Clase práctica 8: Funciones Primitivas Recursivas

Laski (inspirado en Facundo Carreiro y Hernán Czemerinski)

Primer Cuatrimestre 2014

1. Repaso de la teórica

Para probar que una función es primitiva recursiva hace falta mostrar que se puede construir a partir de finitas aplicaciones de:

Funciones Iniciales

- Cero: $n(x) = 0$
- Sucesor: $s(x) = x + 1$
- Proyecciones: $u_i^n(x_1, ..., x_n) = x_i$

Las primeras dos sirven para definir los naturales. Las proyecciones sirven para cuando queremos usar solo algunas de las variables que nos pasan por parámetro.

Composición

Si $f : \mathbb{N}^k \to \mathbb{N}$ y $g_1, ..., g_k : \mathbb{N}^n \to \mathbb{N}$ son p.r., entonces $h : \mathbb{N}^n \to \mathbb{N}$ definida por

$$h(\bar{x}) = f(g_1(\bar{x}), ..., g_k(\bar{x}))$$

es p.r.

En resumen, podemos componer funciones p.r. y el resultado sigue siendo p.r.

Nota: $\bar{x}$ es la forma corta de decir $x_1, ..., x_n$.

Recursión primitiva

Si $g : \mathbb{N}^{n+2} \to \mathbb{N}$ y $f : \mathbb{N}^n \to \mathbb{N}$ son p.r., entonces $h : \mathbb{N}^{n+1} \to \mathbb{N}$ definida

por

$$h(\bar{x}, 0) = f(\bar{x})$$
$$h(\bar{x}, t+1) = g(t, h(\bar{x}, t), \bar{x})$$

es p.r.

Es decir, tenemos que definir una función para el caso base (f) y una función para el caso recursivo (g), que va a tomar como parámetros en qué paso estoy (t), cuál es el resultado (ya calculado) de ese paso ($h(\bar{x}, t)$ y cuáles son los parámetros $\bar{x}$, y debería devolver el valor de la función en el paso $t+1$ (no es necesario que use todos sus parámetros, de hecho en general no va a usarlos todos).

2. Ejercicios

Nota: para los ejercicios vamos a suponer p.r. las siguientes funciones vistas en la teórica.

- $(+)$ (la función suma).
- $(-)$ (la resta truncada, que da 0 cuando la resta normal debería dar un resultado negativo).
- $(=)$ (la comparación por igualdad).
- La división en casos, siempre y cuando sean finitos, con condiciones disjuntas y cuyas guardas (predicado de cada caso) y resultados sean funciones p.r. No se preocupen si no entienden todo lo que acabo de decir, más adelante lo van a entender bien (en particular en el ejercicio 5).
- La minimización acotada (más información en el ejercicio 6).

2.1. Producto

Enunciado: Probar que la función producto $Pr(x, y) = x.y$ es primitiva recursiva.

Resolución: Definamos la función usando recursión primitiva

$$Pr(x, 0) = n(x)$$
$$Pr(x, t+1) = g(t, Pr(x, t), x)$$

donde

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

A primera vista puede resultar extraño que haya usado $n(x)$ en lugar de 0 o que no use v en lugar de $u_2^3(t, v, x)$. Pero formalmente la funcion g toma tres parámetros, y no podemos usar directamente la variable v porque no es una función inicial. En su lugar, usamos el proyector que recibe todos los parámetros y devuelve uno solo, y lo componemos con la suma y con otra proyección para lograr lo que queremos.

No voy a ser estricto con esto en todo el apunte porque se vuelve engorroso muy rápido. Pero sepan que cuando usan una variable suelta en realidad están componiendo con una proyección, y por eso es p.r.

2.2. Potencia

Enunciado: Probar que la función potencia $Pot(x, y) = x^y$ es primitiva recursiva.

Resolución: Definamos la función usando recursión primitiva

$$Pr(x, 0) = s(n(x))$$
$$Pr(x, t + 1) = g(t, Pot(x, t), x)$$

donde

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

De nuevo, acá uso $s(n(x))$ en lugar de 1 para que entiendan lo que estoy haciendo realmente cuando después use simplemente "1″.

2.3. $f^n(x)$

Enunciado: Dada una función $f : \mathbb{N} \to \mathbb{N}$ p.r. fija, probar que la función

$$f^n(x) = \underbrace{f(f(f...(f}_{n \text{ veces}}(x))...))$$

es primitiva recursiva.

Resolución: ¿Estaría bien si simplemente usamos composición de f n veces? No, porque en ese caso estaríamos fijando un n. A lo sumo estaríamos diciendo que todas las funciones fijas f^n para algún n son p.r. Pero lo que queremos nosotros es una función (llamémosla $h : \mathbb{N}^2 \to \mathbb{N}$) que

toma como parámetro la cantidad de veces que debería ser aplicada f y se encarga de aplicarla esa cantidad de veces.

$$h(x, 0) = u_1^1(x)$$
$$h(x, t + 1) = g(t, h(x, t), x)$$

donde

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

Entonces en cada paso recursivo $t + 1$ aplicamos una vez más la f al valor ya calculado de $f(x, t)$, y el caso base es simplemente x, por lo que efectivamente obtuvimos la función que queríamos.

2.4. Par

Enunciado: Probar que el predicado $par(x)$ es p.r.

Nota: Llamamos predicado a cualquier función $p : \mathbb{N}^n \to \{0, 1\}$.

Resolución:

$$h(0) = 1$$
$$h(t + 1) = g(t, h(t))$$

donde

$$g(t, v) = uno(t, v) - u_2^2(t, v)$$

uno es una función definida *ad hoc* que toma dos parámetros y devuelve siempre 1. Lo mismo vale para 1, que es una función 0-aria equivalente a la constante 1.

