LyC 2023-Verano - Resolución Parcial Computabilidad

AVISO: Este documento se entrega sólo a efectos de compartirles cómo se pueden plantear cada uno de los ejercicios. Esto no quiere decir que lo presentado sea la única ni la mejor solución para cada punto, ni que pueda estar exenta de errores involuntarios.

1 Ejercicio 1

1.1 Inciso a

Defino una nueva función F:

$$F(0) =< k_1, k_2 >$$
$$F(t+1) =< r(F(t)), g(max\{l(F(t)), r(F(t))\}, t+1) >$$

Como F queda definida por el esquema de recursión primitiva a partir de funciones primitivas recursivas (observadores y codificadores de tuplas son p.r., máximo de dos números es p.r., composición de p.r. es p.r. y estamos suponiendo g p.r.), entonces F es p.r. Como $h(t) = l(F(t))$, entonces h es p.r. por ser composición de p.r.

1.2 Inciso b-I

Tomando como la definieron en la teórica:

$$g(x, y) = HALT(y, x)$$

Si g fuera computable, entonces $f(x, y) = g(U_2^2(x, y), U_1^2(x, y)) = HALT(x, y)$ sería computable (composición de computables es computable), y ya sabemos que no lo es. Como g no es computable, no puede ser p.r.

1.3 Inciso b-II

Puedo tomar $k_1 = k_2 = 1$. Verificando con la codificación de la teórica, el número 1 codifica al programa cuya lista de instrucciones tiene número 2, es decir, es la lista [1]. Entonces es un programa de una sola instrucción codificada por una tripla tal que $< a, < b, c >>= 1$. Se puede verificar que para que eso pase tiene que suceder que $a = 1$ y $< b, c >= 0$, y entonces $b = c = 0$. Entonces el programa de número 1 es:

$$[A]\ Y \leftarrow Y$$

Es decir, es un programa que está definido para toda entrada. Si verificamos que pasa con h para estos valores de k_1 y k_2, vemos que $h(0) = h(1) = 1$. Por lo tanto, $h(2) = g(max\{1, 1\}, 1) = g(1, 1) = 1$ porque el programa 1 está definido para toda entrada, y siguiendo este misma razonamiento podemos observar que la función h va a ser la función constante 1, que es p.r.

2 Ejercicio 2

2.1 Inciso a

El siguiente programa computa la función que devuelve 1 si $X_!$ es par y se cuelga sino:

$$
\begin{aligned}
& Z_1 \leftarrow X_1 \\
& IF\ Z_1 = 0\ GOTO\ F \\
[A]\ & Z_1 \leftarrow Z_1 - 1 \\
[B]\ & IF\ Z_1 = 0\ GOTO\ B \\
& Z_1 \leftarrow Z_1 - 1 \\
& IF\ Z_1 = 0\ GOTO\ F \\
& GOTO\ A \\
[F]\ & Y \leftarrow 1
\end{aligned}
$$

2.2 Inciso b

Dado el programa P:

$$
\begin{aligned}
\Psi_P^{(1)}(X_1) &= X_1 \\
\Psi_P^{(2)}(X_1, X_2) &= X_1 + 2 * X_2 \\
\Psi_P^{(3)}(X_1, X_2, X_3) &= X_1 + 2 * X_2
\end{aligned}
$$

En el primer caso, como consideramos al programa que toma una sola variable, $X_i = 0$ para todo $i \neq 1$, en particular $X_2 = 0$. Por lo tanto la condición del primer IF va a ser siempre verdadera y el programa va a devolver siempre X_1.
En el segundo caso, la condición del IF va a depender del valor de X_2, que ya no es siempre 0. Por lo tanto, el programa le va a sumar 2 a la variable Y (a la cual empieza asignándole X_1), hasta que X_2 llegue a 0. Es decir, devuelve el resultado de sumarle dos X_2 veces a X_1.
En el tercer caso, como la variable X_3 nunca se menciona en el programa, el resultado es el mismo que en el caso anterior.

2.3 Inciso c

Vamos a demostrarlo por inducción en la cantidad de instrucciones del programa. Llamemos n a esta cantidad:

Caso base n=0: Si tenemos el programa vacío P (que seguro no tiene ninguna instrucción IF), trivialmente $\Psi_P(x) \downarrow$ y $\Psi_P(x) = 0 \leq |\#(P) + 1| = 0$ para todo x.

