Diagonalización y reducciones

Lógica y Computabilidad - Curso de verano 2018

Franco Frizzo

8 de febrero de 2018

Ejercicio 1: Diagonalización

Ejercicio 1.1

Demostrar que la siguiente función no es computable.

$$f_1(x) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x) \downarrow \\ 0 & \text{en otro caso.} \end{cases}$$

Resolución

Haremos una demostración por el absurdo, empleando una técnica conocida como *diagonalización*. Comenzamos suponiendo que f_1 es computable. Es importante notar que f_1 es una función que interpreta su entrada como el número de un programa, y su valor depende de una propiedad de la función computada por este programa.

A partir de este hecho, trataremos de definir un programa P_1 "rebelde". La idea es que P_1 reciba como argumento un número, que interpretará como el número de un programa, y le "pregunte" a f_1 acerca del comportamiento de este programa. Luego, P_1 exhibirá el comportamiento exactamente opuesto. Esto nos permitirá llegar a un absurdo pasándole a P_1 su propio número como argumento.

Sea entonces P_1 el (pseudo-)programa:

$$\begin{array}{rl} P_1\colon & Z_1 \leftarrow f_1(X_1) \\ \text{[A]} & \text{IF } Z_1 \neq 0 \text{ GOTO A} \end{array}$$

que está bien definido porque f_1 es computable, y sea $e_1 = \#(P_1) \in \mathbb{N}$.

Tenemos que

$$\Psi_{P_1}^{(1)}(x) = \Phi_{e_1}^{(1)}(x) = \begin{cases} 0 & \text{si } f_1(x) = 0 \\ \uparrow & \text{en otro caso.} \end{cases}$$

Podemos observar que el programa e_1 tiene dos comportamientos posibles: colgarse o devolver 0. Veamos que para cualquier entrada x, e_1 se comporta de una de estas dos formas si y solo si x exhibe el comportamiento opuesto:

$$\Phi_{e_1}^{(1)}(x) = 0 \underset{\text{(def. } e_1\text{)}}{\iff} f_1(x) = 0 \underset{\text{(def. } f_1\text{)}}{\iff} \Phi_x^{(1)}(x) \uparrow .$$

El problema surge porque el número e_1 es una entrada posible para el programa con número e_1.[1] Al instanciar $x = e_1$, obtenemos:

$$\Phi^{(1)}_{e_1}(e_1) = 0 \quad \iff \quad \Phi^{(1)}_{e_1}(e_1) \uparrow .$$

Esto es absurdo, porque solo hay dos casos posibles: o bien $\Phi^{(1)}_{e_1}(e_1) = 0$ o bien $\Phi^{(1)}_{e_1}(e_1) \uparrow$, y en cualquiera de ellos se llega a una contradicción.[2] El absurdo debe provenir de la única suposición que hicimos: que f_1 es computable. Por lo tanto, podemos concluir que f_1 no es computable.

Ejercicio 1.2

Demostrar que la siguiente función no es computable.

$$f_2(x, y) = \begin{cases} 0 & \text{si } \Phi^{(1)}_x(y) \text{ es par} \\ 1 & \text{en otro caso.} \end{cases}$$

Resolución

Supongamos que f_2 es computable. Queremos definir un programa "rebelde" P_2, que reciba como entrada el número de un programa, le "pregunte" a f_2 por el comportamiento del mismo, y luego exhiba un comportamiento opuesto. Como f_2 habla de la paridad de los resultados, una posibilidad es definir P_2 para que se comporte de esta manera:[3]

$$\Psi^{(1)}_{P_2}(x) = \begin{cases} \text{alguna constante par} & \text{si } f_2(x, x) = 1 \\ \text{alguna constante impar} & \text{en otro caso.} \end{cases}$$

Es sencillo ver que, como f_2 es computable, existe un programa P_2 que computa la función anterior, independientemente de cómo elijamos las dos constantes (se lo puede escribir a modo de ejercicio). Entonces, seguro que existe (por ejemplo) $e_2 \in \mathbb{N}$ tal que

$$\Phi^{(1)}_{e_2}(x) = \begin{cases} 2468 & \text{si } f_2(x, x) = 1 \\ 1 & \text{en otro caso.} \end{cases}$$

