Funciones primitivas recursivas y clases PRC (Parte 2)

María Emilia Descotte

25 de agosto de 2017

Ejercicio 1. *Demostrar que la siguiente función $f : \mathbb{N} \to \mathbb{N}$ es primitiva recursiva:*

$$\begin{aligned} f(0) &= 1 \\ f(1) &= 2 \\ f(2) &= 4 \\ f(n+3) &= f(n) + f(n+1)^2 + f(n+2)^3 \end{aligned}$$

Idea: En el esquema de recursión primitiva visto en clase, para calcular cada paso de la recursión, solo puede usarse el paso anterior. Sin embargo, f utiliza tres pasos anteriores. Para resolver este problema, vamos a construir una función auxiliar h que guarde tres valores de f a la vez y que nos permita recuperar la función f componiéndola con algo p.r. La idea es que esa función auxiliar h sí va a ajustarse a nuestro esquema de recursión primitiva porque al guardar de a tres valores, utilizando únicamente el paso anterior de la recursión, tendremos toda la información que necesitamos. Finalmente probaremos que f es p.r. por ser composición de funciones p.r.

Resolución:

Sea $h : \mathbb{N} \to \mathbb{N}$ definida por:

$$h(n) = [f(n), f(n+1), f(n+2)].$$

Veamos que h se ajusta al esquema de recursión primitiva visto en clase:

$$h(0) = [f(0), f(1), f(2)] = [1, 2, 4] \text{ (es una constante y por lo tanto es p.r.)}$$

$$\begin{aligned} h(n+1) &= [f(n+1), f(n+2), f(n+3)] \text{ (por definición de } h) \\ &= [f(n+1), f(n+2), f(n) + f(n+1)^2 + f(n+2)^3] \text{ (por definición de } f) \\ &= [h(n)[2], h(n)[3], h(n)[1] + (h(n)[2])^2 + (h(n)[3])^3] \text{ (por definición de } h) \end{aligned}$$

Dado que $h(0)$ resultó una función p.r. (es constante) y que $h(n+1)$ resultó una función p.r. (es composición de todas funciones p.r: tomar observadores de listas, sumar, elevar a potencias, construir listas) que sólo depende del valor anterior $h(n)$, la función h se ajusta al esquema de recursión primitiva visto en clase y, por lo tanto, es p.r.

Por último, observemos que $f(n) = h(n)[1]$ y, por lo tanto, es primitiva recursiva ya que es composición de funciones que lo son (vimos que h lo es y las observadoras de listas lo son).

□

Ejercicio 2. *Sea $P : \mathbb{N} \to \{0, 1\}$ un predicado primitivo recursivo. Probar que las siguientes funciones son primitivas recursivas:*

a. $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ tal que $f(l, i) = n$ si la posición del $i+1$-ésimo elemento de la lista (sin ceros al final) codificada por l que cumple con P es n. En caso de que no exista un $i+1$-ésimo elemento

tal, $f(l, i)$ *devuelve* 0 *(incluso si* $l = 0$ *que no codifica ninguna lista). Por ejemplo, si* $P(x)$ *representa 'x es par' entonces*

$$f([1, 8, 3, 6, 6], 0) = 2$$
$$f([1, 8, 3, 6, 6], 1) = 4$$
$$f([1, 8, 3, 6, 6], 2) = 5$$
$$f([1, 8, 3, 6, 6], 3) = 0$$

b. $filter : \mathbb{N} \to \mathbb{N}$ *que interpreta una entrada* l *como lista sin ceros al final y devuelve la lista con los elementos de la misma que cumplen con* P *(en el mismo orden que estaban en la lista original).* $filter(0)$ *se define como* 1*. Por ejemplo, si* $P(x)$ *representa 'x es par' entonces*

$$filter([1, 8, 3, 6, 6]) = [8, 6, 6]$$
$$filter([1, 3, 3, 7, 9]) = [\ ]$$

Idea: a. Vamos a ver que la función se ajusta al esquema de recursión primitiva visto en clase ya que en el caso $f(l, 0)$ lo que querremos calcular es el primer índice donde se cumple P y podemos hacerlo con una minimización acotada; y en el paso recursivo, usando el paso anterior también podemos calcular el valor de f buscando un mínimo pero esta vez también acotado por abajo.

b. Para este ítem podemos usar que la función f del ítem anterior es p.r. y, a partir de ella, recuperar las posiciones de la lista que nos interesan (aquellas donde vale P) pensándolas como la primera posición donde vale P, la segunda posición donde vale P, etc.

