Clase práctica

Computabilidad: diagonalización y reducciones

Nina Pardal

25 de septiembre de 2020

Algunos comentarios previos...

Consejos para encarar una demostración:

1. Escribir formalmente las hipótesis con las que cuentan
2. Escribir formalmente aquello que quieren demostrar
3. Analizar la estructura del problema para entender qué técnicas se pueden/conviene utilizar:
    - Inducción (común o estructural)
    - Demostración constructiva
    - Reducción por el absurdo

Algunos comentarios previos...

Sobre las demostraciones por el absurdo:

¿Cuál es el procedimiento?
Parto de un cierto conjunto de hipótesis $\mathcal{S}$ (o de un cierto marco teórico donde valen ciertas reglas) y quiero demostrar que vale una cierta proposición P
Supongo que valen todas las hipótesis de $\mathcal{S}$ y además supongo que vale $\neg P$, y ahí empiezo a inferir...

¿Qué logramos con eso? $\rightarrow$ si al suponer verdadero todo $\mathcal{S}$ y $\neg P$ llegara a una contradicción (como estoy segura de que todas las suposiciones de $\mathcal{S}$ son verdaderas), entonces la causa de la contradicción es suponer que vale $\neg P$ $\Rightarrow$ si valen todas las hipótesis de $\mathcal{S}$, entonces P debe ser verdadera.

Un ejemp(lit)o:

''Existen infinitos números primos'' $= P$

En este caso, mi conjunto de hipótesis $\mathcal{S}$/marco teórico es:

(I) $\mathbb{N}$ y las características que definen a los números naturales

(II) la definición de número primo

$\neg P =$ *''Existen finitos números primos''*
Sean $p_1, p_2, \ldots, p_k$ todos los números primos naturales.
Defino un nuevo número $p := p_1 p_2 \ldots p_k + 1$.
Como $p > p_i$, entonces $p \neq p_i$, y esto vale para todo $i = 1, \ldots, k$.
Luego, p no es primo.
Entonces, existe algún número primo que lo divida, **es decir, alguno de los p_i lo divide**, pero esto no puede ser porque $p_i \mid p_1 p_2 \ldots p_k$ y $p_i \nmid 1$, y así llegamos a un absurdo (que vino de suponer que existían finitos primos!)

Ejercicio 1

Demostrar que las siguientes funciones no son computables:

Ejercicio 1

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

Para ver que una función no es computable podemos usar una *diagonalización*: asumir que es computable y generar una función computable que falle sobre una entrada particular.

Observemos primero que, por su definición, f_1 es total.
Supongamos que f_1 es computable y lleguemos a un absurdo. Sea P_1 un programa que computa f_1 ($\Psi_{P_1} = f_1$).

Defino el programa P'_1 como:

$$\begin{array}{ll} & P_1 \\ [R] & \text{IF } Y \neq 0 \text{ GOTO } R \end{array}$$

donde R es una etiqueta fresca.

Ejercicio 1

Vemos que para todo x se cumple que

$$\Psi_{P_1'}(x)\downarrow \Leftrightarrow \Psi_{P_1}(x) = 0 \Leftrightarrow f_1(x) = 0$$

Por definición de f_1, para todo x vale

$$f_1(x) = 0 \Leftrightarrow \Phi_x(x)\uparrow$$

Por lo tanto, para todo x tenemos

$$\Psi_{P_1'}(x)\downarrow \Leftrightarrow \Phi_x(x)\uparrow$$

En particular, si tomamos $e = \#P_1'$ nos queda

$$\Phi_e(e)\downarrow \Leftrightarrow \Phi_e(e)\uparrow$$

absurdo! Luego, f_1 no es computable. $\diamond$

Ejercicio 1

$$f_2(x) = \begin{cases} 1 & \Phi_x(x) \uparrow \\ 0 & \text{en otro caso} \end{cases}$$

Opción 1: Podemos hacer una *reducción* a f_1 (que ya vimos que no es computable) aprovechando que hay una relación entre las dos funciones. *¿Cuál?* $f_1(x) = 1 - f_2(x)$

