Lógica y Computabilidad
1er cuatrimestre de 2005
Práctica 9

Enrique A. Tobis

Las explicaciones que siguen son en mayor o menor grado de desarrollo, las ideas que tuve para resolver los ejercicios de las prácticas de Lógica y Computabilidad cuando fui ayudante en la materia. Además de mis ideas, incorporan cambios y correcciones que me fueron sugiriendo los estudiantes. SE PRESENTAN SIN NINGÚN TIPO DE AVAL DE LA FCEN, ni garantía de correctitud. Por supuesto, puse mi mejor fe en hacerlas, pero no me puedo hacer responsable por errores u omisiones que el lector encuentre en ellas.

Desde ya, toda corrección, sugerencia o comentario que desee hacerme llegar será bienvenido. Me puede contactar escribiendo a **etobis@dc.uba.ar**.

Ejercicio 1 Como f es computable, tenemos un programa $\mathcal{P}$ tal que $\psi_\mathcal{P} = f$. Dado n, vamos calculando $f(0)$, $f(1)$, y así siguiendo con el STP. En algún momento, $f(i) = n$, porque la función es total y biyectiva. Devolvemos ese valor. Es fácil armar un programa que haga esto.

Ejercicio 2 Supongamos que $f(x)$ es recursiva. Entonces, $\text{Gf}_f(x, y)$ resulta trivialmente recursiva: basta calcular $f(x)$ y ver si es y. Supongamos ahora que $\text{Gf}_f(x, y)$ es recursiva. Como es la función característica del gráfico de una función, $\forall x \exists! y\ \text{Gr}_f(x, y) = 1$, pues todo punto debe tener una, y solo una, imagen. Entonces, corremos $\text{Gr}_f(x, i)$, aumentando el i, hasta llegar al que nos da 1.

Ejercicio 3 Podemos definir $g(x) = \text{mín}_z(f(z) = x)$. Esta función está bien definida, porque f es suryectiva. Entonces, siempre hay algún z que cumple $f(z) = x$ para cualquier x. Además de estar bien definida, es el resultado de aplicar minimización propia a una función parcialmente computable y total. Entonces, es parcialmente computable y total. Los conjuntos $\{z|f(z) = x\}$ y $\{z|f(z) = y\}$ son disjuntos para $y \neq x$. Entonces, $g(x) \neq g(y)$ para $x \neq y$, por lo que es inyectiva. Si evaluamos $g(f(x)) = \text{mín}_z(f(z) = f(x))$. Como x cumple esto, el mínimo va a ser menor o igual que x, y g cumple todo lo pedido.

Ejercicio 4 Supongamos que $\text{HALT}(X, X)$ es computable. Con él, construimos el programa $\mathcal{P}$

$[A]$ IF $\text{HALT}(X, X)$ GOTO A

$\mathcal{P}$ para con input X sii $\neg\text{HALT}(X, X)$. Sea $y_0 = \#(\mathcal{P})$. Entonces, $\text{HALT}(y_0, y_0) \Leftrightarrow y_0$ para con input $y_0 \Leftrightarrow \neg\text{HALT}(y_0, y_0)$, lo cual nos da un absurdo.

Ejercicio 5 Supongamos que $\text{HALT}(f(x), x)$ sea parcialmente computable. Entonces, tenemos un programa que lo computa. Sabemos que f es biyectiva y computable. Entonces, por el ejercicio 1, sabemos que f^{-1} también es computable. Luego, el siguiente programa computa $\text{HALT}(x, f^{-1}(x))$.

$Z \leftarrow f^{-1}(X)$
$Y \leftarrow \text{HALT}(f(Z), Z)$

Entonces, el programa

$[A]$ IF $\text{HALT}(X, f^{-1}(X))$ GOTO A
$Y \leftarrow 1$

computa la h de la sugerencia. Sea h_0 el número de este programa. Observemos que $h_0 = f^{-1}(f(h_0))$. Veamos qué pasa con $h(f(h_0))$. Esto puede ser 1 o $\uparrow$. Si es 1, entonces $\Phi(f(h_0), h_0) \uparrow$. Entonces $h(f(h_0)) \uparrow$. Pero esto es una contradicción. Ahora, si es $\uparrow$, entonces $\Phi(f(h_0), h_0) \downarrow$. Luego, $h(f(h_0)) \downarrow$. Nuevmente tenemos una contradicción. Como lo único que supusimos era que $\text{HALT}(f(x), x)$ era parcialmente computable, esto es falso.

