Ciencias de la Computación (2021-22)

Algoritmos y funciones computables

Ejercicio 1.– Diseñar programas GOTO que, sin usar macros (salvo quizás GOTO L), calculen las funciones:

$$f_1(x) = 3x \quad f_2(x) = \begin{cases} 1 & \text{si } x \text{ es par} \\ 0 & \text{si } x \text{ es impar} \end{cases} \qquad f_3(x) = \begin{cases} 1 & \text{si } x \text{ es par} \\ \uparrow & \text{si } x \text{ es impar} \end{cases}$$

Ejercicio 2.– Probar que las siguientes funciones son GOTO– computables:

$$g_1(x_1, x_2) = \begin{cases} 1 & \text{si } x_1 = x_2 \\ 0 & \text{si } x_1 \neq x_2 \end{cases} \quad g_2(x_1, x_2) = \begin{cases} 1 & \text{si } x_1 \leq x_2 \\ 0 & \text{si } x_1 > x_2 \end{cases} \quad g_3(x_1, x_2) = \text{máx}\{x_1, x_2\}.$$

Ejercicio 3.– Para cada uno de los siguientes conjuntos, diseñar un programa GOTO sin macros que pruebe que es decidible (es decir, que su función característica es GOTO-computable):

$$A = \{(x_1, x_2) \ : \ x_1 + x_2 \text{ es par}\} \qquad B = \{(x_1, x_2, x_3) \ : \ x_3 \in [x_1, x_2]\}$$

Ejercicio 4.– Diseñar un programa GOTO que calcule la función $f(x) = \text{máx}\{n \ : \ n^2 \leq x\}$.

Ejercicio 5.– Consideremos un predicado computable $\theta(x)$ tal que $\theta(0)$. Diseñar un programa GOTO sin macros (salvo quizás θ, la suma, y GOTO L) que calcule la función

$$f(x) = |\{n \ : \ \theta(n) \wedge n \leq x\}|$$

Ejercicio 6.– Sea $P(x)$ un predicado GOTO–computable. Probar que las siguientes funciones son GOTO–computables:

$$E_1(x) = \begin{cases} 1 & \text{si } P(x) \\ \uparrow & \textit{e.o.c.} \end{cases} \qquad E_2(r) = \begin{cases} 1 & \text{si existen al menos } r \text{ números } n \text{ tales que } P(n) \\ \uparrow & \textit{e.o.c.} \end{cases}$$

Ejercicio 7.– Sean f y g computables (de aridad 1). Probar que la función $h(x) = f(g(x))$ es computable.

Ejercicio 8.– Sea $g : \mathbb{N} \to \mathbb{N}$ biyectiva y GOTO–computable. Demostrar que g^{-1} es GOTO–computable.

Ejercicio 9.– Sea $f : \mathbb{N}- \to \mathbb{N}$ definida por $f(0) = 0$, $f(1) = 1$ y $f(n) = f(n-1) + f(n-2)$ si $n \geq 2$. Demostrar que f es computable, diseñando un programa GOTO que la calcule.

Ejercicio 10.– Diseñar un programa tal que toda computación $s_1, \ldots, s_k$ verifique que $k = 5$

Ejercicio 11.– Diseñar un programa tal que para todo $n > 0$ y toda computación $s_1 = (1, \sigma), s_2, \ldots, s_k$ verificando que $\sigma(X) = n$, se tenga $k = 2n + 1$. ¿Podemos conseguir que $k = 2n + 1$ también se cumpla para $n = 0$?

Ejercicio 12.– Demostrar que para toda función computable existen infinitos programas que calculan dicha función.

Ejercicio 13.– En este ejercicio analizaremos los programas lineales. Los programas lineales son los diseñados utilizando el lenguaje $\texttt{GOTO}_l$, obtenido a partir de `GOTO` eliminando la instrucción condicional.

1. Sea $k \in \mathbb{N}$. Demostrar que la función constante $C_k(x) = k$ es $\texttt{GOTO}_l$–computable.
2. Probar que si `p` es un programa que calcula C_k, entonces la longitud de `p` es mayor o igual que k.
3. Probar que la función $f(x) = x + 1$ no es $\texttt{GOTO}_l$–computable (luego `GOTO` y $\texttt{GOTO}_l$ no son equivalentes).

Ejercicio 14.– Sea $\texttt{GOTO}_f$ el lenguaje que se obtiene a partir de `GOTO` restringiendo el uso de la instrucción

$$\texttt{I} \equiv \texttt{IF V} \neq 0 \texttt{ GOTO L}$$

a etiquetas `L` para instrucciones que aparecen en el programa después de `I` (o son de salida).