Supongamos que f_2 es computable para llegar a un absurdo. Sea P_2 es un programa que computa f_2 ($\Psi_{P_2} = f_2$).
Defino un nuevo programa P_2' como

$$\begin{array}{l} P_2 \\ Y \leftarrow 1 - Y \end{array}$$

Por inspección es claro que

$$\Psi_{P_2'}(x) = 1 - \Psi_{P_2}(x) = 1 - f_2(x) = f_1(x)$$

Como ya demostramos que f_1 no es computable, esto es un absurdo. $\diamond$

Ejercicio 1

Opción 2: Probar que f_2 no es computable diagonalizando en lugar de reduciendo.
¿Cómo?
Definiendo un programa P_2' idéntico a P_1' salvo por cambiar la comparación $Y \neq 0$ por $Y = 0$.

Queda de ejercicio!

Ejercicio 1

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

Supongamos que f_3 es computable. Sea P_3 un programa que computa f_3 ($\Psi_{P_3} = f_3$).
Consideramos el programa P_3' definido como:

$$\begin{array}{ll} & P_3 \\ & Y \leftarrow Y - 2 \times X \\ [R] & \text{IF } Y \neq 0 \text{ GOTO } R \\ & Y \leftarrow X \end{array}$$

donde R es una etiqueta fresca.

Ejercicio 1

Por inspección de P'_3 vemos que para todo x se cumple que

$$\Psi_{P'_3}(x) \downarrow \Leftrightarrow \Psi_{P_3}(x) = 2x \text{ ó } x = 0$$

Por definición de f_3, para todo x vale

$$f_3(x) = 2x \Leftrightarrow \Phi_x(x) \uparrow \text{ ó } \Phi_x(x) \neq x$$

Por lo tanto, como $\Psi_{P_3} = f_3$ vale para todo x que

$$\Psi_{P'_3}(x) \downarrow \Leftrightarrow \Phi_x(x) \uparrow \text{ ó } \Phi_x(x) \neq x \text{ ó } x = 0$$

En particular, si tomamos $e = \#P'_3$ nos queda

$$\Phi_e(e) \downarrow \Leftrightarrow \Phi_e(e) \uparrow \text{ ó } \Phi_e(e) \neq e \text{ ó } e = 0$$

Observar que $e \neq 0$ porque $e = \#P'_3$ y P'_3 no es el programa vacío. Ademas sabemos que $\Phi_e(e) \neq e$ es falso, porque si termina, P'_3 devuelve siempre su entrada. Luego, llegamos al absurdo

$$\Phi_e(e) \downarrow \Leftrightarrow \Phi_e(e) \uparrow$$

$\diamond$

Ejercicio 1

$$f_4(x) = \begin{cases} 1 & \Phi_x(x) \uparrow \text{ ó } \Phi_x(x) \leqslant 2014 \\ 0 & \text{en otro caso} \end{cases}$$

Supongamos f_4 computable. Sea P_4 el programa que la computa ($\Psi_{P_4} = f_4$). Consideramos el siguiente P_4':

$$\begin{array}{ll} & P_4 \\ [R] & \text{IF } Y = 0 \text{ GOTO } R \\ & Y \leftarrow 2015 \end{array}$$

Siguiendo este camino, se termina el ejercicio parecido al anterior. Vamos a hacer las cuentas en otro orden para ver más posibilidades.

Ejercicio 1

Consideremos $e = \#P'_4$. Por inspección del código, hay dos posibilidades:

$$\Psi_{P'_4}(e) \uparrow \quad \text{o bien} \quad \Psi_{P'_4}(e) = 2015$$

Es decir que si $\Phi_e(e) \uparrow$ es porque

$$\Psi_{P_4}(e) = 0 \implies f_4(e) = 0$$

Pero por definición de $f_4 \Rightarrow \Phi_e(e) \downarrow$, *absurdo!*
Si en cambio $\Phi_e(e) = 2015$, esto implica

$$\Psi_{P_4}(e) = 1 \implies f_4(e) = 1$$

Por definición de f_4 obtenemos

$$\Phi_e(e) \uparrow \text{ ó } \Phi_e(e) \leqslant 2014$$

Ambas opciones contradicen la premisa $\Phi_e(e) = 2015$, *absurdo!* $\diamond$

Ejercicio 2

Demostrar que las siguientes funciones no son computables:

