Ejercicio 1

Probar que no son computables usando diagonalización.

A)

Pruebo que la función f no es computable:

$$f(x) = \begin{cases} 1 & \text{si } \Phi_x(x) \downarrow \\ 0 & \text{si no} \end{cases}$$

Supongo que f es computable para llegar a un absurdo. Sea P el programa que computa f. Defino otro programa usando P llamado P':

```
    P
A   IF Y != 0 GOTO A
```

Lo defino de forma tal que cuando P dice que el programa X se define, entonces P' se indefine. Sea e el numero del programa P'

$$\Psi_{P'}(x) \downarrow \iff \Psi_P(x) = 0 \iff f(x) = 0$$

Por definicion de f1:

$$f(x) = 0 \iff \Phi_x(x) \uparrow$$

Entonces tenemos que para todo x:

$$\Psi_{P'}(x) \downarrow \iff \Phi_x(x) \uparrow$$

Tomando x = e = #P':

$$\Phi_e(e) \downarrow \iff \Psi_{P'}(e) \downarrow \iff \Phi_e(e) \uparrow$$

Absurdo por suponer que f es computable.

Ahora va el ejercicio de verdad:

$$f_1(x, y) = \begin{cases} 1 & \text{si } \Phi_x(y) \downarrow \\ 0 & \text{si no} \end{cases}$$

Usando esto voy a ver que f(x) es una reducción de f1(x, y) usando f(x) = f1(x, x).
Supongo f1 computable y sea Q1 un programa que lo computa. Usando Q1 escribo Q1':

```
X2 <- X1
Q1
```

Entonces vemos que $\Psi^{(1)}_{Q1'}(x) = \Psi^{(2)}_{Q1}(x, x)$
Por hipotesis: $\Psi^{(2)}_{Q1}(x, x) = f1(x, x) = f(x)$
Finalmente, por la transitividad de la igualdad tenemos que: $\Psi^{(1)}_{Q1'}(x) = f(x)$
Absurdo ya que la función que da la salida de un programa (Ψ) no puede ser una función no computable (f).
Surge de suponer que f1 es computable.

B)

$$f_2(x, y) = \begin{cases} 1 & \text{si } \Phi_x(y) = 0 \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi_x(y) = 0 \\ 0 & \text{si } \Phi_x(y) \uparrow \vee\ \Phi_x(y) \neq 0 \end{cases}$$

Sin usar reducción (dudosa)

Hago todo el versito de suponer f computable, con P el programa que la computa y otro programa P' que usa la salida de P...

```
    P
A   IF Y != 0 GOTO A
    Y <- 1
```

$\Psi_{P'}(x,y)\downarrow \iff \Psi_P(x,y)=1 \iff f_2(x,y)=1 \iff \Phi_x(y)=0$

Queda como:

$\Psi_{P'}(x,y)\downarrow \iff \Phi^2_e(x,y)\downarrow \iff \Phi^2_e(x,y)=1 \iff \Phi^1_x(y)=0$

Asi de una yo se que las Φ^1 son funciones donde todos los parametros $x_i, i>1$ estan definidos como 0. Puedo reemplazarlas por una Φ^2 que tenga la segunda variable en 0.

$\Phi^2_e(x,y)=1 \iff \Phi^2_x(y,0)=0$

evaluando x = e = #(P'):

$\Phi^2_e(e,y)=1 \iff \Phi^2_e(y,0)=0$

Puedo escribir el programa de forma tal que la salida siempre sea 1, lo cual me lleva a un absurdo.

Voy a demostrar un caso mas simple para después demostrar la reducción.

Primera forma (galerazo?)

