Primitive recursive functions

Aim to give a more abstract, machine-independent description of the collection of computable partial functions.

We will eventually characterize it as the smallest collection containing some basic functions and closed under some fundamental operations for forming new functions from old, viz.

Composition, primitive recursion and minimization.

The characterization is due to Kleene (circa 1936), who made use of earlier, related work by Gödel and Herbrand.

We begin by looking at the basic functions, composition and primitive recursion; minimization will be considered in the section on partial recursive functions.

The basic functions :

Projection functions, $proj_i^n \in Fun(\mathbb{N}^n, \mathbb{N})$

$$proj_i^n(x_1, \ldots, x_n) \stackrel{def}{=} x_i$$

Constant function with value zero, $zero^n \in Fun(\mathbb{N}^n, \mathbb{N})$

$$zero^n(x_1, \ldots, x_n) \stackrel{def}{=} 0$$

Successor function, $suc \in Fun(\mathbb{N}, \mathbb{N})$

$$suc(x) \stackrel{def}{=} x+1$$

PROPOSITION :

The basic functions are computable.

Proof

A register machine for computing

$proj_i^n$ is specified by START → [copy Ri to R0] → HALT

$zero^n$ " " " START → HALT

suc " " " START → R1+ → [copy R1 to R0] → HALT

□

Composition of partial functions

Given $f \in Pfn(\mathbb{N}^n, \mathbb{N})$ & $g_1, \ldots, g_n \in Pfn(\mathbb{N}^m, \mathbb{N})$, define $f \circ (g_1, \ldots, g_n) \in Pfn(\mathbb{N}^m, \mathbb{N})$ by:

$f \circ (g_1, \ldots, g_n)(x_1, \ldots, x_m) = z \overset{\text{def}}{\iff}$ there exists $(y_1, \ldots, y_n) \in \mathbb{N}^n$ so that

$g_1(x_1, \ldots, x_m) = y_1 \;\&\; \cdots \;\&\; g_n(x_1, \ldots, x_m) = y_n$

and $f(y_1, \ldots, y_n) = z$

Thus $f \circ (g_1, \ldots, g_n)$ is the unique m-ary partial function h satisfying $h(x_1, \ldots, x_m) \equiv f(g_1(x_1, \ldots, x_m), \ldots, g_n(x_1, \ldots, x_m))$

(where $\equiv$ denotes Kleene equivalence ...)

NOTE: in case $n = 1$, we write $f \circ g$ instead of $f \circ (g)$

Recall (from p.20) that for a partial function $h \in Pfn(\mathbb{N}^m, \mathbb{N})$, we write "$h(x_1, \ldots, x_m) = z$" for "$((x_1, \ldots, x_m), z) \in h$", i.e. to mean that h is defined at $(x_1, \ldots, x_m)$ and takes value z there.

Thus "$h(x_1, \ldots, x_m)$" is an example of a partially defined expression, i.e. an expression which either denotes a specific value (a number, in this case) or is undefined.

"$f(g_1(x_1, \ldots, x_m), \ldots, g_n(x_1, \ldots, x_m))$" is a more complicated example of such partially defined expressions.

Given two partially defined expressions e and e', the statement

$e \equiv e'$ ("e and e' are Kleene equivalent")

is defined to mean

"either e and e' are both undefined, or they are both defined and the values they denote are equal"

