Resueltos Lógica y Computabilidad

Ignacio E. Losiggio

February 16, 2019

1 Práctica 1 — Funciones primitivas recursivas y clases PRC

1.1 Mostrar que, dado un k fijo, la función constante $f(x) = k$ puede definirse usando las funciones iniciales y composición (sin usar recursión primitiva)

Podemos empezar bien qué funciones nos piden, si $k = 0$ entonces $f_0(x) = n(x)$ (con n la función constante 0, que es primitiva). Si $k = 1$ entonces $f_1 = s(f_0(x))$, que sabemos que es primitiva recursiva porque lo hicimos una composición válida. Podemos intentar entonces búqueda de un patrón dentro de las funciones que nos piden:

k	$f_k(x)$	Expresión expandida
$k = 0$	$f_0(x) = n(x)$	$n(x)$
$k = 1$	$f_1(x) = s(f_0(x))$	$s(n(x))$
$k = 2$	$f_2(x) = s(f_1(x))$	$s(s(n(x))$
$k = n$	$f_n(x) = s(f_{n-1}(x))$	$s(s(\ldots s(n(x))))$

Cómo podemos ver, construir la función constantemente k sólo requiere de aplicar $s(x)$ $k$veces sobre $n(x)$.

1.2 Probar que las siguientes funciones son primitivas recursivas, mostrando que pueden obtenerse a partir de las funciones iniciales usando composición t/o recursión primitiva:

1.2.1 $f_1(x, y) = x + y$

Tomamos la estructura de la recursión primitiva:

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

Con esto tenemos que elegir una $h(x)$ y una $g(n,x,t)$ que nos construyan la suma y copypastear cómo corresponda:

$$\begin{aligned}
f_1(x,0) &= u_1^1(x) \\
f_1(x,t+1) &= g(f_1(x,t),x,t) \\
g(n,x,t) &= s(u_1^3(n,x,t))
\end{aligned}$$

¿No queda claro por qué funciona? Hagamos un par de reemplazos a ver que pasa.

$$\begin{aligned}
g(n,x,t) &= s(u_1^3(n,x,t)) \\
&= s(n) \\
f_1(x,0) &= u_1^1(x) \\
&= x \\
f_1(x,t+1) &= g(f_1(x,t),x,t) \\
&= s(f_1(x,t))
\end{aligned}$$

Ajá!

$$\begin{aligned}
f_1(x,0) &= x \\
f_1(x,t+1) &= s(f_1(x,t))
\end{aligned} \quad \text{es equivalente a } f_1(x,y) = \begin{cases} x & \text{si } y = 0 \\ 1 + f_1(x,y-1) & \text{sino} \end{cases}$$

Bueno, con este truco vamos a meterle a todo el resto de los ejercicios de este punto!

1.2.2 $f_2(x,y) = x \cdot y$

Bien, pongamos de vuelta en práctica lo que acabamos de hacer. Si tu intuición te llama a que vamos a tener que usea $f_1(x,y)$ acá estás más que en lo cierto. Pero primero plantiemos una función partida común y corriente a ver cómo vamos desde ahí a nuestra recursión primitiva:

$$f_2(x,y) = \begin{cases} 0 & \text{si } y = 0 \\ x + f_2(x,y-1) & \text{sino} \end{cases}$$

Y de ahí salimos a buscar nuestro $h(x)$ y nuestro $g(n,x,t)$:

$$\begin{aligned}
f_2(x,0) &= n(x) \\
&= 0 \\
f_2(x,t+1) &= g(f_2(x,t),x,t) \\
g(n,x,t) &= f_1(u_2^3(n,x,t), u_1^3(n,x,t)) \\
&= x + n
\end{aligned}$$

Y para estar del todo seguros hacemos todos los reemplazos cómo la otra vez:

$$\begin{aligned} f_2(x,0) &= 0 \\ f_2(x,t+1) &= x + f_2(x,t) \end{aligned}$$

Justo lo que buscábamos.