1. Probar que si $f : \mathbb{N}- \to \mathbb{N}$ es $\texttt{GOTO}_f$–computable, entonces existe una partición finita $\{A_1, \ldots, A_l\}$ de $\mathbb{N}$ tal que en cada A_i, la función es constante.
2. Probar que $\texttt{GOTO}_f$ y $\texttt{GOTO}_l$ no son equivalentes.
3. Diremos que una función $f : \mathbb{N} \to \mathbb{N}$ es *eventualmente constante* si existe $k \in \mathbb{N}$ tal que

   $$\text{Para todo } i,\, j \geq k,\ f(i) = f(j)$$

   Pruébese que $f : \mathbb{N} \to \mathbb{N}$ es $\texttt{GOTO}_f$–computable si y sólo si f es eventualmente constante.

Ejercicio 15.– Una función $f : \mathbb{N}- \to \mathbb{N}$ se dice n–computable si existe un `GOTO`–programa `p` tal que:

- $[\![\texttt{p}]\!] = f$ y la longitud de `p` es $\leq n$
- En `p` sólo ocurren variables del conjunto $\{\texttt{X}, \texttt{Y}, \texttt{Z}_1, \ldots, \texttt{Z}_n\}$ y etiquetas del conjunto $\{A_1, \ldots, A_n, E\}$.

Probar que

1. Si f es computable por un programa de longitud menor o igual que n, entonces f es n–computable.
2. Dado n, el número de funciones n–computables es finito.
3. Dado n, el número de funciones computables por programas de longitud menor o igual que n es finito.
4. Para cada $n \geq 0$, existe una función computable que no lo es por un programa de longitud $\leq n$ (luego no se puede limitar ni la longitud de los programas ni el número de variables usable en `GOTO`).

Ejercicio 16.– Sea $\texttt{GOTO}_A$ el lenguaje definido como GOTO pero cuyas instrucciones (etiquetadas o no) son:

$$\texttt{V} \leftarrow \texttt{V}'$$
$$\texttt{V} \leftarrow \texttt{V} + 1$$
$$\texttt{IF V} \neq \texttt{V}' \texttt{ GOTO L}$$

que tienen la interpretación obvia. Probar que una función f es $\texttt{GOTO}_A$–computable si y sólo si f es GOTO–computable.

Ejercicio 17.– El modelo $\texttt{GOTO}_{\texttt{SU}}$ es el modelo de computación con semántica similar a GOTO salvo que en los programas cada variable aparece, a lo sumo, en una instrucción

1. Probar que si f es $\texttt{GOTO}_{\texttt{SU}}$-computable entonces $f(0)\downarrow$. Además, $f(0) = 0$, o bien, $f(0) = 1$.
2. ¿Es GOTO L una macro en $\texttt{GOTO}_{\texttt{SU}}$? En caso afirmativo, diséñese una macro-expansión
3. Justificar por qué la función signo

$$sg(x) = \begin{cases} 1 & \text{si } x \neq 0 \\ 0 & \text{si } x = 0 \end{cases}$$

   no es $\texttt{GOTO}_{SU}$-computable

Ejercicio 18.– Sea $\texttt{DO}_1^2$ el modelo de computación definido de como GOTO, pero utilizando como básicas las siguientes instrucciones:

(a) Incremento, decremento, SKIP, y $\texttt{W} \leftarrow \texttt{2} \cdot \texttt{W}$

(b) DO V TIMES I, donde I es una instrucción del tipo (a).

El efecto de la instrucción del tipo (b) es el siguiente: si en un estado σ se verifica $\sigma(V) = n$, entonces se ejecuta n veces la instrucción I partiendo de σ. Se pide:

1. Probar que $f(x) = 2^x$ es $\texttt{DO}_1^2$-computable por un programa de longitud 2 sin macros
2. Sea f la función definida por

$$\begin{cases} f(x,0) = x+1 \\ f(x,y+1) = 2^{f(x,y)} \cdot f(x,y) \end{cases}$$

   Probar que si p es un programa $\texttt{DO}_1^2$, entonces $[\![\texttt{p}]\!](x) \leq f(x,|p|)$ para todo x. (Indicación: probar por inducción en $|p|$ que el valor final de cualquier variable está acotado por $f(x,|p|)$).
3. Probar que la función $g(x) = 2^{f(x,x)}$ no es $\texttt{DO}_1^2$ computable.

Ejercicio 19.– Una *codificación* es una aplicación inyectiva $\#(.) : \texttt{GOTO}_{\texttt{P}} \to \mathbb{N}$. Una envolvente asociada a $\#(.)$ es una función $f : \mathbb{N}^2 \to N$ tal que para todo programa $p \in \texttt{GOTO}_{\texttt{P}}$ tal que $[\![\texttt{p}]\!]$ es total se satisface que

$$[\![\texttt{p}]\!](x) \leq f(\#(p), x)$$

Probar que la función $g(x) = f(x,x) + 1$ no es G-computable