Teoría de la Computabilidad

Santiago Figueira

Departamento de Computación, FCEyN, UBA

Primer cuatrimestre 2017

Contenido

Teoría de la Computabilidad

Clase 0

Repaso de la parte de Computabilidad de LyC

Organización de la materia

- 3 puntos para Licenciatura en Computación, 3 puntos para Doctorado en Computación
- correlativa: LyC
- Lunes de 13 a 17 en aula 11 del pab. I
- Evaluación: seguramente un parcial y un trabajo final.
- http://www.glyc.dc.uba.ar/teocomp/

Bibliografía

En orden de importancia

- Robert I. Soare, Recursively Enumberable Sets and Degrees: A Study of Computable Functions and Computably Generated Sets (Perspectives in Mathematical Logic), Springer, 1987.
- Hartley Rogers, Theory of Recursive Functions and Effective Computability, The MIT Press, 1987.
- Piergiorgio Odifreddi, Classical Recursion Theory, Vol. 1 (Studies in Logic and the Foundations of Mathematics, Vol. 125), North Holland, 1999.
- Piergiorgio Odifreddi, Classical Recursion Theory, Vol. 2 (Studies in Logic and the Foundations of Mathematics, Vol. 143), North Holland, 1999.
- Martin Davis, Ron Sigal, Elaine Weyuker, Computability, Complexity and Languages, fundamentals of theoretical computer science, Elsevier, 1994.

Composición y recursión primitiva

Definición

Sea $f : \mathbb{N}^k \to \mathbb{N}$ y $g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$. $h : \mathbb{N}^n \to \mathbb{N}$ se obtiene a partir de f y $g_1, \ldots, g_k$ por composición si

$$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n))$$

Definición

$h : \mathbb{N}^{n+1} \to \mathbb{N}$ se obtiene de $g : \mathbb{N}^{n+2} \to \mathbb{N}$ y $f : \mathbb{N}^n \to \mathbb{N}$ por recursión primitiva si

$$\begin{aligned} h(x_1, \ldots, x_n, 0) &= f(x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n, t+1) &= g(t, x_1, \ldots, x_n, h(x_1, \ldots, x_n, t)) \end{aligned}$$

Funciones primitivas recursivas $\mathcal{P}_1$

Funciones iniciales

- $\mathcal{O}(x) = 0$
- $\mathcal{S}(x) = x + 1$
- $\mathcal{I}_i^n(x_1, \ldots, x_n) = x_i$ para todo $n \geq 1$ y $i \in \{1, \ldots, n\}$

Definición

La clase $\mathcal{P}_1$ de funciones primitivas recursivas es la clase de funciones más chica:

- conteniendo las funciones iniciales
- cerrada por composición
- cerrada por recursión primitiva

Ejemplos de funciones en $\mathcal{P}_1$

- funciones aritméticas básicas: suma, resta, producto, potencia
- conectivos booleanos: $\wedge$, $\vee$, $\neg$ (verdadero se interpreta como 1 y falso como 0)
- predicados aritméticos básicos: $=$, $\leq$
- definición por casos: si $g_1, \ldots, g_m, h \in \mathcal{P}_1$ y $p_1, \ldots, p_m \in \mathcal{P}_1$ son predicados mutuamente excluyentes entonces también está en $\mathcal{P}_1$ la función

$$f(\vec{x}) = \begin{cases} g_1(\vec{x}) & \text{si } p_1(\vec{x}) \\ \vdots & \\ g_m(\vec{x}) & \text{si } p_m(\vec{x}) \\ h(\vec{x}) & \text{si no} \end{cases}$$

- sumatorias y productorias: si $f \in \mathcal{P}_1$ entonces también están en $\mathcal{P}_1$ las funciones

$$g(y, \vec{x}) = \sum_{t=0}^{y} f(t, \vec{x}) \qquad \text{y} \qquad h(y, \vec{x}) = \prod_{t=0}^{y} f(t, \vec{x})$$

Ejemplos de funciones en $\mathcal{P}_1$

- cuantificadores acotados: si $p \in \mathcal{P}_1$ es un predicado entonces también están en $\mathcal{P}_1$:

$$(\forall t)_{\leq y}\ p(t, \vec{x}) \qquad \text{y} \qquad (\exists t)_{\leq y}\ p(t, \vec{x})$$

- minimización acotada: si $p \in \mathcal{P}_1$ es un predicado entonces también está en $\mathcal{P}_1$ la función

$$\min_{t \leq y} p(t, \vec{x}) = \begin{cases} \text{mínimo } t \leq y \text{ tal que} \\ p(t, \vec{x}) \text{ es verdadero} & \text{si existe tal } t \\ y + 1 & \text{si no} \end{cases}$$

- codificación y decodificación de k-uplas:

$$\langle x_1, \ldots, x_k \rangle \qquad \pi_i(z)$$

- codificación y decodificación de secuencias:

$$[x_1, \ldots, x_n] \qquad z[i]$$

Recursión *course-of-value*

Teorema

$\mathcal{P}_1$ está cerrada por recursión course-of-value en donde la definición de $f(\vec{x}, y+1)$ involucra no solo el último valor $f(\vec{x}, y)$ sino también cualquier número (quizá todos) los valores $(f(\vec{x}, z))_{z \leq y}$. Más formalmente, si definimos la función de historia

$$\hat{f}(\vec{x}, y) = [f(\vec{x}, 0), \ldots, f(\vec{x}, y)],$$

tenemos que si $g, h \in \mathcal{P}_1$ entonces también está en $\mathcal{P}_1$ la función f definida como:

$$\begin{aligned} f(\vec{x}, 0) &= g(\vec{x}) \\ f(\vec{x}, y+1) &= h(\vec{x}, y, \hat{f}(\vec{x}, y)) \end{aligned}$$

Recursión simultánea

Teorema

$\mathcal{P}_1$ está cerrada por recursión simultánea en donde un número finito de funciones $f_1, \ldots, f_n$ se definen simultáneamente y la definición de $f_m(\vec{x}, y+1)$ puede involucrar no solo $f_m(\vec{x}, y)$ sino también cualquier número (quizá todos) los valores $(f_i(\vec{x}, y))_{1 \leq i \leq n}$.

Funciones parciales

Una función parcial es simplemente una función que puede indefinirse en algunos (quizá todos) sus argumentos.

Una función parcial es total si se define para todos sus argumentos. Las funciones primitivas recursivas son totales.

Solo trabajamos con funciones $\mathbb{N}^k \to \mathbb{N}$.

Notamos

- $f(\vec{x}) \downarrow$ cuando f está definida en $\vec{x}$
- $f(\vec{x}) \uparrow$ cuando f está indefinida en $\vec{x}$

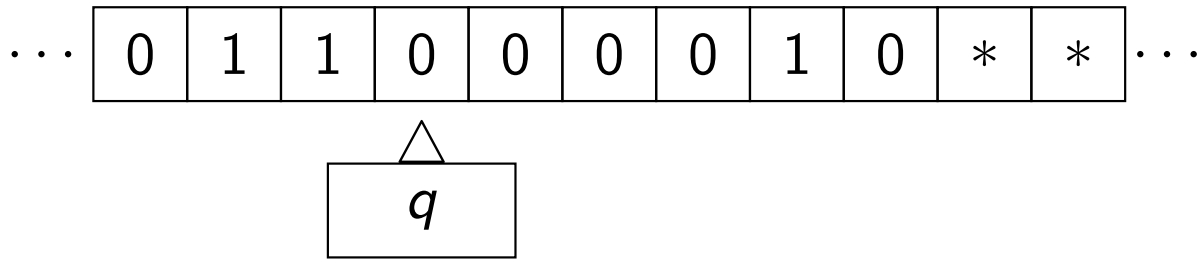
El dominio de una función es conjunto de argumentos en el que la función se define:

$$\text{dom } f = \{\vec{x} \colon f(\vec{x}) \downarrow\}.$$

Si f y g son funciones parciales, por $f(\vec{x}) = g(\vec{x})$ nos referimos a

$$[f(\vec{x}) \uparrow \wedge g(\vec{x}) \uparrow] \vee [f(\vec{x}) \downarrow \wedge g(\vec{x}) \downarrow \wedge \underbrace{f(\vec{x})}_{\in\mathbb{N}} = \underbrace{g(\vec{x})}_{\in\mathbb{N}}].$$

Máquinas de Turing



Se compone de :

- una cinta
  - dividida en celdas
  - infinita en ambas direcciones
  - cada celda contiene un símbolo de un alfabeto dado Σ.
    - $* \in \Sigma$
    - $L, R \notin \Sigma$

      $*$ representa el blanco en una celda

      L y R son símbolos reservados (representarán acciones que puede realizar la cabeza)
- una cabeza
  - lee y escribe un símbolo a la vez
  - se mueve una posición a la izquierda o una posición a la derecha
- una tabla finita de instrucciones
  - dice qué hacer en cada paso

Tabla de instrucciones

- Σ es el alfabeto. $L, R \notin \Sigma$, $* \in \Sigma$.
- Q es el conjunto finito de estados
- $A = \Sigma \cup \{L, R\}$ es el conjunto de acciones
  - un símbolo $s \in \Sigma$ se interpreta como "escribir s en la posición actual"
  - L se interpreta como "mover la cabeza una posición hacia la izquierda"
  - R se interpreta como "mover la cabeza una posición hacia la derecha"

Una tabla de instrucciones T es un subconjunto (finito) de

$$Q \times \Sigma \times A \times Q$$

La tupla

$$(q, s, a, q') \in T$$

se interpreta como

> *Si la máquina está en el estado q leyendo en la cinta el símbolo s, entonces realiza la acción a y pasa al estado q'*

Máquinas de Turing

Una máquina de Turing es una tupla

$$(\Sigma, Q, T, q_0, q_f)$$

donde

- Σ (finito) es el conjunto símbolos $(L, R \notin \Sigma, * \in \Sigma)$
- Q (finito) es el conjunto de estados
  - tiene dos estados distinguidos:
    - $q_0 \in Q$ es el estado inicial
    - $q_f \in Q$ es el estado final
- $T \subseteq Q \times \Sigma \times \Sigma \cup \{L, R\} \times Q$ es la tabla de instrucciones
  - va a ser finita porque Σ y Q lo son

- cuando no hay restricciones sobre T decimos que $\mathcal{M}$ es una máquina de Turing no determinísitca
- cuando no hay dos instrucciones en T que empiezan con las mismas primeras dos coordenadas, decimos que $\mathcal{M}$ es una máquina de Turing determinísitca
- Si no decimos nada, es una máquina de Turing determinística.

Representación de números y tuplas en la cinta

Fijamos $\Sigma = \{*, 1\}$.

- representaremos a los números naturales en unario (con palotes).
  - el número $x \in \mathbb{N}$ se representa como

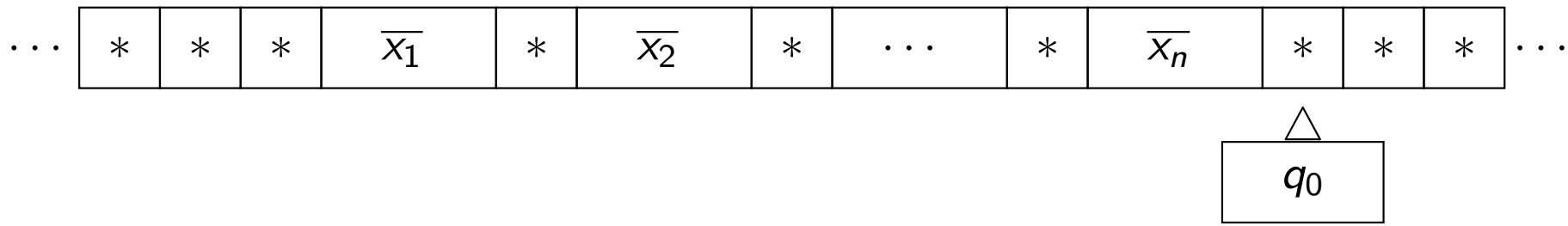
$$\overline{x} = \underbrace{1 \ldots 1}_{x+1}$$

- representamos a las tuplas $(x_1, \ldots, x_n)$ como lista de (representaciones de) los x_i separados por blanco
  - la tupla $(x_1, \ldots, x_n)$ se representa como

$$*\overline{x_1} * \overline{x_2} * \cdots * \overline{x_n} *$$

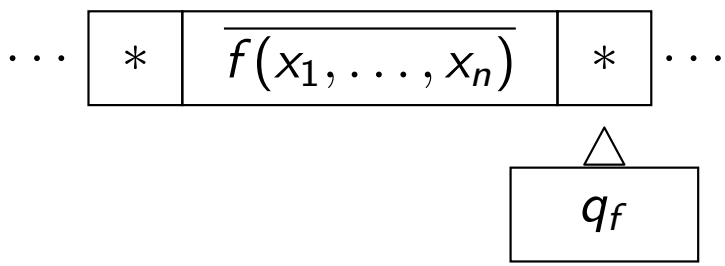
Funciones parciales computables

Una función (parcial) $f : \mathbb{N}^n \to \mathbb{N}$ es parcial computable si existe una máquina de Turing determinística $\mathcal{M} = (\Sigma, Q, T, q_0, q_f)$ con $\Sigma = \{*, 1\}$ tal que cuando empieza en la configuración inicial



(con los enteros x_i representados en unario y nada más en la cinta de entrada salvo la representación de la entrada):

- si $f(x_1, \ldots, x_n) \downarrow$ entonces siguiendo sus instrucciones en T llega a una configuración final de la forma



  (quizá algo más en la cinta)

- si $f(x_1, \ldots, x_n) \uparrow$ entonces nunca termina en el estado q_f.

Una función es computable si es parcial computable y total.

Enumeración de las máquinas de Turing

Habiendo fijado Σ, q_0, q_f, cada máquina de Turing puede ser identificada por su tabla de instrucciones.

Una tabla de instrucciones es un conjunto finito de 4-uplas. Cada tabla de instrucciones (i.e. cada máquina) puede ser codificado de manera efectiva por un número.

Definición

M_e es la máquina de Turing con tabla de instrucciones con código e. $\varphi_e^{(n)}$ es la función parcial de n variables computada por la máquina M_e. Abreviamos $\varphi^{(1)} = \varphi$.

Notación:

- "φ_e es la e-ésima función parcial computable"
- e es el índice o código o número de Gödel de φ_e.

Teoremas de Forma Normal y de Enumeración

Teorema (Forma Normal)

Existe un predicado $T(e, x, t)$ *y una función* $U(y)$ *primitivos recursivos tal que*

$$\varphi_e(x) = U(\min_t T(e, x, t)).$$

Teorema (Enumeración, existencia de máquina universal)

Para cada $n \geq 1$ *hay una función parcial computable de* $n + 1$ *variables,* $\varphi_{z_n}^{(n+1)}$*, tal que*

$$\varphi_{z_n}^{(n+1)}(e, x_1, \ldots, x_n) = \varphi_e^{(n)}(x_1, \ldots, x_n).$$

Teorema del Parámetro (TP)

Teorema

Para cada $m, n \geq 1$ *existe una función primitiva recursiva 1-1* s_n^m *de* $m+1$ *variables tal que para todo* $y_1, \ldots, y_m, z_1, \ldots, z_n \in \mathbb{N}$,

$$\varphi_{s_n^m(x,y_1,\ldots,y_m)}(z_1, \ldots, z_n) = \varphi_x(y_1, \ldots, y_m, z_1, \ldots, z_n).$$

Teorema de la Recursión (TR)

Teorema

Para cualquier función computable f existe un $n \in \mathbb{N}$ (llamado punto fijo de f) tal que $\varphi_n = \varphi_{f(n)}$.

Teorema (Recursión con parámetros)

Para cualquier función computable $f : \mathbb{N}^2 \to \mathbb{N}$ existe una función computable $n : \mathbb{N} \to \mathbb{N}$ tal que $\varphi_{n(y)} = \varphi_{f(n(y),y)}$.

Demostración.

Definir d computable tal que

$$\varphi_{d(x,y)}(z) = \begin{cases} \varphi_{\varphi_x(x,y)}(z) & \text{si } \varphi_x(x,y) \downarrow \\ \uparrow & \text{si no} \end{cases}$$

Sea v tal que $\varphi_v(x,y) = f(d(x,y),y)$. Definir $n(y) = d(v,y)$.

$$\varphi_{n(y)} = \varphi_{d(v,y)} = \varphi_{\varphi_v(v,y)} = \varphi_{f(d(v,y),y)} = \varphi_{f(n(y),y)}.$$

□

Conjuntos, secuencias, cadenas

Cuando hablamos de un conjunto de naturales A pensamos siempre en la función característica de ese conjunto.

$$A(x) = \begin{cases} 1 & \text{si } x \in A \\ 0 & \text{si no} \end{cases}$$

Así, un conjunto puede ser computable o primitivo recursivo.
A puede ser visto también como una secuencia infinita de 0s y 1s:

$$A(0)\ A(1)\ A(2)\ A(3)\ A(4)\ A(5) \cdots \in \{0,1\}^\infty$$

Los primeros n bits de A (visto como tira de 0s y 1s) es

$$A \upharpoonright n = A(0)\ A(1)\ \ldots A(n-1)$$

Las cadenas finitas de 0s y 1s pueden ser vistas como conjuntos finitos.

Ejecución de a pasos

Definición

Notamos $\varphi_{e,s}(x) = y$ si $x, y, e < s$ e y es la salida de la máquina con código e ejecutada por $< s$ pasos.

- si tal y existe, decimos que $\varphi_{e,s}(x)$ converge ($\varphi_{e,s}(x)\downarrow$).
- en caso contrario decimos que $\varphi_{e,s}(x)$ diverge ($\varphi_{e,s}(x)\uparrow$).

Teorema

Los siguientes conjuntos son computables:

- $\{\langle e, x, s\rangle : \varphi_{e,s}(x)\downarrow\}$
- $\{\langle e, x, y, s\rangle : \varphi_{e,s}(x) = y\}$

Conjuntos computablemente enumerables

Definición

- Un conjunto $A \subseteq \mathbb{N}$ es comptuablemente enumerable (c.e.) si A es el dominio de alguna función parcial computable.
- El e-ésimo conjunto c.e. se denota

$$W_e = \text{dom } \varphi_e = \{x : \varphi_e(x) \downarrow\} = \{x : (\exists t)\ T(e, x, t)\}$$

- $W_{e,s} = \text{dom } \varphi_{e,s}$ (por convención si $x \in W_{e,s}$ entonces $x, e < s$).

Teorema

A es computable sii A y $\overline{A}$ son c.e.

Teorema

Si f es parcial computable y A es c.e. entonces $f^{-1}(A)$ es c.e.

Caracterizaciones de los conjuntos c.e.