Para toda entrada $x \in \mathbb{N}$, se cumple que

$$\Phi^{(1)}_{e_2}(x) = 2468 \underset{\text{(def. } e_2\text{)}}{\iff} f_2(x, x) = 1 \underset{\text{(def. } f_2\text{)}}{\iff} \Phi^{(1)}_x(x) \text{ es impar} \quad \text{o} \quad \Phi^{(1)}_x(x) \uparrow .$$

Instanciando $x = e_2$, obtenemos que

$$\Phi^{(1)}_{e_2}(e_2) = 2468 \quad \iff \quad \Phi^{(1)}_{e_2}(e_2) \text{ es impar} \quad \text{o} \quad \Phi^{(1)}_{e_2}(e_2) \uparrow .$$

Esto quiere decir que en los dos casos posibles, $\Phi^{(1)}_{e_2}(e_2) = 2468$ y $\Phi^{(1)}_{e_2}(e_2) = 1$, se llega a una contradicción, lo cual es absurdo. Concluimos que nuestra suposición de que f_2 es computable debe ser falsa.

[1] Este paso de la demostración es el que le da su nombre a la técnica: estamos evaluando un programa con su propio número como entrada. Si imaginamos un plano donde ubicamos en un eje los números de programas y en el otro eje las posibles entradas, al emplear esta técnica estaríamos considerando un elemento de la *diagonal*.

[2] Esto es *muy importante*: para poder afirmar que se llegó un absurdo, debe encontrase una contradicción en *todos* los casos posibles. En este caso, si $\Phi^{(1)}_{e_1}(e_1)$ pudiera tener un resultado distinto que dar 0 o colgarse, esta demostración no sería exhaustiva y habría que analizar este tercer caso para llegar también a una contradicción.

[3] Algunas preguntas para reflexionar: ¿Por qué nos interesa el comportamiento de P_2 cuando recibe **una** entrada, si f_2 es una función de **dos** parámetros? ¿Por qué, en la definición de P_2, la entrada que recibe f_2 es (x, x)? ¿De qué otra manera se puede definir P_2 para que también "contradiga" a f_2? (Pista para la última: pensar en la posibilidad de que P_2 se indefina en algún caso).

Ejercicio 1.3

Demostrar que la siguiente función no es computable.

$$f_3(x) = \begin{cases} 2x & \text{si } \Phi_x^{(1)}(x) = x \\ 3x & \text{en otro caso.} \end{cases}$$

Resolución

Supongamos que f_3 es computable. Sea $e_3 \in \mathbb{N}$ tal que

$$\Phi_{e_3}^{(1)}(x) = \begin{cases} x & \text{si } f_3(x) = 3x \\ x+1 & \text{en otro caso.} \end{cases}$$

Claramente, si f_3 es computable, la función anterior también lo es, por lo que podemos estar seguros de la existencia de tal e_3.

Para toda entrada $x \in \mathbb{N}$, se cumple que

$$\Phi_{e_3}^{(1)}(x) = x \underset{\text{(def. } e_3)}{\iff} f_3(x) = 3x \underset{\text{(def. } f_3)}{\iff} \Phi_x^{(1)}(x) \neq x \quad \text{o} \quad x = 0,$$

donde la última disyunción se debe a que, si $x = 0$, $f_3(x) = 0 = 3x$ sin importar cómo se comporte $\Phi_x^{(1)}(x)$.

Instanciando $x = e_3$, obtenemos que

$$\Phi_{e_3}^{(1)}(e_3) = e_3 \iff \Phi_{e_3}^{(1)}(e_3) \neq e_3 \quad \text{o} \quad e_3 = 0.$$

A priori, esto no es un absurdo: está molestando la disyunción que permite que $e_3 = 0$. Pero, en realidad, es posible deducir que $e_3 \neq 0$, dado que el programa cuyo número es 0 es el programa vacío, que computa la función constante 0, y por lo tanto $\Phi_{e_3}^{(1)} \neq \Phi_0^{(1)}$. Entonces, podemos eliminar la disyunción y reescribir la equivalencia anterior como

$$\Phi_{e_3}^{(1)}(e_3) = e_3 \iff \Phi_{e_3}^{(1)}(e_3) \neq e_3.$$

Ahora sí llegamos a un absurdo, lo cual nos permite afirmar que f_3 no puede ser computable.