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

Ejercicio 2

Observemos que $f_1(x) = g_1(x, x)$. Esta reducción tiene pinta de no ser computable, pero hay que demostrarlo! Supongamos g_1 computable y sea Q_1 un programa que la computa ($\Psi^{(2)}_{Q_1} = g_1$). Queremos dar un programa que computa f_1 usando Q_1. Sea Q'_1:

$$X_2 \leftarrow X_1$$
$$Q_1$$

Es inmediato que

$$\Psi^{(1)}_{Q'_1}(x) = \Psi^{(2)}_{Q_1}(x, x)$$

Por hipótesis,

$$\Psi^{(2)}_{Q_1}(x, x) = g_1(x, x) = f_1(x)$$

Usando ambas igualdades juntas obtenemos

$$\Psi^{(1)}_{Q'_1}(x) = f_1(x)$$

lo cual es un absurdo porque ya vimos que f_1 no es computable. $\diamond$

Ejercicio 2

$$g_2(x,y) = \begin{cases} 1 & \Phi_x(y)\uparrow \text{ ó } \Phi_x(x)\downarrow \\ 0 & \text{en otro caso} \end{cases}$$

En este caso, si reducimos a $x \mapsto g_2(x,x)$ nos queda una función que es constantemente 1, por lo tanto computable, con lo cual no nos va a servir.
Una reducción mas útil es $g_2'(x) = g_2(x,0)$:

$$g_2'(x) = \begin{cases} 1 & \Phi_x(0)\uparrow \text{ ó } \Phi_x(x)\downarrow \\ 0 & \text{en otro caso} \end{cases}$$

Para poder usar esta reducción tenemos que hacer 2 cosas: **(1)** mostrar la no computabilidad de g_2', y **(2)** mostrar que la reducción misma es computable.

Vamos a hacer el primer paso, que es el más complejo, y el resto queda como ejercicio.

Ejercicio 2

Supongamos g_2' computable y Q_2 un programa que la computa ($\Psi^{(2)}_{Q_2} = g_2'$).
Sea Q_2' el siguiente programa

$$
\begin{array}{ll}
 & Q_2 \\
 & \text{IF } X = 0 \text{ GOTO } F \\
[R] & \text{IF } Y \neq 0 \text{ GOTO } R \\
[F] & Y \leftarrow 0
\end{array}
$$

y $e = \#Q_2'$.
Por inspección del programa Q_2',

$$
\Psi_{Q_2'}(x) = \Phi_e(x) = \begin{cases} 0 & x = 0 \text{ ó } g_2'(x) = 0 \\ \uparrow & \text{en otro caso} \end{cases} \qquad (*)
$$

Ejercicio 2

Usando la definición de Q_2' resumida en $(*)$ podemos ver que

$$\Phi_e(e) \downarrow \Leftrightarrow e = 0 \text{ ó } g_2'(e) = 0$$

Como sabemos que $e \neq 0$,

$$e = 0 \text{ ó } g_2'(e) = 0 \Leftrightarrow \text{FALSO ó } g_2'(e) = 0 \Leftrightarrow g_2'(e) = 0$$

Usando la definición de g_2' obtenemos,

$$g_2'(e) = 0 \Leftrightarrow \Phi_e(0) \downarrow \text{ y } \Phi_e(e) \uparrow$$

Usando $(*)$ una vez más vemos que $\Phi_e(0) = 0$ o sea que $\Phi_e(0) \downarrow$, con lo cual

$$\Phi_e(0) \downarrow \text{ y } \Phi_e(e) \uparrow \Leftrightarrow \text{VERDADERO y } \Phi_e(e) \uparrow \Leftrightarrow \Phi_e(e) \uparrow$$