Teorema

Si $A \neq \emptyset$, son equivalentes:

1. *A es r.e.*
2. *A es el rango de una función primitiva recursiva*
3. *A es el rango de una función computable*
4. *A es el rango de una función parcial computable*

Una función computable que no es primitiva recursiva

Se pueden enumerar las funciones primitivas recursivas de 1 variable: $\psi_1, \psi_2, \ldots$

Teorema (Enumeración de funciones primitivas recursivas)

Hay una función f computable de 2 variables tal que

$$f(e, x) = \psi_e(x).$$

Analicemos $g : \mathbb{N} \to \mathbb{N}$ definida como $g(x) = f(x, x)$.

- claramente g es computable porque f lo es.
- supongamos que $g \in \mathcal{P}_1$
  - entonces $h(x) = g(x) + 1 = f(x, x) + 1 \in \mathcal{P}_1$
  - existe un e tal que $\psi_e = h$
  - tenemos $f(e, x) = h(x) = f(x, x) + 1$
  - e está fijo pero x es variable
  - instanciando $x = e$ llegamos a un absurdo
- ¿por qué esto no funciona para funciones parciales computables en vez de primitivas recursivas?

Teoría de la Computabilidad

Clase 1

Funciones Primitivas recursivas, iteración, esqueleto de funciones primitivas recursivas, recursión anidada en varias variables

Composición y recursión primitiva

Definición

Sea $f : \mathbb{N}^k \to \mathbb{N}$ y $g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$. $h : \mathbb{N}^n \to \mathbb{N}$ se obtiene a partir de f y $g_1, \ldots, g_k$ por composición si

$$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n))$$

Definición

$h : \mathbb{N}^{n+1} \to \mathbb{N}$ se obtiene de $g : \mathbb{N}^{n+2} \to \mathbb{N}$ y $f : \mathbb{N}^n \to \mathbb{N}$ por recursión primitiva si

$$\begin{aligned} h(x_1, \ldots, x_n, 0) &= f(x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n, t+1) &= g(t, x_1, \ldots, x_n, h(x_1, \ldots, x_n, t)) \end{aligned}$$

Funciones primitivas recursivas $\mathcal{P}_1$

Funciones iniciales

- $\mathcal{O}(x) = 0$
- $\mathcal{S}(x) = x + 1$
- $\mathcal{I}_i^n(x_1, \ldots, x_n) = x_i$ para todo $n \geq 1$ y $i \in \{1, \ldots, n\}$

Definición

La clase $\mathcal{P}_1$ de funciones primitivas recursivas es la clase de funciones más chica:

- conteniendo las funciones iniciales
- cerrada por composición
- cerrada por recursión primitiva

Recursión anidada en una variable

Definición

f se define a partir de $g_1, \ldots, g_n$ por recursión anidada en una variable si

$$\begin{aligned} f(\vec{x}, 0) &= g(\vec{x}) \\ f(\vec{x}, y) &= h(\vec{x}, y) \end{aligned}$$

donde $h(\vec{x}, y)$ es un término numérico construido por

- números naturales n
- las variables $\vec{x}$ e y
- las funciones $g_1, \ldots, g_m$
- el símbolo f
- los valores de f usados en la definición de $f(\vec{x}, y)$ son de la forma $f(\vec{t}, s)$ donde s tiene un valor numérico $< y$

Teorema

$\mathcal{P}_1$ está cerrada por recursión anidada en una variable.

Caracterización alternativa de $\mathcal{P}_1$

Definición

Una función f se define por iteración a partir de t si

$$f(x, n) = t^{(n)}(x)$$

donde $t^{(n)}$ denota el resultado de n aplicaciones sucesivas de t (por convención $t^{(0)}(x) = x$).

Teorema

La clase de funciones primitivas recursivas es la clase más chica:

- *conteniendo las funciones iniciales y funciones de codificación y decodificación de pares*
- *cerrada por composición*
- *cerrada por iteración*

Caracterización alternativa de $\mathcal{P}_1$ (demo)

Sea $\mathcal{C}$ la clase más chica de funciones que satisfacen las condiciones del teorema.

$\mathcal{C} \subseteq \mathcal{P}_1$ Las funciones de codificación y decodificación son primitivas recursivas. La iteración $f(x, n) = t^{(n)}(x)$ es un caso especial de recursión primitiva:

$$\begin{aligned} f(x, 0) &= x \\ f(x, n+1) &= t(f(x, n)) \end{aligned}$$

$\mathcal{P}_1 \subseteq \mathcal{C}$ Hay que ver que $\mathcal{C}$ es cerrada por recursión primitiva. Observar que $\mathcal{C}$ tiene cod./decod. de pares y composición, por lo tanto tiene cod./decod. de n-uplas para cualquier n fijo. Sean $g, h \in \mathcal{C}$ y sea f definida como

$$\begin{aligned} f(\vec{x}, 0) &= g(\vec{x}) \\ f(\vec{x}, y+1) &= h(\vec{x}, y, f(\vec{x}, y)) \end{aligned}$$

Basta ver que esta función está en $\mathcal{C}$:

$$s(\vec{x}, n) = \langle \vec{x}, n, f(\vec{x}, n) \rangle$$

La función

$$\begin{aligned} s(\vec{x}, 0) &= \langle \vec{x}, 0, f(\vec{x}, 0) \rangle \\ &= \langle \vec{x}, 0, g(\vec{x}) \rangle \end{aligned}$$

está en $\mathcal{C}$ porque g y $\langle \cdot \rangle$ están en $\mathcal{C}$. Podemos transformar

$$s(\vec{x}, n) = \langle \vec{x}, n, \underbrace{f(\vec{x}, n)}_{z} \rangle$$

en

$$\begin{aligned} s(\vec{x}, n+1) &= \langle \vec{x}, n+1, f(\vec{x}, n+1) \rangle \\ &= \langle \vec{x}, n+1, h(\vec{x}, n, \underbrace{f(\vec{x}, n)}_{z}) \rangle \end{aligned}$$

con esta función, que también está en $\mathcal{C}$:

$$t(\vec{x}, n, z) = \langle \vec{x}, n+1, h(\vec{x}, n, z) \rangle.$$

Como $\mathcal{C}$ está cerrada por iteración, está en $\mathcal{C}$ esta función:

$$s(\vec{x}, n) = t^{(n)}(s(\vec{x}, 0)).$$

La familia de funciones $(h_n)_{n\in\mathbb{N}}$

Sea

$$h_0(x) = x + 1 \qquad \text{y} \qquad h_{n+1}(x) = h_n^{(x)}(x)$$

Proposición

$x \leq h_n(x)$.

Demostración.

Por inducción en n. Para $n = 0$ es trivial. Para $n + 1$, suponer $(\forall z)\ z \leq h_n(z)$. Es fácil probar por inducción en m que $x \leq h_n^{(m)}(x)$. En particular

$$x \leq h_n^{(x)}(x) = h_{n+1}(x).$$

□

Propiedades de $(h_n)_{n\in\mathbb{N}}$

Proposición

Si $n \leq m$ *y* $x > 0$ *entonces* $h_n(x) \leq h_m(x)$.

Demostración.

Basta probar $(\forall x > 0)\ h_n(x) \leq h_{n+1}(x) = h_n^{(x)}(x)$.

Basta probar $h_n(z) \leq h_n^{(m)}(z)$ por inducción en $m > 0$.

Para $m = 1$ es trivial. Para $m + 1$,

$$\begin{array}{rcll} h_n(z) & \leq & h_n^{(m)}(z) & \text{por HI} \\ & \leq & h_n(h_n^{(m)}(z)) & \text{por Proposición de pág. 34} \\ & = & h_n^{(m+1)}(z) & \end{array}$$

□

Propiedades de $(h_n)_{n\in\mathbb{N}}$

Proposición

Si $x \leq y$ entonces $h_n(x) \leq h_n(y)$.

Demostración.

Por inducción en n. Para $n = 0$ es trivial. Para $n+1$,

$$\begin{array}{rcll} h_{n+1}(x) & = & h_n^{(x)}(x) & \\ & \leq & h_n^{(y)}(x) & \text{por Proposición de pág. 34 } (y-x \text{ veces}) \\ & \leq & h_n^{(y)}(y) & \text{por HI } (y \text{ veces}) \\ & = & h_{n+1}(y). & \end{array}$$

□

Propiedades de $(h_n)_{n\in\mathbb{N}}$

Proposición

$h_1(x) = 2x$.

Demostración.

Basta probar $h_0^{(m)}(x) = x + m$ por inducción en m. Para $m = 0$ es trivial. Para $m + 1$,

$$\begin{aligned} h_0^{(m+1)}(x) &= h_0(h_0^{(m)}(x)) \\ &= h_0(x + m) && \text{por HI} \\ &= (x + m) + 1 && \text{porque } h_0(z) = z + 1 \\ &= x + (m + 1). \end{aligned}$$

□

Propiedades de $(h_n)_{n\in\mathbb{N}}$

Proposición

$h_2(x) = x \cdot 2^x$.

Demostración.

Basta probar $h_1^{(m)}(x) = x \cdot 2^m$ por inducción en m. Para $m = 0$ es trivial. Para $m+1$,

$$\begin{aligned} h_1^{(m+1)}(x) &= h_1(h_1^{(m)}(x)) \\ &= h_1(x \cdot 2^m) && \text{por HI} \\ &= (x \cdot 2^m) \cdot 2 && \text{porque } h_1(z) = 2z \\ &= x \cdot 2^{m+1} \end{aligned}$$

□

Todas las h_n son primitivas recursivas

Teorema

Para cada n, h_n es primitiva recursiva.

Demostración.

Obvio, pues la función sucesor está en $\mathcal{P}_1$ y $\mathcal{P}_1$ está cerrada por iteración. □

Cada función primitiva recursiva es dominada por alguna h_n

Teorema

Para cada f primitiva recursiva, hay un n tal que $f(\vec{x}) \leq h_n(\sum \vec{x})$ para casi todo $\vec{x}$.

Demostración.

Por inducción en la caracterización alternativa de $\mathcal{P}_1$ (pág. 31).
Veamos que:

- las funciones iniciales y las funciones de codificación y decodificación lo cumplen
- si $h, g_1, \ldots, g_m$ lo cumplen,

$$f(\vec{x}) = h(g_1(\vec{x}), \ldots, g_m(\vec{x}))$$

también lo cumple

- si t lo cumple,

$$f(x, y) = t^{(y)}(x)$$

también lo cumple.

Cada función primitiva recursiva es dominada por alguna h_n

Funciones iniciales, codificación y decodificación

- $\mathcal{O}(x) = 0$ está dominada por h_0
- $\mathcal{S}(x) = x + 1$ está dominada por h_0
- $\mathcal{I}_i^n(x_1, \ldots, x_n) = x_i$ está dominada por h_0 porque

$$\begin{aligned} \mathcal{I}_i^n(x_1, \ldots, x_n) &\leq \sum_j x_j \\ &\leq h_0\left(\sum_j x_j\right) \end{aligned}$$

- $\langle x, y \rangle = 2^x(y+1) - 1$ está dominada por h_2 porque

$$h_2(x+y) = (x+y) \cdot 2^{x+y} \geq 2^x(y+1) - 1$$

  para casi todo x, y.
- $\pi_1(x), \pi_2(x) \leq x$, dominadas por $h_0(x)$

Cada función primitiva recursiva es dominada por alguna h_n

Composición

Supongamos que

$$f(\vec{x}) = h(g_1(\vec{x}), \ldots, g_m(\vec{x}))$$

y por HI tenemos que para un $n \geq 2$ suficientemente grande (usar Proposición de pág. 35), $g_i(\vec{x}) \leq h_n(\sum \vec{x})$ para casi todo $\vec{x}$ y $h(\vec{y}) \leq h_n(\sum \vec{y})$ para casi todo $\vec{y}$.

$$\begin{array}{rcll}
f(\vec{x}) & = & h(g_1(\vec{x}), \ldots, g_m(\vec{x})) & \\
& \leq & h_n(\sum_{1 \leq i \leq m} g_i(\vec{x})) & \text{por HI sobre } h \\
& \leq & h_n(\sum_{1 \leq i \leq m} h_n(\sum \vec{x})) & \text{por HI sobre } g_i \text{ y Prop. pág. 36} \\
& = & h_n(m \cdot h_n(\sum \vec{x})) & \\
& \leq & h_n(h_2(h_n(\sum \vec{x}))) & \text{si } m \leq \sum \vec{x} \text{ y porque } h_2(z) = z \cdot 2^z \\
& \leq & h_n(h_n(h_n(\sum \vec{x}))) & \text{por } n \geq 2 \text{ y Props. págs. 35 y 36} \\
& = & h_n^{(3)}(\sum \vec{x}) & \\
& \leq & h_n^{(\sum \vec{x})}(\sum \vec{x}) & \text{si } 3 \leq \sum \vec{x} \text{ y Prop. pág. 34} \\
& = & h_{n+1}(\sum \vec{x}) &
\end{array}$$

Cada función primitiva recursiva es dominada por alguna h_n

Iteración

Supongamos que

$$f(x, y) = t^{(y)}(x)$$

y por HI para algún n, $t(x) \leq h_n(x)$ para casi todo x.

$$\begin{array}{rcll} f(x,y) & = & t^{(y)}(x) & \\ & \leq & h_n^{(y)}(x) & \text{por HI } y \text{ veces y Prop. pág. 36} \\ & \leq & h_n^{(x+y)}(x+y) & \text{por Props. págs. 34 y 36} \\ & = & h_{n+1}(x+y) & \end{array}$$

La función diagonal domina a cualquier primitiva recursiva

Teorema

La función diagonal $d(x) = h_x(x)$ domina toda función primitiva recursiva $\mathbb{N} \to \mathbb{N}$.

Demostración.

Sea f una función primitiva recursiva. Existe n tal que para casi todo x,

$$\begin{array}{rcll} f(x) & \leq & h_n(x) & \\ & \leq & h_x(x) & \text{si } x > n \text{ y por Prop. de pág. 35} \\ & = & d(x) & \end{array}$$

Entonces para casi todo x, $f(x) \leq d(x)$. □

Corolario

d no es primitiva recursiva.

Demostración.

Si lo fuera, $d(x) + 1$ también lo sería. Por teorema anterior, para casi todo x tenemos $d(x) + 1 \leq d(x)$. Absurdo. □

Recursión anidada en n variables

Sea $\leq_{\text{lex}}$ el orden lexicográfico de $\mathbb{N}^n$.

Definición

f se define a partir de $g_1, \ldots, g_n$ por recursión anidada en n variables si

$$\begin{aligned} f(\vec{x}, \vec{0}) &= g(\vec{x}) \qquad (\vec{0} \in \mathbb{N}^n) \\ f(\vec{x}, \vec{y}) &= h(\vec{x}, \vec{y}) \qquad (\vec{y} \in \mathbb{N}^n, \vec{y} \neq \vec{0}) \end{aligned}$$

donde $h(\vec{x}, \vec{y})$ es un término numérico construido por

- números naturales
- las variables $\vec{x}$ e $\vec{y}$
- las funciones $g_1, \ldots, g_m$
- el símbolo f
- los valores de f usados en la definición de $f(\vec{x}, \vec{y})$ son de la forma $f(\vec{t}, \vec{s})$ donde $\vec{s} \in \mathbb{N}^n$ tiene un valor numérico $<_{\text{lex}} \vec{y}$.

Funciones n-primitivas recursivas $\mathcal{P}_n$

Definición

La clase $\mathcal{P}_n$ de funciones *n-primitivas recursivas* es la clase de funciones más chica:

- conteniendo las funciones iniciales
- cerrada por composición
- cerrada por recursión anidada en a lo sumo n variables

$\mathcal{P}_1$ no está cerrado por recursión anidada en 2 variables

Teorema

$\mathcal{P}_1 \subsetneq \mathcal{P}_2$.

Demostración.

Las funciones $(h_n)_{n\in\mathbb{N}}$ se pueden describir así:

$$\begin{aligned} h_0(x) &= x+1 & h_n^{(0)}(x) &= x \\ h_{n+1}(x) &= h_n^{(x)}(x) & h_n^{(z+1)}(x) &= h_n(h_n^{(z)}(x)) \end{aligned}$$

Se pueden ver como una única función $h(n,z,x) = h_n^{(z)}(x)$ definidas por recursión anidada en las variables n y z (pensar h_n como $h_n^{(1)}$):

$$\begin{aligned} h(n,0,x) &= x & h(n+1,1,x) &= h(n,x,x) \\ h(0,1,x) &= x+1 & h(n,z+1,x) &= h(n,1,h(n,z,x)) \end{aligned}$$

Si h fuese primitiva recursiva, también lo sería

$$d(x) = h_x(x) = h(x,1,x)$$

y esto contradice el Teorema de pág. 44. □

Jerarquía de funciones n-recursivas

El resultado anterior se puede generalizar.

Teorema

Para todo n, $\mathcal{P}_n \subsetneq \mathcal{P}_{n+1}$.

Teoría de la Computabilidad

Clase 2

Halting problem, Reducibilidades one-one y many one, Teorema de Rice, Teorema de Isomorfismo de Myhill, cilindros

K, el Halting Problem

Definición

$K = \{x : \varphi_x(x) \downarrow\} = \{x : x \in W_x\}$.

Teorema

K es c.e.

Demostración.

$g(x) = \varphi_x(x)$ es una función parcial computable. $K = \text{dom } g$. □

Teorema

K es no es computable.

Demostración.

Si K fuera computable, entonces esta función también lo sería:

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{si } x \in K \\ 0 & \text{si no} \end{cases}$$

Pero $f \neq \varphi_x$ para todo x. Absurdo. □

K_0, otro Halting Problem

Definición

$K_0 := \{\langle x, y\rangle : \varphi_x(y) \downarrow\}$

Proposición

K_0 es c.e.

Demostración.

$g(\langle x, y\rangle) = \varphi_x(y)$ es una función parcial computable.
$K_0 = \text{dom } g$. □

Corolario

K_0 es no es computable.

Demostración.

$x \in K$ sii $\langle x, x\rangle \in K_0$. Si K_0 fuera computable, también lo sería K. □

Reducibilidades many-one y one-one

La demostración del corolario anterior sugiere un método indirecto para demostrar resultados de indecidibilidad de nuevos conjuntos: reducir K a esos conjuntos.

Definición

- A es many-one reducible a B ($A \leq_m B$) si hay una función computable f tal que

$$x \in A \qquad \text{sii} \qquad f(x) \in B.$$

- A es one-one reducible a B ($A \leq_1 B$) si $A \leq_m B$ vía una función 1-1 (i.e. inyectiva)

Informalmente, $A \leq_m B$ si A es a lo sumo tan difícil como B. Conociendo B puedo computar A. Para todo x, haciendo una sola pregunta a B, decido si $x \in A$ o $x \notin A$. La pregunta es: ¿$f(x) \in B$?

Propiedades de $\leq_1$ y $\leq_m$

Proposición

$\leq_1$ *y* $\leq_m$ *son reflexivas y transitivas.*

Demostración.