Ejercicio 2: Reducciones

La *reducción* es una técnica que nos permite aprovechar el conocimiento de que cierta función no es computable para demostrar que otra función relacionada tampoco lo es.

En general, al usar esta técnica, diremos que estamos demostrando que cierta función g no es computable *reduciendo* otra función f a g. Es necesario saber de antemano que f no es computable, o poder demostrarlo. Supondremos que existe una manera de computar g y mostraremos que partiendo de este resultado, se obtiene también una manera de computar f (es decir, que computar f *se reduce* a computar g de alguna forma particular). Entonces, habremos llegado a un absurdo que nos permitirá afirmar que g no es computable.

Ejercicio 2.1

Demostrar que la siguiente función no es computable.

$$g_1(x, y) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y) \downarrow \\ 0 & \text{en otro caso.} \end{cases}$$

Resolución

Vamos a demostrar que g_1 no es computable haciendo una *reducción* de f_1 a g_1. Si suponemos que g_1 es computable, podemos definir el siguiente (pseudo-)programa:

$$Q_1: \qquad Y \leftarrow g_1(X_1, X_1)$$

Así,

$$\Psi^{(1)}_{Q_1}(x) \underset{\text{(def. } Q_1\text{)}}{=} g_1(x,x) \underset{\text{(def. } g_1\text{)}}{=} \begin{cases} 1 & \text{si } \Phi^{(1)}_x(x)\downarrow \\ 0 & \text{en otro caso} \end{cases} \underset{\text{(def. } f_1\text{)}}{=} f_1(x).$$

Por lo tanto, f_1 es computable (por el programa Q_1), lo cual es absurdo, puesto que ya habíamos demostrado lo contrario. El absurdo provino de suponer que g_1 es computable. Concluimos que g_1 no es computable.

Ejercicio 2.2

Demostrar que la siguiente función no es computable.

$$g_2(x_1, y_1, x_2, y_2) = \begin{cases} 1 & \text{si } \Phi^{(1)}_{x_1}(y_1) \text{ divide a } \Phi^{(1)}_{x_2}(y_2) \\ 0 & \text{en otro caso.} \end{cases}$$

Resolución

En este caso, reduciremos la función f_2. Suponemos que g_2 es computable. Dadas dos entradas x e y, para poder computar $f_2(x, y)$ utilizando g_2 necesitamos lograr dos cosas:

- que donde dice $\Phi^{(1)}_{x_1}(y_1)$ pase a decir 2, y
- que donde dice $\Phi^{(1)}_{x_2}(y_2)$ pase a decir $\Phi^{(1)}_x(y)$.

El segundo punto es sencillo: basta con que al evaluar g_2 fijemos los parámetros x_2 e y_2 en x e y, respectivamente. Para el primer punto, debemos elegir convenientemente el programa que le pasaremos a g_2 en el parámetro x_1. Una opción sencilla es utilizar un programa que compute la función constante 2; esto nos permite usar cualquier valor para el parámetro y_1, por ejemplo, 42.[4]

Sea entonces $e \in \mathbb{N}$ tal que $\Phi^{(1)}_e(x) = 2$ para todo x. En tal caso,

$$\begin{aligned} g_2(e, 42, x, y) &\underset{\text{(def. } g_2\text{)}}{=} \begin{cases} 1 & \text{si } \Phi^{(1)}_e(42) \text{ divide a } \Phi^{(1)}_x(y) \\ 0 & \text{en otro caso} \end{cases} \\ &\underset{\text{(def. } e\text{)}}{=} \begin{cases} 1 & \text{si 2 divide a } \Phi^{(1)}_x(y) \\ 0 & \text{en otro caso} \end{cases} \\ &\underset{\text{(def. } f_2\text{)}}{=} \ldots \text{¡Ups!} \end{aligned}$$

Tenemos un pequeño detalle técnico que resolver: nuestra instanciación particular de g_2 está devolviendo 1 cuando f_2 devuelve 0, y viceversa. Afortunadamente, eso se arregla fácilmente usando la función α.