Poniendo todos los $\Leftrightarrow$ de arriba juntos usando transitividad nos queda el absudo

$$\Phi_e(e) \downarrow \Leftrightarrow \Phi_e(e) \uparrow$$

que provino de suponer g_2' computable. $\diamond$

Ejercicio 2

$$g_3(x, y, z, w) = \begin{cases} x + w & \Phi_x(z) \uparrow \text{ y } \Phi_y(z) \uparrow \\ w + z & \Phi_x(z) \downarrow \text{ y } \Phi_y(z) \downarrow \text{ y } \Phi_x(z) = \Phi_y(z) \\ w & \text{en otro caso} \end{cases}$$

Hay muchas reducciones que sirven. Una que nos permite aprovechar el trabajo del ejercicio anterior es $g_3'(x) = g_3(x, d, x, 2x)$ donde d es el número del programa

$$Y \leftarrow X.$$

De esta manera, $\Phi_d(x) = x$ para todo x y se verifica que $g_3' = f_3$. Ya sabemos que f_3 no es computable, con lo cual solo falta verificar la reducción.

Ejercicio 2

Supongamos Q_3 un programa que computa g_3 ($\Psi^{(4)}_{Q_3} = g_3$). Sea Q'_3 el siguiente programa

$$
\begin{aligned}
& X_2 \leftarrow d \\
& X_3 \leftarrow X_1 \\
& X_4 \leftarrow 2 \times X_1 \\
& Q_3
\end{aligned}
$$

Recordando que X y X_1 son la misma variable, se ve inmediatamente que Q'_3 computa $g'_3 = f_3$, lo cual es un absurdo. $\diamond$

Ejercicio 3

Dado una función total $f : \mathbb{N} \to \mathbb{N}$, un *aproximador* de f es una función total $g : \mathbb{N}^2 \to \mathbb{N}$ tal que para todo x, $g(x,t) = f(x)$ para todo t salvo finitos valores. Dicho de otra manera, $\lim_{t\to\infty} g(x,t) = f(x)$. Decidir si son verdaderas o falsas las siguientes afirmaciones. Justificar la respuesta.

a. Si f es computable entonces tiene un aproximador computable.

b. Si f tiene un aproximador computable entonces f es computable.

Ejercicio 3

a. *Verdadera*. Si sabemos el verdadero valor de f, es fácil aproximarlo: definimos $g(x, t) = f(x)$. Es fácil ver que "ignorar" el parámetro t es una reducción computable: si P es un programa que computa f ($\Psi_P^{(1)} = f$) entonces se deduce inmediatamente que el mismo P computa g como la definimos arriba ($\Psi_P^{(2)} = g$). $\diamond$

b. *Falsa*. La ventaja que tiene el aproximador g es que cuenta con un parámetro t con el que puede "acotar" las computaciones y así asegurarse terminar siempre. Como toda computación que termina lo hace en una cantidad de pasos fija t, g se va a mantener constante cuando su segundo parámetro sea mas grande que t.

Ejercicio 3

Consideremos una función no computable lo mas simple posible.
Por ejemplo, $f = f_1$

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

Ya sabemos que f es no computable.
Definimos también $g(x, t)$ de una forma parecida, pero acotada a algo computable.

$$g(x, t) = \begin{cases} 1 & \Phi_x(x) \text{ termina en } t \text{ o menos pasos} \\ 0 & \text{en otro caso} \end{cases}$$

Ejercicio 3

Para completar el contraejemplo tenemos que ver que

I. $\lim_{t\to\infty} g(x,t) = f(x)$

II. $g(x,t)$ es computable.

Lo primero se sigue inmediatamente de las definiciones: si $\Phi_x(x)$ termina, entonces hay alguna cantidad de pasos t en la que termina y por lo tanto

$$g(x,t) = g(x,t+1) = g(x,t+2) = ... = f(x)$$

La segunda parte sale inmediatamente de ver que $g(x,t) = \mathrm{STP}^{(1)}(x,x,t)$, con lo cual g es primitiva recursiva, y por lo tanto, computable. $\diamond$