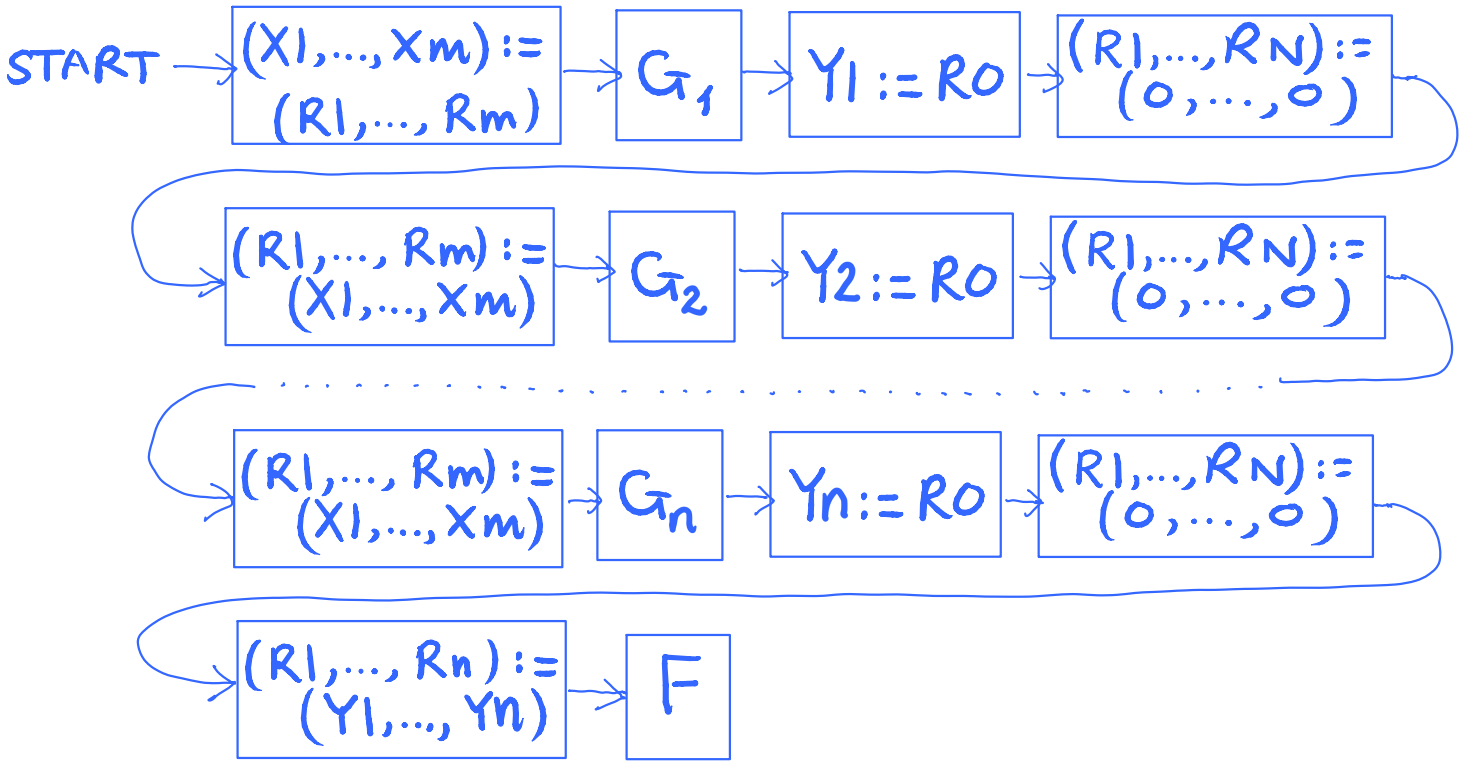
Kleene equivalence allows one to express statements about undefinedness and equality in a convenient & succinct form.

PROPOSITION:
If f and $g_1, \ldots, g_n$ are all computable, then so is $f \circ (g_1, \ldots, g_n)$.

Proof.

Given register machine programs $\begin{cases} F \\ G_i \end{cases}$ computing $\begin{cases} f(y_1, \ldots, y_n) \\ g_i(x_1, \ldots, x_m) \end{cases}$ in $R0$ starting with $\begin{cases} R1, \ldots, Rn \\ R1, \ldots, Rm \end{cases}$ set to $\begin{cases} y_1, \ldots, y_n \\ x_1, \ldots, x_m \end{cases}$, then the following graph specifies a program computing $f \circ (g_1, \ldots, g_n)(x_1, \ldots, x_m)$ in $R0$ starting with $R1, \ldots, Rm$ set to $x_1, \ldots, x_m$. □

Program for $f \circ (g_1, \ldots, g_n)$:



We assume programs $F, G1, \ldots, Gn$ only use registers $R1, \ldots, RN$ (where $N \geq \max\{n, m\}$) and that $X1, \ldots, Xm$ & $Y1, \ldots, Yn$ are not in that list.

