Conjuntos computablemente enumerables

Guillermo Mosse

28 de septiembre de 2018

Sugerencia: leer la clase práctica `https://campus.exactas.uba.ar/pluginfile.php/90823/mod_folder/content/0/conjuntos_ce_y_coce.pdf?forcedownload=1` (2do Cuatrimestre, "conjuntos ce y co-ce"). Especialmente el ejercicio 2.

Definición 1 *Se dice que un conjunto $A \subseteq \mathbb{N}$ es c.e. si existe una función $g : \mathbb{N} \to \mathbb{N}$ parcial computable tal que:*

$$A = \{x : g(x) \downarrow\} = dom\ g$$

Definición 2 *Se dice que un conjunto $A \subseteq \mathbb{N}$ es co-c.e. si su complemento, $\bar{A}$, es c.e.*

Teorema 3 *Sea $A \subseteq \mathbb{N}$. Son equivalentes:*

- *A es c.e.*
- *A es el rango (o sea, la imagen) de una función p.r.*
- *A es el rango de una función computable.*
- *A es el rango de una función parcial computable.*

Ejercicio 1: ¿Es todo conjunto finito c.e.? ¿Y todo conjunto co-finito?

Solución: Dado un conjunto finito $C = \{x_1, \ldots, x_n\}$, podemos armar un programa P que devuelva 0 en esos elementos, y sino se cuelgue. Por ejemplo:

$$IF\ X = x_1\ GOTO\ [E]$$

$$\ldots$$

$$IF\ X = x_n\ GOTO\ [E]$$

$$[A]\ \ GOTO\ [A]$$

Sea f la función que P computa. Es fácil ver que $C = dom\ f$, así que C es el dominio de una función computable, y por lo tanto es c.e.

Otra manera de responder la pregunta es probando que C es computable (sugerencia: armar un programa con n IFs, como el de arriba) y entonces, por la slide 120 de la teórica, es c.e.

Para la otra pregunta, si C es co-finito, es decir, si $\bar{C}$ es finito, entonces $\bar{C}$ es computable. Es decir, existe g computable tal que $g(x) = \begin{cases} 1 \text{ si } x \in \bar{C} \\ 0 \text{ si no} \end{cases}$

y por lo tanto C también es computable, vía $h(x) = \neg\, g(x)$, es decir,

$$h(x) = \begin{cases} 0 \text{ si vale } g(x) \\ 1 \text{ si no} \end{cases}$$

Como C es computable, de nuevo tenemos que es c.e.

Recién usamos que si un conjunto es computable entonces su complemento también. ¿Vale lo mismo para c.e.?

Observación 4 *(Teórica) (Importante) C c.e. y co-c.e. $\Leftrightarrow$ C computable*

(Está bueno pensar cómo probar esto: ¿cómo armarse un programa que se fije si un elemento está en C sabiendo que tanto C como $\bar{C}$ son el rango de funciones computables?)

Si valiera que si C es c.e. entonces su complemento es c.e., entonces todo conjunto c.e. sería computable por la observación de arriba. Pero $K := \{x : \Phi_x(x) \downarrow\}$ es c.e., y no computable, así que:

Observación 5 *En general **no vale** C c.e. $\Rightarrow \bar{C}$ c.e.*

La siguiente definición es nueva:

Definición 6 *Sea $A, B \subset \mathbb{N}$. Se dice que A es reducible a B, y se nota $A \leq B$, si existe una función f **computable** (total) tal que $\forall\, x \in \mathbb{N}, x \in A$ sii $f(x) \in B$. Además decimos que $A \equiv B$ si $A \leq B$ y $B \leq A$.*

Pensemos que un conjunto puede ser computable. Si no es computable, al menos puede ser c.e. O también podría no ser ni siquiera c.e., como TOT. Si $A \leq B$, entonces B "acota" la "dificultad" de A.

Esto puede resultar intuitivo por la siguiente observación: supongamos que B es computable. ¿Cómo decidir si $x \in A$? Como B es computable, podemos simplemente chequear si $f(x) \in B$. Computar A se reduce a computar B. Pero esta cota de dificultad también se da en otros sentidos:

Ejercicio 2: Probar que si $A \leq B$, entonces B c.e implica A c.e.

Solución: Una posibilidad es construirnos una función h parcial computable tal que $A = dom\ h$. B es c.e. así que existe g computable tal que $Dom\ g = B$. Además, como $A \leq B$, existe f computable tal que $\forall\, x \in \mathbb{N}, x \in A$ sii $f(x) \in B$. ¿Cómo podemos armarnos la función h que necesitamos? Queremos que se defina solamente en los elementos de A. Tenemos que f manda elementos de A (y **sólo** los de A) a B, y g sólo se define en los elementos de B. Afirmamos que $Dom\ g \circ f = A$. En efecto, solo hay que seguir los si y solo si que cumple cada función:

$$x \in A \textit{ sii } f(x) \in B \textit{ sii } g(f(x)) \downarrow$$

Ejercicio 3 (tarea, y muy útil): Probar lo mismo, cambiando "c.e." por:

1. computable (formalizar lo discutido antes)
2. co-ce

Ejercicio 4: completar con $\leq, \geq$ ó $\equiv$

(El primer ítem fue agregado después de la clase)

a)

$$\{2 \cdot n : n \in \mathbb{N}\} \square \{4 \cdot n : n \in \mathbb{N}\}$$

Solución: Vale $\equiv$. Llamemos C_1 al primer conjunto, y C_2 al segundo.

