Clase práctica 2 (computabilidad)
Lógica y Computabilidad

Facundo Carreiro

1. Funciones Primitivas Recursivas

Vamos a definir varias funciones y probar que son *primitivas recursivas* (de ahora en más "p.r."). Podemos suponer definidas y p.r. las funciones $(+)$ y $(-)$. Para probar que una función es p.r. en general debemos demostrar que se adecua a un esquema de construcción usando

Funciones iniciales

- Cero: $n(x) = 0$
- Sucesor: $s(x) = x + 1$
- Proyección: $u_i^n(X_1, \ldots, X_n) = X_i$

Composición

Sea $f : \mathbb{N}^k \to \mathbb{N}$ y $g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$. $h : \mathbb{N}^n \to \mathbb{N}$ se obtiene a partir de f y $g_1, \ldots, g_k$ por composición si f, g_i son p.r. y

$$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n))$$

Recursión primitiva

$h : \mathbb{N}^{n+1} \to \mathbb{N}$ se obtiene de $g : \mathbb{N}^{n+2} \to \mathbb{N}$ y $f : \mathbb{N}^n \to \mathbb{N}$ por recursión primitiva si f, g son p.r. y

$$\begin{aligned} h(x_1, \ldots, x_n, 0) &= f(x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n, t+1) &= g(t, h(x_1, \ldots, x_n, t), x_1, \ldots, x_n) \end{aligned}$$

1.1. Producto

Enunciado: Probar que la función producto

$$Pr(x, y) = x * y$$

es primitiva recursiva.

Resolución: Definimos la función usando recursión primitiva

$$\begin{aligned} Pr(x, 0) &= n(x) \\ Pr(x, t+1) &= g(t, Pr(x, t), x) \end{aligned}$$

donde

$$g(t, v, x) = u_2^3(t, v, x) + u_3^3(t, v, x)$$

A primera vista puede resultar extraño que no se use 0 en lugar de $n(x)$ o que no se use v en lugar de $u_2^3(t, v, x)$. Estrictamente la función g toma 3 parámetros y, según el esquema anteriormente presentado, no podemos usar directamente la variable v ya que esta no es una función inicial. En su lugar podemos usar el proyector que recibe todos los parámetros y devuelve sólo uno de ellos.

1.2. Potencia

Enunciado: Probar que la función potencia

$$Pot(x, y) = x^y$$

es primitiva recursiva.

Resolución: Definimos la función usando recursión primitiva

$$\begin{aligned} Pot(x, 0) &= s(n(x)) \\ Pot(x, t+1) &= g(t, Pot(x, t), x) \end{aligned}$$

donde

$$g(t, v, x) = Pr(u_2^3(t, v, x), u_3^3(t, v, x))$$

1.3. Factorial

Enunciado: Probar que la función factorial

$$fact(n) = n!$$

es primitiva recursiva.

Resolución: Definimos la función usando recursión primitiva

$$\begin{aligned} fact(0) &= s(n(x)) \\ fact(t+1) &= g(t, fact(t)) \end{aligned}$$

donde

$$g(t, v) = Pr(s(u_1^2(t, v)), u_2^2(t, v))$$

1.4. $f^n(x)$

Enunciado: Dada una función $f : \mathbb{N} \to \mathbb{N}$ p.r. fija, probar que la función $f^n(x)$ que toma un parámetro x y aplica n veces la función f es primitiva recursiva.

Resolución: ¿Estaría bien si hacemos lo siguiente?

$$f^n(x) = \underbrace{f(\cdots f(x))}_{n \text{ veces}} \tag{1}$$

No. Estaría bien si el n fuera *fijo* pero en este caso no lo es. Si hacemos lo de (1) estaríamos probando que para cada n la función que aplica n veces f es primitiva recursiva, por ejemplo

$$\begin{aligned} f^2(x) &= f(f(x)) \\ f^3(x) &= f(f(f(x))) \\ &\vdots \end{aligned}$$

En este caso nos están pidiendo que probemos que existe una función f' que toma la cantidad de veces que debe ser aplicada f como *parámetro* y que se encarga de aplicarla las veces necesarias. Llamamos $f^n(x) = f'(x, n)$ y la definimos como

$$\begin{aligned} f'(x, 0) &= u_1^1(x) \\ f'(x, t+1) &= g(t, f'(x, t), x) \end{aligned}$$

donde

$$g(t, v, x) = f(u_2^3(t, v, x))$$

Esta diferencia es **muy** importante y no debe ser tomada a la ligera, ante cualquier duda consulte a su docente.