IMT2016050)

- *Proofs for functions used as intermediate steps*

1. *rem*

$$rem(x, y) = remainder\ when\ x\ is\ divided\ by\ y.$$

$$rem(0, y) = 0$$

$$rem(n + 1, y) = h(rem(n, y), n, y)$$

where,

$$h(x_1, x_2, x_3) = \times(sg(\dot{-}(\dot{-}(P_3^3(x_1, x_2, x_3), 1)P_3^1(x_1, x_2, x_3))), S(P_3^1(x_1, x_2, x_3)))$$

Therefore rem is primitive recursive.

2. *sg*

$$sg(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases}$$

$$sg(0) = 0$$

$$sg(n + 1) = h(sg(n), n)$$

where,

$$h(x_1, x_2) = S(Z(P_2^2(x_1, x_2)))$$

Therefore sg is primitive recursive.

3. $\overline{sg}$

$$\overline{sg}(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\overline{sg}(0) = 1$$

$$\overline{sg}(n + 1) = h(\overline{sg}(n), n)$$

where,

$$h(x_1, x_2) = Z(P_2^2(x_1, x_2))$$

Therefore $\overline{sg}$ is primitive recursive.

4. <u>*odd*</u>

$$odd(x) = \begin{cases} 1 & \text{if } x \text{ is odd} \\ 0 & \text{if } x \text{ is even} \end{cases}$$

We will prove that these functions are primitive recursive.

$$odd(0) = 0$$

$$odd(n+1) = h_1(odd(n), n)$$

where

$$h_1(x_1, x_2) = \overline{sg}(P_2^1(x_1, x_2))$$

Therefore *odd* is primitive recursive.

5. <u>*even*</u>

$$even(x) = \begin{cases} 1 & \text{if } x \text{ is even} \\ 0 & \text{if } x \text{ is odd} \end{cases}$$

