Diagonalización y Reducciones

Modificación menor de clase originalmente dada por Ignacio Mollo

Clase del 6 de febrero de 2020

1 Diagonalización

La *diagonalización* es un método que podemos usar para demostrar la no computabilidad de cierto tipo de funciones. Nos va a servir en algunos casos cuando tengamos una función $f : \mathbb{N}^n \to \mathbb{N}$ que trate a algunas de sus entradas como números de programa sobre los cuales tenemos que decidir algún comportamiento. El ejemplo arquetípico de aplicación de esta técnica es la demostración de que la función $\text{Halt}(x, y)$ no es computable.

Pero primero, preguntémonos qué significa demostrar que una función es no computable. Para hacer esto, deberíamos ingeniárnoslas para demostrar que no existe ningún programa que la compute. ¿Cómo podemos demostrar que algo *no existe*?

El procedimiento general que veremos en esta clase es el de suponer que la función sí es computable, y llegar a un absurdo. En el caso del método de la diagonalización, esto consiste de los siguientes pasos:

a. Suponemos que la función f sí es computable.

b. Como es computable, podemos usarla como macro en un programa de $\mathcal{S}$, o usarla directamente para construir otras funciones computables. En particular, construimos un programa o función "rebelde", que de alguna manera hace lo contrario de lo que f dice.

c. Evaluamos la función computada por el programa rebelde en su propio número. Si el programa está bien elegido, ahora deberíamos llegar a un absurdo.

Ejercicio 1. Usando la técnica de diagonalización, demostrar que la siguiente función no es computable:

$$g_1(x) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x)\downarrow \\ 0 & \text{si no.} \end{cases}$$

Resolución. Empezamos haciendo la suposición fundamental: supongamos que g_1 sí es computable. Como es computable, podemos usarla como macro en la escritura de programas. En particular, podemos usarla para construir un "programa rebelde" que le pregunte a g_1 qué comportamiento tiene la entrada vista como número de programa, y proceda a tener el comportamiento opuesto. Una posibilidad es la siguiente:

$$\begin{array}{ll} & Z_1 \leftarrow g_1(X_1) \\ [A] & \text{IF } Z_1 = 1 \text{ GOTO } A \end{array}$$

Como g_1 es computable, el programa anterior está bien definido, así que tiene un número de programa e. Observemos que este programa computa la siguiente función:

$$\begin{aligned} \Phi_e^{(1)}(x) &= \begin{cases} \uparrow & \text{si } g_1(x) = 1 \\ 0 & \text{si no.} \end{cases} \\ &= \begin{cases} \uparrow & \text{si } \Phi_x^{(1)}(x)\downarrow \\ 0 & \text{si no.} \end{cases} \end{aligned}$$

Lo que hace esta función en entrada x es tener un funcionamiento "contrario" al de $\Phi_x^{(1)}(x)$: se indefine cuando $\Phi_x^{(1)}(x)$ está definida, y está definida cuando $\Phi_x^{(1)}(x)$ se indefine. ¿Por qué nos serviría esto? Porque en particular, cuando $x = e$, la función $\Phi_e^{(1)}$ va a contradecir su propio comportamiento, conduciéndonos a un absurdo. Veamos:

$$\Phi_e^{(1)}(e) = \begin{cases} \uparrow & \text{si } \Phi_e^{(1)}(e)\downarrow \\ 0 & \text{si no.} \end{cases}$$

¿Cuánto vale $\Phi_e^{(1)}(e)$? Hay dos posibilidades, dependiendo de cuál de las dos guardas de la expresión de arriba sea verdadera. Separemos en los dos casos:

- Si $\Phi_e^{(1)}(e)\downarrow$, entonces vale la guarda superior de la expresión de arriba, y eso significa que $\Phi_e^{(1)}(e)$ se cuelga. Absurdo. Por lo tanto, no puede ser que termine.
- Si $\Phi_e^{(1)}(e)\uparrow$, entonces vale la guarda inferior de la expresión de arriba. Eso significa que $\Phi_e^{(1)}(e)$ en realidad no se cuelga, y da cero. Absurdo. Por lo tanto, no puede ser que se cuelgue.