[4] Como ejercicio, pensar otros valores que podrían haberse elegido para x_1 e y_1.

$$\alpha(g_2(e, 42, x, y)) \underset{\text{(def. } g_2 \text{ y } \alpha)}{=} \begin{cases} 0 & \text{si } \Phi_e^{(1)}(42) \text{ divide a } \Phi_x^{(1)}(y) \\ 1 & \text{en otro caso} \end{cases}$$
$$\underset{\text{(def. } e)}{=} \begin{cases} 0 & \text{si 2 divide a } \Phi_x^{(1)}(y) \\ 1 & \text{en otro caso} \end{cases}$$
$$\underset{\text{(def. } f_2)}{=} f_2(x, y).$$

Como α es una función primitiva recursiva (y por lo tanto computable), g_2 es computable, y e y 42 son constantes, f_2 tiene que ser computable. Esto es absurdo; concluimos entonces que g_2 no puede ser computable.

Ejercicio 2.3

Demostrar que la siguiente función no es computable.

$$g_3(x, y) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y) \uparrow \text{ o } \Phi_x^{(1)}(x) \downarrow \\ 0 & \text{en otro caso.} \end{cases}$$

Resolución

En este caso, al menos a simple vista, no parece ser posible emplear una reducción de ninguna de las funciones del ejercicio 1. Sin embargo, aún así podemos emplear la técnica de reducción para "simplificar" g_3. Por ejemplo, si encontramos una función no computable que solo tome un parámetro y pueda reducirse a g_3, quizás demostrar que esta nueva función no es computable sea más sencillo que demostrarlo para g_3 directamente.

Un primer intento de función a reducir podría ser, al igual que en el ejercicio 2.1, la que resulta de igualar ambos parámetros de g_3. Es decir, la función $x \mapsto g_3(x, x)$. Sin embargo, esta función no cumple con el requisito de no ser computable: se trata de la función constante 1.

Mas útil resulta fijar en una constante el segundo parámetro de g_3. Por ejemplo, podemos definir

$$h_3(x) = g_3(x, 0) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(0) \uparrow \text{ o } \Phi_x^{(1)}(x) \downarrow \\ 0 & \text{en otro caso} \end{cases}$$

Suponiendo que g_3 es computable, h_3 resulta claramente computable. Si logramos llegar a un absurdo a partir de este último hecho, habremos demostrado que g_3 no es computable. Lo haremos mediante una diagonalización.

Queremos definir un programa, con número e, que se comporte de manera "rebelde" con respecto a h_3. Esto quiere decir que:

- si $h_3(e) = 0$, entonces $\Phi_e^{(1)}(0) \uparrow$ o $\Phi_e^{(1)}(e) \downarrow$.
- si $h_3(e) = 1$, entonces $\Phi_e^{(1)}(0) \downarrow$ y $\Phi_e^{(1)}(e) \uparrow$.

Podemos considerar, entonces, $e \in \mathbb{N}$ tal que

$$\Phi_e^{(1)}(x) = \begin{cases} 0 & \text{si } h_3(x) = 0 \text{ o } x = 0 \\ \uparrow & \text{en otro caso.} \end{cases}$$

Seguro existe tal e, porque es sencillo ver que la función recién presentada es parcial computable. Además, informalmente podemos comprobar que e cumple con los dos puntos anteriores, porque:

- si $h_3(e) = 0$, entonces $\Phi_e^{(1)}(e) = 0$, es decir, $\Phi_e^{(1)}(e) \downarrow$.

- si $h_3(e) = 1$, como claramente $e \neq 0$, entonces $\Phi_e^{(1)}(e) \uparrow$; además, seguro que $\Phi_e^{(1)}(0) \downarrow$.