To motivate the definition of primitive recursion, here are some examples of recursive definitions of partial functions $f \in Pfn(\mathbb{N}, \mathbb{N})$

1. Sum of $0, 1, 2, \ldots, x$

$$\begin{cases} sum(0) = 0 \\ sum(x+1) = sum(x) + x + 1 \end{cases}$$

2. n^{th} Fibonacci number

$$\begin{cases} fib(0) = 0 \\ fib(1) = 1 \\ fib(x+2) = fib(x) + fib(x+1) \end{cases}$$

3. A function that's undefined except when $x=0$

$$\begin{cases} f_3(0) = 0 \\ f_3(x+1) = f_3(x+2) + 1 \end{cases}$$

4. McCarthy's "91" function

$$f_4(x) = \text{if } x > 100 \text{ then } x-10 \text{ else } f_4(f_4(x+11))$$

(f_4 maps x to 91 if $x \leq 100$ and to $x-10$ otherwise)

Primitive recursion

Given $f \in Pfn(\mathbb{N}^n, \mathbb{N})$ and $g \in Pfn(\mathbb{N}^{n+2}, \mathbb{N})$, define $\rho^n(f,g) \in Pfn(\mathbb{N}^{n+1}, \mathbb{N})$ by

$$\rho^n(f,g)(x_1,\ldots,x_n,x) = y \overset{\text{def}}{\iff} \text{there exist } y_0, y_1, \ldots, y_x \text{ such that}$$
$$f(x_1,\ldots,x_n) = y_0$$
$$\& \quad g(x_1,\ldots,x_n,i,y_i) = y_{i+1} \text{ for } i = 0,\ldots,x-1$$
$$\& \quad y_x = y$$

It follows that $\rho^n(f,g)$ is the unique $(n+1)$-ary partial function h satisfying

$$\begin{cases} h(x_1,\ldots,x_n,0) \equiv f(x_1,\ldots,x_n) \\ h(x_1,\ldots,x_n,x+1) \equiv g(x_1,\ldots,x_n,x,h(x_1,\ldots,x_n,x)) \end{cases}$$

PROPOSITION:

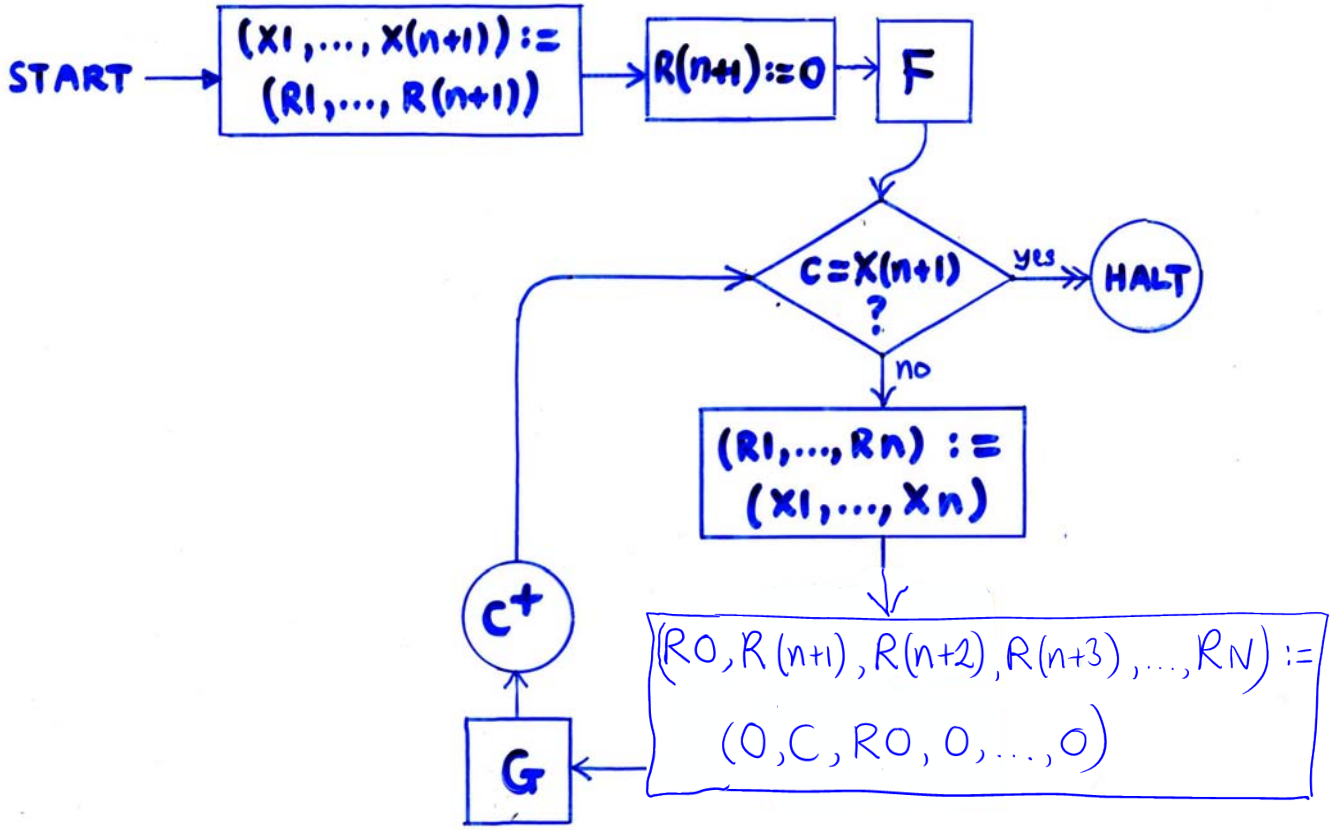
If f and g are computable, then so is $\rho^n(f,g)$.