Los dos casos fallan. Como necesariamente tiene que valer alguno de ellos, llegamos a un absurdo. El absurdo provino de suponer que g_1 es computable, así que concluimos que no lo es. □

Ejercicio 2. Usando diagonalización, demostrar que la siguiente función no es computable:

$$g_2(x, y) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y)\downarrow \text{ y es par,} \\ 0 & \text{si no.} \end{cases}$$

Resolución. Supongamos que g_2 sí es computable. Al igual que en el ejercicio anterior, queremos construir un "programa rebelde" que haga lo contrario de lo que g_2 dice que debería hacer. Una posibilidad es la siguiente:

$$\begin{array}{ll} & \text{IF } g_2(X_1, X_1) \neq 0 \text{ GOTO } A \\ & Y \leftarrow Y + 1 \\ [A] & Y \leftarrow Y + 1 \end{array}$$

Como estamos suponiendo que g_2 es computable, el programa anterior está bien definido, y tiene un número de programa e. Vale que:

$$\Phi_e^{(1)}(x) = \begin{cases} 1 \text{ si } g_2(x,x) = 1 \\ 2 \text{ si no.} \end{cases}$$
$$= \begin{cases} 1 \text{ si } \Phi_x^{(1)}(x)\downarrow \text{ y es par,} \\ 2 \text{ si no.} \end{cases}$$

Ahora evaluamos esta función en e para llegar a un absurdo:

$$\Phi_e^{(1)}(e) = \begin{cases} 1 \text{ si } \Phi_e^{(1)}(e)\downarrow \text{ y es par,} \\ 2 \text{ si no.} \end{cases}$$

¿Cuánto vale $\Phi_e^{(1)}(e)$? De nuevo, hay dos posibilidades, dependiendo de cuál de las guardas es verdadera. Separemos en casos:

- Si $\Phi_e^{(1)}(e)\downarrow$ y es par, entonces $\Phi_e^{(1)}(e) = 1$. Pero entonces no es par, absurdo.
- Si $\Phi_e^{(1)}(e)\uparrow$ o es impar, entonces $\Phi_e^{(1)}(e) = 2$, o sea que ni se cuelga ni es impar. Absurdo.

Como las dos opciones fallan, llegamos a un absurdo, y podemos concluir que g_2 no es computable. □

Observación. En los dos ejercicios anteriores, podríamos habernos ahorrado la escritura del "programa rebelde". En su lugar, podríamos haber definido la función que queríamos utilizar, y demostrado que era computable por otros medios (usando la asumida computabilidad de la función que queremos probar es no computable). En el ejercicio que sigue haremos esto.

Ejercicio 3. Demostrar que la siguiente función no es computable:

$$g_3(x,y,z) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y)\downarrow \ \wedge \ \Phi_x^{(1)}(z)\downarrow \ \wedge \ \Phi_x^{(1)}(y) + \Phi_x^{(1)}(z) = y + z \\ 0 & \text{si no.} \end{cases}$$

Resolución. Como siempre, empezamos suponiendo que g_3 es computable, y la usamos para definir a la siguiente función:

$$f(u) = \begin{cases} u+1 & \text{si } g_3(u,u,u) = 1 \\ u & \text{si no.} \end{cases}$$

Esta función es computable porque es una función partida cuyas guardas y ramas son computables. Por lo tanto, existe un programa de número e que la

computa. Evaluando la función en ese mismo número, obtenemos:

$$\begin{aligned}\Phi_e^{(1)}(e) &= \begin{cases} e+1 & \text{si } g_3(e,e,e)=1 \\ e & \text{si no.} \end{cases} \\ &= \begin{cases} e+1 & \text{si } \Phi_e^{(1)}(e)\downarrow \ \wedge \ \Phi_e^{(1)}(e)+\Phi_e^{(1)}(e) \ = \ e+e \\ e & \text{si no.} \end{cases} \\ &= \begin{cases} e+1 & \text{si } \Phi_e^{(1)}(e)\downarrow \ \wedge \ \Phi_e^{(1)}(e) \ = \ e \\ e & \text{si no.} \end{cases}\end{aligned}$$

¿Cuánto vale $\Phi_e^{(1)}(e)$? Hay dos posibilidades, dependiendo de cuál de las guardas es verdadera:

- si $\Phi_e^{(1)}(e)\downarrow$ y vale e, entonces $\Phi_e^{(1)}(e)$ debería valer $e+1$, que es distinto de e.
- si $\Phi_e^{(1)}(e)\uparrow$ o no vale e, entonces $\Phi_e^{(1)}(e)$ debería valer e.

