Clase práctica 9: Diagonalización e indecidibilidad

Lenguajes Formales, Autómatas y Computabilidad

Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

Primer cuatrimestre 2025

Hay más funciones que naturales

Teorema

Sea $A = \{f \mid f : \mathbb{N} \to \mathbb{N}\}$, no existe ninguna función $h : A \to \mathbb{N}$ biyectiva.

Hay más funciones que naturales

Teorema

Sea $A = \{f \mid f : \mathbb{N} \to \mathbb{N}\}$, no existe ninguna función $h : A \to \mathbb{N}$ biyectiva.

Demostración

Supongamos que sí, que existe una función $h : A \to \mathbb{N}$ biyectiva. Esto quiere decir que puedo enumerar las funciones de naturales en naturales. Considero entonces la enumeración que induce la función h: $f_1, f_2, ...$
Construyo una nueva función $g : \mathbb{N} \to \mathbb{N}$ tal que

$$g(n) = f_n(n) + 1$$

Como g es una función de naturales en naturales, debe existir un $k \in \mathbb{N}$ tal que $g = f_k$. Pero entonces vale que $f_k(k) = g(k) = f_k(k) + 1$ lo cual es un absurdo, que vino de suponer que existía una función $h : A \to \mathbb{N}$ biyectiva. □

Conjunto de partes

Teorema

Sea A un conjunto, no existe una función $h : A \to \mathcal{P}(A)$ sobreyectiva.

Conjunto de partes

Teorema

Sea A un conjunto, no existe una función $h : A \to \mathcal{P}(A)$ sobreyectiva.

Demostración

Supongamos que sí existe una función $h : A \to \mathcal{P}(A)$ sobreyectiva. Dado $a \in A$ hay dos casos posibles: o bien $a \in h(a)$ o bien $a \notin h(a)$. Definimos entonces el conjunto:

$$B = \{a \in A \ : \ a \notin h(a)\}$$

Notar que $B \in \mathcal{P}(A)$ y por lo tanto, como h es sobreyectiva, existe un $a_0 \in A$ tal que $h(a_0) = B$. Entonces:

- Si $a_0 \notin B$, entonces $a_0 \notin h(a_0)$, y por definición de B, $a_o \in B$.
- Si $a_0 \in B$, entonces $a_0 \in h(a_0)$, y por definición de B, $a_o \notin B$.

Esto es un absurdo, que vino de suponer que existía una función $h : A \to \mathcal{P}(A)$ sobreyectiva. □

Comentarios sobre demostraciones formales

Consejos para encarar una demostración:

1. Escribir formalmente las hipótesis que tienen.

Comentarios sobre demostraciones formales

Consejos para encarar una demostración:

1. Escribir formalmente las hipótesis que tienen.
2. Escribir formalmente lo que quieren demostrar.

Comentarios sobre demostraciones formales

Consejos para encarar una demostración:

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

Demostración por el absurdo

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.

Demostración por el absurdo

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.
Si al suponer verdadero todas las hipótesis $\mathcal{S}$ y $\neg P$ llegara a una contradicción, como estoy segura de que todas las suposiciones de $\mathcal{S}$ son verdaderas, entonces el absurdo proviene de suponer que vale $\neg P$.

Demostraciones en computabilidad: absurdo y diagonalización

La demostración por el absurdo va a ser muy útil para demostrar no computabilidad de funciones, y vamos a usar también la técnica de *diagonalización*.
Vamos a querer demostrar que una cierta función f no es computable.
Vamos a suponer entonces que f es computable, y a partir de ahí vamos a construir una nueva función. que resultará computable asumiendo que f lo es, que va a *fallar* sobre una entrada particular.

Demostraciones en computabilidad: absurdo y diagonalización

La demostración por el absurdo va a ser muy útil para demostrar no computabilidad de funciones, y vamos a usar también la técnica de *diagonalización*.
Vamos a querer demostrar que una cierta función f no es computable.
Vamos a suponer entonces que f es computable, y a partir de ahí vamos a construir una nueva función. que resultará computable asumiendo que f lo es, que va a *fallar* sobre una entrada particular.