$$even(0) = 1$$

$$even(n+1) = h_2(even(n), n)$$

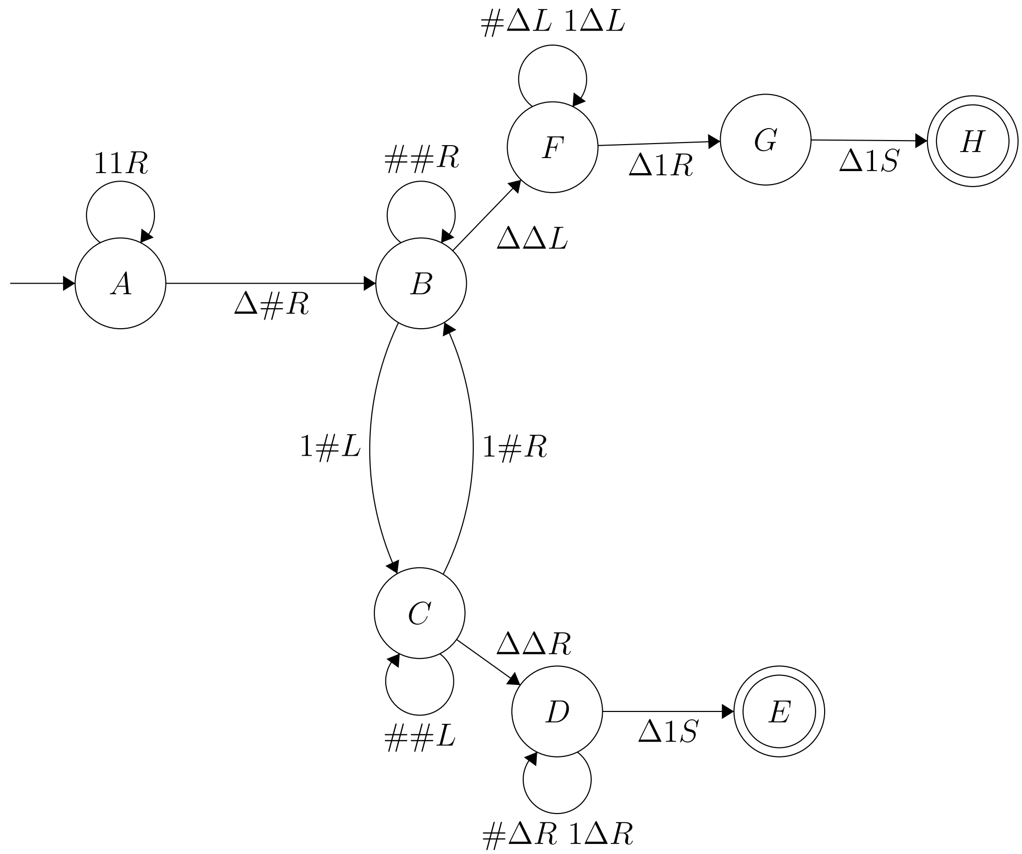
where

$$h_2(x_1, x_2) = \overline{sg}(P_2^1(x_1, x_2))$$

Therefore *even* is primitive recursive.

- *Solution 1.)*

Assuming x and y are in unary notation and are separated by a Δ.



First we replace the Δ in between x and y to a #. Then we go to the first 1 in y and convert it to a #. Then we go to the last 1 in x and convert it to a # and so on. If x becomes all #'s first then it is less that y and we clear all the #'s and 1's and write one 1 denoting 0, else if y becomes all #'s then we clear all #'s and 1's and write two 1's denoting 1.

Say, $x = 2, y = 1$. Then the tape will be as follows,

$$...\Delta\Delta 111\Delta 11\Delta\Delta...$$

$$\Rightarrow ...\Delta\Delta 111\#11\Delta\Delta...$$

$$\Rightarrow ...\Delta\Delta 111\#\#1\Delta\Delta...$$
$$\Rightarrow ...\Delta\Delta 11\#\#\#1\Delta\Delta...$$
$$\Rightarrow ...\Delta\Delta 11\#\#\#\#\Delta\Delta...$$
$$\Rightarrow ...\Delta\Delta 1\#\#\#\#\#\Delta\Delta...$$
$$\Rightarrow ...\Delta\Delta 11\Delta\Delta\Delta\Delta\Delta\Delta\Delta...$$

- *Solution 2.)*

$$geq(0, y) = \overline{sg}(y)$$
$$geq(n + 1, y) = h(geq(x, n), n, y)$$

where

$$h(x_1, x_2, x_3) = \overline{sg}(\dot{-}(P_3^3(x_1, x_2, x_3), +(P_3^2(x_1, x_2, x_3), 1)))$$

and hence, geq is primitive recursive.

- *Solution 3.a.)*

For given function, $f(0) = -1$ as we can't have negative numbers, we keep the base case as 1.

$$f(1) = 3$$
$$f(n + 1) = h_3(f(n), n)$$

where

$$h_3(x_1, x_2) =$$
$$+(\times(+(\times(2, S(P_2^2(x_1, x_2))), 1), odd(S(P_2^2(x_1, x_2)))),$$
$$\times(-(\times(2, S(P_2^2(x_1, x_2))), 1), even(S(P_2^2(x_1, x_2)))))$$

Therefore the function f is primitive recursive.

- ## *Solution 3.b.)*

g is an already known primitive recursive function, there primitive recursive definition of f can be,

$$f(0) = sg(\dot{-}(g(1), g(0)))$$
$$f(n+1) = h(f(n), n)$$

where,

$$h(x_1, x_2) = \times(sg(\dot{-}(g(+(P_2^2(x_1, x_2), 2)), g(S(P_2^2(x_1, x_2)))), P_2^1(x_1, x_2))$$

therefore, f is a primitive recursive function.

- ## *Solution 4.a.)*

$$f(0) = 0$$
$$f(n+1) = h(f(n), n)$$

where,

$$h(x_1, x_2) = +(exp(S(P_2^2(x_1, x_2)), 7), \times(12, exp(S(P_2^2(x_1, x_2)), 5)))$$

therefore f is primitive recursive.

- ## *Solution 4.b.)*

Assuming the length of the vector is fixed as k. We define our function as follows,

$$f(\vec{x}, n) = \sum_{i=1}^{n} x_i \text{ } where \text{ } n \leqslant k$$

So now,

$$f(\vec{x}, 0) = 0$$
$$f(\vec{x}, n+1) = h(f(\vec{x}, n), n, \vec{x})$$

where,

$$h(x_1, x_2, x_3) = +(P_3^1(x_1, x_2, x_3), P_k^{S(P_3^2(x_1, x_2, x_3))}(P_3^3(x_1, x_2, x_3)))$$

Therefore f is primitive recursive.