Ninguna de las dos opciones puede ser verdadera, así que llegamos a un absurdo. Concluimos que g_3 no es computable. □

2 Reducción

Sabemos que la composición de funciones parciales computables resulta parcial computable. Contrarrecíprocamente, si la composición de ciertas funciones resulta en una función que no es parcial computable, al menos una de las funciones utilizadas no era parcial computable.

En el contexto de probar la no-computabilidad de una función h, el método de reducción suele consistir en asumir que h es computable, realizar composiciones con funciones computables, y llegar a una función que ya sabemos no es computable. Esto implicará que la h no podía ser computable. Veamos algunos ejemplos:

Ejercicio 4. Demostrar, reduciendo alguna de las funciones de la sección anterior, que la siguiente función no es computable:

$$h_1(x,y) = \begin{cases} 1 & \text{si } \Phi_y^{(1)}(x)\downarrow \\ 0 & \text{si no.} \end{cases}$$

Resolución. La función h_1 se parece mucho a la función g_1 de la sección anterior. En efecto, vale que:

$$g_1(x) \ = \ h_1(x,x).$$

En otras palabras, la función g_1 se *reduce* a h_1. Entonces, como g_1 no es parcial computable, h_1 tampoco. □

Ejercicio 5. Demostrar, reduciendo alguna de las funciones de la sección anterior, que la siguiente función no es computable:

$$h_2(x,y,z) = \begin{cases} \Phi_z^{(1)}(z) & \text{si } \Phi_{x+y}^{(1)}(x \dot{-} y)\downarrow \\ 0 & \text{si no.} \end{cases}$$

Resolución. Esta función luce más complicada que todas las que vimos hasta ahora. Es la primera que devuelve algo más aparte de 0 y 1. Pero, para empezar, podemos deshacernos del y sin ningún problema, componiendo con la función constante cero. Y además, podemos tomar como z a cualquier número de programa, y eso nos da libertad para poner el valor que queramos en la rama superior de la función. Por ejemplo, podemos tomar e un número de programa que compute la función constantemente uno. Entonces,

$$\begin{aligned} h_2(x,0,e) &= \begin{cases} \Phi_e^{(1)}(e) & \text{si } \Phi_x^{(1)}(x)\downarrow \\ 0 & \text{si no.} \end{cases} \\ &= \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x)\downarrow \\ 0 & \text{si no.} \end{cases} \\ &= g_1(x) \end{aligned}$$

Entonces vemos que h_2 es una reducción de nuestra función g_1, que es no computable. Concluimos que h_2 tampoco lo es. □

Ejercicio 6. Demostrar, reduciendo alguna de las funciones de la sección anterior, que la siguiente función no es computable:

$$h_3(x_1,y_1,x_2,y_2) = \begin{cases} y_1^2 \dot{-} 33 & \text{si } \Phi_{x_1}^{(1)}(y_1)\downarrow \ \wedge \ \Phi_{x_2}^{(1)}(y_2)\downarrow \wedge \\ & \wedge \ \left(\Phi_{x_1}^{(1)}(y_1)+2\right) \text{ divide a } \left(\Phi_{x_2}^{(1)}(y_2)+6\right) \\ 0 & \text{si no.} \end{cases}$$

Resolución. Ya que g_2 "habla" de divisibilidad, probemos reducir la función g_2 de la sección anterior a h_3. Para que resulte más intuitivo qué composiciones hacer, reescribimos la función g_2 en términos de variables que no aparecen en h_3:

$$\begin{aligned} g_2(u,v) &= \begin{cases} 1 & \text{si } \Phi_u^{(1)}(v)\downarrow \text{ y es par,} \\ 0 & \text{si no.} \end{cases} \\ &= \begin{cases} 1 & \text{si } \Phi_u^{(1)}(v)\downarrow \text{ y 2 divide a } \Phi_u^{(1)}(v), \\ 0 & \text{si no.} \end{cases} \end{aligned}$$

Sea e el número de un programa que computa la función constantemente cero. En particular, vale que $\Phi_e^{(1)}(y)\downarrow$ para cualquier y, y que $\Phi_e^{(1)}(y)+2 = 2$. Entonces:

$$h_3(e,y_1,x_2,y_2) = \begin{cases} y_1^2 \dot{-} 33 & \text{si } \Phi_{x_2}^{(1)}(y_2)\downarrow \ \wedge \ \ 2 \text{ divide a } \left(\Phi_{x_2}^{(1)}(y_2)+6\right) \\ 0 & \text{si no.} \end{cases}$$