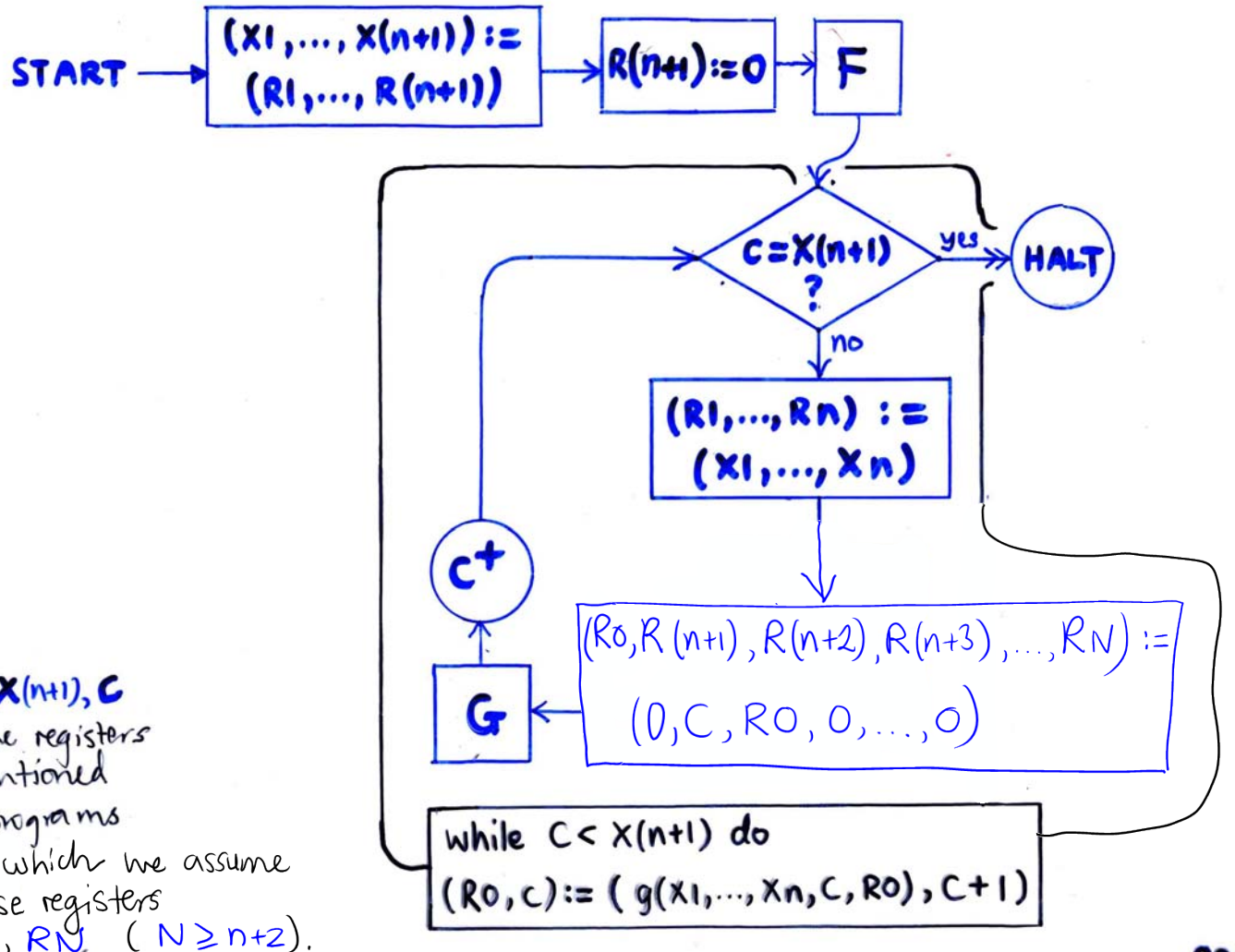
Proof

Given register machine programs

$$\begin{cases} F \\ G \end{cases} \text{ computing } \begin{cases} f(x_1,\ldots,x_n) \\ g(x_1,\ldots,x_{n+2}) \end{cases} \text{ in } R0 \text{ starting with } \begin{cases} R1,\ldots,Rn \\ R1,\ldots,R(n+2) \end{cases} \text{ set to } \begin{cases} x_1,\ldots,x_n \\ x_1,\ldots,x_{n+2} \end{cases},$$

then the following diagram specifies a register machine program that computes $\rho^n(f,g)(x_1,\ldots,x_n)$ in $R0$ starting with $R1,\ldots,R(n+1)$ set to $x_1,\ldots,x_{n+1}$:





$X1, \ldots, X(n+1), C$ are some registers not mentioned in the programs **F** & **G**, which we assume only use registers $R0, \ldots, RN$ $(N \geq n+2)$.

DEFINITION :

A function is **primitive recursive** if it can be built up from the basic functions by repeated use of the operations of composition and primitive recursion.

In other words, the set **PRIM** of primitive recursive functions is the **smallest** set of partial functions containing the basic functions and closed under the operations of composition and primitive recursion.

EXAMPLES of primitive recursive functions

Ex.1 Addition

Recall the inductive definition of $add(x,y) \overset{def}{=} x+y$ in terms of the successor function and zero:

