Lenguajes Formales, Autómatas y Computabilidad

Práctica 7: Funciones primitivas recursivas

Versión del 23 de octubre de 2024

Ejercicio 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).

Ejercicio 2. Probar que las siguientes funciones son primitivas recursivas, mostrando que pueden obtenerse a partir de las funciones iniciales usando composición y/o recursión primitiva:

a. $f(x,y) = x + y$ *b.* $f(x,y) = x \cdot y$ *c.* $f(x,y) = x^y$ *d.* $f(x,y) = \underbrace{x^{x^{x^{\cdot^{\cdot^{x}}}}}}_{y \text{ veces}}$

e. $f(x) = x \dot{-} 1$ *f.* $f(x,y) = x \dot{-} y$ *g.* $f(x,y) = \text{máx}(x,y)$ *h.* $f(x,y) = \text{mín}(x,y)$

Aclaraciones:

- En el inciso *d*, $f(x,0) = 1$.
- $x \dot{-} y = \begin{cases} x - y & \text{si } x \geq y, \\ 0 & \text{si } x < y. \end{cases}$

Ejercicio 3. Sea $f : \mathbb{N}^{n+1} \to \mathbb{N}$ una función primitiva recursiva. Demostrar que son primitivas recursivas las siguientes funciones:

a. $g(y, x_1, ..., x_n) = \sum_{t=0}^{y} f(t, x_1, ..., x_n)$ *b.* $h(y, x_1, ..., x_n) = \prod_{t=0}^{y} f(t, x_1, ..., x_n)$

Ejercicio 4. Llamamos *predicado* a cualquier función $p : \mathbb{N}^n \to \{0, 1\}$. Si p es un predicado y $p(a_1, ..., a_n) = 1$, escribimos simplemente $p(a_1, ..., a_n)$ y decimos informalmente que "$p(a_1, ..., a_n)$ es verdadero". Mostrar que los predicados binarios $\leq$, $\geq$, $=$, $\neq$, $<$ y $>$ son primitivos recursivos.

Ejercicio 5. Sean $p, q : \mathbb{N}^n \to \{0, 1\}$ predicados primitivos recursivos. Mostrar que son primitivos recursivos los siguientes predicados:

a. $\neg p$ *b.* $p \wedge q$ *c.* $p \vee q$

Ejercicio 6. Sean $f_1, ..., f_k, g : \mathbb{N}^n \to \mathbb{N}$ funciones primitivas recursivas y sean $p_1, ..., p_k : \mathbb{N}^n \to \{0, 1\}$ predicados primitivos recursivos disjuntos (es decir, tales que para todo $i \neq j$ y $(a_1, ..., a_n) \in \mathbb{N}^n$, no sucede que $p_i(a_1, ..., a_n) = p_j(a_1, ..., a_n) = 1$). Mostrar que también es primitiva recursiva cualquier función h que cumpla:

$$h(x_1, ..., x_n) = \begin{cases} f_1(x_1, ..., x_n) & \text{si } p_1(x_1, ..., x_n), \\ \vdots & \\ f_k(x_1, ..., x_n) & \text{si } p_k(x_1, ..., x_n), \\ g(x_1, ..., x_n) & \text{en otro caso.} \end{cases}$$

Observar que h queda completamente determinada por este esquema.

Ejercicio 7.

a. Demostrar que el predicado

$$\mathrm{par}(x) = \begin{cases} 1 \text{ si } x \text{ es par} \\ 0 \text{ si no} \end{cases}$$

es primitivo recursivo.

b. Demostrar que la función $f(x) = \lfloor \frac{x}{2} \rfloor$ es primitiva recursiva.

c. Sean $f : \mathbb{N}^n \to \mathbb{N}$, $g_1, g_2 : \mathbb{N}^{n+2} \to \mathbb{N}$ funciones primitivas recursivas. Mostrar que también es primitiva recursiva cualquier función h que cumpla:

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

Observar que h queda completamente determinada por este esquema.

Ejercicio 8. Sea $p : \mathbb{N}^{n+1} \to \{0, 1\}$ un predicado primitivo recursivo. Mostrar que también son primitivas recursivas las siguientes funciones:

a. $\mathrm{cantidad}_p(x_1, ..., x_n, y, z) = |\{t \mid y \leq t \leq z \wedge p(x_1, ..., x_n, t)\}|$

b. $\mathrm{todos}_p(x_1, ..., x_n, y, z) = \begin{cases} 1 & \text{si } \forall t \text{ tal que } y \leq t \leq z \wedge p(x_1, ..., x_n, t) \\ 0 & \text{si no} \end{cases}$

c. $\mathrm{alguno}_p(x_1, ..., x_n, y, z) = \begin{cases} 1 & \text{si } \exists t \text{ tal que } y \leq t \leq z \wedge p(x_1, ..., x_n, t) \\ 0 & \text{si no} \end{cases}$