El número 6 que aparece sumando en la guarda superior nos molesta, pues no parece haber forma de eliminarlo. Si no estuviera ahí, bastaría con colocar u y v en lugar de x_2 y y_2 respectivamente, para obtener la guarda superior de $g_2(u,v)$. Sin embargo, luego de pensarlo un rato nos percatamos de que no hace falta eliminarlo. Como 6 es un número par, para todo número n vale que:

$$2 \text{ divide a } n \Leftrightarrow \ 2 \text{ divide a } (n+6)$$

Por lo tanto, la guarda superior de $h_3(e, y_1, u, v)$ es equivalente a la de $g_2(u, v)$.

El último problema que nos queda solucionar es el $y_1^2 \dot{-} 33$. No existe ningún número natural y_1 que haga que $y_1^2 \dot{-} 33$ sea 1. Para elegir algo, hagamos la composición con 6, de modo que nos quede un número distinto de 0. Así:

$$h_3(e, 6, u, v) = \begin{cases} 3 & \text{si } \Phi_u^{(1)}(v)\downarrow \ \wedge \ \ 2 \text{ divide a } \Phi_u^{(1)}(v) \\ 0 & \text{si no.} \end{cases}$$

Estamos muy cerca de la función que queremos. Vamos a tener que hacerle "algo computable" a esta función, para cambiar el 3 por un 1. Tenemos varias opciones:

- Podemos restar 2. Nos queda:

$$\begin{aligned} h_3(e, 6, u, v) \dot{-} 2 \ &= \ \begin{cases} 1 & \text{si } \Phi_u^{(1)}(v)\downarrow \ \wedge \ \ 2 \text{ divide a } \Phi_u^{(1)}(v) \\ 0 & \text{si no.} \end{cases} \\ &= \ g_2(u, v) \end{aligned}$$

- Podemos aprovechar que $h_3(e, 6, u, v)$ alcanza sólo dos valores para crear una función partida:

$$\begin{aligned} H(u, v) \ &= \ \begin{cases} 1 & \text{si } h_3(e, 6, u, v) = 3 \\ 0 & \text{si no.} \end{cases} \\ &= \ g_2(u, v) \end{aligned}$$

- Podemos usar la función α. Recordemos que la función α representa el predicado "ser igual a cero". Entonces,

$$\begin{aligned} \alpha \circ \alpha(h_3(e, 6, u, v)) = \ &\begin{cases} 1 & \text{si } \Phi_u^{(1)}(v)\downarrow \ \wedge \ \ 2 \text{ divide a } \Phi_u^{(1)}(v) \\ 0 & \text{si no.} \end{cases} \\ = \ &g_2(u, v) \end{aligned}$$

En cualquier caso, logramos reducir g_2 a h_3, y como g_2 no es computable, h_3 tampoco. □

Ejercicio 7. Supongamos que $h_1, h_2 : \mathbb{N} \to \mathbb{N}$ son funciones computables (totales). Supongamos que f y g son funciones que verifican:

$$f(h_1(x), h_2(x)) \ = \ g(x)$$

Decidir si las siguientes afirmaciones son verdaderas o falsas:

a. Si g no es computable, entonces f no es computable.

b. Si f no es computable, entonces g no es computable.

Resolución.

a. Supongamos por el absurdo que f es computable. Entonces, como g se define a partir de composición de funciones computables, debe ser computable, lo cual contradice nuestra hipótesis. Luego, f debe ser no computable, y esta afirmación es Verdadera.

b. Esta es la afirmación recíproca de **a**, y es Falsa. Efectivamente, es posible que f sea no computable y g sí lo sea. Veamos un ejemplo:

   Tomemos $h_1(x) = 107$ y $h_2(x) = 42$; o sea, dos funciones constantes, que en particular son computables. Tomemos $f(x, y) = \text{Halt}(x, y)$ y g el resultado de componer f con h_1 y h_2, es decir:

$$g(x) \;=\; \text{Halt}(107, 42).$$

   Entonces cumplimos con las hipótesis de **b**, y sin embargo g es una función computable. Esto se sigue de que $\text{Halt}(107, 42)$ es independiente de la variable x. Por lo tanto, la función g es constante, y en particular es computable. (Observemos que no estamos dando ningún método efectivo para *computar* el valor de $\text{Halt}(107, 42)$, pero eso no nos importa. Lo importante es que es o constantemente 0 o bien constantemente 1).

□