- ## *Solution 4.c.)*

Assuming the length of the vector is fixed as k. We define our function as follows,

$$f(\vec{x}, n) = \prod_{i=1}^{n} x_i \text{ } where\ n \leqslant k$$

So now,

$$f(\vec{x}, 0) = 0$$
$$f(\vec{x}, n+1) = h(f(\vec{x}, n), n, \vec{x})$$

where,

$$h(x_1, x_2, x_3) = \times(P_3^1(x_1, x_2, x_3), P_k^{S(P_3^2(x_1,x_2,x_3))}(P_3^3(x_1, x_2, x_3)))$$

Therefore f is primitive recursive.

- ## *Solution 4.d.)*

$$f(0) = 1$$
$$f(n+1) = h(f(n), n)$$

where,

$$h(x_1, x_2) = \overline{sg}(rem(S(P_2^2(x_1, x_2)), 3))$$

Hence f is primitive recursive.

- ## *Solution 5.)*

First we show that the number of primitive recursive definitions of a primitive recursive function f are infinite.

Say h is a primitive recursive definition of f, then we can construct infinitely many primitive recursive definitions of f as follows,

$$h' = +(h, 0)$$

Therefore, there exist infinite primitive recursive definitions of f.

Now say the set of all primitive recursive recursive definitions defined in the above way be C. We can define a one to one map from C to natural numbers as the number of times we do an addition with 0. And we know the set of natural numbers is countably infinite therefore C is countably infinite.
Now the set of all primitive recursive definitions of f be B. Clearly $C \subseteq B$.
Now let the set of all strings be A. Clearly $B \subseteq A$.

Now we know that set of all strings (A) is countably infinite and also C is countably infinite and $A \supseteq B \supseteq C$.

Therefore B is countably infinite, that is the number of primitive recursive definitions of f is countably infinite.

- *Solution 6.)*

To Prove: $f(x) == \triangleq \min_{i=0}^{x} g(i)$ is a primitive recursive function.

Proof: We will give a primitive recursive definition for f.

$$f(0) = g(0)$$
$$f(n+1) = h(f(n), n)$$

where,

$$h(x_1, x_2) = \dot{-}(P_2^1(x_1, x_2), \dot{-}(P_2^1(x_1, x_2), g(S(P_2^2(x_1, x_2)))))$$

Therefore f is a primitive recursive function.

- *Solution 7.)*

To Prove: The set of all natural numbers not divisible by 10 is primitive recursive.

We define the characteristic function C_f for our given condition f.

$$C_f(x) = \begin{cases} 1 & \text{if } x \text{ is not divisible by } 10 \\ 0 & \text{if } x \text{ otherwise} \end{cases}$$

We prove that C_f is primitive recursive as follows,

$$C_f(0) = 0$$
$$C_f(n+1) = h(C_f(n), n)$$

where,

$$h(x_1, x_2) = sg(rem(S(P_2^2(x_1, x_2)), 10))$$

Therefore C_f is recursive implying that the set of all numbers not divisible by 10 is primitive recursive.

- *Solution 8.)*

To Prove: The predicate that is true of all natural numbers divisible by 6 is primitive recursive.

We define the characteristic function C_p for our given predicate p.

$$C_p(x) = \begin{cases} 1 & \text{if } x \text{ is divisible by } 6 \\ 0 & \text{if } x \text{ otherwise} \end{cases}$$