Veamos que $par = h$ por inducción en t. Parece tonto, pero sirve para ver cómo se demuestran estas cosas para ejercicios más complejos.

Caso base:

$$h(0) = 1 = par(0)$$

Paso inductivo:

Hipótesis inductiva: $h(n) = par(n)$. Queremos ver que $h(n + 1) = par(n + 1)$.

$$h(n + 1) = g(n, h(n)) = uno(n, h(n)) - u_2^2(n, h(n)) = 1 - h(n).$$

Si n es par, por hipótesis inductiva $h(n) = 0$ y por lo tanto $h(n+1) = 1 = par(n+1)$.

Si n es impar, por hipótesis inductiva $h(n) = 1$ y por lo tanto $h(n+1) = 0 = par(n+1)$.

Probamos entonces que $h(x) = par(x)$ para todo x. Fácil, ¿no?

Algunos ejercicios un poco más difíciles piden probar no que una función es p.r., sino que un esquema de recursión es equivalente al esquema p.r. Por ejemplo:

2.5. Varios casos base

Enunciado: Dado k fijo, $g : \mathbb{N}^{n+2} \to \mathbb{N}$ y $g_0, ..., g_k : \mathbb{N}^n \to \mathbb{N}$ todas p.r., probar que el siguiente esquema de recursión es primitivo recursivo:

$$\begin{aligned} h(\bar{x}, 0) &= g_0(\bar{x}) \\ h(\bar{x}, 1) &= g_1(\bar{x}) \\ &\vdots \\ h(\bar{x}, k) &= g_k(\bar{x}) \\ h(\bar{x}, t+1) &= g(t, h(\bar{x}, t), \bar{x}) \quad \text{si } t+1 > k \end{aligned}$$

Resolución: Primero, como siempre, escribamos h respetando el esquema de recursión primitiva para después llenar los huecos. El caso para $t = 0$ ya lo tenemos.

$$\begin{aligned} h(\bar{x}, 0) &= g_0(\bar{x}) \\ h(\bar{x}, t+1) &= g'(t, h(\bar{x}, t), \bar{x}) \end{aligned}$$

Una posible idea es usar una función de selección, para pasarle los resultados posibles a los k casos base y al llamado recursivo y que ella se encargue de elegir cuál devuelve.

$$select_k(t, y_0, ..., y_k) = \begin{cases} y_0 & \text{si } t = 0 \\ y_1 & \text{si } t = 1 \\ & \vdots \\ y_{k-1} & \text{si } t = k-1 \\ y_k & \text{si no} \end{cases}$$

Sabemos que $select_k$ es p.r. porque está dividida en casos y:

- Los casos son finitos (k).
- Los casos son disjuntos.
- Las guardas son predicados p.r. (comparar por igualdad es p.r.)
- Los resultados son funciones p.r. (cada una es una proyección de un parámetro de entrada distinto).

Ahora ya podemos definir g' como

$$g'(t, v, \bar{x}) = select_k(t, g_1(\bar{x}), ..., g_k(\bar{x}), g(t, h(\bar{x}, t), \bar{x}))$$

Que es p.r. porque usamos solamente composición, proyecciones, y funciones que ya eran p.r. por hipótesis.

Varios se habrán dado cuenta de que para cualquier $t > k$ la función termina siempre deteniéndose en el caso base k y nunca mira los anteriores. Queda como ejercicio entonces mostrar que el esquema de recursión en el cual el "salto" se hace de a más de una unidad es p.r. Es decir:

$$h(\bar{x}, 0) = f(\bar{x})$$
$$h(\bar{x}, t) = g(t - n, h(\bar{x}, t - n), \bar{x})$$

para algún n.

2.6. Maximización acotada

Enunciado: Mostrar que, dado un predicado $P(t, x_1, ..., x_n)$ p.r., la siguiente función es p.r.:

$$max\_acot(b, \bar{x}) = \begin{cases} b + 1 & \text{si } \forall(y)_{y \leq b} \neg P(y, \bar{x}) \\ max_{t \leq b} P(t, \bar{x}) & \text{si no} \end{cases}$$

Resolución: Sabemos si un predicado Q es p.r. entonces la minimización acotada

$$min\_acot(b, \bar{x}) = \begin{cases} b + 1 & \text{si } \forall(y)_{y \leq b} \neg Q(y, \bar{x}) \\ min_{t \leq b} Q(t, \bar{x}) & \text{si no} \end{cases}$$

es p.r. Y esto, aunque suene irónico, puede ayudarnos a definir la maximización acotada. Miren:

$$max\_acot = min_{t \leq b} Q(t, \bar{x})$$

donde

$$Q(t, \bar{x}) = P(t, \bar{x}) \wedge \forall(t')_{t<t'\leq b} \neg P(t', \bar{x})$$

Es decir, nuestra $Q(t, \bar{x})$ dice exactamente "t cumple P y además no hay otro más grande que lo cumpla (hasta la cota)", es decir que es el máximo que cumple P. Y como la cuantificación acotada, los operadores lógicos, las comparaciones entre números y nuestra cota son p.r. tenemos Q p.r. y por lo tanto nuestra función max_acot es p.r.

En resumen, inventando un predicado auxiliar y usando minimización acotada conseguimos definir una nueva función p.r., sin tener que pelearnos con esquemas de recursión ni nada parecido. Bien usada, la minimización acotada es una herramienta muy poderosa. Siempre y cuando podamos conseguir un predicado p.r. que exprese lo que buscamos y una cota p.r. que nos asegure que lo encontremos, la minimización acotada se encarga de hacer el resto del trabajo por nosotros, "iterando" entre los números hasta encontrar el que queremos.