Ejercicio. □

Proposición

1. $A \leq_1 B \Rightarrow A \leq_m B$
2. $A \leq_1 B \Rightarrow \overline{A} \leq_1 \overline{B}$
3. $A \leq_m B \Rightarrow \overline{A} \leq_m \overline{B}$
4. *Si* $A \leq_m B$ *y* B *computable, entonces* A *es computable*
5. *Si* $A \leq_m B$ *y* B *c.e., entonces* A *es c.e.*

Demostración

1,2,3. Ejercicio.

4. Si $A \leq_m B$ y B computable, entonces A es computable:
   Supongamos que f es computable tal que

$$x \in A \quad \text{sii} \quad f(x) \in B.$$

   Como B es computable, la función característica de A, definida como $A(x) = B(f(x))$ también lo es.

5. Si $A \leq_m B$ y B c.e., entonces A es c.e.:
   Supongamos $A \leq_m B$ vía f. Sea R computable tal que

$$x \in B \quad \text{sii} \quad (\exists y)\ R(x, y).$$

   Entonces

$$x \in A \quad \text{sii} \quad f(x) \in B \quad \text{sii} \quad (\exists y)\ \overbrace{R(f(x), y)}^{\text{computable}}.$$

□

Más propiedades de $\leq_1$ y $\leq_m$

Proposición

1. *Existen A, B incomparables con respecto a $\leq_m$ ($A \not\leq_m B$ y $B \not\leq_m A$)*
2. *Existen A, B no computables e incomparables con respecto a $\leq_m$*

Demostración.

1. $\emptyset \not\leq_m \mathbb{N}$ y $\mathbb{N} \not\leq_m \emptyset$.
2. Considerar K y $\overline{K}$. K es c.e. pero $\overline{K}$ no es c.e. Por teorema anterior (5) $\overline{K} \not\leq_m K$. Por teorema anterior (3) $K \not\leq_m \overline{K}$.

□

Corolario

$A \leq_m B \not\Rightarrow A \leq_m \overline{B}$.

Demostración.

Tomar $A = B = K$. □

Grados m y grados 1

Definición

- $A \equiv_m B$ si $A \leq_m B$ y $B \leq_m A$
- $A \equiv_1 B$ si $A \leq_1 B$ y $B \leq_1 A$
- $\deg_1(A) = \{B : B \equiv_1 A\}$ es el grado 1 de A
- $\deg_m(A) = \{B : B \equiv_m A\}$ es el grado m de A

Semiretículo superior (upper semilattice)

Un semiretículo superior $\mathcal{L} = (L, \leq, \vee)$ es un conjunto parcialmente ordenado tal que cada par de elementos tiene un least upper bound (lub).

$c \in L$ es un lub de $a, b \in L$ si:

- $a \leq c$
- $b \leq c$
- si $d \in L$ verifica $a \leq d$ y $b \leq d$ entonces $c \leq d$

Al lub de a y b también se lo llama supremo o join. Si a y b son elementos de $\mathcal{L}$ entonces $a \vee b$ denota el lub de a y b.

$\leq_m$ forma un semiretículo superior

Definición

$A \oplus B = \{2x : x \in A\} \cup \{2x+1 : x \in B\}$.

Proposición

El orden de $\leq_m$ *forma un semiretículo superior sobre* $\deg_m$. *Es decir, cualesquiera dos grados* m *tienen un lub.*

Demostración.

Veamos que $\deg_m(A \oplus B)$ es el lub de $\deg_m(A)$ y $\deg_m(B)$ en el orden $\leq_m$.

- $A \leq_m A \oplus B$ vía $\lambda x.2x$
- $B \leq_m A \oplus B$ vía $\lambda x.2x+1$
- supongamos C tal que $A \leq_m C$ vía f y $B \leq_m C$ vía g. Entonces $A \oplus B \leq_m C$ vía h definida como

$$\begin{aligned} h(2x) &= f(x) \\ h(2x+1) &= g(x) \end{aligned}$$

□

Más conjuntos indecidibles

Teorema

$K \leq_1 \text{Tot} := \{x : \varphi_x \textit{ es una función total}\}$

Demostración.

Definir la función parcial computable

$$g(x,y) = \begin{cases} 1 & \text{si } x \in K \\ \uparrow & \text{sino} \end{cases}$$

Por el Teorema del Parámetro, existe una función 1-1 computable f tal que $\varphi_{f(x)}(y) = g(x,y)$.

- $x \in K \Rightarrow \varphi_{f(x)} = \lambda y.1 \Rightarrow f(x) \in \text{Tot}$
- $x \notin K \Rightarrow \varphi_{f(x)} = \lambda y.\uparrow \Rightarrow f(x) \notin \text{Tot}$

Entonces $x \in K$ sii $f(x) \in \text{Tot}$. □

Teorema de Rice

Definición

$A \subseteq \mathbb{N}$ es un conjunto de índices si para todo x, y, si $x \in A$ y $\varphi_x = \varphi_y$ entonces $y \in A$.

Teorema (Rice)

Si A es un conjunto de índices no trivial (i.e. $A \neq \emptyset, \mathbb{N}$) entonces A no es computable.

Demostración.

Sea b tal que $(\forall x)\ \varphi_b(x) \uparrow$. Supongamos que $b \in \overline{A}$ y probemos que $K \leq_1 A$ (del mismo modo, si $b \in A$ entonces $K \leq_1 \overline{A}$).
Como $A \neq \emptyset$, sea $a \in A$. Como A es un conjunto de índices, $\varphi_a \neq \varphi_b$. Por el TR, sea f una función computable 1-1 tal que

$$\varphi_{f(x)}(y) = \begin{cases} \varphi_a(y) & \text{si } x \in K \\ \uparrow & \text{si no} \end{cases}$$

- $x \in K \Rightarrow \varphi_{f(x)} = \varphi_a \Rightarrow f(x) \in A$
- $x \in \overline{K} \Rightarrow \varphi_{f(x)} = \varphi_b \Rightarrow f(x) \in \overline{A}$

Permutaciones computables

Definición

- Una permutación computable es una función $\mathbb{N} \to \mathbb{N}$ computable biyectiva.
- Una propiedad es computablemente invariante si es invariante bajo cualquier permutación computable.

Ejemplos de propiedades computablemente invariantes:

- A es c.e.
- $\#A = n$
- A es computable

Ejemplos de propiedades no computablemente invariantes:

- $2 \in A$
- A contiene solo números pares
- A es un conjunto de índices

Definición

A es computablemente isomorfo a B ($A \equiv B$) si hay una permutación computable p tal que $A = p(B)$. Es decir, $x \in A$ sii $p(x) \in B$.

Teorema de Isomorfismo de Myhill

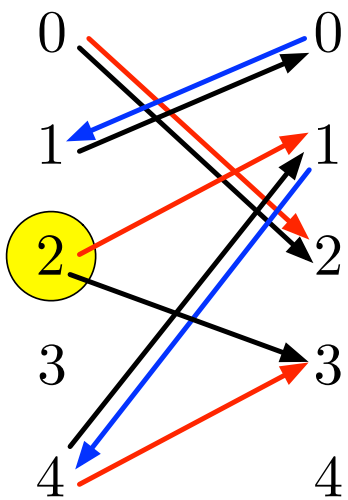
Teorema

$A \equiv B$ *sii* $A \equiv_1 B$.

Demostración

$\Rightarrow$ Trivial.

$\Leftarrow$ Sea $A \leq_1 B$ vía f y $B \leq_1 A$ vía g. Definimos permutación computable h por etapas. $h = \bigcup_s h_s$. $h_0 = \emptyset$. Cada h_s es finita, 1-1, y cumple $x \in A$ sii $h_s(x) \in B$ para $x \in \text{dom } h_s$.



- Paso 1: $f(0) = 2 \quad h_1(0) = 2$
- Paso 2: $g(0) = 1 \quad h_2^{-1}(0) = 1$ o sea $h_2(1) = 0$
- Paso 3: $h_3(1)$ ya está definido: $h_3(1) = h_2(1) = 0$
- Paso 4: $g(1) = 4 \quad h_4^{-1}(1) = 4$
- Paso 5: $f(2) = 1 \quad f(4) = 3 \quad h_5(2) = f \circ h_1^{-1} \circ f(1) = 3$

Formalizamos la demostración de $\Leftarrow$ Supongamos definida h_s y definamos h_{s+1} de esta manera:

- si $s+1 = 2x+1$, definir $h_{s+1}(x)$. Si $h_s(x)$ está definido, no hacer nada. Si no, enumerar

$$\{f(x), f(h_s^{-1}f(x)), \ldots, f(h_s^{-1}f)^n(x), \ldots\}$$

  hasta encontrar el primer $y \notin \text{rg } h_s$. Definir $h_{s+1}(x) = y$. Observar que h_{s+1} es finita, 1-1 y $x \in A$ sii $y \in B$.
- si $s+1 = 2x+2$, definir $h^{-1}(x)$ de forma similar, cambiando
  - $f \longleftrightarrow g$
  - $h_s \longleftrightarrow h_s^{-1}$
  - dom $h_s \longleftrightarrow$ rg h_s

□

Vamos a clasificar conjuntos módulo permutaciones computable (lo mismo pasa en algebra: clasifica estructuras módulo isomorfismo).

Cilindros

Definición

A es un cilindro si $A \equiv B \times \mathbb{N}$ para algún B.

Teorema

1. $A \leq_1 A \times \mathbb{N}$
2. $A \times \mathbb{N} \leq_m A$ *(por lo tanto* $A \equiv_m A \times \mathbb{N}$*)*
3. A *es un cilindro sii* $(\forall B)\ [B \leq_m A \Rightarrow B \leq_1 A]$
4. $A \leq_m B$ *sii* $A \times \mathbb{N} \leq_1 B \times \mathbb{N}$

Demostración

1. Ejercicio.
2. Ejercicio.

3. A es un cilindro sii $(\forall B)\ [B \leq_m A \Rightarrow B \leq_1 A]$:

   $\Rightarrow$ Sea $A \equiv C \times \mathbb{N}$ y supongamos que $B \leq_m A$. Entonces $B \leq_m C$ vía alguna función computable f. Sea $g(x) = \langle f(x), x \rangle$.

   - g es 1-1
   - $x \in B$ sii $g(x) \in C \times \mathbb{N}$

   Entonces $B \leq_1 C \times \mathbb{N} \equiv_1 A$.

   $\Leftarrow$ De 2 sabemos que $A \times \mathbb{N} \equiv_m A$. Por hipótesis, $A \times \mathbb{N} \leq_1 A$. De 1 sabemos $A \leq_1 A \times \mathbb{N}$. Luego $A \equiv_1 A \times \mathbb{N}$. Por Teorema de Isomorfismo, $A \equiv A \times \mathbb{N}$, de modo que A es un cilindro.

4. $A \leq_m B$ sii $A \times \mathbb{N} \leq_1 B \times \mathbb{N}$:

   $\Rightarrow$ Supongamos $A \leq_m B$. Entonces

   $$A \times \mathbb{N} \leq_m A \leq_m B \leq_1 B \times \mathbb{N}.$$

   Luego $A \times \mathbb{N} \leq_m B \times \mathbb{N}$. Por 3 (y porque $B \times \mathbb{N}$ es trivialmente un cilindro), $A \times \mathbb{N} \leq_1 B \times \mathbb{N}$.

   $\Leftarrow$ Supongamos que $A \times \mathbb{N} \leq_1 B \times \mathbb{N}$. Tenemos

   $$A \leq_1 A \times \mathbb{N} \leq_1 B \times \mathbb{N} \leq_m B.$$

Teoría de la Computabilidad

Clase 3

Conjuntos 1-completos, conjuntos productivos y creativos

Conjuntos 1-completos

Definición

Un conjunto c.e. A es r-completo ($r \in \{1, m\}$) si $W_e \leq_r A$ para todo e.

Recordar

$$K_0 := \{\langle x, y\rangle : \varphi_x(y) \downarrow\}.$$

Proposición

K_0 es 1-completo.

Demostración.

$x \in W_e$ sii $\langle e, x\rangle \in K_0$. □

Como K es c.e., tenemos $K \leq_1 K_0$ (esto ya lo habíamos probado por otro lado antes).

$K \equiv_1 K_0$

Ya vimos $K \leq_1 K_0$.

Proposición

$K_0 \leq_1 K$.

Demostración.

Misma idea que demostración de $K \leq_1$ Tot funciona.
Definir la función parcial computable

$$g(x,z) = \begin{cases} 1 & \text{si } x \in K_0 \\ \uparrow & \text{sino} \end{cases}$$

Por el Teorema del Parámetro, existe una función 1-1 computable f tal que $\varphi_{f(x)}(z) = g(x,z)$.

- $x \in K_0 \Rightarrow \varphi_{f(x)} = \lambda z.1 \Rightarrow f(x) \in K$
- $x \notin K_0 \Rightarrow \varphi_{f(x)} = \lambda z.\uparrow \Rightarrow f(x) \notin K$

Entonces $x \in K_0$ sii $f(x) \in K$. □

K, K_0 y K_1 son 1-completos

Definición

$K_1 := \{x : W_x \neq \emptyset\}$.

Proposición

$K \leq_1 K_1$.

Demostración.

Ejercicio. □

Como K_1 es c.e. y K_0 es 1-completo, $K_1 \leq_1 K_0$.
Ya habíamos visto $K \equiv_1 K_0$ y $K \leq_1 K_1$.
Entonces

$$K \equiv_1 K_0 \equiv_1 K_1$$

Corolario

K, K_0 y K_1 son 1-completos.

Conjuntos productivos y creativos

Una forma de ver que $\overline{K}$ no es c.e. es el hecho de que dado el índice de un conjunto W c.e. incluido en $\overline{K}$ uno puede obtener un número que está en $\overline{K}$ pero no en W:

$$W_x \subseteq \overline{K} \Rightarrow x \in \overline{K} \setminus W_x.$$

Definición

- Un conjunto P es productivo si existe una función parcial computable ψ, llamada función productiva para P, tal que

$$(\forall x)[W_x \subseteq P \Rightarrow [\psi(x)\downarrow \ \wedge \psi(x) \in P \setminus W_x]].$$

- Un conjunto C es creativo si es c.e. y $\overline{C}$ es productivo.

Por ejemplo, K es creativo vía la identidad $\psi(x) = x$.

Idea: un conjunto C es creativo si es "efectivamente" no computable: para cualquier candidato W_x para $\overline{C}$, $\psi(x)$ es un contraejemplo efectivo.

Conjuntos productivos

Proposición

Cualquier conjunto productivo es productivo vía una función total.

Demostración.

Sea P productivo vía ψ. Por el TP, sea g computable tal que

$$\varphi_{g(x)}(y) = \begin{cases} 1 & \text{si } \psi(x)\downarrow \wedge \varphi_x(y)\downarrow \\ \uparrow & \text{si no} \end{cases}$$

Entonces

$$W_{g(x)} = \begin{cases} W_x & \text{si } \psi(x)\downarrow \\ \emptyset & \text{si no} \end{cases}$$

Definir $q(x)$ como $\psi(x)$ o $\psi(g(x))$, la que converja primero.

- $W_x \subseteq P \Rightarrow \psi(x)\downarrow \Rightarrow W_{g(x)} = W_x \Rightarrow P$ es productivo vía q
- q es total:
  - $\psi(x)\downarrow \Rightarrow q(x)\downarrow$
  - $\psi(x)\uparrow \Rightarrow W_{g(x)} = \emptyset \subseteq P \Rightarrow \psi(g(x))\downarrow \Rightarrow q(x)\downarrow$

Conjuntos productivos

Proposición

Cualquier conjunto productivo es productivo vía una función 1-1 total.

Demostración

Sea P productivo vía q total. Veamos cómo convertir q en p 1-1. Sea h computable tal que

$$\varphi_{h(x)}(y) = \begin{cases} 1 & \text{si } \varphi_x(y)\downarrow \vee y = q(x) \\ \uparrow & \text{si no} \end{cases}$$

Tenemos $W_{h(x)} = W_x \cup \{q(x)\}$. Observar que

$$W_x \subseteq P \Rightarrow W_{h(x)} \subseteq P \Rightarrow (\forall n)\ W_{h^{(n)}(x)} \subseteq P$$

y

$$W_{h^{(n)}(x)} = W_x \cup \{q(x), qh(x), \ldots, qh^{(n-1)}(x)\}$$

Transformamos q total en p total y 1-1. Definimos $p(0), p(1), \ldots$ en ese orden.

Definir $p(0) = q(0)$. Para $x > 0$, computar $p(x)$ así: enumerar

$$\{q(x), qh(x), qh^{(2)}(x), \ldots\}$$

hasta que ocurre alguno de estos dos casos:

1. algún $qh^{(n)}(x) \notin \{p(0), \ldots, p(x-1)\}$. Definir $p(x) = qh^{(n)}(x)$.
   - supongamos $W_x \subseteq P$. Como $W_{h^{(n)}(x)} \subseteq P$ entonces
   $$p(x) = qh^{(n)}(x) \in P \setminus W_{h^{(n)}(x)} \subseteq P \setminus W_x$$
2. existe $m < n$ tal que $qh^{(m)}(x) = qh^{(n)}(x)$
   - Veamos que $W_x \not\subseteq P$: Supongamos $W_x \subseteq P$
     $W_x \subseteq P \Rightarrow$ (mismo argumento de arriba) $\Rightarrow$
     $$(\forall k)\ qh^{(k)}(x) \in P \setminus W_{h^{(k)}(x)}$$
     En particular, $qh^{(n)}(x) \in P \setminus W_{h^{(n)}(x)}$.
     Pero $qh^{(m)}(x) \in W_{h^{(n)}(x)}$ por definición. Absurdo.
   - Entonces da lo mismo cómo definir $p(x)$. Basta con que sea un número adecuado para que p no deje de ser 1-1. Por ejemplo, $p(x) = \text{mín}\{y : y \notin \{p(0), \ldots, p(x-1)\}\}$.

Conjuntos Productivos

Teorema

Si P es productivo entonces

1. *P no es c.e.*
2. *P contiene un conjunto c.e. infinito W*
3. *si $P \leq_m A$ entonces A es productivo*

Demostración

1. Inmediato (ejercicio).
2. Sea P productivo vía p total y 1-1. Sea $W_n = \emptyset$. Por TP sea h 1-1 tal que $W_{h(x)} = W_x \cup \{p(x)\}$. Definir

$$W = \{p(n), ph(n), ph^{(2)}(n), \dots\}$$

- W es c.e.
- W es infinito porque p y h son 1-1
- $W \subseteq P$ porque P es productivo vía p (ver observación p. 72)

3. Sea $P \leq_m A$ vía f.
   Sea P productivo vía p total. Sea g computable tal que

$$\varphi_{g(x)}(y) = \begin{cases} 0 & \text{si } f(y) \in W_x \\ \uparrow & \text{si no} \end{cases}$$

   Tenemos

$$W_{g(x)} = f^{-1}(W_x) = \{y : f(y) \in W_x\}$$

   Veamos que A es productivo vía fpg:

   - $W_x \subseteq A \Rightarrow f^{-1}(W_x) \subseteq f^{-1}(A)$
   - como $x \in P \Leftarrow f(x) \in A$, entonces $f^{-1}(A) \subseteq P$
   - entonces $W_{g(x)} \subseteq P$
   - luego $pg(x) \in P \setminus W_{g(x)}$
   - veamos que $fpg(x) \in A \setminus W_x$:
     - $pg(x) \in P \Rightarrow fpg(x) \in A$
     - $pg(x) \notin W_{g(x)} \Rightarrow fpg(x) \notin W_x$

Productivo implica $\geq_1 \overline{K}$

Teorema

Si P es productivo entonces $\overline{K} \leq_1 P$.

