

# Lógica y Computabilidad

## Práctica 1: Funciones primitivas recursivas y clases PRC

2do cuatrimestre 2022

### Ejercicio 1

Para construir una constante  $k$ , aplicamos la función  $s$  (sucesor) unas  $k$  veces, partiendo inicialmente de la función  $n$  que nos devuelve el 0.

$$f(x) = k = (\underbrace{s \circ \dots \circ s}_{k \text{ veces}} \circ n)(x) = s^k(n(x))$$

### Ejercicio 2

- $f_1(x, y) = \text{suma}(x, y) = x + y$   
 $\text{suma}(x, 0) = u_1^1(x) = x$   
 $\text{suma}(x, y + 1) = g(\text{suma}(x, y), x, y)$  donde  $g(x_1, x_2, x_3) = s(u_1^3(x_1, x_2, x_3))$   
 $\Rightarrow \text{suma}(x, y + 1) = s(\text{suma}(x, y))$
- $f_2(x, y) = \text{prod}(x, y) = x \cdot y$   
 $\text{prod}(x, 0) = n(x) = 0$   
 $\text{prod}(x, y + 1) = g(\text{prod}(x, y), x, y)$  donde  $g(x_1, x_2, x_3) = \text{suma}(u_1^3(x_1, x_2, x_3), u_2^3(x_1, x_2, x_3))$   
 $\Rightarrow \text{prod}(x, y + 1) = \text{suma}(\text{prod}(x, y), x)$
- $f_3(x, y) = \text{pot}(x, y) = x^y$   
 $\text{pot}(x, 0) = s(n(x)) = 1$   
 $\text{pot}(x, y + 1) = g(\text{pot}(x, y), x, y)$  donde  $g(x_1, x_2, x_3) = \text{prod}(u_1^3(x_1, x_2, x_3), u_2^3(x_1, x_2, x_3))$   
 $\Rightarrow \text{pot}(x, y + 1) = \text{prod}(\text{pot}(x, y), x)$
- $f_4(x, y) = \underbrace{x^{x^{\cdot^{\cdot^{\cdot^x}}}}}_{y \text{ veces}}$   
 $f_4(x, 0) = 1$   
 $f_4(x, y + 1) = g(f_4(x, y), x, y)$  donde  $g(x_1, x_2, x_3) = \text{pot}(u_2^3(x_1, x_2, x_3), u_1^3(x_1, x_2, x_3))$   
 $\Rightarrow f_4(x, y + 1) = \text{pot}(x, f_4(x, y))$   
Esta función a veces se la llama “Power Tower” ([Wikipedia](#))
- $g_1(x) = \text{pred}(x) = x \div 1$   
 $\text{pred}(0) = n() = 0$  Permitimos utilizar la función nula  $n$  sin parámetros.  
 $\text{pred}(x + 1) = g(\text{pred}(x), x)$  donde  $g(x_1, x_2) = u_2^2(x_1, x_2) = x_2$   
 $\Rightarrow \text{pred}(x + 1) = x$
- $g_2(x, y) = \text{resta}(x, y) = x \div y$   
 $\text{resta}(x, 0) = u_1^1(x) = x$   
 $\text{resta}(x, y + 1) = g(\text{resta}(x, y), x, y)$  donde  $g(x_1, x_2, x_3) = \text{pred}(u_1^3(x_1, x_2, x_3))$   
 $\Rightarrow \text{resta}(x, y + 1) = \text{pred}(\text{resta}(x, y))$

- $g_3(x, y) = \max\{x, y\}$

$$g_3(x, y) = \text{suma}(\text{resta}(x, y), y) = (x \dot{-} y) + y$$

Si  $x \geq y$ , entonces  $g_3$  simplemente resta y suma  $y$  a un  $x$  que es más grande, y en efecto terminamos con  $x$  que era el máximo. En el otro caso  $x < y$ , al hacer la resta en  $\mathbb{N}$  obtenemos  $x \dot{-} y = 0$ , luego al sumar  $y$  obtenemos nuevamente  $y$  que era el máximo.

- $g_4(x, y) = \min\{x, y\}$

$$g_4(x, y) = \text{resta}(\text{suma}(x, y), \max\{x, y\}) = x + y - \max\{x, y\}$$

## Ejercicio 3

a)

Para la ida ( $\Rightarrow$ ) hacemos una demostración por inducción estructural. Primero probamos que todas las funciones iniciales cumplen la propiedad.

- Función nula

$f(x) = n(x) = 0$ . La función nula cae en el caso  $f(x) = k$  donde  $k = 0$ .

- Función sucesor

$f(x) = s(x) = x + 1$ . La función sucesor cae en el caso  $f(x) = x + k$  donde  $k = 1$ .

- Función proyector

$f(x_1, \dots, x_n) = u_i^n(x_1, \dots, x_n) = x_i$ . La función proyector cae en el caso  $f(x_1, \dots, x_n) = x_i + k$  donde  $k = 0$ .

Paso inductivo. Supongamos que existe  $h_m \in \mathcal{C}_c$  generada a partir de  $m$  composiciones, tal que  $h_m(x_1, \dots, x_n) = k$  o bien  $h_m(x_1, \dots, x_n) = x_i + k$ . Queremos ver si cualquier  $h_{m+1} \in \mathcal{C}_c$  también cumple la propiedad. Para generar  $h_{m+1}$  componemos  $h_m$  con alguna función  $f \in \mathcal{C}_c$ .

