Conjuntos computablemente enumerables

Lógica y Computabilidad

Julián Dabbah (sobre una clase de María Emilia Descotte)

22 de septiembre de 2017

Repasito

- ¿ Qué quiere decir que un conjunto A sea c.e.?

  Respuesta: Varias definiciones equivalentes:

  - A es el dominio de una función parcial computable, o sea, existe $g : \mathbb{N} \to \mathbb{N}$ parcial computable tal que $A = Dom(g)$
  - A es la imagen (o rango) de una función parcial computable, o sea, existe $g : \mathbb{N} \to \mathbb{N}$ parcial computable tal que $A = Im(g)$
  - A es la imagen (o rango) de una función computable, o sea, existe $g : \mathbb{N} \to \mathbb{N}$ computable tal que $A = Im(g)$
  - A es la imagen (o rango) de una función p.r., o sea, existe $g : \mathbb{N} \to \mathbb{N}$ p.r. tal que $A = Im(g)$

- ¿ Y co-c.e.?

  Respuesta: Que $\overline{A}$ es c.e.

- Decidir si las siguientes afirmaciones son verdaderas o falsas:

  1. A computable $\Rightarrow$ A c.e.
     Respuesta: Verdadero (demo en la teórica)
  2. A c.e. $\Rightarrow$ A computable.
     Respuesta: Falso. Por ejemplo $A = K$
  3. A computable $\Rightarrow$ A co-c.e.
     Respuesta: Verdadero. Si A es computable, $\overline{A}$ también. Entonces por 1, $\overline{A}$ es c.e.
  4. A computable $\Leftrightarrow$ A c.e. y co-c.e.
     Respuesta: Verdadero (demo de $\Leftarrow$ en la teórica)
  5. K es c.e.
     Respuesta: Verdadero (demo en la teórica)
  6. K es co-c.e.
     Respuesta: Falso. Si lo fuese sería computable y ya sabemos que no lo es
  7. TOT no es ni c.e. ni co-c.e.
     Respuesta: Verdadero (demo en la teórica)

- **Reducciones de conjuntos** Si A y B son conjuntos de naturales, decimos que A es reducible a B ($A \leq B$) si existe $f : \mathbb{N} \to \mathbb{N}$ una función total computable tal que $x \in A$ sii $f(x) \in B$.

  Ejercicio (tarea) : Si $A \leq B$, entonces valen las siguientes implicaciones:

- B computable $\Rightarrow$ A computable.
- B c.e. $\Rightarrow$ A c.e.
- B co c.e. $\Rightarrow$ A co c.e.

Y por lo tanto, A no computable (resp. c.e., co-c.e.) implica B no computable (resp. c.e., co-c.e.).

Ejercicio 1

- Sea A un conjunto c.e., sea f una función parcial computable. Probar que $f(A) = \{f(x) : x \in A\}$ es c.e. ¿Y si f es total computable?
- Si ahora A es computable y f es total computable. ¿$f(A)$ es computable? ¿Y si A y f fueran p.r.?
- ¿Vale lo mismo para co-c.e.? Es decir, si A es co-c.e. y f es parcial computable, ¿$f(A)$ es co-c.e.?

Solución

- Como A es c.e., entonces para alguna función parcial computable g_A tenemos que $A = \{g_A(0), g_A(1), g_A(2), \dots$ $Im(g_A)$. Luego, $f(A) = \{f(g_A(0)), f(g_A(1)), f(g_A(2)), \dots\}$, o lo que es lo mismo, $f(A) = \{(f \circ g_A)(0), (f \circ g_A)(1), (f \circ g_A)(2), \dots\} = Im(f \circ g_A)$. Como f y g_A son parciales computables, tenemos que $f \circ g_A$ es parcial computable, y por lo tanto, su imagen es un conjunto c.e.
  Si f es total computable, en particular, es parcial computable, por lo tanto, la demostración anterior sigue valiendo.
- No. Recordemos que los conjuntos c.e. también se pueden enumerar con funciones totales computables. Luego, si tomamos $A = \mathbb{N}$, y B cualquier conjunto c.e., tenemos que $B = f(A)$, con A un conjunto computable y f una función total computable. Si la afirmación fuera cierta, tendríamos entonces que B es computable, o sea que habríamos probado que todos los conjuntos c.e. son computables, y ya sabemos que esto no es así. Por ejemplo, con este argumento, si tomáramos $B = K$ tenemos $K = f(\mathbb{N})$ para alguna f computable, sin embargo, ya sabemos que K no es computable.
  Recordemos también que los conjuntos c.e. también se pueden enumerar con funciones primitivas recursivas. Luego, como $\mathbb{N}$ es un conjunto p.r., vale exactamente la misma demostración que acabamos de dar.
  Lo que sí es cierto, es que como A p.r. implica A computable, que implica A c.e. y que f p.r. implica f total computable, que implica f parcial computable, podemos aplicar el resultado del ítem anterior y tenemos que $f(A)$ es un conjunto c.e. en cualquiera de todos estos casos.
- Observemos que para que $f(A)$ fuera co-c.e., deberíamos poder reconocer cuándo un número x es distinto de todas las imágenes de los elementos del conjunto. Esto suena un poco ambicioso, aún si pudiéramos reconocer si x no está en A. Tomemos $\mathbb{N}$ como antes, y, dado un i, sea