Demostración.

Sea P productivo vía p total y 1-1. Por TP definir f computable tal que

$$W_{f(x,y)} = \begin{cases} \{p(x)\} & \text{si } y \in K \\ \emptyset & \text{si no} \end{cases}$$

Por TR con parámetros, existe una función computable n 1-1 tal que

$$W_{n(y)} = W_{f(n(y),y)} = \begin{cases} \{pn(y)\} & \text{si } y \in K \\ \emptyset & \text{si no} \end{cases}$$

- $y \in K \Rightarrow W_{n(y)} = \{pn(y)\} \Rightarrow W_{n(y)} \not\subseteq P \Rightarrow pn(y) \in \overline{P}$
- $y \in \overline{K} \Rightarrow W_{n(y)} = \emptyset \Rightarrow W_{n(y)} \subseteq P \Rightarrow pn(y) \in P$

Entonces $\overline{K} \leq_1 P$ vía $\lambda y.pn(y)$. □

Creativo implica 1-completo

Corolario

Si C es creativo, entonces C es 1-completo.

Demostración.

C es c.e. y $\overline{C}$ es productivo. Entonces por teorema anterior $\overline{K} \leq_1 \overline{C}$. Entonces $K \leq_1 C$.

Como K es 1-completo, C es 1-completo.

□

Equivalencias

Corolario

Los siguientes son equivalentes:

1. P *es productivo*
2. $\overline{K} \leq_1 P$
3. $\overline{K} \leq_m P$

Demostración.

$1 \Rightarrow 2$ sale de teorema de pág. 76.

$2 \Rightarrow 3$ es trivial.

$3 \Rightarrow 1$ sale del hecho de que $\overline{K}$ es productivo y teorema de pág. 74. □

Corolario

Los siguientes son equivalentes:

1. C *es creativo*
2. C *es 1-completo*
3. C *es m-completo*

Teoría de la Computabilidad

Clase 4

Reducibilidad tt, cómputos con oráculo, principio del uso

Tablas de verdad

Definición

- Una tabla de verdad k-aria es una función $\{0,1\}^k \to \{0,1\}$.
- Un par $\langle \vec{x}, \alpha \rangle$, $\vec{x} \in \mathbb{N}^k$ y α una tabla de verdad k-aria es una tt-condición
- Una tt-condición $\langle (x_1, \ldots, x_n), \alpha \rangle$ es satisfecha por A si $\alpha(A(x_1), \ldots, A(x_k)) = 1$.

Por ejemplo,

$$\alpha = \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

- $\langle (n, m), \alpha \rangle$ es satisfecha por A sii $n \in A$ y $m \in A$

Las tablas de verdad son objetos finitos. Las tt-condiciones también.
Suponemos una codificación de las tt-condiciones con $\mathbb{N}$.

Reducibilidad tabla de verdad (*truth table*)

Sabemos que K, $\overline{K}$ y $K \oplus \overline{K}$ están en clases m distintas. Pero intuitivamente son interreducibles.

Definición

A es *truth table* reducible a B ($A \leq_{tt} B$) si hay una función computable f tal que para todo x

$x \in A$ sii la tt-condición (codificada por) $f(x)$ es satisfecha por B

Por ejemplo, para todo A:

- $A \leq_{tt} \overline{A}$ vía la función $f(x)$ que codifica la tt-condición $\langle x, \alpha \rangle$, donde $\alpha(0) = 1$, $\alpha(1) = 0$.
- $\overline{A} \leq_{tt} A \oplus \overline{A}$ vía la función $f(x)$ que codifica la tt-condición $\langle 2x + 1, \alpha \rangle$, donde $\alpha(0) = 0$, $\alpha(1) = 1$.
- $A \oplus \overline{A} \leq_{tt} A$ vía $f(2x) = \langle x, \alpha \rangle$, donde $\alpha(0) = 0$, $\alpha(1) = 1$; $f(2x + 1) = \langle x, \alpha \rangle$, donde $\alpha(0) = 1$, $\alpha(1) = 0$.

Propiedades de $\leq_{tt}$

Definición

$A \equiv_{tt} B$ si $A \leq_{tt} B$ y $B \leq_{tt} A$.

Definición

Un conjunto c.e. A es tt-completo si $W_e \leq_{tt} A$ para todo e.

Teorema

1. $A \leq_m B \Rightarrow A \leq_{tt} B$
2. $A \leq_{tt} \overline{A}$ *(por lo tanto* $A \equiv_{tt} \overline{A}$*)*
3. $\leq_{tt}$ *forma un semiretículo superior*
4. *Si* $A \leq_{tt} B$ *y* B *computable, entonces* A *es computable*
5. *Si* A *es computable entonces* $(\forall B)\ A \leq_{tt} B$

Demostración.

Ejercicio. □

$\leq_{tt}$ tampoco alcanza

Consideremos

$$C = \{x : \varphi_x(x)\downarrow \text{ y la tt-condición } \varphi_x(x) \text{ es satisfecha por } K\}$$

Intuitivamente C es reducible a K, es decir

$$C \leq_{\text{intuitivamente}} K,$$

pero...

Proposición

$C \not\leq_{tt} K$.

Demostración.

Supongamos $C \leq_{tt} K$. Entonces $\overline{C} \leq_{tt} K$ vía φ_e total.

$e \in \overline{C}$	sii	$\varphi_e(e)$ es satisfecho por K	porque $\overline{C} \leq_{tt} K$
	sii	$e \in C$	por definición de C

□

Máquinas de Turing

- una cinta
  - dividida en celdas
  - infinita en ambas direcciones
  - cada celda contiene un símbolo de un alfabeto dado Σ.
    - $* \in \Sigma$
    - $L, R \notin \Sigma$

      $*$ representa el blanco en una celda

      L y R son símbolos reservados (representarán acciones que puede realizar la cabeza)
- una cabeza
  - lee y escribe un símbolo a la vez
  - se mueve una posición a la izquierda o una posición a la derecha
- una tabla finita de instrucciones
  - dice qué hacer en cada paso

Máquinas de Turing

Una máquina de Turing (determinística) es una tupla

$$(\Sigma, Q, \delta, q_0, q_f)$$

donde

- Σ (finito) es el conjunto símbolos (Fijamos $\Sigma = \{*, 1\}$)
- Q (finito) es el conjunto de estados
  - tiene dos estados distinguidos:
    - $q_0 \in Q$ es el estado inicial
    - $q_f \in Q$ es el estado final
- $\delta : Q \times \Sigma \to Q \times \Sigma \times \{R, L\}$ es la tabla de instrucciones o programa
  - va a ser finita porque Σ y Q lo son
  - $\delta(q, s) = (q', s', d)$ se interpreta como

    Si la máquina está en el estado q leyendo en la cinta el símbolo s, entonces escribe s' sobre s, se mueve hacia la izquierda si $d = L$ o derecha si $d = R$, y pasa al estado q'

Máquinas de Turing con oráculo

Es una Máquina de Turing con una cinta adicional que es solo de lectura.

- la nueva cinta se llama cinta oracular (fijamos $\Sigma' = \{*, 0, 1\}$)
- la vieja cinta se llama cinta de trabajo y funciona igual que antes (fijamos $\Sigma = \{*, 1\}$)

La nueva tabla de instrucciones o programa de una máquina de Turing oracular es una función

$$\delta : Q \times \Sigma \times \Sigma' \to Q \times \Sigma \times \{R, L\}$$

$\delta(q, s, x) = (q', s', d)$ se interpreta como

Si la máquina está en el estado q leyendo en la cinta el símbolo s y en la cinta oracular está el símbolo x, entonces escribe s' sobre s en la cinta de trabajo, pasa al estado q' y se mueve hacia la izquierda si $d = L$ o derecha si $d = R$ en ambas cintas

Enumeración de máquinas con oráculo

Igual que antes, los programas se pueden enumerar: Φ_e es el e-ésimo programa, o la e-ésima máquina de Turing oracular.

Φ_e necesita dos entradas

- una entrada finita $x_1, \ldots, x_n$ para la cinta de trabajo
- una entrada posiblemente infinita $A \subseteq \mathbb{N}$ para la cinta oracular

Se suele escribir $\Phi_e^A(x_1, \ldots, x_n)$ para referirse a la e-ésima máquina de Turing oracular con oráculo A y entrada $x_1, \ldots, x_n$.

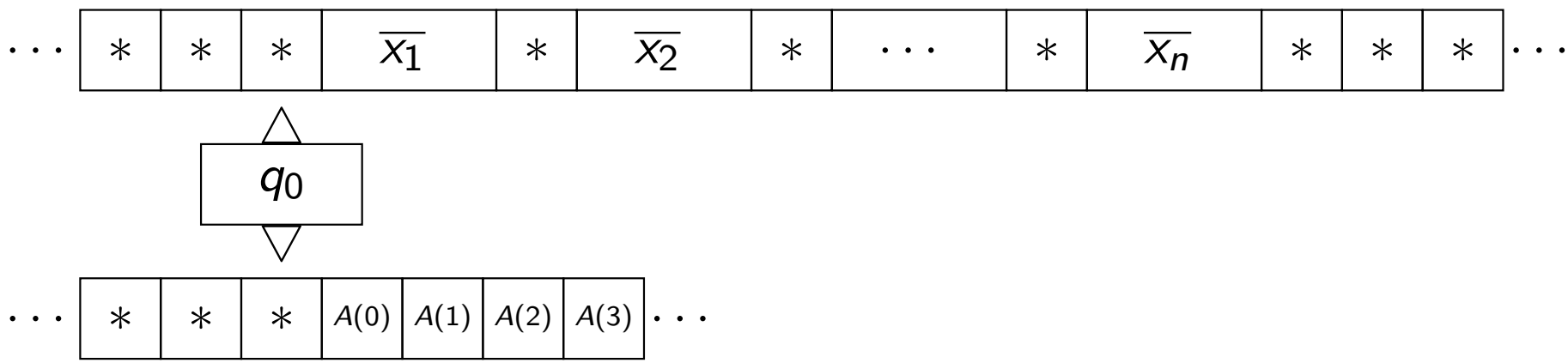
Configuración inicial

Se codifica

- la entrada $x_1, \ldots, x_n$ como antes (sobre alfabeto $\Sigma = \{*, 1\}$)
- el oráculo A como una lista de ceros y unos $A(0), A(1), A(2) \ldots$ (sobre alfabeto $\Sigma' = \{*, 0, 1\}$). Como siempre

$$A(n) = \begin{cases} 1 & \text{si } n \in A \\ 0 & \text{si no} \end{cases}$$

La configuración inicial es esta:



(con los números x_i representados en unario)

Cómputos relativos a oráculos

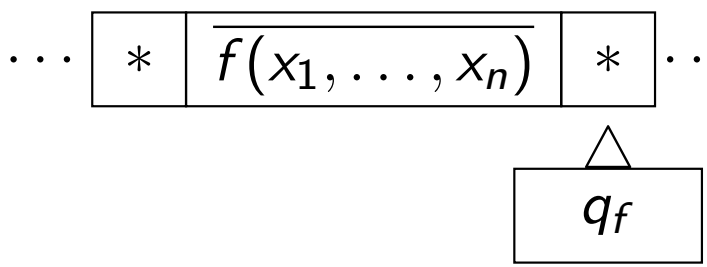
Definición

Una función parcial f es Turing computable en A (o A-computable), notado $f \leq_T A$, si existe un programa e tal que

$$f(x_1, \ldots, x_n) = \Phi_e^A(x_1, \ldots, x_n),$$

es decir:

- si $f(x_1, \ldots, x_n) \downarrow$ entonces $\Phi_e^A(x_1, \ldots, x_n)$, partiendo de la configuración inicial y siguiendo sus instrucciones, llega a una configuración final de la forma



(quizá algo más en la cinta de trabajo; recordar que la cinta oracular es solo de lectura y se mueve a la par de la cinta de trabajo)

- si $f(x_1, \ldots, x_n) \uparrow$ entonces $\Phi_e^A(x_1, \ldots, x_n)$ nunca llega al estado final.

El *uso*

En cualquier cómputo $\Phi_e^A(x_1, \ldots, x_n) \downarrow$ la máquina e solo mira una cantidad finta de elementos del oráculo. Sea $y+1$ el máximo número de celdas no $*$ consultadas durante ese cómputo. Decimos que los elementos $z \leq y$ son usados en el cómputo.

Definición

- Notamos $\Phi_{e,s}^A(x) = y$ para $x, y, e < s, s > 0$, si $\Phi_e^A(x) = y$ en $< s$ pasos y usando solo números $z < s$ del oráculo.
- Notamos $\Phi_{e,s}^A(x) \downarrow$ para $x, y, e < s, s > 0$, si $\Phi_{e,s}^A(x) = y$ para algún y.
- El uso $u(A, e, x, s)$ se define como

$$u(A, e, x, s) = \begin{cases} 1+\text{el máximo número usado} \\ \text{en el cómputo } \Phi_{e,s}^A(x) & \text{si } \Phi_{e,s}^A(x) \downarrow \\ 0 & \text{si no} \end{cases}$$

- Para $\sigma \in \{0,1\}^*$, notamos $\Phi_{e,s}^\sigma(x) = y$, si $\Phi_{e,s}^A(x) = y$ para algún $A \supset \sigma$, y solo elementos menores a $|\sigma|$ son usados en este último cómputo.

Propiedades del uso

Si $\Phi^A_{e,s}(x) = y$ entonces

- $x, y, e < s$ y $u(A, e, x, s) \leq s$
- $\Phi^\sigma_{e,s}(x) = y$ donde $\sigma = A \upharpoonright u(A, e, x, s)$
- $(\forall t \geq s)\ [\Phi^A_{e,t}(x) = y \wedge u(A, e, x, s) = u(A, e, x, t)]$

Habíamos definido $u(A, e, x, s)$.

Definición

$u(A, e, x)$ es

$$u(A, e, x) = \begin{cases} u(A, e, x, s) & \text{si } \Phi^A_{e,s}(x) \downarrow \text{ para algún } s \\ \uparrow & \text{si no} \end{cases}$$

El principio del uso (*use principle*)

Teorema

1. $\Phi_e^A(x) = y \Rightarrow (\exists s)(\exists \sigma \subset A)\ \Phi_{e,s}^{\sigma}(x) = y$
2. $\Phi_{e,s}^{\sigma}(x) = y \Rightarrow (\forall t \geq s)(\forall \tau \supset \sigma)\ \Phi_{e,t}^{\tau}(x) = y$
3. $\Phi_e^{\sigma}(x) = y \Rightarrow (\forall A \supset \sigma)\ \Phi_e^A(x) = y$

Corolario

Si $\Phi_{e,s}^A(x) = y$ *y* $B \upharpoonright u = A \upharpoonright u$ *entonces* $\Phi_{e,s}^B(x) = y$, *donde* $u = u(A, e, x, s)$.

Propiedades de Φ_e^A y Φ_e^σ

Teorema

1. $\{\langle e, \sigma, x, s\rangle : \Phi^\sigma_{e,s}(x) \downarrow\}$ *es computable*
2. $\{\langle e, \sigma, x\rangle : \Phi^\sigma_{e}(x) \downarrow\}$ *es c.e.*

Teorema

Son funciones parciales A-computables:

1. $\lambda e, s, x.\Phi^A_{e,s}(x)$ *(esta no es total)*
2. $\lambda e, s, x. \begin{cases} 1 & \text{si } \Phi^A_{e,s}(x) \downarrow \\ 0 & \text{si no} \end{cases}$ *(esta es total)*

Teoría de la Computabilidad

Clase 5

Teorema del Salto, Lema del Límite, grados Turing

Relativizaciones de resultados conocidos

Teorema (Teorema del Parámetro Relativizado - TPR)

Para cada $m, n \geq 1$ existe una función primitiva recursiva 1-1 s_n^m de $m+1$ variables tal que para todo $A \subseteq \mathbb{N}$ y para todo $y_1, \ldots, y_m, z_1, \ldots, z_n \in \mathbb{N}$,

$$\Phi^A_{s_n^m(x,y_1,\ldots,y_m)}(z_1, \ldots, z_n) = \Phi^A_x(y_1, \ldots, y_m, z_1, \ldots, z_n)$$

Teorema (Teorema de la Recursión Relativizado - TRR)

Para cualquier función A-computable $f : \mathbb{N}^2 \to \mathbb{N}$ existe una función computable n tal que

$$\Phi^A_{n(y)} = \Phi^A_{f(n(y),y)}.$$

Es más, la definición de n no depende de A. Es decir, si $f(x, y) = \Phi^A_e(x, y)$ entonces n se puede definir uniformemente en e.

Conjuntos A-computables, A-c.e. y Σ_1^A

Definición

- $W_e^A = \text{dom } \Phi_e^A$
- $W_{e,s}^A = \text{dom } \Phi_{e,s}^A$
- $W_e^\sigma = \text{dom } \Phi_e^\sigma$
- $W_{e,s}^\sigma = \text{dom } \Phi_{e,s}^\sigma$

Definición

- B es computable en A (o A-computable), notado $B \leq_T A$, si $B = \Phi_e^A$ para algún e.
  - notamos $A \equiv_T B$ cuando $A \leq_T B$ y $B \leq_T A$
  - notamos $A <_T B$ cuando $A \leq_T B$ y $B \not\leq_T A$
  - notamos $\deg_T(A) = \{B : B \equiv_T A\}$
- B es c.e. en A (o A-c.e.) si $B = W_e^A$ para algún e
- B es de la forma Σ_1^A (abreviado $B \in \Sigma_1^A$) cuando $B = \{x : (\exists \vec{y})\ R^A(x, \vec{y})\}$ para algún predicado A-computable R^A.