1.2.3 $f_3(x,y) = x^y$

Bueno, supongo que ya te diste cuenta de cuál es el patrón, te paso cuál es la $h(x)$ y la $g(n,x,t)$:

$$\begin{aligned} h(x) &= s(n(x)) \\ &= 1 \\ g(n,x,t) &= f_2(u_2^3(n,x,t), u_1^3(n,x,t)) \\ &= x \cdot n \end{aligned}$$

$$\begin{aligned} f_3(x,0) &= 1 \\ f_3(x,t+1) &= x \cdot f_3(x,t) \end{aligned}$$

1.2.4 $f_4(x,y) = x^{x^{\cdot^{\cdot^{\cdot^{x}}}}}$

Observación: se asume que $f_4(x,0) = 1$

$$\begin{aligned} h(x) &= s(n(x)) \\ &= 1 \\ g(n,x,t) &= f_3(u_2^3(n,x,t), u_1^3(n,x,t)) \\ &= x^n \end{aligned}$$

$$\begin{aligned} f_4(x,0) &= 1 \\ f_4(x,t+1) &= x^{f_4(x,t)} \end{aligned}$$

1.2.5 $g_1(x) = x \dot{-} 1$

Cómo sólo estamos operando en los $\mathbb{N}$ el predecesor de 0 es 0. Una cosa a notar es que (según la teórica, filmina 25, primera clase de computabilidad) "En este contexto, una función 0-aria es una constante k. Si f es 1-aria y $t = 0$ entonces $h(t) = k = s^{(k)}(n(t))$.".

$$
\begin{aligned}
g_1(0) &= n(0) \\
&= 0 \\
g_1(t+1) &= u_2^2(g_1(t), t) \\
&= t
\end{aligned}
$$

1.2.6 $g_2(x, y) = x \dot{-} y$

Observación:

$$
x \dot{-} y = \begin{cases} x - y & \text{si } y \leq x \\ 0 & \text{si } y > x \end{cases}
$$

Bueno, entonces podemos empezar a intentar construir la resta.

$$
\begin{aligned}
f(x) &= u_1^1(x) \\
&= x \\
g(n, x, t) &= g_1(u_1^3(n, x, t)) \\
&= n \dot{-} 1
\end{aligned}
$$

¿Está bien esto? Probemos algún número:

$$
\begin{aligned}
g_2(5,3) &= g(g_2(5,2), 5, 3) && = g_2(5,2) - 1 \\
g_2(5,3) &= g(g_2(5,1), 5, 3) - 1 && = g_2(5,1) - 1 - 1 \\
g_2(5,3) &= g(g_2(5,0), 5, 3) - 1 - 1 && = g_2(5,0) - 1 - 1 - 1 \\
g_2(5,3) &= 5 - 1 - 1 - 1 \\
g_2(5,3) &= 2
\end{aligned}
$$

¡Bien!

1.2.7 $g_3(x, y) = \max\{x, y\}$

Como operamos con los naturales sabemos que si $x < y$ entonces $x - y = 0$ podemos usar la recursión primitiva para construir una función de decisión.

$$
\begin{aligned}
d(x, y, 0) &= u_2^2(x, y) \\
&= y \\
d(x, y, t+1) &= u_2^4(d(x, y, t), x, y, t) \\
&= x
\end{aligned}
$$

Ahora, esto no es $g_3(x, y)$ pero está peligrosamente cerca, podemos usar la composición para finalmente construirlo.

$$\begin{aligned} g_3(x, y) &= d(u_1^2(x, y), u_2^2(x, y), g_2(x, y)) \\ &= d(x, y, x \dot{-} y) \end{aligned}$$

1.2.8 $g_4(x, y) = \min\{x, y\}$

La idea es la misma que recién, pero parametrizamos la decisión al revés:

$$\begin{aligned} g_3(x, y) &= d(u_2^2(x, y), u_1^2(x, y), g_2(x, y)) \\ &= d(y, x, x \dot{-} y) \end{aligned}$$

1.3 Sea $\mathcal{C}_i$ la clase de funciones iniciales y $\mathcal{C}_c$ la (mínima) clase que extiende a $\mathcal{C}_i$ y se encuentra cerrada por composición:

Es decir, $\mathcal{C}_i$ aquella que contiene a:

$$n(x) = 0 \qquad s(x) = x + 1 \qquad u_i^n(x_1, \ldots x_n) = x_i \text{ para cada } n \in \mathbb{N} \text{ e } i \in \{1, \ldots, n\}$$

Y $\mathcal{C}_c$ aquella que si $f, g_1, \ldots, g_m$ están en $\mathcal{C}_c$, entonces $h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$ también lo está.

1.3.1 Demostrar que para toda $f : \mathbb{N}^n \to \mathbb{N}$, f, está en $\mathcal{C}_c$ sii existe $k \geq 0$ tal que, o bien sucede $f(x_1, \ldots, x_n) = k$, o bien para algún i fijo, se tiene $f(x_1, \ldots, x_n) = x_i + k$.