Ejercicio 6 Es computable. Vamos simulando los pasos de $f(x)$. Si en algún momento para, entonces x pertenece al dominio de f y devolvemos un 1. Si nunca para, entonces, x no pertenecía al dominio de f, y está bien que se cuelgue.

Ejercicio 7 Vamos simulando $\Phi(x, x)$, si para, devolvemos el resultado más 1. Si nunca para, entonces está bien que se cuelgue.

Ejercicio 8 Supongamos que $f(x, y)$ es computable. Sea y_0 un natural. Tenemos que $\chi_A(x) = f(x, y_0)$ también tiene que ser computable. Pero χ_A es la función característica del conjunto $A = \{x \in \mathbb{N} | y_0 \in \text{Im } \Phi_x\}$, que tiene que ser recursivo. Entonces, también tiene que ser recursivo $B = \{\Phi_x | y \in \text{Im } \Phi_x\}$. Pero la función que devuelve constantemente $y_0 + 1$ no está en B, y la función que devuelve constantemente y_0 sí. Entonces, por el Teorema de Rice, B no es recursivo. Por ende tampoco lo son A, χ_A y f.

Ejercicio 9

Ejercicio 10

Ejercicio 11

Ejercicio 12

Ejercicio 13

1. Si miramos el conjunto $\{\Phi_x | \text{Dom}\Phi_x = \emptyset\}$, por Rice no es recursivo. Luego, tampoco lo es el que nos dieron.

Ejercicio 14

1. La función característica de este conjunto es

$$f(x,y) = \begin{cases} 1 & \text{si } y \in \text{Im}_{\Phi_x} \\ 0 & \text{si no} \end{cases}$$

   En el ejercicio 8 vimos que esta función no es computable. Por lo tanto, el conjunto no es recursivo.

2. Si llamamos A al conjunto y suponemos que es recursivo, entonces

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

   también tiene que ser recursiva. Pero entonces, si y_0 es tal que $\Phi_{y_0} = 2$, la función constante 2., entonces

$$\chi_B = \chi_A(x, y_0) = \begin{cases} 1 & \text{si } \Phi_x = 2 \\ 0 & \text{si no} \end{cases}$$

   también tiene que ser recursiva. Pero esta es la función característica del conjunto $B = \{x \in \mathbb{N} | \Phi_x = 2\}$. Si consideramos $\mathcal{F}_B = \{\Phi_x | \Phi_x = 2\}$, es fácil ver que este conjunto no contiene a todas las funciones parcialmente computables. Entonces, por Rice, no es recursivo. Esto nos lleva a un absurdo, que dice que suponer que A era recursivo estuvo mal.

3. Consideramos $\mathcal{F} = \{\Phi_x | \text{Im}\Phi_x \text{ es infinito}\}$. Por Rice, este conjunto no es recursivo, luego tampoco lo es el original.

Ejercicio 15

1. Si f fuera recursiva, tendría que existir e tal que $\Phi_e = f$. Entonces

$$\Phi_e(e) = \begin{cases} 1 \text{ si } \Phi_e(e) = 0 \\ 0 \text{ si } \Phi_e(e) \neq 0 \end{cases}$$

   lo cual nos muestra que no puede existir un tal e. Luego, f no es computable.

2. Supongamos que f sea recursiva. Entonces, sea x_0 tal que Φ_{x_0} es la función constante 0. Entonces, si f es recursiva, $g(y) = f(x_0, y)$ también lo es. Pero

$$g(y) = \begin{cases} 1 & \text{si } \Phi_y(y) \neq 0 \\ 0 & \text{si } \Phi_y(y) = 0 \end{cases}$$

   Si esta función fuera recursiva, también lo sería $\alpha(g)$. Pero esta es la función del ejercicio anterior, que ya vimos que no es recursiva. Entonces, nuestra f tampoco lo era.

3. Si fuera computable, también lo sería $g(x,y) = f(x,y,z_0)$, donde z_0 es tal que Φ_{z_0} es la función constante 0. Y por consecuencia, también lo sería $g(x,x)$. Pero esta función es la del inciso a, que vimos que no era computable. Entonces, nuestra f tampoco lo es.

Ejercicio 16 Para probar esto, primero probamos un resultado adicional. Si $B = \{f(n) | n \in \mathbb{N}\}$, con f una función estrictamente creciente, parcialmente computable y total, entonces B es recursivo. Para verlo, basta ver que el siguiente programa

$[A]$ IF $f(Z) = X$ GOTO B
IF $f(Z) > X$ GOTO E
$Z \leftarrow Z + 1$
GOTO A
$[B]$ $Y \leftarrow 1$
GOTO E

computa χ_B. Este programa termina siempre, así que B es recursivo.