Convención

Definimos $\Phi^A_{e,s}$ de modo que

$$\Phi^A_{e,s}(x)\downarrow \Rightarrow x, e < s$$

Si B es A-c.e., entonces $B = \text{dom } \Phi^A_e$ para algún e. Tenemos una A-enumeración natural de B. En general se la nota $(B_s)_{s\in\mathbb{N}}$ y está definida como

$$B_s = \text{dom } \Phi^A_{e,s} \subseteq \{0, \ldots, s-1\}.$$

En realidad, B_s abrevia una función A-computable $b : \mathbb{N} \to \mathbb{N}$ tal que

$$b(s) = \text{codificación de } B_s \text{ usando } \mathbb{N}$$

B_s satisface

- para todo x, $B(x) = \lim_s B_s(x)$
- $(\forall n)(\exists s)\ B \upharpoonright n = B_s \upharpoonright n$

Equivalencias entre A-c.e. y Σ_1^A

Teorema

Son equivalentes:

1. *B es A-c.e.*
2. *B es Σ_1^A*
3. *$B = \emptyset$ o B es el rango de alguna función A-computable*

Demostración

$1 \Rightarrow 2$ Sea $B = \text{dom } \Phi_e^A$ entonces $B = \{x : (\exists s)\ \Phi_{e,s}^A(x)\downarrow\}$.
Recordar que

$$\lambda e, s, x. \begin{cases} 1 & \text{si } \Phi_{e,s}^A(x)\downarrow \\ 0 & \text{si no} \end{cases}$$

es A-computable.

$2 \Rightarrow 3$ Sea $b \in B$ y sea $B = \{x : (\exists \vec{y})\ R^A(x, \vec{y})\}$. Supongamos R^A es $(n+1)$-ario. Definimos la función A-computable

$$f(\langle x, y_1, \ldots, y_n \rangle) = \begin{cases} x & \text{si } R^A(x, y_1, \ldots, y_n) \\ b & \text{si no} \end{cases}$$

- veamos que $x \in B \Rightarrow (\exists z)\ f(z) = x$:
  $x \in B \Rightarrow (\exists \vec{y})\ R^A(x, \vec{y}) \Rightarrow f(\langle x, \vec{y} \rangle) = x$
- veamos que $(\forall z)\ f(z) \in B$:
  Sea $z = \langle x, \vec{y} \rangle$.
  - $\neg R^A(x, \vec{y}) \Rightarrow f(z) = b \in B$
  - $R^A(x, \vec{y}) \Rightarrow x \in B$. Luego $f(z) = f(\langle x, \vec{y} \rangle) = x \in B$

$3 \Rightarrow 1$ Sea e tal que $\Phi_e^A(\mathbb{N}) = B$. Definir la función parcial

$$f(y) = \begin{cases} 0 & \text{si } (\exists x, s)\ \Phi_{e,s}^A(x) = y \\ \uparrow & \text{si no} \end{cases}$$

f es parcial A-computable y dom $f = B$.

Relativizaciones de resultados anteriores

Teorema

Si B es A-computable entonces $\overline{B}$ también.

Teorema

Si B es A-computable entonces B es A-c.e.

Teorema (Complementación)

Si B y $\overline{B}$ son A-c.e. entonces B es A-computable.

El salto

Recordar que

$$\begin{aligned} K &= \{x : \varphi_x(x)\downarrow\} = \{x : x \in W_x\} \\ K_0 &= \{\langle x, y\rangle : \varphi_x(y)\downarrow\} \end{aligned}$$

Se pueden relativizar

$$\begin{aligned} K^A &= \{x : \Phi_x^A(x)\downarrow\} = \{x : x \in W_x^A\} \\ K_0^A &= \{\langle x, y\rangle : \Phi_x^A(y)\downarrow\} \end{aligned}$$

Definición

El salto (*jump*) de A, notado A', se define como

$$A' = K^A = \{x : \Phi_x^A(x)\downarrow\} = \{x : x \in W_x^A\}$$

$A^{(n)}$ es el *n*-esimo salto de A. Se define como $A^{(0)} = A$ y $A^{(n+1)} = (A^{(n)})'$.

Observar que $\emptyset' = K$.

Teorema

$K^A \equiv_1 K_0^A$

El Teorema del Salto (TS)

Teorema

1. A' *es* A*-c.e.*
2. $A' \not\leq_T A$
3. B *es* A*-c.e. sii* $B \leq_1 A'$
4. *si* A *es* B*-c.e. y* $B \leq_T C$ *entonces* A *es* C*-c.e.*
5. $B \leq_T A$ *sii* $B' \leq_1 A'$
6. *si* $B \equiv_T A$ *entonces* $B' \equiv_1 A'$
7. A *es* B*-c.e. sii* A *es* $\overline{B}$*-c.e.*

Demostración del Teorema del Salto

1. A' es A-c.e.:
   La función parcial $f(x) = \Phi_x^A(x)$ es A-computable y dom $f = A'$. Entonces A' es A-c.e.
2. $A' \not\leq_T A$:
   Supongamos que A' es A-computable. Entonces $\overline{A'}$ es A-c.e. Luego existe e tal que
   $$W_e^A = \overline{A'} = \{x : x \notin W_x^A\}.$$
   Entonces $e \in W_e^A$ sii $e \notin W_e^A$. Absurdo.
3. B es A-c.e. sii $B \leq_1 A'$:
   - $\Rightarrow$ Existe e tal que $B = W_e^A = \text{dom } \Phi_e^A$. Entonces $x \in B$ sii $\Phi_e^A(x) \downarrow$ sii $\langle x, e\rangle \in K_0^A$. Como $\lambda x.\langle x, e\rangle$ es 1-1, entonces $B \leq_1 K_0^A$. Además, $K_0^A \equiv_1 K^A = A'$.
   - $\Leftarrow$ Supongamos f computable tal que $x \in B$ sii $f(x) \in A'$. Entonces $x \in B$ sii $\Phi_{f(x)}^A(f(x)) \downarrow$. La función parcial $\lambda x.\Phi_{f(x)}^A(f(x))$ es A-computable, entonces su dominio (que es B) es A-c.e.

Demostración del Teorema del Salto

4. si A es B-c.e. y $B \leq_T C$ entonces A es C-c.e.:
   Por Teorema pág. 97, si $A \neq \emptyset$, A es el rango de una función f tal que $f \leq_T B$. Como $B \leq_T C$ entonces $f \leq_T C$. Luego A es el rango de una función C-computable.
5. $B \leq_T A$ sii $B' \leq_1 A'$:
   - $\Rightarrow$ Por TS1 B' es B-c.e. Como $B \leq_T A$, por TS4, B' es A-c.e. Por TS3, $B' \leq_1 A'$.
   - $\Leftarrow$ $B, \overline{B} \leq_1 A'$ (ejercicio). Por TS3 B y $\overline{B}$ son A-c.e. Por Teorema de Complementación (pág. 101), $B \leq_T A$.
6. si $B \equiv_T A$ entonces $B' \equiv_1 A'$:
   Directo de TS5.
7. A es B-c.e. sii A es $\overline{B}$-c.e.
   Directo de TS4.

Lema del límite

Teorema

Para toda función total f, $f \leq_T A'$ sii hay una función A-computable g de dos variables tal que $f(x) = \lim_s g(x,s)$.

Demostración

$\Rightarrow$ Supongamos que $f = \Phi_e^{A'}$. Por TS1 A' es A-c.e. Sea $(A'_s)_{s\in\mathbb{N}}$ una enumeración A-computable de A'. Definir:

$$g(x,s) = \begin{cases} \Phi_{e,s}^{A'_s}(x) & \text{si } \Phi_{e,s}^{A'_s}(x)\downarrow \\ 0 & \text{si no} \end{cases}$$

Claramente $g \leq_T A$. Sea t suficientemente grande tal que

$$(\exists z \leq t)\ [\Phi_{e,t}^{A'_t\restriction z}(x)\downarrow\ \wedge\ A'_t \restriction z = A' \restriction z]$$

Entonces para todo $s \geq t$ tenemos

$$\Phi_{e,s}^{A'_s\restriction z}(x) = \Phi_e^{A'_s\restriction z}(x) = \Phi_e^{A'\restriction z}(x) = \Phi_e^{A'}(x) = f(x)$$

Luego $\lim_s g(x,s) = f(x)$.

$\Leftarrow$ Sup. $f(x) = \lim_s g(x,s)$ para g A-computable. Definir

$$A_x = \{s : (\exists t)\ [s \leq t\ \wedge\ g(x,t) \neq g(x,t+1)]\}$$

- A_x es finito porque existe el límite $\lim_s g(x,s)$
- $s \notin A_x \Rightarrow (\forall t \geq s)\ g(x,t) = g(x,t+1) \Rightarrow g(x,s) = f(x)$

$$\begin{aligned} B &= \{\langle s,x\rangle : s \in A_x\} \\ &= \{z : (\exists t)\ [\pi_1(z) \leq t\ \wedge\ g(\pi_2(z),t) \neq g(\pi_2(z),t+1)]\} \end{aligned}$$

es claramente Σ_1^A, luego A-c.e., luego $B \leq_T A'$ (TS).
La siguiente función es B-computable:

$$m(x) = \min\{s : s \notin A_x\} = \min\{s : \langle s,x\rangle \notin B\}.$$

Entonces $g(x,m(x)) = f(x)$. Como $g \leq_T A \leq_T A'$ y $m \leq_T B \leq_T A'$ entonces $f \leq A'$.

$\square$

Grados Turing

Recordar que

$$\deg_T(A) = \{B : A \equiv_T A\}$$

Si $\mathbf{a}$ es un grado Turing, $\mathbf{a}' = \deg_T(A')$ para $A \in \mathbf{a}$.
Sabemos que

- $\mathbf{a}' >_T \mathbf{a}$
- $\mathbf{a}'$ es c.e. en $\mathbf{a}$

Sea $\mathbf{0^{(n)}} = \deg_T(\emptyset^{(n)})$. Entonces tenemos una jerarquía de grados

$$\mathbf{0} <_T \mathbf{0}' <_T \mathbf{0}'' <_T \mathbf{0}''' \ldots$$

Ejemplos

$$
\begin{aligned}
\mathbf{0} &= \deg_T(\emptyset) = \{B : B \text{ es computable}\} \\
\mathbf{0}' &= \deg_T(\emptyset') = \deg_T(K) = \deg_T(K_0) = \deg_T(K_1) \\
\mathbf{0}'' &= \deg_T(\emptyset'') = \deg_T(\text{Tot}) = \deg_T(\text{Fin}) \\
\mathbf{0}''' &= \deg_T(\emptyset''') = \deg_T(\text{Cof}) = \deg_T(\text{Rec}) = \deg_T(\text{Ext})
\end{aligned}
$$

donde

$$
\begin{aligned}
\text{Tot} &:= \{x : W_x = \mathbb{N}\} = \{x : \Phi_x \text{ es total}\} \\
\text{Fin} &:= \{x : W_x \text{ es finito}\} \\
\text{Cof} &:= \{x : W_x \text{ es cofinito}\} \\
\text{Rec} &:= \{x : W_x \text{ es computable}\} \\
\text{Ext} &:= \{x : \Phi_x \text{ es extendible a una función total computable}\}
\end{aligned}
$$

Teoría de la Computabilidad

Clase 6

Jerarquía Aritmética, Teorema de Post, Conjuntos Σ_n-completos

Conjuntos Σ_n y Π_n

Definición

- Un conjunto B es Σ_0 o Π_0 sii B es computable
- Para $n \geq 1$, un conjunto B es Σ_n (notado $B \in \Sigma_n$) sii existe una relación computable $R(x, y_1, \ldots, y_n)$ tal que

$$x \in B \quad \text{sii} \quad \underbrace{(\exists y_1)(\forall y_2)(\exists y_3)\ldots(Qy_n)}_{n-1 \text{ alternancias}} \; R(x, y_1, \ldots, y_n)$$

  donde $Q = \exists$ si n es par y $Q = \forall$ si n es impar.
- Para $n \geq 1$, un conjunto B es Π_n (notado $B \in \Pi_n$) sii existe una relación computable $R(x, y_1, \ldots, y_n)$ tal que

$$x \in B \quad \text{sii} \quad \underbrace{(\forall y_1)(\exists y_2)(\forall y_3)\ldots(Qy_n)}_{n-1 \text{ alternancias}} \; R(x, y_1, \ldots, y_n)$$

  donde $Q = \forall$ si n es par y $Q = \exists$ si n es impar.
- B es Δ_n (notado $B \in \Delta_n$) sii $B \in \Sigma_n \cap \Pi_n$.

Relaciones y fórmulas Σ_n y Π_n

Definición

- Una relación $R \subseteq \mathbb{N}^k$ es Σ_n (Π_n) si es Σ_n (Π_n) el conjunto

$$\{\langle r_1, \ldots, r_k \rangle : (r_1, \ldots, r_k) \in R\}$$

- Una fórmula $\varphi(x_1, \ldots, x_n)$ con variables libres $x_1, \ldots, x_n$ es Σ_n (Π_n) si es Σ_n (Π_n) la relación

$$\{(a_1, \ldots, a_n) : \mathbb{N} \models \varphi(a_1, \ldots, a_n)\}$$

Por ejemplo,

- $R \subseteq \mathbb{N}^3$ es Σ_2 si existe una relación R' computable tal que

$$(x, y, z) \in R \quad \text{sii} \quad (\exists u)(\forall v)\ R'(x, y, z, u, v).$$

- si R es computable, es Π_3 la siguiente fórmula:

$$(\forall x)(\exists y)(\forall z)\ R(x, y, z, w)$$

Propiedades de Σ_n y Π_n

Teorema

1. $A \in \Sigma_n \Rightarrow \overline{A} \in \Pi_n$
2. $A \in \Sigma_n(\Pi_n) \Rightarrow (\forall m > n)\ A \in \Sigma_m \cap \Pi_m$
3. $A, B \in \Sigma_n(\Pi_n) \Rightarrow A \cup B, A \cap B \in \Sigma_n(\Pi_n)$
4. $R \in \Sigma_n \wedge n > 0 \wedge A = \{x : (\exists y)\ R(x, y)\} \Rightarrow A \in \Sigma_n$

4'. $R \in \Pi_n \wedge n > 0 \wedge A = \{x : (\forall y)\ R(x, y)\} \Rightarrow A \in \Pi_n$

5. $B \leq_m A \wedge A \in \Sigma_n(\Pi_n) \Rightarrow B \in \Sigma_n(\Pi_n)$
6. *Si* $R \in \Sigma_n(\Pi_n)$ *entonces están en* $\Sigma_n(\Pi_n)$*:*

$$\begin{aligned} A &= \{\langle x, y\rangle : (\forall z < y)\ R(x, y, z)\} \\ B &= \{\langle x, y\rangle : (\exists z < y)\ R(x, y, z)\} \end{aligned}$$

1. $A \in \Sigma_n \Rightarrow \overline{A} \in \Pi_n$:
   Si
   $$A = \{x : (\exists y_1)(\forall y_2)(\exists x_3) \dots \ R(x, y_1, \dots, y_n)\}$$
   entonces
   $$\overline{A} = \{x : (\forall y_1)(\exists y_2)(\forall y_3) \dots \ \neg R(x, y_1, \dots, y_n)\}$$

2. $A \in \Sigma_n(\Pi_n) \Rightarrow (\forall m > n)\ A \in \Sigma_m \cap \Pi_m$:
   Observar que
   $$(\exists x_1)(\forall x_2)(\exists x_3) \dots (Qx_n)\ R(x, x_1, \dots, x_n)$$
   sii
   $$(\forall x_0)(\exists x_1)(\forall x_2)(\exists x_3) \dots (Qx_n)\ \underbrace{R(x, x_1, \dots, x_n)}_{R'(x, x_0, x_1, \dots, x_n)}$$
   sii
   $$(\exists x_1)(\forall x_2)(\exists x_3) \dots (Qx_n)(\tilde{Q}x_{n+1})\ \underbrace{R(x, x_1, \dots, x_n)}_{R'(x, x_1, \dots, x_n, x_{n+1})}$$
   donde $\tilde{Q} = \exists$ si $Q = \forall$ y viceversa.
   Lo mismo para fórmulas Π_n.

3. $A, B \in \Sigma_n(\Pi_n) \Rightarrow A \cup B, A \cap B \in \Sigma_n(\Pi_n)$:
Supongamos

$$\begin{array}{lll} x \in A & \text{sii} & (\exists y_1)(\forall y_2) \ldots R(x, \vec{y}) \\ x \in B & \text{sii} & (\exists z_1)(\forall z_2) \ldots S(x, \vec{z}) \end{array}$$

Entonces

$$\begin{array}{lll} x \in A \cup B & \text{sii} & (\exists y_1)(\forall y_2) \ldots R(x, \vec{y}) \vee (\exists z_1)(\forall z_2) \ldots S(x, \vec{z}) \\ & \text{sii} & (\exists y_1)(\exists z_1)(\forall y_2)(\forall z_2) \ldots R(x, \vec{y}) \vee S(x, \vec{z}) \\ & \text{sii} & (\exists u_1)(\forall u_2) \ldots R(x, \pi_1(u_1), \pi_1(u_2), \ldots) \vee \\ & & S(x, \pi_2(u_1), \pi_2(u_2), \ldots) \end{array}$$

El mismo truco para $A \cap B$ y para Π_n.

4. $R \in \Sigma_n \wedge n > 0 \wedge A = \{x : (\exists y)\ R(x,y)\} \Rightarrow A \in \Sigma_n$
   Supongamos

$$(x,y) \in R \quad \text{sii} \quad (\exists z_1)(\forall z_2)\ldots R'(x,y,z_1,z_2,\ldots).$$

   Entonces

$$\begin{array}{lll} x \in A & \text{sii} & (\exists y)\ R(x,y) \\ & \text{sii} & (\exists y)(\exists z_1)(\forall z_2)\ldots R'(x,y,z_1,z_2,\ldots) \\ & \text{sii} & (\exists u)(\forall z_2)\ldots R'(x,\pi_1(u),\pi_1(u),z_2,\ldots) \end{array}$$

4'. $R \in \Pi_n \wedge n > 0 \wedge A = \{x : (\forall y)\ R(x,y)\} \Rightarrow A \in \Pi_n$
   Análogo a 4.

5. $B \leq_m A \wedge A \in \Sigma_n(\Pi_n) \Rightarrow B \in \Sigma_n(\Pi_n)$
   Sea $A = \{x : (\exists y_1)(\forall y_2)\ldots R(x,\vec{y})\}$ y sea $B \leq_m A$ via f.
   Entonces

$$B = \{x : (\exists y_1)(\forall y_2)\ldots \underbrace{R(f(x),\vec{y})}_{\text{computable}}\}$$

   Lo mismo para el caso Π_n.