d. $\mathrm{mínimo}_p(x_1, ..., x_n, y, z) = \begin{cases} \mathrm{mín}\{t \mid y \leq t \leq z \wedge p(x_1, ..., x_n, t)\} & \text{si existe tal } t \\ 0 & \text{si no} \end{cases}$

e. $\mathrm{máximo}_p(x_1, ..., x_n, y, z) = \begin{cases} \mathrm{máx}\{t \mid y \leq t \leq z \wedge p(x_1, ..., x_n, t)\} & \text{si existe tal } t \\ 0 & \text{si no} \end{cases}$

f. $\mathrm{único}_p(x_1, ..., x_n, y, z) = \begin{cases} u & \text{si } \{u\} = \{t \mid y \leq t \leq z \wedge p(x_1, ..., x_n, t)\} \\ z+1 & \text{si no} \end{cases}$

Ejercicio 9. Mostrar que las siguientes funciones son primitivas recursivas:

a. $\mathrm{cociente}(x, y) = \lfloor \frac{x}{y} \rfloor$

b. $\mathrm{resto}(x, y) = x \bmod y$

c. $\mathrm{divide}(x, y) = \begin{cases} 1 \text{ si } x \text{ es divisor de } y \\ 0 \text{ si no} \end{cases}$

d. $\mathrm{primo}(x) = \begin{cases} 1 \text{ si } x \text{ es un número primo} \\ 0 \text{ si no} \end{cases}$

e. $\mathrm{raíz}(x, y) = \begin{cases} \lfloor \sqrt[x]{y} \rfloor & \text{si } x \neq 0 \\ 0 & \text{si } x = 0 \end{cases}$

f. $\mathrm{nprimo}(x) = k$ donde k es primo y hay solo $x - 1$ primos positivos menores que k

Aclaración: Se asume que $\mathrm{cociente}(x, 0) = 0$ y $\mathrm{resto}(x, 0) = x$.

Ejercicio 10. Considerar la codificación de pares de naturales dada por $\langle x, y \rangle = 2^x(2y+1) \dot{-} 1$. Mostrar que las funciones observadoras $l, r : \mathbb{N} \to \mathbb{N}$ tales que $l(\langle x, y \rangle) = x$ y $r(\langle x, y \rangle) = y$ son primitivas recursivas.

Ejercicio 11. Mostrar que fib : $\mathbb{N} \to \mathbb{N}$, la función de Fibonacci, es primitiva recursiva, donde:

$$\begin{aligned} \mathrm{fib}(0) &= 0 \\ \mathrm{fib}(1) &= 1 \\ \mathrm{fib}(n+2) &= \mathrm{fib}(n+1) + \mathrm{fib}(n) \end{aligned}$$

Ejercicio 12. Considerar la codificación de secuencias finitas de números naturales dada por $[a_1, ..., a_n] = \prod_{i=1}^{n} \mathrm{nprimo}(i)^{a_i}$, donde nprimo es la función definida en el ejercicio 9.

a. Mostrar que la codificación dada forma una biyección entre el conjunto de secuencias finitas que no terminan en cero y los números naturales mayores que cero.

b. Determinar qué valor codifica la secuencia vacía y mostrar que las siguientes funciones son primitivas recursivas:

   i. *Longitud*: $|\bullet| : \mathbb{N} \to \mathbb{N}$ tal que $|[a_1, ..., a_n]| = n$.

   ii. *Observador*: $\bullet\,[i] : \mathbb{N} \to \mathbb{N}$ tal que $[a_1, ..., a_n][i] = \begin{cases} a_i & \text{si } 1 \leq i \leq n \\ 0 & \text{si no.} \end{cases}$

   iii. *Creación*: $[\bullet] : \mathbb{N} \to \mathbb{N}$ tal que $[x]$ es la secuencia cuyo único elemento es x.

   iv. *Concatenación*: $\circ : \mathbb{N}^2 \to \mathbb{N}$ tal que $[a_1, ..., a_n] \circ [b_1, ..., b_m] = [a_1, ..., a_n, b_1, ..., b_m]$.

   v. *Subsecuencia*: sub : $\mathbb{N}^3 \to \mathbb{N}$ tal que $\mathrm{sub}([a_1, ..., a_n], i, j) = \left[a_i, ..., a_j\right]$.

c. Proponer una codificación de secuencias $\rho : \mathbb{N}^* \to \mathbb{N}$ que forme una biyección entre los números naturales (incluyendo el cero) y el conjunto de *todas* las secuencias finitas de naturales, de forma tal que todas las funciones del ítem anterior sean primitivas recursivas.

