

PRÁCTICA 2 - FUNCIONES  $\mathcal{S}$ -COMPUTABLES -**Ejercicio 1.**

- a. Definir *macros* para las siguientes pseudo-instrucciones (con su interpretación natural) e indicar en cada caso qué variables y qué etiquetas se asumen “frescas”:
- $V_i \leftarrow k$
  - $V_i \leftarrow V_j + k$
  - IF  $V_i = 0$  GOTO  $L$
  - GOTO  $L$
- b. Definir dos pseudo-programas distintos en el lenguaje  $\mathcal{S}$  (usando las macros convenientes del punto anterior) que computen la función de dos variables  $f(x_1, x_2) = x_1 + x_2$ . Para alguno de los dos, expandir las macros utilizadas prestando atención a la instanciación de variables y etiquetas frescas.
- c. Sea  $P$  el programa en  $\mathcal{S}$  que resulta de expandir todas las macros en alguno de los códigos del punto anterior. Determinar cuál es la función computada en cada caso:
- $\Psi_P^{(1)} : \mathbb{N} \rightarrow \mathbb{N}$
  - $\Psi_P^{(2)} : \mathbb{N}^2 \rightarrow \mathbb{N}$
  - $\Psi_P^{(3)} : \mathbb{N}^3 \rightarrow \mathbb{N}$

**Ejercicio 2.**

- a. Sea  $\mathcal{C}_{\mathcal{S}} = \{\Psi_P^{(n)} \mid P \text{ es un programa en } \mathcal{S}, n \geq 1\}$  la clase de funciones  $\mathcal{S}$ -parciales computables. Mostrar que  $\mathcal{C}_{\mathcal{S}}$  es una clase *PRC*.
- b. Demostrar (sin definir un programa en  $\mathcal{S}$ ) que la función  $*$  :  $\mathbb{N}^2 \rightarrow \mathbb{N}$  definida por  $*(x, y) = x \cdot y$  es  $\mathcal{S}$ -computable.
- c. Si  $f : \mathbb{N}^n \rightarrow \mathbb{N}$  es una función primitiva recursiva. ¿Qué podemos decir acerca de la existencia de un programa en el lenguaje  $\mathcal{S}$  que la compute?

**Ejercicio 3.** Decimos que un programa  $P$  es *autocontenido* si en cada instrucción *IF*  $V \neq 0$  *GOTO*  $L$  que ocurre en  $P$ ,  $L$  es una etiqueta definida en  $P$ .

- a. Demostrar que todo programa  $P$  tiene un programa autocontenido  $P'$  equivalente ( $P$  y  $P'$  son programas equivalentes si  $\Psi_P^{(n)} = \Psi_{P'}^{(n)} \forall n \geq 1$ ).
- b. Sean  $P$  y  $Q$  dos programas autocontenidos con etiquetas disjuntas y sea  $r : \mathbb{N}^n \rightarrow \{0, 1\}$  un predicado primitivo recursivo. Definir macros para las siguientes pseudo-instrucciones (con su interpretación natural):
- IF  $r(V_1, \dots, V_n)$  GOTO  $L$
  - IF  $r(V_1, \dots, V_n)$  THEN  $P$  ELSE  $Q$
  - WHILE  $r(V_1, \dots, V_n)$   $P$

c. Dadas las funciones  $f, g : \mathbb{N} \rightarrow \mathbb{N}$  definidas por

$$f(x) = \begin{cases} 1 & \text{si } x = 3 \\ \uparrow & \text{en otro caso} \end{cases} \quad \text{y} \quad g(x) = 2x$$

Demostrar que es  $\mathcal{S}$ -parcial computable la función

$$h(x) = \begin{cases} f(x) & \text{si } x \geq 5 \vee x = 3 \\ g(x) & \text{en otro caso} \end{cases}$$

**Ejercicio 4.**

a. Se definen las siguientes variantes del lenguaje  $\mathcal{S}$ :

- $\mathcal{S}_1$ : Igual que  $\mathcal{S}$  pero sin la instrucción  $V \leftarrow V + 1$
- $\mathcal{S}_2$ : Igual que  $\mathcal{S}$  pero sin la instrucción IF  $V \neq 0$  GOTO  $L$
- $\mathcal{S}_3$ : Igual que  $\mathcal{S}$  pero sin la instrucción  $V \leftarrow V \div 1$

Demostrar que para cada uno de estos lenguajes existe al menos una función  $\mathcal{S}$ -parcial computable que no es computable en este nuevo lenguaje.

b. Sea  $\mathcal{S}'$  el lenguaje de programación definido como  $\mathcal{S}$  salvo que sus instrucciones (etiquetadas o no) son de los siguientes tres tipos (con su interpretación natural):

$$\begin{aligned} V &\leftarrow V' \\ V &\leftarrow V + 1 \\ \text{IF } V \neq V' &\text{ GOTO } L \end{aligned}$$

Demostrar que una función es parcial computable en  $\mathcal{S}'$  si solamente si lo es en  $\mathcal{S}$ .