6. Si $R \in \Sigma_n(\Pi_n)$ entonces están en $\Sigma_n(\Pi_n)$:

$$\begin{aligned} A &= \{\langle x, y\rangle : (\forall z < y)\ R(x, y, z)\} \\ B &= \{\langle x, y\rangle : (\exists z < y)\ R(x, y, z)\} \end{aligned}$$

Por inducción en n. Si $n = 0$, A y B son computables. Sea $n > 0$ y supongamos cierta la propiedad para todo $< n$.
Supongamos $R \in \Sigma_n$. Por 4, $B \in \Sigma_n$. Existe $S \in \Pi_{n-1}$ tal que

$$R(x, y, z) \quad \text{sii} \quad (\exists u)\ S(x, y, z, u).$$

Entonces

$$\begin{aligned} \langle x, y\rangle \in A \quad & \text{sii} \quad (\forall z < y)\ R(x, y, z) \\ & \text{sii} \quad (\forall z < y)(\exists u)\ S(x, y, z, u) \\ & \text{sii} \quad (\exists \sigma)(\forall z < y)\ S(x, y, z, \pi_z(\sigma)) \end{aligned}$$

donde σ se interpreta como una y-upla.
Por HI

$$\{\langle x, y, \sigma\rangle : (\forall z < y)\ S(x, y, z, \pi_z(\sigma))\} \in \Pi_{n-1}.$$

Entonces $A \in \Sigma_n$.
El caso $R \in \Pi_n$ es análogo.

Ejemplos

$$\begin{aligned} \text{Fin} &:= \{x : W_x \text{ es finito}\} \\ \text{Cof} &:= \{x : W_x \text{ es cofinito}\} \end{aligned}$$

Proposición

$\text{Fin} \in \Sigma_2$.

Demostración.

$x \in \text{Fin}$ sii W_x es finito sii

$$(\exists s)(\forall t)\ [t > s \Rightarrow \overbrace{W_{x,s} = W_{x,t}}^{\text{computable}}]$$

□

Proposición

$\text{Cof} \in \Sigma_3$.

Demostración.

$x \in \text{Cof}$ sii $\overline{W_x}$ es finito sii

$$(\exists y)(\forall z)\ [z > y \Rightarrow z \in W_x] \quad \text{sii} \quad (\exists y)(\forall z)(\exists s)\ [z > y \Rightarrow \overbrace{z \in W_{x,s}}^{\text{computable}}\ ]$$

□

Conjuntos Σ_n y Π_n-completos y Teorema de Post (TP)

Definición

Un conjunto A es Σ_n-completo (Π_n-completo) si $A \in \Sigma_n$ (Π_n) y $B \leq_1 A$ para todo $B \in \Sigma_n$ (Π_n).

Teorema (Post)

Para todo $n \geq 0$:

1. $B \in \Sigma_{n+1}$ *sii* B *es c.e. en algún conjunto* Π_n *sii* B *es c.e. en algún conjunto* Σ_n
2. $\emptyset^{(n)}$ *es* Σ_n*-completo, para* $n > 0$
3. $B \in \Sigma_{n+1}$ *sii* B *es c.e. en* $\emptyset^{(n)}$
4. $B \in \Delta_{n+1}$ *sii* $B \leq_T \emptyset^{(n)}$

1. $B \in \Sigma_{n+1}$ sii B es c.e. en algún conjunto Π_n sii B es c.e. en algún conjunto Σ_n:
   Habíamos visto que B es C-c.e. sii B es $\overline{C}$-c.e. (TS 7). El último 'sii' sale de ahí. Veamos el primer 'sii':
   - $\Rightarrow$ Sea $B \in \Sigma_{n+1}$. Entonces existe $R \in \Pi_n$ tal que
   $$x \in B \quad \text{sii} \quad (\exists y)\ R(x,y)$$
   Entonces B es Σ_1^R y por lo tanto B es R-c.e.
   - $\Leftarrow$ Supongamos que B es C-c.e. para $C \in \Pi_n$. Entonces existe un e tal que
   $$\begin{array}{lll} x \in B & \text{sii} & x \in W_e^C \\ & \text{sii} & (\exists s)(\exists \sigma)\ [\sigma \subset C \wedge \underbrace{x \in W_{e,s}^{\sigma}}_{\text{computable}}\ ] \end{array}$$
   Basta ver que '$\sigma \subset C$' es Σ_{n+1}
   $$\begin{array}{lll} \sigma \subset C & \text{sii} & (\forall i < |\sigma|)\ [\sigma(i) = C(i)] \\ & \text{sii} & (\forall i < |\sigma|)\ [\underbrace{[\sigma(i) = 1 \wedge i \in C]}_{\Pi_n} \vee \underbrace{[\sigma(i) = 0 \wedge i \notin C]}_{\Sigma_n}] \end{array}$$
   Sabemos $\Pi_n, \Sigma_n \subseteq \Sigma_{n+1}$ y que $\Sigma_{n+1} \vee \Sigma_{n+1} \subseteq \Sigma_{n+1}$. Como el $\forall$ externo está acotado, $C \in \Sigma_{n+1}$.

2. $\emptyset^{(n)}$ es Σ_n-completo, para $n > 0$:
   Por inducción en n. Es claro que vale para $n = 1$.
   Supongamos $\emptyset^{(n)}$ es Σ_n-completo.

   $B \in \Sigma_{n+1}$ sii B es c.e. en algún $C \in \Sigma_n$ por TP1
   sii B es c.e. en $\emptyset^{(n)}$ por HI, $C \leq_1 \emptyset^{(n)}$
   sii $B \leq_1 \emptyset^{(n+1)}$ por TS3

3. $B \in \Sigma_{n+1}$ sii B es c.e. en $\emptyset^{(n)}$:

   $B \in \Sigma_{n+1}$ sii B es c.e. en algún conjunto Σ_n por TP1
   sii B es c.e. en $\emptyset^{(n)}$ por TP2

4. $B \in \Delta_{n+1}$ sii $B \leq_T \emptyset^{(n)}$:

   $B \in \Delta_{n+1}$ sii $B, \overline{B} \in \Sigma_{n+1}$
   sii $B, \overline{B}$ son c.e. en $\emptyset^{(n)}$ por TP3
   sii $B \leq_T \emptyset^{(n)}$

Jerarquía estricta

Sabemos

$$\begin{aligned} \Pi_n, \Sigma_n &\supseteq \bigcup_{m<n} \Sigma_m \cup \Pi_m \\ \Pi_n, \Sigma_n &\supseteq \Delta_n \end{aligned}$$

Corolario

Si $n > 0$ *entonces* $\Delta_n \subsetneq \Pi_n, \Sigma_n$.

Demostración.

$\emptyset^{(n)} \in \Sigma_n$. Supongamos que $\emptyset^{(n)} \in \Delta_n$. Por TP4 $\emptyset^{(n)} \leq_T \emptyset^{(n-1)}$ y esto contradice TS2. Entonces $\emptyset^{(n)} \in \Sigma_n \setminus \Delta_n$.

Análogamente $\overline{\emptyset^{(n)}} \in \Pi_n \setminus \Delta_n$. □

Ejemplo

Proposición

Fin *es* Σ_2*-completo.*

Demostración.

Ya vimos que Fin $\in \Sigma_2$. Sea $A \in \Sigma_2$ y veamos $A \leq_1$ Fin. Por definición

$$x \in \overline{A} \quad \text{sii} \quad (\forall y)(\exists z)\ R(x, y, z).$$

Por el Teorema del Parámetro, existe f computable 1-1 tal que

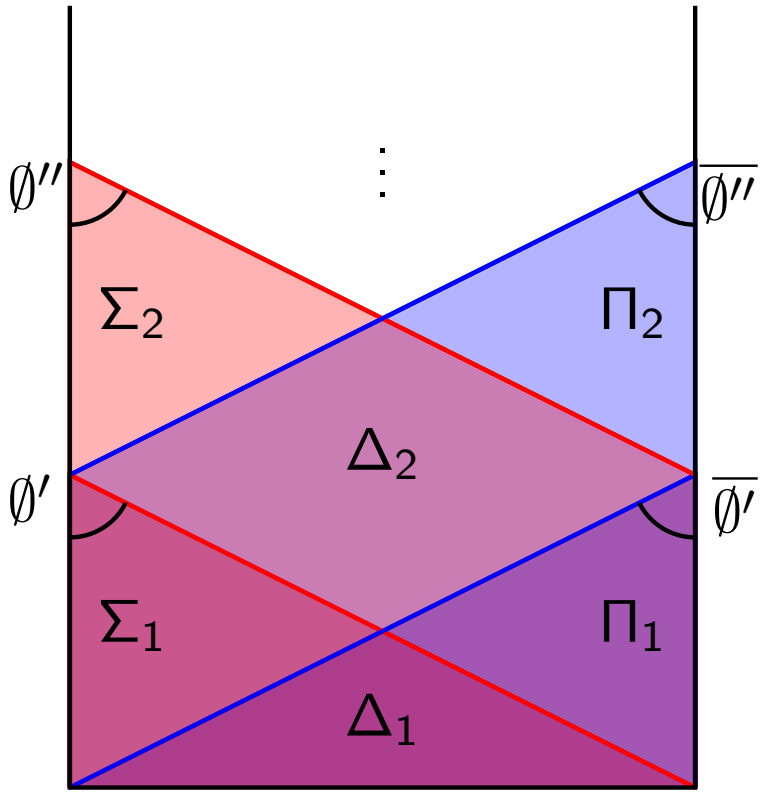
$$\varphi_{f(x)}(u) = \begin{cases} 0 & \text{si } (\forall y \leq u)\ (\exists z)\ R(x, y, z) \\ \uparrow & \text{si no} \end{cases}$$

Tenemos

- si $x \in A$ entonces existe y' tal que no es cierto $(\exists z)\ R(x, y', z)$. Luego, no es cierto $(\forall y \leq u)\ (\exists z)\ R(x, y, z)$ para $u \geq y'$. Entonces $W_{f(x)}$ es finito, o sea $f(x) \in$ Fin.
- $x \in \overline{A} \Rightarrow W_{f(x)} = \mathbb{N} \Rightarrow f(x) \notin$ Fin

□

La jerarquía aritmética



Conjuntos Σ_n^A y Π_n^A

Definición

- Un conjunto B es Σ_0^A (Π_0^A) sii B es A-computable
- Para $n \geq 1$, un conjunto B es Σ_n^A (notado $B \in \Sigma_n^A$) sii existe una relación A-computable $R(x, y_1, \ldots, y_n)$ tal que

$$x \in B \quad \text{sii} \quad \underbrace{(\exists y_1)(\forall y_2)(\exists y_3)\ldots(Qy_n)}_{n-1 \text{ alternancias}} \; R(x, y_1, \ldots, y_n)$$

  donde $Q = \exists$ si n es par y $Q = \forall$ si n es impar.
- Para $n \geq 1$, un conjunto B es Π_n^A (notado $B \in \Pi_n^A$) sii existe una relación A-computable $R(x, y_1, \ldots, y_n)$ tal que

$$x \in B \quad \text{sii} \quad \underbrace{(\forall y_1)(\exists y_2)(\forall y_3)\ldots(Qy_n)}_{n-1 \text{ alternancias}} \; R(x, y_1, \ldots, y_n)$$

  donde $Q = \forall$ si n es par y $Q = \exists$ si n es impar.
- B es Δ_n^A (notado $B \in \Delta_n^A$) sii $B \in \Sigma_n^A \cap \Pi_n^A$.

Jerarquía Aritmética relativizada

Definición

Un conjunto A es Σ_n^C-completo (Π_n^C-completo) si $A \in \Sigma_n^C$ (Π_n^C) y $B \leq_1 A$ para todo $B \in \Sigma_n^C$ (Π_n^C).

Teorema (Post relativizado, TPR)

Para todo $n \geq 0$:

1. $B \in \Sigma_{n+1}^A$ *sii* B *es c.e. en algún conjunto* Π_n^A *sii* B *es c.e. en algún conjunto* Σ_n^A
2. $A^{(n)}$ *es* Σ_n^A*-completo, para* $n > 0$
3. $B \in \Sigma_{n+1}^A$ *sii* B *es c.e. en* $A^{(n)}$
4. $B \in \Delta_{n+1}^A$ *sii* $B \leq_T A^{(n)}$

Observaciones sobre Σ_n^A y Π_n^A

Proposición

1. *si* $A \leq_T B$ *entonces* $\Sigma_n^A \subseteq \Sigma_n^B$
2. *si* $A \leq_T B$ *entonces* $\Pi_n^A \subseteq \Pi_n^B$
3. $\Sigma_{n+1} = \Sigma_n^{\emptyset'}$
4. $\Pi_{n+1} = \Pi_n^{\emptyset'}$

Demostración.

1 y 2 son triviales.
Para 3, de TP3 $A \in \Sigma_{n+1}$ sii A es c.e. en $\emptyset^{(n)}$. Por TPR3, $A \in \Sigma_n^{\emptyset'}$ sii A es c.e. en $(\emptyset')^{(n-1)} = \emptyset^{(n)}$.
Análogo para 4. □

Teoría de la Computabilidad

Clase 7

Conjuntos simples

Problema de Post

Problema de Post:

¿Existe un conjunto A c.e. tal que $\emptyset <_T A <_T \emptyset'$?

Idea: creemos que sí, entonces construyamos un conjunto A c.e. tal que

- A sea incompleto ($K \not\leq_T A$).
  - para esto, definir una propiedad sencilla sobre $\overline{A}$ que diga que $\overline{A}$ es muy "fino", tiene muy poca información.
- A no sea computable ($A \not\leq_T \emptyset$)
  - la propiedad que definamos tiene que ser compatible con ser no computable.

Inmunidad y simpleza

Definición

- Un conjunto es inmune si es infinito pero no contiene a ningún conjunto infinito c.e.
- Un conjunto A es simple si A es c.e. y $\overline{A}$ es inmune

Proposición

Si A es simple entonces:

1. *A no es computable*
2. *A no es creativo*
3. *A no es m-completo*

Demostración.

Ejercicio. □

Existe un conjunto simple

Vamos a ver dos pruebas:

- Una larga pero educativa
  - nos va a servir para lo que viene
- Una corta y elegante

Primero la larga.

Existencia de conjuntos simples (I)

Requerimientos

Tratamos de construir un conjunto A c.e. y satisfacer los siguientes requerimientos, para todo $e \in \mathbb{N}$:

- $P_e : W_e$ es infinito $\Rightarrow W_e \cap A \neq \emptyset$
- $N_e : \|\overline{A}\| \geq e$

Intuitivamente,

- P_e es un requerimiento "positivo": trata de poner elementos en A, así A crece
- N_e es un requerimiento "negativo": trata de evitar poner elementos en A, así $\overline{A}$ crece

Si el conjunto c.e. A que construimos satisface P_e y N_e para todo $e \in \mathbb{N}$ entonces A es simple:

- supongamos que W_e es infinito y $W_e \subseteq \overline{A}$. Entonces $W_e \cap A = \emptyset$, o sea, no se cumple P_e.
- supongamos que $\|\overline{A}\| = e$, entonces no se cumple N_{e+1}.

Existencia de conjuntos simples (I)

Construcción

Recordar:

- $P_e : W_e$ es infinito $\Rightarrow W_e \cap A \neq \emptyset$
- $N_e : \|\overline{A}\| \geq e$

Definimos una enumeración de A. Llamamos A_s a la aproximación de A en el paso s. Cada A_s es finito, y definimos

$$\overline{A_s} = \{a_{0,s} < a_{1,s} < \dots\}.$$

Finalmente $A = \bigcup_s A_s$.

Paso 0 Definir $A_0 := \emptyset$.

Paso $s+1$ Ya está definido A_s.

Buscar el primer $e \leq s$ tal que

1. $W_{e,s} \cap A_s = \emptyset$ y
2. para algún $m \geq e$, $a_{m,s} \in W_{e,s}$

Si no hay ningún tal e, pasar al paso $s+2$. Si no,

- definimos $A_{s+1} := A_s \cup \{a_{n,s}\}$ (n es el mínimo m que satisface 2)
- decimos que "atendemos a P_e en el paso $s+1$"

Existencia de conjuntos simples (I)

Verificación

Hecho

Una vez que P_e es atendido en el paso s no vuelve a ser atendido.

Hecho

Los $a_{n,s}$ van creciendo a medida que aumenta s ($a_{n,s} \leq a_{n,s+1}$).
Cuando atendemos P_e en el paso $s+1$ y metemos $a_{n,s}$ en A,
crecen los $a_{m,s}$ para $m \geq n$, pero quedan iguales $a_{0,s}, \ldots, a_{n-1,s}$.

$$\overline{A_s} = \{a_{0,s} < \cdots < a_{e,s} < \cdots < a_{n-1,s} < \overbrace{a_{n,s}}^{\in A_{s+1}} < a_{n+1,s} < a_{n+2,s} \ldots\}$$

$$\overline{A_{s+1}} = \{a_{0,s} < \cdots < a_{e,s} < \cdots < a_{n-1,s} \quad < \quad a_{n+1,s} < a_{n+2,s} \ldots\}$$

$$\begin{array}{ccccccc} \| & & \| & & \| & & \| \\ a_{0,s+1} & & a_{e,s+1} & & a_{n-1,s+1} & & a_{n,s+1} \quad a_{n+1,s+1} \end{array}$$

Existencia de conjuntos simples (I)

Verificación

Hecho

Para todo n, $\lim_t a_{n,t} < \infty$.

Demostración.

Supongamos lo contrario. Existe n y sucesión $s_1 < s_2 < \dots$ tal que para todo i, $a_{n,s_i} < a_{n,s_i+1}$. Entonces para todo $i \in \mathbb{N}$, en el paso $s_i + 1$ atendemos a cierto P_{e_i} y metemos a_{n,s_i} en A. Cada P_{e_i} es atendido una sola vez. Entonces si $i \neq j$, $e_i \neq e_j$. Pero por la condición 2 de la construcción, $n \geq e_i$. Entonces $\{e_i : i \in \mathbb{N}\}$ es finito. Absurdo. □

Existencia de conjuntos simples (I)

Verificación

Hecho

Para todo e, se satisface N_e. Es decir, $\overline{A}$ es infinito.

Demostración.

Fijemos e. Sea s suficientemente grande tal que $a_{0,s}, \ldots, a_{e-1,s}$ son estables (i.e. para todo $t \geq s$ y $i = 0, \ldots, e-1$, tenemos $a_{i,s} = a_{i,t}$). Entonces

$$\{a_{0,s}, \ldots a_{e-1,s}\} \subseteq \overline{A}.$$

De esta manera, $\|\overline{A}\| \geq e$. □

Existencia de conjuntos simples (I)

Verificación

Hecho

Para todo e, se satisface P_e.

Demostración.

Supongamos que $\|W_e\| = \infty$ y $W_e \cap A = \emptyset$. Sea

$$\begin{aligned} \overline{A} &= \{a_0 < a_1 < a_2 < \dots\} \\ y &= \text{mín}\{x : x \in W_e \wedge x > a_e\} \end{aligned}$$

Sea s suficientemente grande tal que

- $y \in W_{e,s}$
- $a_{n,s} = a_n$ para $n \in \{0, \dots, e\}$
- si P_i, $i < e$, es atendido, ya fue atendido para el paso $s+1$

En el paso $s+1$, el item 1 de la construcción vale trivialmente.