Ahora, sea A r.e. e infinito. Tenemos una función primitiva recursiva $g(x)$ que nos va dando los elementos de A. Definimos $f(0) = g(0)$, y suponiendo definida $f(k)$ para todo $k < n$, definimos $f(n) = g(\text{mín}_z(g(z) > f(n-1)))$. Esta minimización es propia, pues A es infinito. Entonces, siempre podemos encontrar un elemento de A mayor que todos los que tengamos. Esto nos da una f parcialmente computable, total, y estrictamente creciente. Si tomamos $B = \{f(n) | n \in \mathbb{N}\}$, tenemos que B es recursivo y que $B \subseteq A$.

Ejercicio 17

1. $A \cup B$ es r.e.: voy dando un elemento de A y uno de B. Para que sea más bonito, guardo los que di y no repito.

   $A \cap B$ también es r.e.: Voy guardando dos listas, la de elementos de A y la de elementos de B. Genero un número de cada conjunto. Cada vez que genero, me fijo si está en la lista del otro. Si está, lo doy. Si no, por el momento no.

2. Es falsa. Si tomamos B los programas que paran cuando se les da como entrada su número de programa, y A todos los programas, entonces $A \backslash B$ son los programas que no se detienen cuando se les da su número de programa como entrada. Si este conjunto fuera r.e., $\text{HALT}(X, X)$ sería computable.

3. Es falso. Sea $B = \mathbb{N}$, pensados como números de programas. Sea, A el conjunto de números de programas que computan la función vacía (la que no está definida en ningún lado). En el ejercicio 20.c veremos que este subconjunto no es r.e.

4. Es falso. Ni $\text{HALT}(X, X)$ es recursivo, ni su complemento, pero es r.e.

5. Es verdadero. Con un argumento diagonal, voy dando un elemento del primero, uno del segundo, otro del primero, otro del segundo, uno del tercero, etc.

Ejercicio 18

Ejercicio 19 Para ver que B es r.e., vamos simulando todos los programas, y listamos los que van parando. Para ver que no es recursivo, usamos Rice con el conjunto $\{\Phi_x | \Phi_x(0) = 1\}$.

Ejercicio 20

1. Es r.e.: voy simulando $\Phi_x(0)$ para todos los x, en orden. A medida que paran, los voy listando. Para ver que no es recursivo, aplico Rice al conjunto $\{\Phi_x | \Phi_x(0) \downarrow\}$

2. Es r.e.: idem el inciso anterior, pero con entrada x. No es recursivo, pues es $\text{HALT}(x, x)$.

3. Aplicando Rice al conjunto $\{\Phi_x | \text{Dom}\Phi_x = \emptyset\}$ se ve que no es recursivo. Ahora, en forma análoga al inciso 1, se puede ver que $A_k = \{x \in \mathbb{N} | \Phi_x(k) \downarrow\}$ es r.e. para todo $k \in \mathbb{N}$. Por el ejercicio 17, inciso 5, el conjunto $\bigcup_{k=0}^{\infty} A_k$ es r.e. Este conjunto es el complemento del que nos piden estudiar. Si el que nos piden fuera r.e., como su complemento también lo es, sería recursivo. Como no lo es, no es tampoco r.e.

4. El conjunto es completamente análogo al del inciso a, pero cambiando 0 por 1.

Ejercicio 21 Se deduce de combinar los teoremas 4.9 y 4.10 del libro de Davis.

Ejercicio 22 En palabras, por el ejercicio anterior tenemos una f parcialmente computable y total que nos va dando los elementos de B. Entonces, armamos un programa que vaya enumerando esos elementos, pero que guarde los elementos que va dando como salidas. El siguiente programa nos da, entonces, esa tal función modificada.

$[A]$ $Z_2 \leftarrow f(Z)$
IF Z_2 NO ESTA EN LA LISTA Z_3 GOTO B // Z_3 guarda los elementos que doy como salida
$Z \leftarrow Z + 1$ // Si ya estaba en la lista,
GOTO A //genero otro elemento
$[B]$ IF $\mathrm{Lt}(Z_3) = X$ GOTO C // Si ya generé tantos elementos como me pidieron
AGREGO Z_2 A LA LISTA Z_3 // Si no, genero uno más
$Z \leftarrow Z + 1$
GOTO A
$[C]$ $Y \leftarrow Z_2$

Como este programa termina para todo X de entrada, tenemos que nuestro conjunto es la imagen de una función parcialmente computable, total e inyectiva.

Ejercicio 23

Ejercicio 24

