Lógica y Computabilidad

Primer Cuatrimestre – 2024

Práctica 9 – Funciones Computables

Una función parcial $f : \mathbb{N}^k \to \mathbb{N}$ se dice $\mathcal{S}$-*computable*, *Turing computable* o *computable* a secas si existe un programa $\mathcal{P}$ en $\mathcal{S}$ que *computa* f, esto quiere decir que $\mathcal{P}$ devuelve $f(x_1, \ldots, x_k)$ con la entrada $x_1, \ldots, x_k$ cuando $(x_1, \ldots, x_k) \in \mathrm{Dom}(f)$ y se indefine en caso contrario. El conjunto de todas las funciones parciales computables se nota COMP.

1. Probar que las siguientes funciones están en COMP.

   a) La función $suc : \mathbb{N} \to \mathbb{N}$ definida por $suc\,(x) = x + 1$.

   b) Las proyecciones $P_i^n : \mathbb{N}^n \to \mathbb{N}$, $P_i^n(x_1, \ldots, x_n) = x_i$.

   c) Las constantes $C_k^n : \mathbb{N}^n \to \mathbb{N}$, $C_k^n(x_1, \ldots, x_n) = k$.

   d) La función $h : \mathbb{N}^r \to \mathbb{N}$ construida a partir de $f : \mathbb{N}^k \to \mathbb{N}$ y $g_i : \mathbb{N}^r \to \mathbb{N}, 1 \le i \le k$, que están en COMP, de la siguiente forma:

   $$h(x_1, \ldots, x_r) = f(g_1(x_1, \ldots, x_r), \ldots, g_k(x_1, \ldots, x_r)).$$

   e) La función $h : \mathbb{N}^{k+1} \to \mathbb{N}$ construida a partir de $g : \mathbb{N}^k + 2 \to \mathbb{N}$ y $f : \mathbb{N}^k \to \mathbb{N}$, que están en COMP, definida de la siguiente forma:

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

   Concluir que COMP es una clase PRC y contiene a las funciones recursivas primitivas.

2. Dada una función parcial computable $f : \mathbb{N} \to \mathbb{N}$, considerar las funciones parciales

   $$g\,(x) = \begin{cases} 1 & \text{si } f\,(x) = 1, \\ \uparrow & \text{si } f\,(x) \ne 1, \end{cases} \quad \text{y} \quad h(x) = \begin{cases} 1 & \text{si } f\,(x)\downarrow, \\ \uparrow & \text{si } f\,(x)\uparrow. \end{cases}$$

   Decidir si es posible escribir programas en $\mathcal{S}$ que computen g y h.

3. Recordar que el lenguaje $\mathcal{S}$ contiene sólo las instrucciones

   $$V \leftarrow V + 1 \quad , \quad V \leftarrow V - 1 \quad \text{e} \quad \text{IF } V \ne 0 \text{ GOTO } L.$$

   Se definen las siguientes variantes de $\mathcal{S}$:

   $\mathcal{S}_1$ igual a $\mathcal{S}$ pero sin la instrucción $V \leftarrow V + 1$,

   $\mathcal{S}_2$ igual a $\mathcal{S}$ pero sin la instrucción IF $V \ne 0$ GOTO L,

   $\mathcal{S}_3$ igual a $\mathcal{S}$ pero sin la instrucción $V \leftarrow V - 1$,

   $\mathcal{S}_4$ reemplazando las tres instrucciones de $\mathcal{S}$ por

   $$V \leftarrow W \quad , \quad V \leftarrow V + 1 \quad \text{e} \quad \text{IF } V \ne W \text{ GOTO } L.$$

   En cada caso, decidir si $\mathcal{S}$-computable implica $\mathcal{S}_i$-computable y viceversa.