**Ejercicio 5.**

a. Demostrar que si  $p : \mathbb{N}^{n+1} \rightarrow \{0, 1\}$  es un predicado  $\mathcal{S}$ -computable (total), entonces es  $\mathcal{S}$ -parcial computable:

$$\text{minimoNA}_p(x_1, \dots, x_n, y) = \begin{cases} \min\{t \mid y \leq t \wedge p(x_1, \dots, x_n, t)\} & \text{si existe algún tal } t \\ \uparrow & \text{en otro caso} \end{cases}$$

b. Mostrar, usando el resultado anterior, que si  $f : \mathbb{N} \rightarrow \mathbb{N}$  es biyectiva y  $\mathcal{S}$ -computable (total), entonces también lo es su inversa,  $f^{-1}$ .

**Ejercicio 6.** Un programa  $P$  en el lenguaje  $\mathcal{S}$  con instrucciones  $I_1, I_2, \dots, I_n$  se dice *optimista* si  $\forall i = 1, \dots, n$ , si  $I_i$  es la instrucción IF  $V \neq 0$  GOTO  $L$  entonces  $L$  no aparece como etiqueta de ninguna instrucción  $I_j$  con  $j \leq i$ .

Demostrar que el siguiente predicado es primitivo recursivo:

$$r(x) = \begin{cases} 1 & \text{si el programa cuyo número es } x \text{ es optimista} \\ 0 & \text{caso contrario} \end{cases}$$

**Ejercicio 7.** Utilizando las funciones primitivas-recursivas  $\text{STP}^{(n)}$  y  $\text{SNAP}^{(n)} : \mathbb{N}^{n+2} \rightarrow \mathbb{N}$  vistas en clase, mostrar que las siguientes son funciones  $\mathcal{S}$ -parciales computables:

$$\begin{aligned} f_1(x, y) &= \begin{cases} 1 & \text{si } y \in \text{Dom } \Phi_x^{(1)} \\ \uparrow & \text{si no} \end{cases} & f_2(x) &= \begin{cases} 1 & \text{si } \text{Dom } \Phi_x^{(1)} \neq \emptyset \\ \uparrow & \text{si no} \end{cases} \\ f_3(x, y) &= \begin{cases} 1 & \text{si } y \in \text{Im } \Phi_x^{(1)} \\ \uparrow & \text{si no} \end{cases} & f_4(x, y) &= \begin{cases} 1 & \text{si } \text{Dom } \Phi_x^{(1)} \cap \text{Im } \Phi_y^{(1)} \neq \emptyset \\ \uparrow & \text{si no} \end{cases} \end{aligned}$$

**Ejercicio 8.** Sea  $f : \mathbb{N} \rightarrow \mathbb{N}$  una función  $\mathcal{S}$ -parcial computable en tiempo polinomial (i.e., existe un programa  $P$  tal que  $\Psi_P^{(1)}(x) = f(x)$  y tal que, para algún polinomio  $Q(x)$ ,  $P$  no requiere más que  $Q(\lceil \log_2 x \rceil)$  pasos para terminar).

- a. Mostrar que  $f$  es primitiva recursiva.
- b. ¿Sucedre lo mismo si la cota es exponencial, doblemente exponencial, etc.?
- c. ¿Qué podemos decir, en general, sobre la complejidad temporal de una función computable que no sea primitiva recursiva?

**Ejercicio 9.** \* Se dice que un programa  $P$  en el lenguaje  $\mathcal{S}$  *se pisa con  $n$  entradas* si para alguna entrada  $x_1, x_2, \dots, x_n$  y algún tiempo  $t$ , la variable de salida  $Y$  luego de  $t$  pasos de la ejecución de  $P$  con entradas  $x_1, x_2, \dots, x_n$  vale  $\#P$ .

Demostrar que para cualquier  $n \in \mathbb{N}$  es  $\mathcal{S}$ -parcial computable la función:

$$f_n(x) = \begin{cases} 1 & \text{si el programa cuyo número es } x \text{ se pisa con } n \text{ entradas} \\ \uparrow & \text{caso contrario} \end{cases}$$

\*Este ejercicio puede ser entregado, de manera opcional, como se resolvería en un examen, a modo de práctica para el parcial.