Caso inductivo: nuestra hipótesis inductiva es la siguiente: para todos los programas de longitud menor o igual a n que no tienen instrucciones IF, esos programas terminan para toda entrada y cumplen que $\Psi_P(x) \leq |\#(P) + 1|$ para todo $x \in \mathbb{N}$.
Sea entonces un programa P de $n+1$ instrucciones que no tiene instrucciones IF. Podemos escribir a P como:

$$
\begin{aligned}
& Q \\
& R
\end{aligned}
$$

donde Q es un programa de longitud n y R es un programa de una sola instrucción. Como el programa P no tiene instrucciones IF, seguro Q tampoco, y la instrucción de R no puede ser un IF. Por hipótesis inductiva, vale que el programa Q está definido para toda entrada y $\Psi_Q(x) \leq |\#(Q) + 1|$ para todo $x \in \mathbb{N}$. Analicemos qué puede pasar con la única instrucción de R:

1. Si la única instrucción de R no involucra a la variable Y, o involucra a la variable Y pero es sólo una asignación ($Y \leftarrow Y$ con alguna etiqueta para que el programa sea válido), como sabemos que la instrucción no puede ser un IF, entonces debe ser una suma, resta o asignación, y el programa P va a terminar siempre exactamente un paso después que el programa Q. Además, $\Psi_P(x) = \Psi_Q(x) \leq |\#(Q) + 1| = n < n + 1 = |\#(P) + 1|$ para todo x, y la propiedad vale.

2. Si la única instrucción de R involucra a Y y es una resta ($Y \leftarrow Y - 1$), entonces nuevamente el programa P siempre termina exactamente un paso después que Q y $\Psi_P(x) \leq \Psi_Q(x) \leq |\#(Q) + 1| = n < n + 1 = |\#(P) + 1|$ para todo x.

3. Si la única instrucción de R es $Y \leftarrow Y + 1$, entonces de vuelta el programa P siempre termina exactamente un paso después que Q y $\Psi_P(x) = \Psi_Q(x) + 1 \leq |\#(Q) + 1| + 1 = n + 1 = |\#(P) + 1|$ para todo x.

Como vimos todas las opciones posibles ya que la única instrucción de R no puede ser un IF, demostramos que seguro $\Psi_P(x) \downarrow$ y $\Psi_P(x) \leq |\#(P) + 1|$ para todo $x \in \mathbb{N}$.

3 Ejercicio 3

Supongo que g es computable y defino la siguiente función:

$$h(x) = \begin{cases} 1 & si \quad g(|x+1|, x) > 0 \; (\iff \; \Phi_x(2023)\downarrow) \\ 0 & sino \end{cases}$$

Noto primero que, suponiendo que g es computable, h también lo es, ya que está definida por casos y eso, suponiendo que los predicados de las condiciones son totales computables, entonces también es computable. Noto también que $g(|x+1|, x) > 0 \iff \Phi_x(2023)\downarrow$, ya que al evaluar $g(|x+1|, x)$, la condición $|x+1| = |x+1|$ va a ser siempre trivialmente cierta. Propongo ahora tres formas distintas de demostrar que h en realidad no es computable.

3.1 Teorema del Parámetro

Defino la siguiente función:

$$f(x, y) = \begin{cases} 1 & si \quad HALT(x, x) \\ \uparrow & sino \end{cases}$$

f es parcial computable, ya que el siguiente pseudo-programa la computa:

$$Z_1 \leftarrow \Phi_x(x)$$
$$Y \leftarrow 1$$

Entonces hay algún número de programa e tal que $\Phi_e(x, y) = f(x, y)$. Por lo tanto, por teorema del parámetro, existe alguna función p.r. S tal que $\Phi_e(x, y) = \Phi_{S(x,e)}(y)$. Ahora:

$$h(S(x, e)) = 1 \iff \Phi_{S(x,e)}(2023)\downarrow \iff \Phi_e(x, 2023)\downarrow \iff f(x, 2023)\downarrow \iff HALT(x, x)$$

Por lo tanto, como h es un predicado, $h(S(x, e)) = HALT(x, x)$, entonces h no puede ser computable, pues si lo fuera, como S es p.r., la composición sería computable, y ya sabemos que HALT no es computable.
Como h no es computable, g tampoco puede serlo, pues si lo fuera, como h es un caso particular de g, h también lo sería.

3.2 Teorema de la Recursión

Defino ahora:

$$f(x,y) = \begin{cases} 1 & si \quad h(x) = 0 \\ \\ \uparrow & si \quad h(x) = 1 \end{cases}$$

Como estamos suponiendo que h es computable, entonces seguro f también lo es. Por lo tanto, por teorema de la recursión, existe un número e tal que

$$\Phi_e(y) = f(e,y) = \begin{cases} 1 & si \quad h(e) = 0 \\ \\ \uparrow & si \quad h(e) = 1 \end{cases}$$

Ahora:

$$h(e) = 1 \iff \Phi_e(2023)\downarrow \iff f(e,2023)\downarrow \iff h(e) = 0$$

ya que $f(e,y)$ va a estar definida únicamente en el primer caso, es decir, si $h(e) = 0$. Llegamos a un absurdo, que vino de suponer que h era computable, y por lo mismo que antes g no es computable.