$$\begin{cases} add(x,0) = x \\ add(x,y+1) = add(x,y)+1 \end{cases}$$

Thus $add = \rho^1(f,g)$ where $f(x) \overset{def}{=} x$ and $g(x,y,z) \overset{def}{=} z+1$.

Since $f = proj^1_1$ and $g = suc \circ proj^3_3$,

$$add = \rho^1(proj^1_1, suc \circ proj^3_3)$$

is primitive recursive.

Ex.2 Multiplication $mult(x,y) \overset{def}{=} x.y$ can be inductively defined from addition by $\begin{cases} mult(x,0) = 0 \\ mult(x,y+1) = mult(x,y) + x \end{cases}$

Thus $mult = \rho'(f,g)$ with $f(x) \overset{def}{=} 0$ and $g(x,y,z) \overset{def}{=} z + x$.

Hence $mult = \rho'(zero^1, add \circ (proj^3_3, proj^3_1))$

$= \rho'(zero^1, \rho'(proj^1_1, suc \circ proj^3_3) \circ (proj^3_3, proj^3_1))$ by Ex.1

is primitive recursive.

Ex.3 Exponentiation $exp(x,y) \overset{def}{=} x^y$ can be inductively defined from multiplication by $\begin{cases} exp(x,0) = 1 \\ exp(x,y+1) = exp(x,y).x \end{cases}$

Thus $exp = \rho'(f,g)$ where $f(x) \overset{def}{=} 1$ and $g(x,y,z) \overset{def}{=} z.x$,

i.e. $f = suc \circ zero^1$ and $g = mult \circ (proj^3_3, proj^3_1)$.

Hence by Ex.2,

$exp = \rho'(suc \circ zero^1, \rho'(zero^1, \rho'(proj^1_1, suc \circ proj^3_3) \circ (proj^3_3, proj^3_1)) \circ (proj^3_3, proj^3_1))$ [!!]

is primitive recursive.