We prove that C_p is primitive recursive as follows,

$$C_p(0) = 0$$
$$C_p(n+1) = h(C_p(n), n)$$

where,

$$h(x_1, x_2) = \overline{sg}(rem(S(P_2^2(x_1, x_2)), 6))$$

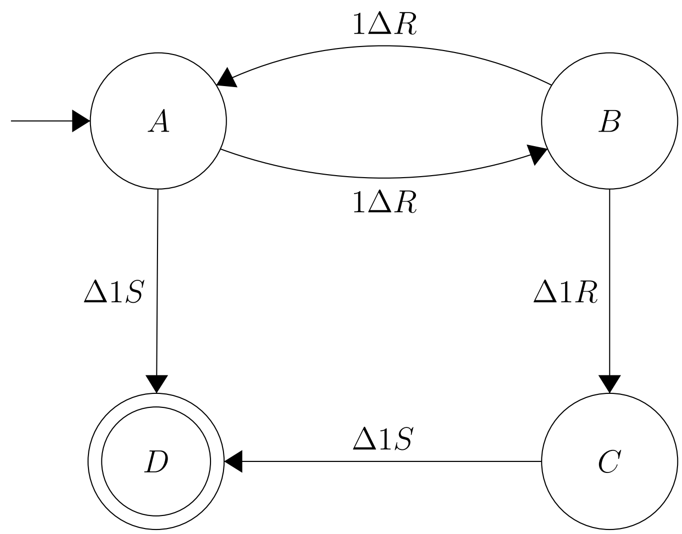
Therefore C_p is recursive implying that the predicate that is true of all natural numbers divisible by 6 is primitive recursive.

- *Solution 9.)*

The characteristic function of the set of all even numbers is follows,

$$C_f(x) = \begin{cases} 1 & \text{if } x \text{ is even} \\ 0 & \text{if } x \text{ otherwise} \end{cases}$$

TM:



- *Solution 10.a.)*

Let this problem be decidable. Say there exists a TM as follows,

$$M''(M) = \begin{cases} Y & \textit{if } M \textit{ halts only on non palindromic inputs.} \\ N & \textit{otherwise} \end{cases}$$

Let us define a new TM M' that takes M_w and any string y as input as follows,

$$M'(M_w, y) = \begin{cases} \textit{Halts} & \textit{if } M_w \textit{ halts} \\ \textit{Does not halt} & \textit{otherwise} \end{cases}$$

Here M_w denotes a TM M which is run on a given string w.

Now we define M''' as,

$$M''' = \begin{cases} Y & \textit{if } M'' \textit{ run with } M' \textit{ as input gives } N \\ N & \textit{if } M'' \textit{ run with } M' \textit{ as input gives } Y \end{cases}$$

Now We claim that M''' is a halting tester.

If M halts on w, M' halts on every input string y and when passed to M'', it will output N as M' will halt on palindromic strings as well. Therefore M''' will output Y.

If M does not halt on w, M' does not halt on any input string y and when passed to M'', it will output Y as M' will not halt on any palindromic string which is equivalent to halting only on non-palindromic strings. Therefore M'' will output Y and M''' will output N.

M''' will output Y if M halts on w and N otherwise. We have constructed a halting tester from M'' which we know cannot exist, therefore M'' cannot exist. And hence the given problem is undecidable.

- *Solution 10.b.)*

Let this problem be decidable. Say there exists a TM as follows,

$$M''(A, B) = \begin{cases} Y & \textit{if } L(A) \backslash L(B) = \varnothing \\ N & \textit{otherwise} \end{cases}$$

Let us define a new TM M' that takes M_w and any string y as input as follows,

$$M'(M_w, y) = \begin{cases} \textit{Accepts} & \textit{if } M_w \textit{ halts} \\ \textit{Does not accept} & \textit{otherwise} \end{cases}$$

Here M_w denotes a TM M which is run on a given string w.

Now we define E as a TM that accepts nothing ie. $L(E) = \varnothing$

Now we define M''' as,

$$M''' = \begin{cases} Y & \textit{if } M'' \textit{ accepts when run with input M' and E} \\ \textit{Does not accept} & \textit{if } M'' \textit{ does not accept when run with input M' and E} \end{cases}$$

We claim that M''' is a halting tester.

If M halts on w, M' accepts everything, so when you pass M' and E to M'', it gives Y and therefore M''' gives Y as $L(M') \setminus L(E) \neq \varnothing$.

If M does not halt on w, M' accepts nothing, so when you pass M' and E to M'', it gives N and therefore M''' gives N as $L(M') \setminus L(E) = \varnothing$.

M''' will output Y if M halts on w and N otherwise. We have constructed a halting tester from M'' which we know cannot exist, therefore M'' cannot exist. And hence the given problem is undecidable.