$$f_i(x) = \begin{cases} x & \text{si } \Phi_i(x)\downarrow \\ \uparrow & \text{en otro caso} \end{cases}$$

  Observemos que f_i es parcial computable sin importar quién sea el i, y además, $f_i(\mathbb{N}) = Dom\Phi_i$. Si fuera cierta la afirmación, debería ser $f_i(\mathbb{N})$ co-c.e.. Pero, también, $f_i(\mathbb{N})$ es el dominio de una función parcial computable, y por lo tanto, es c.e., con lo cual, tendríamos que $f_i(\mathbb{N})$ es computable para cualquier i, o lo que es lo mismo, al igual que antes, que todos los conjuntos c.e. son computables. En particular, si tomamos k tal que $\Phi_k(x)\downarrow \Leftrightarrow x \in K$, que existe por ser K c.e., tenemos que $f_k(\mathbb{N}) = Dom(\Phi_k) = K$. Luego, K sería co-c.e., y entonces sería computable, lo cual es absurdo.

1. Ejercicio 2

Decidir si los siguientes conjuntos son sólo c.e., sólo co-c.e., ambas o ninguna y justificar la respuesta:

- $VAC = \{i : W_i = \emptyset\}$
- $M = \{\text{mín}\, W_i : i \in \mathbb{N} \text{ y } W_i \neq \emptyset\}$

Solución:

- Recordemos que $W_i = \{x : \Phi_i(x) \downarrow\} = Dom(\Phi_i)$. Luego, por el Teorema de Rice, VAC no es computable, pues $i \in VAC$ si y sólo si $Dom\Phi_i = \emptyset$, con lo cual VAC es un conjunto de índices no trivial (queda como ejercicio la justificación detallada).
  Veamos ahora si puede ser c.e. o co-c.e.. De ser c.e., debería existir un programa que pueda terminar siempre que otro programa esté indefinido para todas las entradas, lo cual deberá sonar un poco pretencioso a esta altura. Pensemos ahora qué debería pasar para que VAC fuera co-c.e.: debería existir un programa que se detenga cuando encuentre que otro, para alguna entrada, está definido. Esto suena más razonable y, de hecho, podemos escribir la siguiente función parcial computable que lo realiza:
  $$NoVacio(p) = \exists_{\langle x,t\rangle} STP(x, p, t)$$
  $NoVacio$ es parcial computable por ser un existencial no acotado sobre un predicado total computable (ver práctica 2) y además está definida sólo en los casos en los que existe alguna entrada para la cual el programa p termina. En definitiva:
  $$NoVacio(p) \downarrow \Leftrightarrow Dom(\Phi_p) \neq \emptyset \Leftrightarrow p \notin VAC \Leftrightarrow p \in \overline{VAC}$$
  y por lo tanto, VAC es co-c.e. Y también, entonces, como ya sabemos que no es computable, podemos afirmar que no es c.e.

- Decidir si un número es el mínimo del dominio de una función parcial computable cualquiera debería sonarnos muy incomputable (y está bien que así sea). Sin embargo, si miramos atentamente quién es en realidad M, vemos que no sólo es computable, sino que también es p.r. pues es $\mathbb{N}$.
  Para ver esto, dado que ya sabemos que $M \subseteq \mathbb{N}$, veamos que $\mathbb{N} \subseteq M$, es decir, que para cualquier $n \in \mathbb{N}$ tenemos que n es el mínimo del dominio de alguna función parcial computable, lo cual ya suena mucho más razonable. Hay infinitas, pero nos alcanza con mostrar una, por ejemplo:
  $$f_n(x) = \begin{cases} 1 & \text{si } x = n \\ \uparrow & \text{si no} \end{cases}$$
  Como f_n es parcial computable, entonces la computa Φ_e para algún $e \in \mathbb{N}$, y por cómo la definimos, tenemos que $n = \text{mín}\, W_e$ y $W_e \neq \emptyset$, con lo cual $n \in M$. Como n puede ser cualquier natural, tenemos $\mathbb{N} \subseteq M$, que era lo que nos faltaba probar para ver que $M = \mathbb{N}$.