El Halting problem

Enunciado

Demostrar que el predicado $HALT : \mathbb{N} \to \mathbb{N}$ no es computable, donde

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

El Halting problem

Enunciado

Demostrar que el predicado $HALT : \mathbb{N} \to \mathbb{N}$ no es computable, donde

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

Demostración

Supongamos que HALT es computable y sea P un programa que la computa. Construimos la siguiente función:

$$f(x) = \begin{cases} 0 & \text{si HALT(x)=0} \\ \uparrow & \text{si no} \end{cases}$$

El Halting problem

Continuación demostración

La función f resulta computable suponiendo que $HALT$ lo es, pues el siguiente programa la computa:

$$Q: \ \mathit{While}\ \Psi_P^{(1)}(X_1) \neq 0\ \mathit{do}\ \{\}$$

Sea e el número del programa anterior. Vale que:

$$\Phi_e(e)\downarrow \ \mathit{sii}\ \Psi_P^{(1)}(e) = 0\ \mathit{sii}\ \mathit{Halt}(e) = 0\ \mathit{sii}\ \Phi_e(e)\uparrow$$

Lo cual es un absurdo, que vino de suponer que $HALT$ era computable. □

Ejercicio 1

Demostrar que las siguientes funciones no son computables:

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

Ejercicio 1

Demostrar que las siguientes funciones no son computables:

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

Una forma de resolverlo es nuevamente con diagonalización. Asumimos entonces que f_1 es computable y sea P un programa que la computa. Notemos que la función es muy parecida a $HALT(x, x)$, así que podemos plantear una nueva función muy parecida a la que planteamos antes:

$$g(x) = \begin{cases} 1 & si\ f_1(x) = 1 \\ \uparrow & si\ f_1(x) = 0 \end{cases}$$

Y para ver que es computable podemos dar, como antes, un programa que la computa:

$$\Psi_P^{(1)}(X_1)$$
$$\textbf{WHILE}\ Y = 0\ DO\ \{\}$$

Y la demostración prosigue con una análisis análogo al anterior.

Ejercicio 1 - otra forma

Otra forma de demostrar que no es computable es mediante una **reducción**, que ya vimos la clase pasada. La idea es aprovechar que la función es muy parecida a $Halt(x, x)$, y que ya sabemos que HALT no es computable, para demostrar que f_2 tampoco lo es. Primero, ¿qué relación exactamente tienen f_2 y HALT? $HALT(x, x) = \alpha(f_2(x)) = 1 - f_2(x)$. Suponiendo que f_2 es computable, debe existir algún programa P_2 que la computa. Podemos definir entonces un nuevo programa P':

$$\Psi^{(1)}_{P_2}(X_1)$$
$$Y := 1 - Y$$

y por inspección es claro que

$$\Psi_{P'}(x) = 1 - \Psi_{P_2}(x) = 1 - f_2(x) = HALT(x, x)$$

Y esto resulta un absurdo, porque ya sabemos que HALT no es computable. Por lo tanto, f_2 tampoco puede serlo.

Ejercicio 2

$$f_2(x) = \begin{cases} 1 & si\ \Phi_x(x) \uparrow \text{ ó } \Phi_x(x) \leqslant 2014 \\ 0 & \text{en otro caso} \end{cases}$$

Ejercicio 2

$$f_2(x) = \begin{cases} 1 & si\ \Phi_x(x) \uparrow \text{ ó } \Phi_x(x) \leqslant 2014 \\ 0 & \text{en otro caso} \end{cases}$$

Supongamos f_2 computable. Planteamos:

$$\text{f'}_2(x) = \begin{cases} 2015 & si\ f_2(x) = 1 \\ \uparrow & \text{en otro caso} \end{cases}$$

Que resulta computable porque puedo dar el siguiente programa que la computa, siendo P_4 un programa que computa f_2:

$$\Psi^{(1)}_{P_2}(X_1)$$
$$\textbf{WHILE } Y = 0 \textbf{ DO } \{\}$$
$$Y := 2015$$