4. Si P es un predicado computable, probar que IF $P(V)$ GOTO L puede simularse en $\mathcal{S}$.

5. Probar que $\neg P$, $P \vee Q$, $P \wedge Q$ y $P \to Q$ son predicados computables si P y Q lo son.

6. Sean $g_1, \ldots, g_k$ funciones computables y sean $P_1, \ldots P_k$ predicados computables totales *mutuamente excluyentes y exhaustivos*, es decir, tales que para cada $x \in \mathbb{N}$ se tiene $P_i(x) = 1$ para un único $1 \leq i \leq k$. Probar que la función $f : \mathbb{N} \to \mathbb{N}$ dada por

$$f(x) = g_i(x) \ \text{ si } P_i(x) = 1$$

está bien definida y es computable. ¿Qué sucede si los predicados no son mutuamente excluyentes y exhaustivos? ¿Y si no son totales?

7. Probar que toda f parcial computable es computada por infinitos programas en $\mathcal{S}$.

8. Probar que f^{-1} es total computable cuando $f : \mathbb{N} \to \mathbb{N}$ es una biyección computable.

9. Un número es *computable* si existe algún programa que pueda aproximarlo con un error arbitrariamente pequeño. Precisamente, $x \in \mathbb{R}$ es *computable* si existe una función total computable $f : \mathbb{N} \to \mathbb{N}$ que para todo $n \in \mathbb{N}$ verifica $f(n) = [s, p, q]$ con $q \neq 0$ y

$$\left| x - (-1)^s \left(\frac{p}{q} \right) \right| \leq \frac{1}{n+1}.$$

La definición expresa que la función computable f calcula, para cada $n \in \mathbb{N}$, un número racional $(-1)^s(p/q)$ que aproxima x con error a lo sumo $(n+1)^{-1}$. Notar que dicha cota de error puede hacerse arbitrariamente chica tomando n suficientemente grande.

   a) Probar que todo número racional es computable.
   b) Probar que el número real $\sqrt{2} + \sqrt{3}$ es computable.
   c) Probar que $x + y$ y $x \cdot y$ son computables si x e y lo son.

10. Calcular los códigos de los siguientes programas y escribir las funciones que computan.

```
      IF X ≠ 0  GOTO B
[A]   X ← X + 1
      IF X ≠ 0  GOTO A
[B]   Y ← Y + 1
```

```
[B]   IF X ≠ 0  GOTO B
[A]   X ← X − 1
      IF X ≠ 0  GOTO A
      Z ← Z + 1
```

11. Escribir el programa $\mathcal{P}$ de código 324 y dar una expresión de la función que computa.

12. Probar que toda función computable es computada por algún programa de código par.

13. Sea $f : \mathbb{N} \to \mathbb{N}$ la función que asigna a cada número natural n el número de instrucciones del programa que tiene código n. Probar que f es recursiva primitiva.

14. Utilizando las funciones recursivas primitivas $\mathrm{STP}^{(n)}$ y $\mathrm{SNAP}^{(n)} : \mathbb{N}^{n+2} \to \mathbb{N}$, mostrar que las siguientes funciones parciales son computables:

$$f_1(x,y) = \begin{cases} 1 & \text{si } y \in \operatorname{Dom}\left(\Phi_x^{(1)}\right), \\ \uparrow & \text{en caso contrario.} \end{cases} \qquad f_2(x) = \begin{cases} 1 & \text{si } \operatorname{Dom}\left(\Phi_x^{(1)}\right) \neq \emptyset, \\ \uparrow & \text{en caso contrario.} \end{cases}$$

$$f_3(x,y) = \begin{cases} 1 & \text{si } y \in \operatorname{Im}\left(\Phi_x^{(1)}\right), \\ \uparrow & \text{en caso contrario.} \end{cases} \qquad f_4(x,y) = \begin{cases} 1 & \text{si } \operatorname{Dom}\left(\Phi_x^{(1)}\right) \cap \operatorname{Im}\left(\Phi_y^{(1)}\right) \neq \emptyset, \\ \uparrow & \text{en caso contrario.} \end{cases}$$