Resolución:

a. Es fácil convencerse (puede verse por inducción) de que

$$f(l, 0) = min_{1 \leq t \leq |l|} P(l([t]))$$
$$f(l, i + 1) = min_{1 \leq t \leq |l|} P(l([t])) \wedge t > f(l, i)$$

Usando que la minimización acotada, la longitud de una lista, los observadores de listas, la comparación por mayor y el predicado P son p.r., es inmediato que f es p.r.

b. De nuevo, es fácil convencerse de que

$$filter(l) = \prod_{i=1}^{|l|} p_i^{l[f(l,i)]}$$

Luego, como tomar longitud de una lista, calcular el i-ésimo primo, f, los observadores de listas, las potencias y tomar productoria son funciones p.r., $filter$ resulta p.r.

Observación (en ambos ítems): Los casos con $l = 0$ se comportan bien pues $|0| = 0$. □

Ejercicio 3. *Sea $k \in \mathbb{N}$, $f : \mathbb{N} \to \mathbb{N}$ una función tal que $f(x+1) < x+1$ para todo x, y sea $h : \mathbb{N}^2 \to \mathbb{N}$ definida como*

$$\begin{aligned} h(x,0) &= k \\ h(x,t+1) &= g(x,t,h(x,f(t+1))) \end{aligned}$$

Demostrar que si f y g pertenecen a una clase PRC $\mathcal{C}$, entonces también h pertenece a $\mathcal{C}$.

Idea: La idea es similar a la del Ejercicio 1 pero en este caso, el problema no es que necesitemos más de un valor anterior para el paso recursivo, sino que necesitamos uno que sabemos que es anterior pero no es necesariamente el inmediatamente anterior y no sabemos exactamente cuál es. Entonces vamos a definir una función auxiliar $\tilde{h}$ a partir de la cual podamos recuperar h componiendo con funciones p.r. y que guarde todos los valores de h hasta el paso actual.

Resolución: Sea $\tilde{h} : \mathbb{N}^2 \to \mathbb{N}$ definida por:

$$\tilde{h}(x,t) = [h(x,0), h(x,1), \cdots, h(x,t)].$$

Veamos que $\tilde{h}$ se ajusta al esquema de recursión primitiva visto en clase:

$$\tilde{h}(x,0) = [h(x,0)] = [k] \text{ (es una constante y por lo tanto es p.r.)}$$

$$\begin{aligned} \tilde{h}(x,t+1) &= [h(x,0), h(x,1), \cdots, h(x,t+1)] \text{ (por definición de } \tilde{h}) \\ &= [h(x,0), h(x,1), \cdots, h(x,t), g(x,t,h(x,f(t+1)))] \text{ (por definición de } h) \\ &= [\tilde{h}(x,t)[1], \cdots, \tilde{h}(x,t)[t+1], g(x,t,\tilde{h}(x,t)[f(t+1)+1])] \text{ (por definición de } \tilde{h}) \end{aligned}$$

Observar que el último paso usa la hipótesis de que $f(t+1) < t+1$.

Usando que la clase $\mathcal{C}$ contiene a todas las funciones p.r. y es cerrada por composición y recursión primitiva (por ser PRC), podemos concluir que $\tilde{h}$ pertenece a la clase $\mathcal{C}$ (sumas, observadoras de listas y constructoras de listas son p.r.).

Por último, observemos que $h(x,t) = \tilde{h}(x,t)[t+1]$ y, por lo tanto, pertenece a la clase $\mathcal{C}$ ya que es composición de funciones que pertencen (vimos que $\tilde{h}$ pertence y las observadoras de listas pertencen por ser p.r.).

□