Para este e y cualquier entrada x, tenemos que

$$\Phi_e^{(1)}(x) = 0 \underset{\text{(def. } e)}{\iff} h_3(x) = 0 \quad \text{o} \quad x = 0 \underset{\text{(def. } h_3)}{\iff} \left(\Phi_x^{(1)}(0) \downarrow \quad \text{y} \quad \Phi_x^{(1)}(x) \uparrow\right) \quad \text{o} \quad x = 0.$$

Instanciando $x = e$, tenemos que

$$\Phi_e^{(1)}(e) = 0 \iff \left(\Phi_e^{(1)}(0) \downarrow \quad \text{y} \quad \Phi_e^{(1)}(e) \uparrow\right) \quad \text{o} \quad e = 0,$$

aunque como sabemos que $e \neq 0$ (ya que $\Phi_e^{(1)} \neq \Phi_0^{(1)}$), nos deshacemos de la disyunción y nos queda

$$\Phi_e^{(1)}(e) = 0 \iff \Phi_e^{(1)}(0) \downarrow \quad \text{y} \quad \Phi_e^{(1)}(e) \uparrow,$$

y a su vez, usando que $\Phi_e^{(1)}(0) \downarrow$ por definición de e, eliminamos la conjunción y obtenemos

$$\Phi_e^{(1)}(e) = 0 \iff \Phi_e^{(1)}(e) \uparrow .$$

Esto último es un absurdo, ya que en los dos casos posibles ($\Phi_e^{(1)}(e) = 0$ y $\Phi_e^{(1)}(e) \uparrow$) se llega a una contradicción. Esto concluye nuestra demostración de que g_3 no puede ser computable.

Ejercicio 3

Demostrar o refutar las siguientes afirmaciones.

(a) Si $f : \mathbb{N}^n \to \mathbb{N}$ es una función total tal que, para alguna constante $k \in \mathbb{N}$, $f(x_1, \ldots, x_n) \leqslant k$ para todo $(x_1, \ldots, x_n) \in \mathbb{N}^n$, entonces f es computable.

(b) La función

$$g(x) = \begin{cases} 1 & \text{si Halt}(69, 21^{20}) \\ 0 & \text{en otro caso} \end{cases}$$

es computable.

(c) La función

$$D(x) = \begin{cases} 1 & \text{si Dios existe} \\ 0 & \text{en otro caso} \end{cases}$$

es computable.

Resolución

(a) **Falso**.

Podemos tomar como contraejemplo a nuestra vieja conocida

$$f(x) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x) \downarrow \\ 0 & \text{en otro caso.} \end{cases}$$

En este caso $f(x) \leqslant 1$ para todo x. Sin embargo, como ya demostramos en el ejercicio 1, f no es computable.

(b) **Verdadero**.

Al analizar la guarda de g, observamos que

$$\text{Halt}(69, 21^{20}) \iff \text{el programa número } 69 \text{ con entrada } 21^{20} \text{ se detiene,}$$

¡condición que no depende de x! Es decir, la sentencia expresada por $\text{Halt}(69, 21^{20})$ o bien es verdadera o bien es falsa para todo x. A pesar de que no sepamos cuál es el resultado exacto del problema de la parada para el programa 69 y la entrada 21^{20}, sí sabemos que es constante: 1 o 0.

Entonces, hay dos casos posibles:

- $g(x) = 1$ para todo x, en cuyo caso g es computada por el programa

$$Y \leftarrow 1$$

- $g(x) = 0$ para todo x, en cuyo caso g es computada por el programa vacío.

En cualquier caso, g es una función constante, por lo que es computable y, más aún, primitiva recursiva.

(c) **Verdadero**.

La resolución es análoga al ítem anterior: o bien Dios existe o bien Dios no existe. Independientemente de cuál sea La Verdad, la función D es o bien la constante 1 o bien la constante 0. En cualquiera de ambos casos, D es computable. Queda como ejercicio determinar qué programa computa efectivamente D.