Ejercicio 2

Consideremos e el número de algún programa que computa f_2'. Podemos ver que para todo x vale que:

$$\Phi_e(x) \downarrow \iff f_2(x) = 1$$

Además, si $\Phi_e(x) \downarrow$, entonces $\Phi_e(x) = 2015$ siempre, por lo que vale que:

$$\Phi_e(x) \downarrow \wedge\ \Phi_e(x) = 2015 \iff f_2(x) = 1$$

Ahora evaluando Φ_e en e, por definición de f_2, llegamos a que:

$$\Phi_e(e) \downarrow \wedge\ \Phi_e(e) = 2015 \iff f_2(e) = 1 \iff \Phi_e(e) \uparrow \vee\ \Phi_e(e) \leqslant 2014$$

Lo cual es un absurdo, que vino de suponer que f_2 es computable.

El Halting problem revisitado

Enunciado

Consideremos ahora el predicado $HALT : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ definido como

$$HALT(x, y) = \begin{cases} 1 & \text{si } \Phi_x(y) \downarrow \\ 0 & \text{si no} \end{cases}$$

Demostrar que $HALT$ no es computable.

El Halting problem revisitado

Enunciado

Consideremos ahora el predicado $HALT : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ definido como

$$HALT(x, y) = \begin{cases} 1 & \text{si } \Phi_x(y) \downarrow \\ 0 & \text{si no} \end{cases}$$

Demostrar que $HALT$ no es computable.

Resolución

Supongamos que sí lo es. Si $Halt(x, y)$ es computable, entonces la función $H : \mathbb{N} \to \mathbb{N}$ definida como $H(x) = HALT(x, x)$ también sería computable, puesto que sería composición de funciones computables, y ya vimos en el ejercicio anterior que H no es computable. Por lo tanto, $HALT$ tampoco puede serlo. □

Ejercicio 3

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

Ejercicio 3

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

Notar que en este caso la reducción $f_1(x) = g_1(x,x)$ es la función constantemente 1, con lo cual resulta computable y no nos sirve para demostrar lo que queremos. Otra posibilidad para hacer una reducción es con una **constante**. Podemos tomar:

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

Vamos a demostrar que g_1' no es computable. Queda de tarea demostrar que la reducción es computable.

Ejercicio 3

Planteamos la siguiente función:

$$f_1(x) = \begin{cases} \uparrow & g_1'(x) = 1 \text{ y } x \neq 0 \\ 1 & \text{en otro caso} \end{cases}$$

Supongamos g_1' computable y Q_1 un programa que la computa ($\Psi^{(1)}_{Q_1} = g_1'$). Sea Q_1' el siguiente programa:

$\Psi^{(1)}_{Q_1}(X_1)$
IF $X_1 = 0$ **then** $Y := 0$ **else** *pass*
WHILE $Y \neq 0$ **DO** $\{\}$
$Y := 1$

Por inspección del programa Q_1', vemos que efectivamente computa la función f_1.

Ejercicio 3

Sea entonces e el número del programa Q_1'. Podemos ver que

$$\Phi_e(e) \downarrow \iff e = 0 \text{ ó } g_1'(e) = 0$$

Como sabemos que $e \neq 0$, pues el programa de número 0 computa la función constante 0,

$$e = 0 \text{ ó } g_1'(e) = 0 \iff \text{FALSO ó } g_1'(e) = 0 \iff g_1'(e) = 0$$

Usando la definición de g_1' obtenemos,

$$g_1'(e) = 0 \iff \Phi_e(0) \downarrow \text{ y } \Phi_e(e) \uparrow$$

Usando la definición de f_1 una vez más vemos que $\Phi_e(0) = 0$, o sea que $\Phi_e(0) \downarrow$, con lo cual

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

Poniendo todos los $\iff$ de arriba juntos usando transitividad nos queda:

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

que provino de suponer g_1' computable. Por lo tanto g_1 tampoco lo es.

Ejercicio 4

Enunciado

Sea $g : \mathbb{N} \to \mathbb{N}$ una función total que crece asintóticamente más rápido que cualquier función computable unaria, es decir, para toda $f : \mathbb{N} \to \mathbb{N}$ computable, existe un n_f tal que $g(n) \geqslant f(n)$ para todo $n \geqslant n_f$.
Demostrar que g no es computable.