- Caso  $f(x) = n(x)$

$$h_{m+1} = f(h_m(x_1, \dots, x_n)) = n(h_m(x_1, \dots, x_n)) = 0.$$

No importa la forma de  $h_m$  pues  $n(x) = 0$  para cualquier  $x$ .

- Caso  $f(x) = s(x)$

- Caso  $h_m(x_1, \dots, x_n) = x_i + q$

$$h_{m+1} = f(h_m(x_1, \dots, x_n)) = s(x_i + q) = x_i + q + 1 = x_i + k \text{ donde } k = q + 1.$$

- Caso  $h_m(x_1, \dots, x_n) = q$

$$h_{m+1} = f(h_m(x_1, \dots, x_n)) = s(q) = q + 1 = k \text{ donde } k = q + 1.$$

- Caso  $f(x) = u_i^n(x)$

Como  $h_m : \mathbb{N}^n \rightarrow \mathbb{N}$ , necesariamente  $f(x) = u_1^1(x)$  para poder componer  $f \circ h_m$ .

- Caso  $h_m(x_1, \dots, x_n) = x_i + k$

$$h_{m+1} = f(h_m(x_1, \dots, x_n)) = u_1^1(x_i + k) = x_i + k.$$

- Caso  $h_m(x_1, \dots, x_n) = k$

$$h_{m+1} = f(h_m(x_1, \dots, x_n)) = u_1^1(k) = k.$$

- Caso  $f(x) = x + r$

- Caso  $h_m(x_1, \dots, x_n) = x_i + q$

$$h_{m+1} = f(h_m(x_1, \dots, x_n)) = x_i + q + r = x_i + k \text{ donde } k = q + r.$$

- Caso  $h_m(x_1, \dots, x_n) = q$

$$h_{m+1} = f(h_m(x_1, \dots, x_n)) = q + r = k \text{ donde } k = q + r.$$

- Caso  $f(x) = k$

- Caso  $h_m(x_1, \dots, x_n) = x_i + q$   
 $h_{m+1} = f(h_m(x_1, \dots, x_n)) = f(x_i + q) = k.$
- Caso  $h_m(x_1, \dots, x_n) = q$   
 $h_{m+1} = f(h_m(x_1, \dots, x_n)) = f(q) = k.$

Por lo tanto, partiendo de una función  $h_m \in \mathcal{C}_c$ , vemos que al realizar una composición con alguna función  $f \in \mathcal{C}_c$  obtenemos una función  $h_{m+1} \in \mathcal{C}_c$  (pues  $\mathcal{C}_c$  es cerrado por composición) que mantiene la propiedad enunciada.

Para la vuelta ( $\Leftarrow$ ) mostramos que podemos construir cualquier  $f(x_1, \dots, x_n) = k$  o  $f(x_1, \dots, x_n) = x_i + k$  a partir de composición de las funciones iniciales, y por lo tanto  $f \in \mathcal{C}_c$ .

- $f(x_1, \dots, x_n) = k = s^k(n(x_1, \dots, x_n))$
- $f(x_1, \dots, x_n) = x_i + k = s^k(u_i^n(x_1, \dots, x_n))$

b)

En el ejercicio 2 vimos que la función  $\text{suma}(x, y) = x + y$  es P.R. pero  $\text{suma} \notin \mathcal{C}_c$  pues no cumple con la propiedad.

## Ejercicio 4

Cualquier clase PRC contiene las funciones iniciales y está cerrada por recursión primitiva y composición. Para mostrar que los predicados están en cualquier clase PRC, es suficiente con mostrar que se pueden construir a partir de las funciones iniciales utilizando recursión primitiva y/o composición.

Para simplificar la escritura vamos a utilizar la función  $\alpha(x)$  la cual es PR a partir de las iniciales y por lo tanto pertenece a cualquier clase PRC.

$$\alpha(x) = \begin{cases} 1 & \text{si } x = 0 \\ 0 & \text{si no} \end{cases}$$

Además, podemos definir el predicado  $\neg(x) = \alpha(x)$  que niega otro predicado.

$$\leq) \quad p(x, y) = \begin{cases} 1 & \text{si } x \leq y \\ 0 & \text{si no} \end{cases} = \alpha(x \dot{-} y)$$

$$\geq) \quad p(x, y) = \begin{cases} 1 & \text{si } x \geq y \\ 0 & \text{si no} \end{cases} = \alpha(y \dot{-} x)$$

$$=) \quad p(x, y) = \begin{cases} 1 & \text{si } x = y \\ 0 & \text{si no} \end{cases} = (x \leq y) \cdot (x \geq y)$$

$$\neq) \quad p(x, y) = \begin{cases} 1 & \text{si } x \neq y \\ 0 & \text{si no} \end{cases} = \neg(x = y)$$

$$<) \quad p(x, y) = \begin{cases} 1 & \text{si } x < y \\ 0 & \text{si no} \end{cases} = \neg(x \geq y)$$

$$>) \quad p(x, y) = \begin{cases} 1 & \text{si } x > y \\ 0 & \text{si no} \end{cases} = \neg(x \leq y)$$

## Ejercicio 5

Pendiente

## **Ejercicio 6**

Pendiente

## **Ejercicio 7**

Pendiente

## **Ejercicio 8**

Pendiente

## **Ejercicio 9**

Pendiente

## **Ejercicio 10**

Pendiente

## **Ejercicio 11**

Pendiente

## **Ejercicio 12**

Pendiente

## **Ejercicio 13**

Pendiente

## **Ejercicio 14**

Pendiente

## **Ejercicio 15**

Pendiente