El enunciado nos propone demostrar que $\mathcal{C}_c$ sólo tiene funciones que devuelven constantes y funciones que suman constantes a *uno* de sus parámetros (y siempre el mismo).

Demostrar la vuelta es particularmente fácil, primero construyo la función 1-aria que suma k con el método del primer ejercicio (teniendo $u_1^1(x)$ cómo caso base en lugar de $n(x)$ y la llamo $k(x)$. También construyo una función constantemente 0 que tome n argumentos.

$$\begin{aligned} k(x) &= s(\ldots s(u_1^1(x))) \qquad\qquad \text{Nota: si } k = 0 \text{ entonces } k(x) = u_1^1(x) \\ z(x_1, \ldots, x_n) &= n(u_1^n(x_1, \ldots, x_n)) \end{aligned}$$

Ahora sólo tengo que cubrir cada caso:

- Si tengo una $f : \mathbb{N}^n \to \mathbb{N}$ que es constantemente k.

$$f(x_1, \ldots, x_n) = k(z(x_1, \ldots, x_n))$$

- Si tengo una $f : \mathbb{N}^n \to \mathbb{N}$ que es constantemente $x_i + k$.

$$f(x_1, \ldots, x_n) = k(u_i^n(x_1, \ldots, x_n))$$

Cómo sólo hice composiciones para construir la f entonces esto debería bastar para demostrar que las funciones constantes y las que suman constantes pertenecen todas a $\mathcal{C}_c$.

Para demostrar la ida necesito asegurarme que las funciones en $\mathcal{C}_c$ sólo pueden ser constantes (o sumar constantes). Si empezamos examinando los casos base…

Primitiva	Forma	k
$n(x) = 0$	$f(x) = k$	0
$s(x) = x + 1$	$f(x) = x + k = x + 1$	1
$u_i^n(x_1, \ldots, x_n) = x_i$	$f(x_1, \ldots, x_n) = x_i + k$	0

…tenemos que todos ellos tienen o la forma k o la de $x_i + k$. Ahora examinemos nuestra regla de composición:

Sea:

$f : \mathbb{N}^m \to \mathbb{N}$

$g_1, \ldots, g_m : \mathbb{N}^n \to \mathbb{N}$

Podemos construir $h : \mathbb{N}^n \to \mathbb{N}$ de la siguiente manera:

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

Puedo tomar $f_1, \ldots, f_n$ que tengan la forma k o la forma $x_i + k$ y analizar qué pasa si las compongo (paso inductivo).

	$f_i(x_1, \ldots, x_n) = k$		$f_i(x_1, \ldots, x_n) = x_j + k$	
Composición	Forma	k'	Forma	k'
$n(f_i(x_1, \ldots, f_n))$	k	0	k	0
$s(f_i(x_1, \ldots, f_n))$	k	$k+1$	$x_j + k$	$k+1$
$u_i^n(f_1(\ldots),\ dots, f_i(\ldots), \ldots, f_n(\ldots))$	k	k	$x_j + k$	k
$f_i(f_1(\ldots), \ldots, f_n(\ldots))$	k	k	$f_j(\ldots) + k$	$f_j(\ldots) + k$

Si bien el último caso puede parecer molesto, por la hipótesis inductiva sabemos que f_k tiene o bien forma k o bien forma $x_i + k$, entonces $k' = k_{f_j} + k_{f_i}$ y la forma se mantiene.

Entonces, si parto de funciones en $\mathcal{C}_c$ y las compongo siempre voy a llegar a una función de la forma k o $x_i + k$. Y dado que las funciones iniciales tienen también esa forma entonces todo $\mathcal{C}_c$ la tiene también.

1.3.2 Mostrar que existe una función primitiva recursiva que no está en $\mathcal{C}_c$

$f_1(x, y)$ del ejercicio 2. Es primitiva recursiva porque está construida en base a la recursión primitiva y a la composición de funciones. Pero depende de ambos parámetros para emitir su resultado (a diferencia de todas las de $\mathcal{C}_c$).

1.4 Mostrar que los predicados $\leq$, $\geq$, $=$, $\neq$, $<$ t $>$: $\mathbb{N}^2 \to \{0,1\}$ están en cualquier clase *PRC*

Llamamos *predicado* a cualquier función $p : \mathbb{N}^n \to \{0,1\}$, escribimos $p(a_1, \ldots, a_n)$ en lugar de $p(a_1, \ldots, a_n) = 1$ y decimos, informalmente, en este caso, que "$p(a_1, \ldots, a_n)$ es verdadero".