Ejercicio 4

Enunciado

Sea $g : \mathbb{N} \to \mathbb{N}$ una función total que crece asintóticamente más rápido que cualquier función computable unaria, es decir, para toda $f : \mathbb{N} \to \mathbb{N}$ computable, existe un n_f tal que $g(n) \geqslant f(n)$ para todo $n \geqslant n_f$.
Demostrar que g no es computable.

Demostración

Supongamos que sí lo es, y definamos una nueva función h como:

$$h(n) = g(n) + 1$$

Claramente la función h es computable suponiendo que g lo es, y vale que $h(n) > g(n)$ para todo n. Esto es un absurdo, que vino de suponer que g era computable. □

Ejercicio 5

Enunciado

Decidir si la siguiente afirmación es verdadera o falsa y demostrar: sea $f : \mathbb{N}^2 \to \mathbb{N}$ una función no computable, entonces la función $g : \mathbb{N} \to \mathbb{N}$ definida como:

$$g_n(x) = f(x, n)$$

tampoco es computable.

Ejercicio 5

Enunciado

Decidir si la siguiente afirmación es verdadera o falsa y demostrar: sea $f : \mathbb{N}^2 \to \mathbb{N}$ una función no computable, entonces la función $g : \mathbb{N} \to \mathbb{N}$ definida como:

$$g_n(x) = f(x, n)$$

tampoco es computable.

Resolución

La afirmación es **Falsa**. Consideremos la siguiente función:

$$f(x, y) = \begin{cases} 1 & \Phi_y(y) \downarrow \\ 0 & \text{sino} \end{cases}$$

Ejercicio 5 continuación

Resolución

Si consideramos la función $g_n(x) = f(x, n)$, esta función resulta ser la función

$$g_n(x) = \begin{cases} 1 & \Phi_n(n) \downarrow \\ 0 & \text{sino} \end{cases}$$

Como n es en este caso una **constante**, la función g resulta ser una función constante para cualquier n (la función constante 0 o la función constante 1 dependiendo n), y en cualquier caso resulta computable.

Ejercicio 6

Enunciado

Decidir si la siguiente función es computable o no y justificar:

$$f(x) = \begin{cases} 1 & \Phi_x^{(1)} \text{ es parcial computable} \\ 0 & \text{sino} \end{cases}$$

Ejercicio 6

Enunciado

Decidir si la siguiente función es computable o no y justificar:

$$f(x) = \begin{cases} 1 & \Phi_x^{(1)} \text{ es parcial computable} \\ 0 & \text{sino} \end{cases}$$

Resolución

f es parcial computable! Sabemos que $\Phi_x^{(1)}$ siempre es parcial computable (lo vimos en la clase de intérprete universal), con lo cual f es la función constante 1.

Ejercicio 7

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.

Resolución ejercicio 7

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 ($\Psi_P^{(2)} = g$).

Resolución ejercicio 7

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 ($\Psi_P^{(2)} = g$).

b. *Falsa*. La ventaja que tiene el aproximador g es que cuenta con un parámetro t con el que puede "acotar" los cómputos y así asegurarse de terminar siempre. Como todo cómputo que termina lo hace en una cantidad fija de pasos t, g se va a mantener constante cuando su segundo parámetro sea más grande que ese t. Veamos un contraejemplo.

Resolución ejercicio 7-b: contraejemplo

Consideremos una función no computable lo mas simple posible. Por ejemplo, $f(x) = HALT(x, x)$. Ya sabemos que f es no computable. Definimos ahora $g(x, t)$ de una forma parecida, pero acotada a algo computable.

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

Resolución ejercicio 7-b: contraejemplo

Para completar el contraejemplo tenemos que ver que

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

Resolución ejercicio 7-b: contraejemplo

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_0 en la que termina y por lo tanto

$$g(x,t_0) = g(x,t_0+1) = g(x,t_0+2) = \ldots = f(x)$$

para todo $t \geqslant t_0$. La segunda parte sale inmediatamente de ver que $g(x,t) = STEP^{(1)}(x,t,x)$,

FIN

★ ★ ★