- si $y \in A_s$ entonces $y \in A \cap W_e$. Absurdo.
- si $y \in \overline{A_s}$, tenemos $y = a_{m,s}$ para algún $m > e$. Entonces vale el item 2 y $a_{n,s} \in A_{s+1} \cap W_{e,s} \subseteq A \cap W_e$. Absurdo. □

Existencia de conjuntos simples (II)

Definir f parcial computable como:

$$f(e) = \textit{primer elemento} > 2e \textit{ que aparece en } W_e$$

Sea $S = \text{rg } f$.

- S es c.e. porque es el rango de una función parcial computable
- $\overline{S}$ es infinito: como $f(e) > 2e$, S contiene a lo sumo e elementos de entre $\{0, 1, \ldots, 2e\}$, a saber: $f(0), f(1), \ldots, f(e-1)$.
  Entonces

$$\begin{aligned} \|\overline{S} \upharpoonright 2e+1\| &= 2e+1 - \|\{f(0), f(1), \ldots, f(e-1)\}\| \\ &\geq 2e+1-e = e+1, \end{aligned}$$

  de modo que $\overline{S}$ es infinito.
- si W_e es infinito, $f(e) \in S \cap W_e$, de modo que $S \cap W_e \neq \emptyset$.
  Entonces S es simple.

Los conjuntos simples no nos alcanzan

Sabemos que si A es simple entonces

- A es c.e. (y por lo tanto $A \leq_m \emptyset'$)
- no computable ($A \not\leq_T \emptyset$)
- m-incompleto ($\emptyset' \not\leq_m A$)

Sin embargo A puede ser T-completo ($A \geq_T \emptyset'$) (lo veremos más adelante).

Nueva idea: Definir $\overline{A}$ más y más "fino".

Teoría de la Computabilidad

Clase 8

Conjuntos hipersimples

Codificación de conjuntos finitos y arreglos

Definición

Dado un conjunto finito $A = \{a_1 < a_2 < \cdots < a_k\}$,

- el número $y = \sum_{1 \leq i \leq k} 2^{a_i}$ es el índice canónico de A
- notamos D_y al conjunto finito cuyo índice canónico es y, y D_0 denota $\emptyset$.

Si y se escribe en binario, D_y son las posiciones donde el dígito 1 ocurre.
Por ejemplo, si $y = 5 = 2^0 + 2^2$, que en binario es 101, entonces $D_y = D_5 = \{0, 2\}$.

Definición

- Una secuencia $\{F_n\}_{n \in \mathbb{N}}$ de conjuntos finitos es un arreglo si hay una función computable f tal que $F_n = D_{f(n)}$.
- Un arreglo es disjunto si todos sus miembros son disjuntos dos a dos.

Hipersimpleza

Definición

- Un conjunto infinito B es hiperinmune (h-inmune) si no existe un arreglo disjunto $\{F_n\}_{n\in\mathbb{N}}$ tal que $(\forall n)\ F_n \cap B \neq \emptyset$.
- Un conjunto A c.e. es hipersimple (h-simple) si $\overline{A}$ es h-inmune.

Nueva idea: "más fino" = hipersimple
Si A es simple entonces no hay un conjunto infinito c.e.

$$\{a_n\}_{n\in\mathbb{N}} \subseteq \overline{A}.$$

Si A es hipersimple entonces no hay un arreglo disjunto

$$\{F_n\}_{n\in\mathbb{N}} \quad \text{tal que} \quad (\forall n)\ F_n \cap \overline{A} \neq \emptyset$$

- $\overline{A}$ contiene a algún elemento de F_n, pero no sabemos a cuál.

Teorema

Si A es hipersimple entonces A es simple.

Demostración.

Ejercicio.

Mayorar funciones y función principal

Definición

- Una función f mayora a una función g si $(\forall x)\ f(x) \geq g(x)$.
- Si $A = \{a_0 < a_1 < \dots\}$ es un conjunto infinito, la función principal de A es $p_A(n) = a_n$.
- Una función f mayora un conjunto infinito A si f mayora a p_A.

h-inmune sii no computablemente mayorado

Teorema

Un conjunto infinito A es h-inmune sii ninguna función computable mayora a A.

Corolario

Un conjunto coinfinito c.e. A es h-simple sii ninguna función computable mayora a $\overline{A}$.

Demostración (del Teorema)

$\Leftarrow$ Sea $\{D_{g(x)}\}_{x\in\mathbb{N}}$ un arreglo disjunto tal que

$$(\forall x)\ D_{g(x)} \cap A \neq \emptyset.$$

Sea

$$f(x) = \text{máx} \bigcup \{D_{g(y)}\}_{y\leq x}.$$

- f es computable
- como $\{D_{g(x)}\}_{x\in\mathbb{N}}$ es disjunto, $f(x) \geq p_A(x)$

$\Rightarrow$ Sea f computable tal que $(\forall x)\ f(x) \geq p_A(x)$. Definir g de esta manera:

- $g(0)$ es tal que $D_{g(0)} = \{0, \ldots, f(0)\}$
- supongamos que tenemos definido $g(0), \ldots, g(n)$. Sea

$$k_n = 1 + \text{máx} \bigcup \{D_{g(i)}\}_{i \leq n}$$

y definir $g(n+1)$ tal que

$$D_{g(n+1)} = \{k_n, \ldots, f(k_n)\}$$

Claramente $\{D_{g(n)}\}_{n \in \mathbb{N}}$ es un arreglo disjunto.
Veamos que $(\forall n)\ D_{g(n)} \cap A \neq \emptyset$:

- es claro para $n = 0$ pues $p_A(0) \leq f(0)$ y $D_{g(0)} = \{0, \ldots, f(0)\}$.
- basta ver que $p(k_n) \in D_{g(n+1)} = \{k_n, \ldots, f(k_n)\}$
  - como p_A es función principal, $p_A(k_n) \geq k_n$
  - por hipótesis $p_A(k_n) \leq f(k_n)$

$\square$

Existe un conjunto h-simple

Teorema

Para cada conjunto c.e. no computable A hay un conjunto h-simple $B \equiv_T A$.

Demostración

Sea f computable 1-1 tal que rg $f = A$. Sea $a_s = f(s)$. Entonces

$$A = \{a_0, a_1, a_2, \dots\}.$$

Definir el conjunto de deficiencia de A para la enumeración f:

$$B = \{s : (\exists t > s)\ a_t < a_s\}$$

- $B \in \Sigma_1$, de modo que B es c.e.
- $\overline{B}$ es infinito (ejercicio)
- $a_{p_{\overline{B}}(x)} < a_{p_{\overline{B}}(x+1)}$ (y luego $a_{p_{\overline{B}}(x)} \geq x$):
  - $p_{\overline{B}}(x) \in \overline{B}$, de modo que $(\forall t > p_{\overline{B}}(x))\ a_t \geq a_{p_{\overline{B}}(x)}$
  - como f es 1-1, $(\forall t > p_{\overline{B}}(x))\ a_t > a_{p_{\overline{B}}(x)}$
  - instanciar $t = p_{\overline{B}}(x+1) > p_{\overline{B}}(x)$

- $A \leq_T B$:
  - basta ver que $x \in A$ sii $x \in \{a_0, a_1, \ldots, a_{p_{\overline{B}}(x)}\}$
  - sabemos que $(\forall t > p_{\overline{B}}(x))\ a_t > a_{p_{\overline{B}}(x)} \geq x$
  - si $x \in A$ entonces $x = a_s$ para $s \leq p_{\overline{B}}(x)$
  - dado B puedo computar $\overline{B}$ y de ahí $p_{\overline{B}}$
- $B \leq_T A$:
  - definir $A_s = \{a_0, a_1, \ldots, a_s\}$ (observar $\#A_s = s + 1$)
  - dado s enumerar $A_0, A_1, A_2, \ldots$ hasta encontrar el primer t tal que $A_t \upharpoonright 1 + a_s = A \upharpoonright 1 + a_s$. Esto se puede hacer de manera A-computable.
  - es claro que $t \geq s$, pues al menos tengo que 'encontrar' a_s
  - por construcción, $a_t \leq a_s$ y $(\forall t' > t)\ a_{t'} > a_s$
  - si $t = s$ entonces $s \notin B$
  - si $t > s$ entonces $a_t < a_s$. Luego $s \in B$.
- $\overline{B}$ es h-inmune:
  - supongamos $p_{\overline{B}}$ es mayorada por g
  - entonces $x \in A$ sii $x \in \{a_0, a_1, \ldots, a_{g(x)}\}$
  - entonces A sería computable. Absurdo.

Simpleza e hipersimpleza no alcanzan

Queríamos resolver el problema de Post:

¿Existe un conjunto A c.e. tal que $\emptyset <_T A <_T \emptyset'$?

Buscábamos una respuesta afirmativa tratando de encontrar una propiedad sobre A que garantizara

1. no computabilidad ($A \not\leq_T \emptyset$)
2. T-incompletitud ($\emptyset' \not\leq_T A$)

Propuestas:

- A es simple
- A es hipersimple
- (hay más: A es hiperhipersimple)

Garantizan 1 pero no pueden garantizar 2: por el teorema anterior, hay un conjunto hipersimple $\equiv_T \emptyset'$

Teoría de la Computabilidad

Clase 9

Conjuntos bajos, solución al problema de Post

Conjuntos altos y bajos

Sabemos que si $\emptyset \leq_T A \leq_T \emptyset'$ entonces $\emptyset' \leq_T A' \leq_T \emptyset''$.

Definición

- A es bajo (*low*) si $A' \equiv_T \emptyset'$.
- A es alto (*high*) si $A' \equiv_T \emptyset''$.

Intuitivamente,

- los conjuntos bajos están cerca de ser computables
- los conjuntos altos están cerca de ser completos

Equivalencias de conjuntos bajos

Teorema

Si $A \leq_T \emptyset'$, los siguientes son equivalentes:

1. *A es bajo*
2. $\Sigma_1^A \subseteq \Pi_2$
3. $A' \leq_1 \overline{\emptyset''}$

Demostración.

$$
\begin{array}{lll}
A \text{ es bajo} & \text{sii} & A' \leq_T \emptyset' \\
 & \text{sii} & A' \in \Delta_2 \qquad \text{por TPR4} \\
 & \text{sii} & \Sigma_1^A \subseteq \Delta_2 \qquad \text{porque } A' \text{ es } \Sigma_1^A\text{-completo} \\
 & \text{sii} & \Sigma_1^A \subseteq \Pi_2 \qquad \text{porque } \Sigma_1^A \subseteq \Sigma_1^{\emptyset'} = \Sigma_2 \\
 & \text{sii} & A' \leq_1 \overline{\emptyset''}
\end{array}
$$

último sii:

$\Rightarrow$ Como $A' \in \Sigma_1^A$, entonces $A' \in \Pi_2$. Como $\overline{\emptyset''}$ es Π_2-completo, $A' \leq_1 \overline{\emptyset''}$.

$\Leftarrow$ $B \in \Sigma_1^A \overset{\text{TPR}}{\Rightarrow} B$ es A-c.e. $\overset{\text{TS5}}{\Rightarrow} B \leq_1 A' \leq_1 \overline{\emptyset''} \Rightarrow B \in \Pi_2$ $\square$

Equivalencias de conjuntos altos

Teorema

Si $A \leq_T \emptyset'$, los siguientes son equivalentes:

1. *A es alto*
2. $\Sigma_2 \subseteq \Pi_2^A$
3. $\emptyset'' \leq_1 \overline{A''}$

Demostración.

$$
\begin{array}{lll}
A \text{ es alto} & \text{sii} & \emptyset'' \leq_T A' \\
 & \text{sii} & \emptyset'' \in \Delta_2^A \qquad \text{por TPR4} \\
 & \text{sii} & \Sigma_2 \subseteq \Delta_2^A \qquad \text{porque } \emptyset'' \text{ es } \Sigma_2\text{-completo} \\
 & \text{sii} & \Sigma_2 \subseteq \Pi_2^A \qquad \text{porque } \Sigma_2 \subseteq \Sigma_2^A \\
 & \text{sii} & \emptyset'' \leq_1 \overline{A''}
\end{array}
$$

último sii:

$\Rightarrow$ Como $\emptyset'' \in \Sigma_2$, entonces $\emptyset'' \in \Pi_2^A$. Como $\overline{A''}$ es Π_2^A-completo, $\emptyset'' \leq_1 \overline{A''}$.

$\Leftarrow$ $B \in \Sigma_2 \Rightarrow B$ es $\emptyset'$-c.e. $\overset{\text{TS5}}{\Rightarrow} B \leq_1 \emptyset'' \leq_1 \overline{A''} \Rightarrow B \in \Pi_2^A$ $\square$

Solución al problema de Post

Friedberg y Muchnik dieron una solución al problema de Post: Construir un conjunto A que sea simple y bajo.

- si A es simple entonces no es computable ($\emptyset <_T A$)
- si A es bajo entonces es incompleto ($A <_T \emptyset'$)
  - en efecto, supongamos que $A \equiv_T \emptyset'$. Entonces $A' \equiv_T \emptyset''$. Como A es bajo, $A' \equiv_T \emptyset'$. Absurdo.

El problema está en construir un conjunto que cumpla con las dos condiciones.

Lesiones finitas (*finite injury*)

Como en la demostración (I) de existencia de conjuntos simples, vamos a tener dos tipos de requerimientos. Informalmente:

- P_e trata de poner elementos en A para que A sea simple
- N_e trata de no poner elementos en A para que A sea bajo

Los P_e y N_e tienen objetivos en conflicto.

Cada P_e y cada N_e tiene asignada una prioridad.

Cuando un P_i actúa, puede destruir (lesionar) un N_j solo si la prioridad de P_i es mayor que la prioridad de de N_j.

La construcción va a garantizar que:

- solo los N_j pueden ser lesionados por P_i de mayor prioridad
- para cada N_j hay una cantidad finita de i tal que P_i tiene más prioridad que N_j. Entonces cada N_j puede ser lesionado solo una cantidad finita de veces.

Existe un conjunto simple y bajo

Basta construir un conjunto A c.e. y coinfinito que cumpla estos requerimientos:

- $P_e : W_e$ es infinito $\Rightarrow$ $W_e \cap A \neq \emptyset$
- $N_e : (\exists^\infty s)\ \Phi_{e,s}^{A_s}(e)\downarrow \Rightarrow \Phi_e^A(e)\downarrow$

Está claro que los P_e quieren agregar elementos a A para que A se interseque con todos los W_e infinitos.

En cambio, los N_e no quieren que A cambie. Si N_e vale y cambiamos A, corremos el riesgo de que N_e deje de valer. En efecto si $A_{t+1} = A_t \cup \{x\}$ podría pasar que $\Phi_e^{A_t}(e)\downarrow$ pero $\Phi_e^{A_{t+1}}(e)\uparrow$. Entonces hay que tener cuidado con qué x podemos agregar a A.

Los P_e garantizan simpleza; los N_e garantizan bajura

Los requerimientos P_e son los mismos que vimos para probar la existencia de conjuntos simples (demostración (I)).

Hecho

Si $\overline{A}$ es infinito y para todo e vale P_e entonces A es simple.

Hecho

Si para todo e vale N_e entonces A es bajo ($A' \leq_T \emptyset'$).

Demostración.

Definir la función computable g de esta manera:

$$g(e,s) = \begin{cases} 1 & \text{si } \Phi_{e,s}^{A_s}(e)\downarrow \\ 0 & \text{si no} \end{cases}$$

Si para todo e vale N_e entonces $\hat{g}(e) = \lim_s g(e,s)$ existe para todo e. Por el Lema del Límite, $\hat{g} \leq_T \emptyset'$. Pero $\hat{g}$ es la función característica de A', de modo que $A' \leq_T \emptyset'$. □

Cómo ayudamos a los N_e

Recordar:

- $N_e : (\exists^\infty s)\ \Phi_{e,s}^{A_s}(e)\downarrow \quad \Rightarrow \quad \Phi_e^A(e)\downarrow$

Vimos que si N_e vale y agregamos un elemento x a A, el cómputo $\Phi_e^A(e)$ se puede destruir (N_e se puede lesionar). Es decir, si $A_{t+1} = A_t \cup \{x\}$ podría pasar que $\Phi_e^{A_t}(e)\downarrow$ pero $\Phi_e^{A_{t+1}}(e)\uparrow$.
Recordar que la función de uso es computable y está definida como

$u(A_s; e, x, s) = 1+$máximo elemento usado en el cómputo de $\Phi_{e,s}^{A_s}(x)$

y definimos $r(e, s) = u(A_s; e, e, s)$.

Cómo ayudamos a los N_e

Ayudamos a N_e si evitamos agregar en A elementos $x \leq r(e,s)$.

1. si ponemos en A un elemento $x \leq r(e,s)$ estamos amenazando N_e y posiblemente lo lesionamos
2. si ponemos en A un elemento $x > r(e,s)$, a N_e no le importa y seguro que no lo lesionamos

Si siempre pudiéramos garantizar 2, si $\Phi_e^{A_t}(e)\downarrow$ y $r = r(e,t)$ entonces $A_t \upharpoonright r = A \upharpoonright r$ y luego $\Phi_e^A(e)\downarrow$.
No podemos garantizar 2 para todo e, pero fijamos esta regla de prioridad:

> *En el paso t, P_i puede meter elementos $\leq r(e,t)$ en A (lesionando a N_e) solo si $i < e$.*

Para $i < e$, P_i tiene mayor prioridad que N_e. Así, N_e puede ser lesionado por $P_0, \ldots, P_{e-1}$ pero nunca por $P_e, P_{e+1}, \ldots$

Construcción de A

Paso 0 $A_0 = \emptyset$.

Paso s+1 Dado A_s podemos computar $r(e, s)$ para el e que queramos. Elegir el menor $i \leq s$ tal que cumpla estas dos condiciones

1. $W_{i,s} \cap A_s = \emptyset$ y
2. $(\exists x)\ [x \in W_{i,s} \wedge x > 2i \wedge (\forall e \leq i)\ r(e, s) < x]$

Si tal i existe, definir $A_{s+1} := A_s \cup \{x\}$. En este caso decimos que atendemos a P_e. Si tal i no existe, definir $A_{s+1} := A_s$.

Finalmente $A = \bigcup_s A_s$.

Decimos que x lesiona a N_e en el paso $s + 1$ si $x \in A_{s+1} \setminus A_s$ y $x \leq r(e, s)$.

Verificación

Lema

P_e es atendido a lo sumo una sola vez.

Demostración.

Una vez que P_e es atendido en el paso $s+1$, la condición 1 de la construcción es falsa para pasos $> s+1$. □

Lema

N_e es lesionado a lo sumo una cantidad finita de veces.

Demostración.

Por la condición 2 de la construcción, N_e puede ser lesionado solo por un x agregado a A cuando un P_i es atendido, con $i < e$. Como cada P_i es atendido a lo sumo una vez, N_e es lesionado a lo sumo e veces. □

Verificación

Lema

Para cada e, N_e es verdadero y $r(e) = \lim_s r(e,s)$ existe.