Vamos a empezar poniéndoles nombres a las funciones que queremos construir y vamos a tener en cuenta las funciones que construímos en el ejercicio 2 para facilitarnos la vida:

$$f_1(x,y) = \begin{cases} 1 & \text{si } x \leq y \\ 0 & \text{sino} \end{cases} \quad f_2(x,y) = \begin{cases} 1 & \text{si } x \geq y \\ 0 & \text{sino} \end{cases} \quad f_3(x,y) = \begin{cases} 1 & \text{si } x = y \\ 0 & \text{sino} \end{cases}$$

$$f_4(x,y) = \begin{cases} 1 & \text{si } x \neq y \\ 0 & \text{sino} \end{cases} \quad f_5(x,y) = \begin{cases} 1 & \text{si } x < y \\ 0 & \text{sino} \end{cases} \quad f_6(x,y) = \begin{cases} 1 & \text{si } x > y \\ 0 & \text{sino} \end{cases}$$

Y para hacernos la vida más fácil vamos a construir $\alpha(x)$ que es la negación lógica en nuestro modelo.

$$\begin{aligned} \alpha(0) &= s(n(0)) \\ \alpha(t+1) &= n(t) \end{aligned}$$

¡Y ahora sí!

$$\begin{aligned}
f_1(x,y) &= \alpha(x \dot{-} y) \\
f_2(x,y) &= \alpha(y \dot{-} x) \\
f_3(x,y) &= (x \leq y) \cdot (y \leq x) \\
f_4(x,y) &= \alpha(x = y) \\
f_5(x,y) &= \alpha(x \geq y)) \\
f_6(x,y) &= \alpha(x \leq y))
\end{aligned}$$

Nota: f_3 puede parecer raro al no parecer una composición tan simple, plantearlo prolijamente requiere una funcion auxiliar que cambia el orden de los parámetros.

1.5 Sean $\mathcal{C}$ una clase *PRC*, $f_1, \ldots, f_k, g : \mathbb{N}^n \to \mathbb{N}$ funciones en $\mathcal{C}$ y $p_1, \ldots, p_k : \mathbb{N}^n \to \{0,1\}$ predicados disjuntos en $\mathcal{C}$. Mostrar que la siguiente función también está en $\mathcal{C}$:

$$h(x_1,\ldots,x_n) = \begin{cases} f_1(x_1,\ldots,x_n) & \text{si } p_1(x_1,\ldots,x_n) \\ \qquad\vdots & \\ f_k(x_1,\ldots,x_n) & \text{si } p_k(x_1,\ldots,x_n) \\ g(x_1,\ldots,x_n) & \text{sino} \end{cases}$$

Observar que h queda completamente determinada por este esquema.

Nota: Al ser $p_1, \ldots, p_k$ disjuntos no sucede $p_i(a_1,\ldots,a_n) = p_j(a_1,\ldots,a_n) = 1$ con $i \neq j$ para ningún $(a_1,\ldots,a_n) \in \mathbb{N}^n$.

Si podemos resolverlo para el caso de $k = 1$ entonces podremos resolverlo para cualquier k arbitrario. Podemos construir $h(x_1,\ldots,x_n)$ de la siguiente forma:

$$\begin{aligned}
h(x_1,\ldots,x_n) &= h_1(x_1,\ldots,x_n) \\
h_i(x_1,\ldots,x_n) &= \begin{cases} f_i(x_1,\ldots,x_n) & \text{si } p_i(x_1,\ldots,x_n) \\ h_{i+1}(x_1,\ldots,x_n) & \text{sino} \end{cases} \qquad \forall i \neq k+1 \\
h_{k+1}(x_1,\ldots,x_n) &= g(x_1,\ldots,x_n)
\end{aligned}$$

Ahora sólo queda resolverlo para el caso de $k = 1$. ¡Cosa que ya hicimos en el ejercicio 2 con $\max\{x,y\}$ y $\min\{x,y\}$! Repasemos esa solución: construimos una función de decisión d_i y la encapsulamos para construir el h_i.

$$\begin{aligned} d_i(x_1,\dots,x_n,0) &= f_i(x_1,\dots,x_n) \\ d_i(x_1,\dots,x_n,t+1) &= h_{i+1}(x_1,\dots,x_n) \end{aligned}$$

$$\begin{aligned} h_i(x_1,\dots,x_n) &= d_i(x_1,\dots,x_n,p(x_1,\dots,x_n)) && \forall i \neq k+1 \\ h_{k+1}(x_1,\dots,x_n) &= g(x_1,\dots,x_n) \end{aligned}$$