$\leq$: Sea $f(x) := 2x$, claramente computable. Tenemos que ver que $\forall\ x \in \mathbb{N}$, $x \in C_1 \Leftrightarrow f(x) \in C_2$. Pero esto es fácil, porque $x \in C_1 \Leftrightarrow \exists\ n \in \mathbb{N}$ tal que $x = 2n \Leftrightarrow \exists\ n \in \mathbb{N}$ tal que $f(x) = 2 \cdot 2n = 4n \Leftrightarrow x \in C_2$, por definición de C_2.

$\geq$: para esta hay que tener un poco más de cuidado, porque dividir por 2 no funciona (¿por qué?). Pero aprovechando todo lo que hicimos en la práctica 1, podemos definir por ejemplo:

$$g(x) = \begin{cases} 2 \text{ si } x \in C_2 \\ 1 \text{ si no} \end{cases}$$

porque en la definición de reducción no se pide que la función sea *inyectiva*.

Claramente $x \in C_2 \Leftrightarrow g(x) \in C_1$, porque $\forall\ x \in C_2, g(x) = 2 \in C_1$, y $\forall\ x \notin C_2, g(x) = 1 \notin C_1$.

b)

$$\mathbb{N} \square \{2 \cdot n : n \in \mathbb{N}\}$$

Solución: Llamemos C_3 al conjunto de la derecha. Para ver que vale $\leq$ podemos hacer algo parecido a lo anterior, y usar la función $f(x) = 2x$. Trivialmente, todo $x \in \mathbb{N}$ cumple que $x \in \mathbb{N}$, así que lo que hay que ver es que $\forall\ x \in \mathbb{N}, f(x) \in C_3$, pero esto es obvio porque el conjunto es el de *los pares*, y la función *multiplica por 2*.

No vale $\geq$, y esto es triqui; en realidad $\mathbb{N}$ es un caso borde, medio patológico de este orden. Para probar que $\mathbb{N} \not\geq \{2 \cdot n : n \in \mathbb{N}\}$, hay que ver que $\forall\ g$ función computable, NO vale $x \in C_3 \Leftrightarrow g(x) \in \mathbb{N}$. Esto es más fácil de lo que parece. Sea $x = 1$. Tenemos que $x \notin C_3$, pero siempre, cualquiera sea g, $g(x) \in \mathbb{N}$, así que no se cumple el si y solo sí (más específicamente, no se cumple $\Leftarrow$, por contrarrecíproco).

c)

$$K \square \{x : \phi_x(x) \downarrow \wedge \phi_x(x) = 42^{42}\}$$

Idea: si un programa está definido en x puedo cambiar su valor a otro número.

Llamemos K_2 al conjunto de la derecha. Veamos que vale $\leq$. Queremos una función f computable tal que $x \in K \Leftrightarrow f(x) \in K_2$, es decir, queremos que $\phi_x(x) \downarrow \Leftrightarrow \phi_{f(x)}(f(x)) = 42$.

Citando a María Emilia Descotte, "estamos buscando entonces una f computable tal que cuando la uso como número de programa le pase algo particular, eso suena a teorema del parámetro."

Sea

$$g(x, y) = \begin{cases} 42^{42} & \text{si } \phi_x(x) \downarrow \\ \uparrow & \text{si no} \end{cases}$$

parcial computable. La variable y no hace nada, está simplemente para que cuando usemos el Teorema del Parámetro no obtengamos una función 0-aria (lo cuál sería un "abuso", como hicimos en clase).

Sea e tal que $\Phi_e^{(2)}(x,y) = g(x,y)$.

Por el teorema del parámetro, existe S_1^1 p.r. que fija la variable x, es decir, tal que $\Phi_e^{(2)}(x,y) = \Phi^{(1)}_{S_1^1(x,e)}(y)$.

Tenemos que $x \in K \Leftrightarrow S_1^1(x,e) = 42^{42} \Leftrightarrow S_1^1(x,e) \in \{x : \phi_x(x)\downarrow \ \wedge \phi_x(x) = 42^{42}\} = K_2$. La función que nos sirve, entonces, es $g(x) = S_1^1(x,e)$. (Notar que e es un número fijo.)

Ver si vale $\geq$ queda de ejercicio.

d) Algo un poquito más difícil de ver es lo siguiente, y lo dejamos para pensar:

$$K \square TOT$$

Para ver que **no** vale $\geq$, supongamos que vale. Entonces, por el **Ejercicio 3**, como K es c.e., tendríamos que TOT es c.e. Pero en la teórica vimos que esto no es cierto, absurdo! Así que $K \not\geq TOT$.