[For the following examples, we leave as an exercise the working out of an explicit description of the function witnessing its primitive recursivity.]

Ex.4 Predecessor function $pred(x) \overset{def}{=} \begin{cases} 0 & \text{if } x = 0 \\ x-1 & \text{if } x > 0 \end{cases}$ is primitive recursive because it satisfies $\begin{cases} pred(0) = 0 \\ pred(x+1) = x \end{cases}$.

Ex.5 Truncated subtraction $minus(x,y) \overset{def}{=} x \dot{-} y = \begin{cases} 0 & \text{if } x < y \\ x-y & \text{if } x \geq y \end{cases}$

satisfies $\begin{cases} minus(x,0) = x \\ minus(x,y+1) = pred(minus(x,y)) \end{cases}$ and hence is prim. rec. by Ex.4.

Ex.6 Conditional function $ifzero(x,y,z) \overset{def}{=} \begin{cases} y & \text{if } x = 0 \\ z & \text{if } x > 0 \end{cases}$

Note that $ifzero = C \circ (proj^3_2, proj^3_3, proj^3_1)$, where $C \in Fun(\mathbb{N}^3, \mathbb{N})$ satisfies $\begin{cases} C(x_1,x_2,0) = x_1 \\ C(x_1,x_2,x+1) = x_2 \end{cases}$.

Thus C, and hence also ifzero, are primitive recursive.

Ex.7 Bounded Summation

If $f \in Fun(\mathbb{N}^{n+1}, \mathbb{N})$ is prim. rec., then so is

$$g(x_1,\dots,x_n,x) \triangleq \sum_{y<x} f(x_1,\dots,x_n,y)$$

$$= \begin{cases} f(x_1,\dots,x_n,0)+\cdots+f(x_1,\dots,x_n,x-1) & \text{if } x>0 \\ 0 & \text{if } x=0 \end{cases}$$

(For note that g satisfies

$$\begin{cases} g(x_1,\dots,x_n,0) = 0 \\ g(x_1,\dots,x_n,x+1) = g(x_1,\dots,x_n,x)+f(x_1,\dots,x_n,x) \end{cases}$$

with f prim.rec (by assumption) & add prim.rec. by Ex. 1 .)

PROPOSITION:

Every primitive recursive function is both computable and total

(Recall that $f \in Pfn(\mathbb{N}^n, \mathbb{N})$ is total if & only if $f(x_1,\dots,x_n)\downarrow$ for all $(x_1,\dots,x_n) \in \mathbb{N}^n$.)

NOTE that by definition of PRIM (see p.94), if P is some property of partial functions, <u>to prove that every member of PRIM has property P, it suffices to show that</u>

(a) the basic functions ($proj^n_i$, $zero^n$, Suc) satisfy P ; and

(b) if $f, g_1, \ldots, g_n$ satisfy P, then so does $f \circ (g_1, \ldots, g_n)$
[assuming of course that the functions have arities for which the composition makes sense] ; and

(c) if f & g satisfy P, then so does $\rho^n(f, g)$
[where n = arity of f & g has arity n+2].

For if (a), (b) & (c) hold, then $\{f \in \text{PRIM} \mid f \text{ satisfies P}\}$ contains the basic functions and is closed under composition and primitive recursion, and hence contains all primitive recursive functions : so they all satisfy P.

<u>Proof</u> of the Proposition on p.99

We have already verified that (a), (b) & (c) hold when P is the property "f is computable" : hence every $f \in \text{PRIM}$ is computable.

Now taking P to be the property "f is a total function", clearly (a) & (b) hold ; and to see that (c) holds, note that if f & g are total then for all $(x_1, \ldots, x_n) \in \mathbb{N}^n$ the definition of $\rho^n(f,g)$ gives

$$\rho^n(f,g)(x_1, \ldots, x_n, 0)\downarrow$$

and $$\rho^n(f,g)(x_1, \ldots, x_n, x)\downarrow \Rightarrow \rho^n(f,g)(x_1, \ldots, x_n, x+1)\downarrow$$

so that $\forall x.\ \rho^n(f,g)(x_1, \ldots, x_n, x)$ by Mathematical Induction on x.
Hence every $f \in \text{PRIM}$ is total. □