Ahora tenemos todo listo, sólo queda enunciar a nuestro h:

$$h(x_1,\dots,x_n) = h_1(x_1,\dots,x_n)$$

Nota: Una forma alternativa (y mucho más simple) que me dieron fué construir $h(x_1,\dots,x_n)$ de la siguiente manera:

$$\begin{aligned} h(x_1,\dots,x_n) = {} & f_1(x_1,\dots,x_n)\cdot p_1(x_1,\dots,x_n) \\ + {} & f_2(x_1,\dots,x_n)\cdot p_2(x_1,\dots,x_n) \\ & \vdots \\ + {} & f_{k-1}(x_1,\dots,x_n)\cdot p_{k-1}(x_1,\dots,x_n) \\ + {} & f_k(x_1,\dots,x_n)\cdot p_k(x_1,\dots,x_n) \\ + {} & g(x_1,\dots,x_n)\cdot \alpha(p_1(x_1,\dots,x_n)+\dots+p_k(x_1,\dots,x_n)) \end{aligned}$$

1.6 Demostrar las siguientes afirmaciones

1.6.1 El predicado $par(x) = \begin{cases} 1 & \text{si } x \text{ es par} \\ 0 & \text{sino} \end{cases}$ está en toda clase PRC

La menor clase PRC es la que contiene a las primitivas recursivas, si podemos demostrar $par(x)$ cómo primitiva recursiva tenemos el ejercicio hecho.

Intentemos definir $par(x)$ con una recursión primitiva:

$$\begin{aligned} par(0) &= 1 \\ par(t+1) &= \alpha(par(t)) \end{aligned}$$

Podemos ver que si el número es par nos va a quedar una cantidad par de α por lo que se cancelan y queda 1 de respuesta, y en el caso contrario un 0 cómo era esperado.

Cómo α es primitiva recursiva y la clase de las primitivas recursivas está cerrada por composición y recursión primitiva entonces $par(x)$ pertenece a esa clase. Cómo esa clase es la menor clase PRC entonces $par(x)$ pertenece a *toda* clase PRC.

1.6.2 Demostrar que la función $f(x) = \lfloor x/2 \rfloor$ está en toda clase PRC

Podemos intentar lo mismo que en el punto anterior:

$$\begin{aligned} f(0) &= 0 \\ f(t+1) &= par(t) + f(t) \end{aligned}$$

1.6.3 Sea $\mathcal{C}$ una clase PRC, y sean $f : \mathbb{N}^n \to \mathbb{N}$ y g_1, $g_2 : \mathbb{N}^{n+2} \to \mathbb{N}$ funciones en $\mathcal{C}$. Mostrar que también está en $\mathcal{C}$ cualquier h que cumpla:

$$h(x_1, \ldots, x_n, t) = \begin{cases} f(x_1, \ldots, x_n) & \text{si } t = 0 \\ g_1(x_1, \ldots, x_n, k, h(x_1, \ldots, x_n, t-1)) & \text{si } t = 2 \cdot k + 1 \\ g_2(x_1, \ldots, x_n, k, h(x_1, \ldots, x_n, t-1)) & \text{si } t = 2 \cdot k + 2 \end{cases}$$

Observar que h queda completamente determinada por este esquema.

Podemos usar una recursión primitiva no tan trivial:

$$\begin{aligned} h(x_1, \ldots, x_n, 0) &= f(x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n, t+1) &= g_1(x_1, \ldots, x_n, \lfloor x/2 \rfloor, h(x_1, \ldots, x_n, t)) \cdot par(t) \\ &\quad + g_2(x_1, \ldots, x_n, \lfloor x/2 \rfloor, h(x_1, \ldots, x_n, t)) \cdot \alpha(par(t)) \end{aligned}$$

Si bien el término para $t+1$ es complejo podemos ver que usamos funciones de $\mathcal{C}$ (f, g_1, g_2), que llamamos a h con el valor anterior de t y que usamos funciones que ya demostramos como primitivas recursivas (y por ende pertenecen a $\mathcal{C}$) siendo estas par, α, la suma, la multiplicación y la división truncada.

El reordenamiento de los parámetros también es primitivo recursivo, dado que nos lo regalan los proyectores (u_i^n).