Más para pensar: ¿Siempre se pueden comparar dos conjuntos? ¿Es *total* el orden?

Un posible contraejemplo es $\emptyset$ y $\mathbb{N}$. Pero piensen uno menos trivial! Sugerencia: usen que c.e. y co-c.e. implica computable, y que si $A \leq B$ entonces B c.e. implica A c.e. y B co-c.e. implica B co-c.e.

Ejercicio 5: sea $A = \{x : \phi_x(x)\downarrow \wedge \phi_x(x) = 42^{42}\}$, como arriba. ¿Es A c.e.? ¿Es A co-c.e.?

Tenemos que $K \leq A$, así que A no puede ser co-c.e., porque eso implicaría que K es co-c.e., y por lo tanto computable.

Además es claro que A es c.e., porque es el dominio de la función:

$$g(x) = \begin{cases} 0 \text{ si } \phi_x(x)\downarrow \wedge\ \phi_x(x) = 42^{42} \\ \uparrow \text{ en caso contrario} \end{cases}, \text{ que es parcial computable.}$$

Ejercicio 6. Decidir V ó F y justificar.

- Sea A computable. Entonces existe f tal que el conjunto $f(A)$ es no computable.

  Falso, si $A = \emptyset$ entonces $f(A) = \emptyset$ y por lo tanto es computable.

- Sea $\emptyset \neq A$ computable y no vacío. Entonces existe f tal que $f(A)$ es no computable.

  Parecido. Falso. Si A es finito entonces $f(A)$ es finito y por lo tanto computable.

- Sea A computable e infinito. Entonces existe f tal que $f(A)$ es no computable.

  No le estamos pidiendo ninguna hipótesis a f. Si A es infinito entonces lo podemos biyectar con K. Numeremos a los elementos de los conjuntos. Por ejemplo, escribiéndolos de menor a mayor, tenemos que $A = \{a_1, a_2, \dots\}$ y $K = \{k_1, k_2, \dots\}$. Entonces podemos definirnos:

$$f(x) = \begin{cases} k_i \text{ si } x = a_i \\ 42^{42} \text{ en caso contrario} \end{cases}$$

Tal vez esto confunde porque estamos acostumbrados a funciones definidas por alguna fórmula, como $2x, 2^x, 3x + 1$, o por cómputos de programas, pero una función cualquiera no es nada más que un objeto matemático que le asigna un valor de la imagen ($\mathbb{N}$) a todo valor del dominio (también $\mathbb{N}$). [1]

- Sea A computable e infinito. Entonces existe f computable total tal que $f(A)$ no es computable.

  Este es un poco más triqui. Pensemos un poco...si $A = \mathbb{N}$ entonces la conjetura es cierta, porque cualquier conjunto c.e. es la imagen de una función computable. En otras palabras, $\exists f : \mathbb{N} \to \mathbb{N}$ computable tal que el conjunto c.e. es igual a $f(\mathbb{N})$. Por ejemplo, para el conjunto K, que es c.e., pero no computable, existe una función f de esta pinta (que "enumera" a K). Entonces $f(\mathbb{N}) = K$ y estamos.

  ¿Y si A no es $\mathbb{N}$? Bueno, si primero pudiéramos mandar A a $\mathbb{N}$ vía una función computable, despueś mandamos $\mathbb{N}$ a K y estamos. Es decir, queremos esto:

  $$A \xrightarrow{h} \mathbb{N} \xrightarrow{f} K$$

  Como A es computable, $\forall\ x \in \mathbb{N}$ podemos decidir si $x \in A$ o no. Esto hace las cosas muy fáciles. h puede ser la función que computa el siguiente programa:

  $$\begin{aligned} &WHILE(Z \leq X) \\ &\quad IF\ A(Z)\ THEN\ Y = Y + 1 \\ &\quad Z = Z + 1 \\ &Y = Y \dot{-} 1 \end{aligned}$$

  El programa no hace más que contar la cantidad de elementos que están en A que sean menores o iguales a x, y luego restar 1. (Ese restar uno está por el detalle de que queremos que algún elemento vaya al 0).

  Dicho más matematicosamente[2], para cada entrada x_1 devolvemos $|\{z : z \leq x \wedge z \in A\}| \dot{-} 1$. Notemos que la función, expresada así, resulta claramente computable porque es composición de funciones computables.

  Si nuevamente escribimos a A de manera ordenada, $A = \{a_1, a_2, \dots\}$, cuando el programa reciba como entrada a a_1 va a devolver 0, y en general va a mandar $a_i \mapsto i - 1$.

  Sea f la función computada por este programa. Como dijimos, $f(A) = \mathbb{N}$. Repitiéndonos un poco, para concluir el ejercicio basta agarrar un conjunto c.e. no computable cualquiera, como K. Sea g una función enumeradora de K. Entonces $g \circ f(A) = K$, y estamos.

[1] La definición formal que se suele ver en Álgebra 1 es con producto cartesiano...pero no importa.

[2] es más gracioso que decir "matemáticamente"