Ejercicio 3

Decidir si los siguientes conjuntos son sólo c.e., sólo co-c.e., ambas o ninguna y justificar la respuesta:

a) $A = \{\langle \#P, k\rangle \mid \Psi_P^{(1)}(k) \downarrow \text{ y } \Psi_P^{(1)}(k+1) \downarrow \text{ y } \Psi_P^{(1)}(k) + 1 = \Psi_P^{(1)}(k+1)\}$.

b) $B = \{\#P \mid \Psi_P^{(1)} \text{ es total y } \Psi_P^{(1)}(x) < \Psi_P^{(1)}(x+1)\ \forall\ x \in \mathbb{N}\}$.

Solución:

a) Veamos que es c.e. pero no co-c.e. pues no es computable.

Lo primero, y más fácil, es probar que no es computable, pues podemos aplicar el teorema de Rice. Para eso, reduzcámoslo computablemente a un conjunto que sólo dependa de números de programa. Por ejemplo, fijando k en 35, definimos $\tilde{A} = \{x | \langle x, 35\rangle \in A\}$ y tenemos que $\tilde{A} \leq A$ tomando $f(x) = \langle x, 35\rangle$. Pero resulta también que, por cómo lo definimos:

$$\tilde{A} = \{x |\ \Psi_x^{(1)}(35)\downarrow \text{ y } \Psi_x^{(1)}(36)\downarrow \text{ y } \Psi_x^{(1)}(35)+1 = \Psi_x^{(1)}(36)\}$$

Con lo cual, $\tilde{A}$ es el conjunto de todos los programas que computan funciones que están definidas en 35 y en 36, y que en 36 valen lo que valen en 35 más 1. Queda como ejercicio convencerse formalmente de que esto es un conjunto de índices no trivial, y después podemos afirmar que, por el Teorema de Rice, $\tilde{A}$ no es computable y entonces tampoco puede serlo A.

Ahora nos queda decidir si puede ser c.e. o co-c.e.. Parece más razonable que sea c.e., dado que si la propiedad vale para algún número, entonces puede existir un programa que se detenga para esa entrada. Por el contrario, para ser co-c.e., deberíamos poder detectar indefiniciones antes de efectivamente chequear la segunda parte de la condición.

Convencidos de esto, escribimos el siguiente programa, que, efectivamente, se detiene siempre que $X_1 = \langle \#P, k\rangle$ pertenece a A.

$$\begin{array}{ll} [A] & \text{IF } \Phi^1_{l(X_1)}(r(X_1))+1 = \Phi^1_{l(X_1)}(r(X_1)+1) \text{ GOTO } E \\ & \text{GOTO } A \end{array}$$

Observemos que tanto como si P no está definido en k o en $k+1$ como si la condición sobre el resultado de P en estos números no se cumple, el programa anterior no termina, lo cual es exactamente lo que queríamos (pues además, en los otros casos, es decir, cuando vale la propiedad entera, termina siempre). Luego, tenemos que $A = Dom\Psi_P^{(1)}$, con lo cual A es c.e..
Finalmente, como ya sabemos que A no es computable, podemos concluir que, dado que es c.e., no puede ser co-c.e..

b) Si observamos la condición de pertenencia de B vemos que B es el conjunto de los programas que computan funciones totales y estrictamente crecientes. Esto ya nos da la idea de que B no puede ser computable (por el Teorema de Rice), pero el panorama parece un poco peor que en el ejercicio anterior. Si quisiéramos probar que B es c.e., deberíamos poder decidir, primero, si x se corresponde con una función total (lo cual ya sabemos que no es posible) y, después, probar sobre Φ_x una propiedad sobre los infinitos números de su dominio. Por otro lado, si quisiéramos probar que es co-c.e., deberíamos probar que su complemento es c.e., o sea que para

$$\overline{B} = \{\#P | \Psi_P^{(1)} \text{ no es total ó } \Psi_P^{(1)}(x) \geq \Psi_P^{(1)}(x+1) \text{ para algún } x \in \mathbb{N}\}$$

existe un programa que termina siempre que $\#P \in \overline{B}$. Este programa debería detectar que P se va a indefinir en alguna de sus entradas y, además, si esto no pasara, debería detenerse si existe algún x para el que no se cumple la segunda parte de la condición. Tampoco parece muy razonable que este programa exista.
Luego, dado que queremos probar que no es c.e. ni co-c.e., podemos usar una reducción a un conjunto que no sea c.e. ni co-c.e.. TOT parece un gran candidato, pues no sólo no es c.e. ni co-c.e., sino que dado que ya habla de la totalidad de las funciones, nos debería facilitar la parte de armar la función de la reducción. Esta función, llamémosla f, debe ser tal que:

$$\Phi^{(1)}_{(x)} \text{ es total} \quad \Leftrightarrow \quad \Phi^{(1)}_{f(x)} \text{ es total y estrictamente creciente}$$