Ejercicio 13. Sea Σ un alfabeto cualquiera, y sea $\#_\Sigma : \Sigma \to \{1, 2, ..., |\Sigma|\}$ una enumeración de los símbolos de Σ (es decir, una función biyectiva que asigna a cada símbolo un número natural entre 1 y $|\Sigma|$).

a. Proponer una codificación $\rho_\Sigma : \Sigma^* \to \mathbb{N}$ que forme una biyección entre las cadenas definidas sobre el alfabeto Σ y los números naturales, de forma que tal que las siguientes funciones sean primitivas recursivas:

   i. *Cabeza*: $\mathrm{cab}_\Sigma : \mathbb{N} \to \mathbb{N}$ tal que $\mathrm{cab}_\Sigma(\rho_\Sigma(x.\omega)) = \#_\Sigma(x)$ y $\mathrm{cab}_\Sigma(\rho_\Sigma(\lambda)) = 0$.

   ii. *Cola*: $\mathrm{cola}_\Sigma : \mathbb{N} \to \mathbb{N}$ tal que $\mathrm{cola}_\Sigma(\rho_\Sigma(x.\omega)) = \rho_\Sigma(\omega)$ y $\mathrm{cola}_\Sigma(\rho_\Sigma(\lambda)) = 0$.

   iii. *Longitud*: $|\bullet|_\Sigma : \mathbb{N} \to \mathbb{N}$ tal que $|\rho_\Sigma(\omega)|_\Sigma = |\omega|$.

   iv. *Concatenación*: $\mathrm{concat}_\Sigma : \mathbb{N}^2 \to \mathbb{N}$ tal que $\mathrm{concat}_\Sigma(\rho_\Sigma(\omega_1), \rho_\Sigma(\omega_2)) = \rho_\Sigma(\omega_1.\omega_2)$.

   v. *Reversa*: $\mathrm{rev}_\Sigma : \mathbb{N} \to \mathbb{N}$ tal que $\mathrm{rev}_\Sigma(\rho_\Sigma(\omega)) = \rho_\Sigma(\omega^{\mathrm{r}})$.

b. Un lenguaje $\mathcal{L} \subseteq \Sigma^*$ se dice primitivo recursivo si el predicado $p_\mathcal{L} : \Sigma^* \to \{0, 1\}$ tal que:

$$p_\mathcal{L}(x) = \begin{cases} 1 & \text{si la cadena codificada por } x \text{ pertenece a } \mathcal{L} \\ 0 & \text{si no} \end{cases}$$

es primitivo recursivo.

Mostrar que $\mathcal{L} = \{\omega \in \{a, b\}^* \mid |\omega|_a \text{ es par}\}$ es un lenguaje primitivo recursivo.

Ejercicio 14.

a. Sea $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ un autómata finito determinístico. Mostrar que, si se define una enumeración $\#_Q : Q \to \{1, 2, ..., |Q|\}$ de los estados de M, entonces la función $\tilde{\delta} : \mathbb{N}^2 \to \mathbb{N}$ tal que:

$$\tilde{\delta}(x, y) = \begin{cases} \delta(q, s) & \text{si existen } q \in Q, s \in \Sigma \text{ tales que } x = \#_Q(q) \wedge y = \#_\Sigma(s) \\ 0 & \text{si no} \end{cases}$$

es primitiva recursiva.

b. Demostrar que todo lenguaje regular es primitivo recursivo.

Ejercicio 15. Sea $G = \langle V_N, V_T, P, S\rangle$ una gramática independiente del contexto, y sea $V = V_N \cup V_T$.

a. Mostrar que es primitiva recursiva la función $\text{step}_G : \mathbb{N} \to \mathbb{N}$ tal que, si $\omega \in V^*$,

$$\text{step}_G(\rho_V(\omega)) = [\rho_V(\alpha_1), ..., \rho_V(\alpha_n)],$$

donde $\alpha_1, ..., \alpha_n$ son todas las cadenas de V^* tales que $\omega \underset{L}{\Rightarrow} \alpha_i$, es decir, que pueden obtenerse a partir de ω reescribiendo el primer no terminal desde la izquierda (si ω no tiene ningún no terminal, entonces $\text{step}_G(\rho_V(\omega)) = [\,]$).

b. Mostrar que es primitiva recursiva la función $\text{sent}_G : \mathbb{N} \to \mathbb{N}$ tal que, si $\omega \in V^*$,

$$\text{sent}_G(t) = [\rho_V(\alpha_1), ..., \rho_V(\alpha_n)],$$

donde $\alpha_1, ..., \alpha_n$ son todas las cadenas de V^* tales que $S \overset{t}{\underset{L}{\Rightarrow}} \alpha_i$, es decir, que pueden obtenerse a partir de S tras t pasos de una derivación más a la izquierda.

c. Demostrar que si G no es recursiva a izquierda, entonces $\mathcal{L}(G)$ es un lenguaje primitivo recursivo.

Sugerencia: Tener en cuenta que si G no es recursiva a izquierda y una cadena $\omega \in V^*$ se obtiene desde S tras $k|V_N|$ pasos de derivación (para $k = 1, 2, ...$), necesariamente sus primeros k símbolos son terminales.