Demostración.

Fijemos e. Sea s_e suficientemente grande tal que N_e no es lesionado en etapas $> s_e$.
Recordar:

- $N_e : (\exists^\infty s)\ \Phi_{e,s}^{A_s}(e)\downarrow \quad \Rightarrow \quad \Phi_e^A(e)\downarrow$

Supongamos cierto el antecedente de N_e. Tenemos $\Phi_{e,s}^{A_s}(e)\downarrow$ para algún $s > s_e$. Entonces (por inducción en t) para todo $t \geq s$ tenemos

$$r(e,t) = r(e,s) \qquad \text{y} \qquad \Phi_e^{A_t}(e) = \Phi_e^{A_s}(e).$$

De modo que para $r = r(e,s)$ tenemos $A_s \upharpoonright r = A \upharpoonright r$ y $\Phi_e^A(e) = \Phi_{e,s}^{A_s}(e)\downarrow$. □

Verificación

Lema

Para cada i, P_i es verdadero.

Demostración.

Fijemos i. Recordar:

- P_i : W_i es infinito $\quad \Rightarrow \quad W_i \cap A \neq \emptyset$

Supongamos que W_i es infinito. Sea s tal que

$$(\forall t \geq s)(\forall e \leq i)\ r(e,t) = r(e).$$

Sea $s' \geq s$ tal que ningún P_j, $j < i$ recibe atención después de la etapa s'. Sea $t > s'$ tal que

$$(\exists x)\ [x \in W_{i,t} \wedge x > 2i \wedge (\forall e \leq i)\ r(e) < x]$$

O bien $W_{i,t} \cap A_t \neq \emptyset$ o bien P_i es atendido en la etapa $t+1$. En cualquier caso, P_i es verdadero al finalizar la etapa $t+1$. □

Observar que $\overline{A}$ es infinito por la cláusula $x > 2i$ de la condición 2.

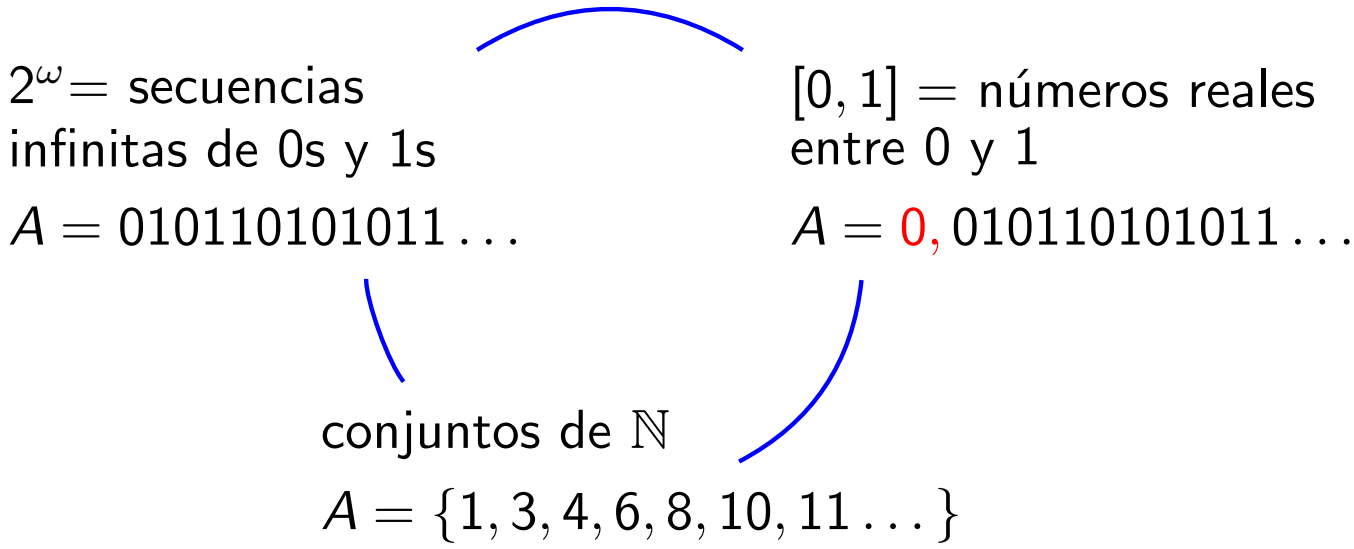
Teoría de la Computabilidad

Clase 10 (Opcional)

Aleatoriedad y Computabilidad

Notación

- Números reales = secuencias = conjuntos de naturales



- $2 = \{0,1\}$
- $2^{<\omega}$ = conjunto de todas las cadenas binarias
- $A \restriction n$ = primeros n dígitos de A
- $A(i) \in 2$ es el i-ésimo bit de A. Entonces $A(i) = 1$ sii $i \in A$.
- si σ es una cadena, $|\sigma|$ es la longitud de σ

Números normales

Experimento: tirar una moneda y anotar 0 = cara y 1 = ceca.

- Resultado 1

  $$0001110110101101010110100111101101\ldots$$

- Resultado 2

  $$0101010101010101010101010101010101\ldots$$

No nos llama la atención 1, peor no creemos en 2.

¿Cómo podemos definir formalmente la idea de "secuencia infinita de 0s y 1s aleatoria"?

Aleatoriedad según el paradigma de teoría de la medida

Idea:

- los números aleatorios deben ser aquellos que no tengan propiedades "efectivamente" raras
- si una propiedad es un conjunto efectivamente nulo, entonces un número aleatorio no debe tener esa propiedad
- o sino, el paradigma de estocasticidad: un aleatorio debe pasar todos los test estadísticos efectivos

Martin-Löf aleatoriedad

Definición

$\mathcal{C} \subseteq 2^\omega$ es una clase Σ^0_1 si existe e tal que

$$\mathcal{C} = \{Z \mid (\exists \sigma \in W_e)\ \sigma \prec Z\}.$$

Definición (Martin-Löf, 1966)

- un test de Martin-Löf es una secuencia $(\mathcal{V}_n)_{n\in\mathbb{N}}$ de clases Σ^0_1 uniformemente c.e. tal que $\mu(\mathcal{V}_n) \leq 2^{-n}$
- A pasa el test de Martin-Löf $(\mathcal{V}_n)_{n\in\mathbb{N}}$ si $A \notin \bigcap_{n\in\mathbb{N}} \mathcal{V}_n$
- A es Martin-Löf aleatorio (ML aleatorio) si y solo si A pasa todo test de Martin-Löf

Test universal

Teorema

Existe un test universal de Martin-Löf; i.e. existe $(\mathcal{U}_n)_{n\in\mathbb{N}}$ *tal que A es ML aleatorio sii A pasa* $(\mathcal{U}_n)_{n\in\mathbb{N}}$*.*

En otras palabras, los ML aleatorios son $2^\omega \setminus \bigcap_{n\in\mathbb{N}} \mathcal{U}_n$

Proposición

Casi todos los reales son ML aleatorios.

Aleatoriedad según el paradigma de impredecibilidad

Idea: El conocimiento de los primeros n bits no ayuda para conocer el $n+1$-ésimo.

Definición (Ville, 1939)

- Una martingala es una función $d : 2^{<\omega} \to \mathbb{R}^{\geq 0}$ tal que $d(\lambda) > 0$ y para cada $\sigma \in 2^{<\omega}$

$$d(\sigma) = \frac{d(\sigma 0) + d(\sigma 1)}{2}.$$

- d tiene éxito en $A \in 2^{\omega}$ si

$$\limsup_{n \to \infty} d(A \upharpoonright n) = \infty.$$

- d tiene éxito en una clase $\mathcal{C} \subseteq 2^{\omega}$ si d tiene éxito en cada $A \in \mathcal{C}$.

La interpretación es:

- juego justo de apuestas en bits sucesivos de una secuencia escondida
- d representa el capital. El apostador empieza con capital $d(\lambda) > 0$.
- en cada vuelta, apuesta $\alpha d(\sigma)$ al 0 y $(1-\alpha)d(\sigma)$ al 1 para $\alpha \in [0,1]$
- si acierta, duplica; sino pierde. $d(\sigma 0) = 2\alpha d(\sigma)$; $d(\sigma 1) = 2(1-\alpha)d(\sigma)$.

Caracterización de clases nulas usando martingalas

Una clase $\mathcal{C}$ es nula cuando la medida de Lebesgue de $\mathcal{C}$ es 0 ($\mu(\mathcal{C}) = 0$).

Teorema (Ville, 1939)

Para cualquier clase $\mathcal{C} \subseteq 2^\omega$, $\mu(\mathcal{C}) = 0$ sii hay una martingala que tiene éxito en $\mathcal{C}$.

En otras palabras

> *Para cualquier clase $\mathcal{C} \subseteq 2^\omega$, $\mu(\mathcal{C}) \neq 0$ sii ninguna martingala tiene éxito en $\mathcal{C}$.*

Caracterización de aleatoriedad de ML aleatoriedad martingalas

Definición

Un martingala d es c.e. si existe una función computable de dos variables

$$d : \mathbb{N} \times 2^{<\omega} \to \mathbb{Q}$$

tal que para todo $\sigma \in 2^{<\omega}$,

$$d(\sigma) = \lim_{s} d(s, \sigma)$$

Proposición

A es ML aleatorio sii ninguna martingala c.e. tiene éxito en A.

Es una versión efectiva del teorema de caracterización de clases nulas usando martingalas que vimos antes.

La crítica de Schnorr - aleatoriedad computable

Según Scnorr

- el teorema de caracterización anterior muestra una falla en la intuición detrás de la noción de ML aleatoriedad
- la aleatoriedad debe estar relacionada con vencer estrategias computables y no c.e.
- de la misma manera, en la definición de Martin-Löf, los tests $\mathcal{V}_n$ son c.e. pero no computables

Definición (Schnorr, 1971)

Lo mismo que Martin-Löf, pero con $\mu(\mathcal{V}_n) = 2^{-n}$.

Así, los $\mathcal{V}_n$ se vuelven computables (i.e. $\sigma 2^\omega \subseteq \mathcal{V}_n$ es computable).

Teorema

No hay test universal de Schnorr.

Computablemente aleatorio

Definición

Un martingala d es computable si

$$d : 2^{<\omega} \to \mathbb{Q}^{\geq 0}$$

es una función computable.

Definición

A es computablemente aleatorio (comp. aleatorio) si ninguna martingala computable tiene éxito en A.

Jerarquía de nociones:

$$\text{ML aleatorio} \Rightarrow \text{comp. aleatorio} \Rightarrow \text{Schnorr aleatorio}$$

No vale ninguna de las implicaciones $\Leftarrow$.

Aleatoriedad según el paradigma computacional

Idea: los números aleatorios son algorítmicamente incompresibles

Sea $\mathbf{T}_0, \mathbf{T}_1, \mathbf{T}_2, \ldots$ una enumeración de todas las máquinas de Turing libres de prefijos ($\operatorname{dom} T_e$ es libre de prefijos).
Fijamos $\mathbf{U}$ = máquina universal libre de prefijos, por ejemplo:

$$\mathbf{U}(0^e 1\sigma) = \mathbf{T}_e(\sigma).$$

Definición (Levin, 1973; Schnorr, 1973; Chaitin, 1975)

$K : 2^{<\omega} \to \mathbb{N}$, la complejidad de largo de programa con respecto a la máquina universal $\mathbf{U}$, es

$$K(\sigma) = \text{mín}\{|p| : \mathbf{U}(p) = \sigma\}.$$

K es una noción de complejidad absoluta (no depende de la máquina universal que elegimos).

K no es computable, pero es aproximable desde arriba

K no es computable, pero es aproximable desde arriba.
Existe f computable y no creciente

$$f : \sigma \times \mathbb{N} \to \mathbb{N} \cup \{\infty\}$$

tal que para todo $\sigma \in 2^{<\omega}$

$$\lim_{s} f(\sigma, s) = K(\sigma).$$

En efecto, definir

$$f(\sigma, s) = K_s(\sigma) = \min\{|p| : |p| \leq s \wedge \mathbf{U}_s(p) = \sigma\} \cup \{\infty\}.$$

ML aleatoriedad usando la complejidad de largo de programa

Teorema (Schnorr, 1971)

A es ML aleatorio sii $(\exists c)(\forall n)\, K(A \upharpoonright n) > n - c$.

Entonces ML aleatoriedad es una noción robusta. Puede ser caracterizada usando

- tests: no es atrapado por ningún test de Martin-Löf
- martingalas: ninguna martingala c.e. tiene éxito
- compresión: sus prefijos son algorítmicamente incompresibles

Probabilidad de detención

Dada una máquina libre de prefijos $\mathbf{M}$,

$$\Omega_{\mathbf{M}} = \sum_{p:\mathbf{M}(p)\downarrow} 2^{-|p|} = \mu\left(\mathrm{dom}\mathbf{M}2^{\omega}\right).$$

es la probabilidad de que $\mathbf{M}$ se detenga.

Por ejemplo, si $\mathbf{M}(p)\downarrow$ sólo cuando $p = 0^i1$ con i par, entonces

$$\Omega_{\mathbf{M}} = 0{,}1010101\ldots = 2/3.$$

Teorema (Chaitin, 1975)

Para cualquier máquina universal $\mathbf{U}$, $\Omega_{\mathbf{U}}$ *es aleatorio.*

Fijamos $\mathbf{U}$ = máquina universal. Escribimos Ω en lugar de $\Omega_{\mathbf{U}}$.

Reales aleatorios left-c.e.

Definición

Un real A es left-c.e. si existe una función computable $f : \mathbb{N} \to \mathbb{Q}$ no decreciente f tal que

$$\lim_s f(s) = A.$$

Proposición

Ω *es left-c.e.*

Teorema (Slaman y Kučera, 2001)

Para cualquier A left-c.e. Son equivalentes:

- *A es ML aleatorio*
- *Para todo B left-c.e.* $(\exists c)(\forall n)\ K(B \upharpoonright n) \leq K(A \upharpoonright n) + c$
- *A es la probabilidad de detención de alguna máquina universal*

ML aleatoriedad y clases Π^0_1

Definición

$\mathcal{C} \subseteq 2^\omega$ es una clase Π^0_1 si es el complemento de una clase Σ^0_1.

Si A no es aleatorio vía c, tenemos

$$(\exists n)\ K(A \upharpoonright n) \leq n - c \quad \text{sii} \quad (\exists n, s)\ K_s(A \upharpoonright n) \leq n - c$$

Sea G_c el siguiente conjunto c.e.

$$\{\sigma : (\exists n, s)\ K_s(\sigma) \leq n - c\}$$

- $A \in G_c 2^\omega$ (que es una clase Σ^0_1) sii A no es aleatorio vía c entonces
- A es aleatorio vía c sii $A \in 2^\omega \setminus G_c 2^\omega$ (que es una clase Π^0_1)

ML aleatoriedad y reducciones

Teorema (Kučera 1985; Gács 1986)

Sea $\mathcal{Q}$ es una clase Π^0_1 no vacía de conjuntos ML aleatorios. Para todo A existe un $Z \in \mathcal{Q}$ tal que $A \leq_{wtt} Z \leq_{wtt} A \oplus \emptyset'$.

($\leq_{wtt}$ es una reducción que implica $\leq_T$.)

Teorema (Calude y Nies, 1997)

$\Omega \equiv_{wtt} \emptyset'$.

Proposición

Ningún conjunto c.e. es ML aleatorio. ML aleatorio implica no computable.

Teorema (Low Basis Theorem, Jockusch y Soare 1972)

Cualquier clase Π^0_1 no vacía tiene un elemento bajo.

¡Entonces existen ML aleatorios bajos!

Reales anti-aleatorios: K-trivialidad y C-trivialidad

A es aleatorio cuando es algorítmicamente incompresible: la complejidad de $A \upharpoonright n$ es mayor que n.

A es anti-aleatorio cuando A es altamente compresible: la complejidad de $A \upharpoonright n$ es parecida a la complejidad de 0^n.

$$\begin{array}{rcl} A \upharpoonright n & = & A_0\ A_1\ A_2 \ldots\ A_{n-1} \\ 0^n & = & 0\ \ 0\ \ 0 \ldots\ \ 0 \end{array}$$

Definición (Kolmogorov, 1965; Solomonoff, 1964; Chaitin 1975)

$C : 2^{<\omega} \to \mathbb{N}$ es la complejidad plana (como K pero sin trabajar con máquinas libres de prefijos).

$C < K$ y C no sirve para definir aleatoriedad.

Definición

- Un real A es C-trivial cuando $(\exists c)(\forall n)\ C(A \upharpoonright n) \leq C(0^n) + c$.
- Un real A es K-trivial cuando $(\exists c)(\forall n)\ K(A \upharpoonright n) \leq K(0^n) + c$.

Reales anti-aleatorios: K-trivialidad y C-trivialidad

Teorema (Chaitin, 1976)

A es C-trivial sii A es computable.

Teorema (Chaitin, 1975)

Si A es K-trivial entonces $A \leq_T \emptyset'$ (i.e. $A \in \Delta^0_2$).

Teorema (Solovay, 1975; Downey et al. 2002)

Existen K-triviales no computables.

Nociones de bajura y K-trivialidad

Una noción de *bajura (lowness)* de un conjunto $A \subseteq \mathbb{N}$ dice que A es computacionalmente débil cuando se lo usa como oráculo. Por lo tanto A está cerca de ser computable. Algunas son nociones de anti-aleatoriedad.

bajura.

1. f. Falta de elevación.

2. f. ant. **bajeza** (‖ acción indigna).

3. f. ant. **bajeza** (‖ abatimiento, humillación).

□ V.

pesca de bajura

Real Academia Española © Todos los derechos reservados

Algunas nociones de bajura

La función de complejidad K y la definición de ML aleatorio se pueden relativizar a oráculos. Algunos oráculos pueden ser útiles y otros no.

- A es K-trivial: algorítmicamente muy compresible.
  $(\exists c)(\forall n)\ K(A \upharpoonright n) \leq K(0^n) + c$.
- A es bajo para K (*low for K*): no es útil para reducir K.
  $(\exists c)(\forall \sigma)\ K(\sigma) \leq K^A(\sigma) + c$.
- A es bajo para aleatorios (*low for random*): no es útil para detectar regularidades.
  Para todo B, si B es aleatorio entonces B es A-aleatorio

Teorema (Nies, 2003)

Las tres clases coinciden.

Referencias


André Nies.
Computability and Randomness
Oxford University Press, 2009.


Rod G. Downey and Denis R. Hirschfeldt.
Algorithmic randomness and complexity.
Springer, to appear.


Ming Li and Paul Vitányi.
An introduction to Kolmogorov complexity and its applications.
Springer, 2nd edition, 1997.


Cristian Calude.
Information and Randomness, an Algorithmic Perspective.
Springer-Verlag, Berlin, 1994.