O sea, que queremos una f computable que tome un número de programa y lo convierta en el número de otro programa que se comporta de una manera particular. Ya sabemos que contamos con la función del parámetro para hacer este tipo de cosas, veamos cómo podemos aplicarla.
En particular, queremos construir, utilizando el Teorema del Parámetro, una función que tome un número de programa x y sea total y creciente si y solo si x, mirado como programa, es total. Para eso, necesitamos partir de una función parcial computable que haga algo parecido. Parece razonable, entonces, definir una g de la siguiente manera:

$$g(x,y) = \begin{cases} y & \text{si } \Phi^{(1)}_x(y)\downarrow \\ \uparrow & \text{en otro caso} \end{cases}$$

Observemos, que fijado x, si x como programa (es decir, Φ_x) es total, siempre vale el primer caso, y por lo tanto, g es la identidad sobre y, que es estrictamente creciente, mientras que si esto no pasa, la función se indefine para todos los y y por lo tanto, no es total. Como g es parcial computable, existe e tal que $g(x,y) = \Phi^{(2)}_e(x,y)$. Luego, por el Teorema del Parámetro, tenemos que

$$\Phi^{(1)}_{S^1_1(x,e)}(y) = \Phi^{(2)}_e(x,y) = g(x,y) = \begin{cases} y & \text{si } \Phi^{(1)}_x(y)\downarrow \\ \uparrow & \text{en otro caso} \end{cases}$$

donde podemos observar que $S^1_1(x,e)$ mirado como número de programa es total y estrictamente creciente sii x como número de programa es total. Luego, tomemos $f(x) = S^1_1(x,e)$ como la función de la reducción a TOT y veamos que $x \in TOT \Leftrightarrow f(x) \in B$.

$x \in TOT \Rightarrow \quad \Phi^{(1)}_x(y)\downarrow \forall y \in \mathbb{N} \Rightarrow \quad g(x,y) = y \ \forall y \in \mathbb{N} \Rightarrow \quad \Phi^{(2)}_e(x,y) = y \ \forall y \in \mathbb{N} \Rightarrow$
$\Rightarrow \quad \Phi^{(1)}_{f(x)}(y) = y \ \forall y \in \mathbb{N} \Rightarrow \quad \Phi^{(1)}_{f(x)}$ es total y creciente $\Rightarrow \quad f(x) \in B$

$x \notin TOT \Rightarrow \quad \exists y \in \mathbb{N} \ |\Phi^{(1)}_x \uparrow \Rightarrow \quad \exists y \in \mathbb{N} \mid g(x,y)\uparrow \Rightarrow \exists y \in \mathbb{N} \mid \Phi^{(2)}_e(x,y)\uparrow \Rightarrow$
$\Rightarrow \quad \exists y \in \mathbb{N} \ |\Phi^{(1)}_{f(x)}(y)\uparrow \Rightarrow \quad f(x)$no es total $\Rightarrow \quad f(x) \notin B$

Como además S^1_1 es primitiva recursiva, tenemos que f es computable, con lo cual tenemos que $TOT \leq B$, y por lo tanto, B no puede ser c.e. ni co-c.e. pues TOT no es ninguno de los dos.

Ejercicio 4

Decidir si las siguientes afirmaciones son verdaderas o falsas y justificar la respuesta.

1. Si $f :\rightarrow$ y tanto su dominio como su imagen son computables, entonces f es computable.
2. Si $A \subset$ tal que $\forall\ S \subset A$ finito, S es computable, entonces A es computable.
3. Si $f_i : \mathbb{N} \to \mathbb{N}$ (con $i \in \mathbb{N}$) es una familia de funciones computables, entonces $F(i) = f_i(i)$ es computable.
4. Si A es infinito y no computable, entonces para todo subconjunto $B \subset A$ infinito, se tiene que B no es computable.

*Solución

1. Falso. Por ejemplo $HALT$.
2. Falso. Todos los conjuntos finitos son computables, luego la hipótesis vale para todos los conjuntos, por ejemplo, para K que ya sabemos que no es computable.
3. Falso. Si tomamos $f_i(x) = \begin{cases} 1 & \text{si } \Phi_i(i)\downarrow \\ 0 & \text{en otro caso} \end{cases}$, todas estas son computables pues son constantes, pero $F(i) = HALT(i,i)$.
4. Falso. Contraejemplo: $A = K$, y $B = \{\#P_i : i \in \mathbb{N}\}$, donde P_i es el programa: $X_i \leftarrow X_i + 1$. Ejercicio: justificar que $B \subset K$ y que B es computable (de hecho, p.r.).