3.3 Teorema de Rice

Considero el conjunto

$$A = \{x \mid \Phi_x(2023)\downarrow\}$$

Se ve inmediatamente que la indicadora de A es la función h. Alcanza entonces con ver que A no es computable, y para eso podemos ver que es un conjunto de índices no trivial:

- $\star$ $0 \in A$ pues $\Phi_0(x)\downarrow$ para todo x.
- $\star$ Sea e el número de algún programa que se cuelga para toda entrada, entonces $\Phi_e(2023)\uparrow$ y $e \notin A$.
- $\star$ Sea P un programa tal que $\#(P) \in A$ y sea Q otro programa tal que $\Psi_P = \Psi_Q$ y $\#(P) \neq \#(Q)$. $\#(P) \in A \iff \Psi_P(2023)\downarrow \iff \Psi_Q(2023)\downarrow \iff \#(Q) \in A$, pues computan la misma función ($\Psi_P(x)\downarrow \iff \Psi_Q(x)\downarrow$), entonces $\#(Q) \in A$.

Demostramos que A es un conjunto de índices no trivial, entonces por Rice no es computable, es decir, su indicadora, que es h, no es computable, y por lo mismo que antes g tampoco.

4 Ejercicio 4

4.1 Inciso a-I

Notamos que $A_k = Im(\Phi_k)$, pues es justamente los z para los cuales existe algún x tal que $\Phi_k(x) = z$, que es justo la definición de la imagen. Por una propiedad vista en la teórica, un conjunto es c.e. si y solo si es la imagen de una función parcial computable, entonces seguro todos los A_k deben ser c.e.
Para ver que no todos los A_k son computables podemos dar un contraejemplo. Consideremos el conjunto K visto en la teórica. Por la propiedad que mencionamos antes, como K es c.e., K es la imagen de alguna función parcial computable, es decir, existe k_0 tal que $K = Im(\Phi_{k_0}) = A_{k_0}$. Como K no es computable, entonces este A_{k_0} particular tampoco lo será.

4.2 Inciso a-II

Sea $z \in \mathbb{N}$, considero la función constante z que seguro es computable. Sea e el número de algún programa que la computa, seguro $z \in Im(\Phi_e) = A_e$. Es decir, para todo $z \in \mathbb{N}$ puedo encontrar algun A_e tal que $z \in A_e$, por lo tanto la unión de todos los A_k debe ser todos los naturales. Como la indicadora de los naturales es la función constante 1, es computable, y al ser computable también es c.e.

4.3 Inciso b

Llamemos D al conjunto. Veamos que no es computable usando Rice, demostrando que es un conjunto de índices no trivial:

- ⋆ Sea e el número de algún programa que computa la función constante 314 (seguro es computable porque las funciones constantes son p.r.), $\Phi_e(0) \downarrow$ y $\Phi_e(0) = 314$, entonces $e \in D$.
- ⋆ Sea e el número de algún programa que computa la función constante 0, $\Phi_e(0) \neq 314$, entonces $e \notin D$.
- ⋆ Sea P un programa tal que $\#(P) \in D$ y sea Q otro programa tal que $\Psi_P = \Psi_Q$ y $\#(P) \neq \#(Q)$. $\#(P) \in D \iff \Psi_P(0) \downarrow$ y $\Psi_P(0) = 314 \iff \Psi_Q(0) \downarrow$ y $\Psi_Q(0) = 314 \iff \#(Q) \in D$, ya que P y Q computan la misma función.

Entonces, como D es un conjunto de índices no trivial, por Rice no es computable.

Veamos que es c.e. Defino la siguiente función:

$$f(x) = (\exists t)(stp^{(1)}(0, x, t) = 1 \wedge r(snap^{(1)}(0, x, t))[1] = 314)$$

Como f es un existencial no acotado sobre un predicado total computable (de hecho es p.r. porque stp y snap lo son, observadores de tuplas y listas lo son y el y lógico lo es), entonces f es parcial computable. Además, f va a estar definida si y solo si en algún momento el programa x con entrada 0 termina y su resultado es 314, y se va a indefinir sino, es decir, $D = Im(f)$. Como D es la imagen de una función parcial computable, entonces debe ser c.e.

Como D es c.e., pero no es computable, no puede ser co-c.e., ya que si lo fuera sería también computable.