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

Para probar que f no es computable, primero supongo que **si** lo es. Por ser computable, existe un programa P que la computa: $\Psi_P(x)=f(x)$. Como P existe, puedo definir otro programa que lo use y lo llamo P':

```
    P
A   IF Y = 0 GOTO A
    Y <- 1
```

$$\Psi_{P'}(x)\downarrow \iff \Psi_P(x)\downarrow \wedge\ \Psi_P(x)\neq 0$$

Como las unicas salidas del programa P son 1 o 0, estamos pidiendo que la salida sea 1:

$$\Psi_P(x)\downarrow \wedge\ \Psi_P(x)\neq 0 \iff \Psi_P(x)=1 \iff f(x)=1$$

Por definición de f, solo se cumple si estamos en la primera guarda:

$$f(x)=1 \iff \Phi_x(x)=0$$

Por transitividad, finalmente llegamos a:

$$\Psi_{P'}(x)\downarrow \iff \Phi_x(x)=0$$

Pero por inspección de P' podemos ver que $\Psi_{P'}(x)=1$ siempre que termine la ejecución del programa. Como P' existe a partir de lo que supusimos, entonces también podemos calcular su numero de programa. Llamamos e = #(P').

$$\Psi_{P'}(x)\downarrow \iff \Psi_{P'}(x)=1 \iff \Phi_x(x)=0$$
$$\Phi*e(e)=1 \iff \Phi*e(e)=0$$

Absurdo que parte de suponer que la función f es computable.

Segunda forma

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

Supongo f computable. Sea P el programa que la computa, voy a escribir una P' con la intención de que esté definida sii la universal se indefina. Para que se indefina la universal necesito que el programa se cuelgue sii la salida es 1 (analizando la guarda de f).

```
    P
A   IF Y != 0 GOTO A
    Y <- ¿?
```

$$\Psi_{P'}(x)\downarrow \iff \Psi_P(x) = 0 \iff f(x) = 0 \text{Por definición:} \iff \Phi_x(x)\uparrow \vee \Phi_x(x) \neq 0$$

Mas o menos quedó lo que quería, estaría bueno no tener el $\Phi_P(x) \neq 0$ que no me permite llegar directo a la contradicción. Sabiendo que para la contradiccción voy a evaluar x = e = #(P') puedo modificar P' para que dé una salida 0 y que solo "sobreviva" la primera parte del $\vee$.

```
    P
A   IF Y != 0 GOTO A
    Y <- 0
```

$$\Psi_{P'}(x)\downarrow \iff \Phi_x(x)\uparrow$$

Evaluando x = e:

$$\Phi_e(e)\downarrow \iff \Phi_e(e)\uparrow$$

Absurdo, por suponer que f era computable.

Reducción

Usando esto voy a ver que f(x) es una reducción de f2(x, y) usando f(x) = f2(x, x).
Supongo f2 computable y sea Q2 un programa que lo computa. Usando Q2 escribo Q2':

```
    X2 <- X1
    Q2
```

Entonces vemos que $\Psi^{(1)}_{Q2'}(x) = \Psi^{(2)}_{Q2}(x, x)$
Por hipotesis: $\Psi^{(2)}_{Q2}(x, x) = f_2(x, x) = f(x)$
Finalmente, por la transitividad de la igualdad tenemos que: $\Psi^{(1)}_{Q2'}(x) = f(x)$
Absurdo ya que la función que da la salida de un programa (Ψ) no puede ser una función no computable (f).
Surge de suponer que f2 es computable.

C)

$$f_3(x, y, z) = \begin{cases} 1 & \text{si } \Phi^{(1)}_x(y)\downarrow \wedge \Phi^{(1)}_x(y) > z \\ 0 & \text{si } \Phi^{(1)}_x(y)\uparrow \vee \Phi^{(1)}_x(y) \leq z \end{cases}$$

Hago P3' un programa que usa P3 (el programa de f3):

Quiero que: $\Psi_{P3'}(x, y, z)\downarrow \iff \Phi^{(1)}_x(y)\uparrow \vee \Phi^{(1)}_x(y) \leq z \iff \Phi^{(3)}_x(y, 0, 0)\uparrow \vee \Phi^{(3)}_x(y, 0, 0) \leq z$
$\Phi^{(3)}_e(x, y, z)\downarrow \iff \Phi^{(3)}_x(y, 0, 0)\uparrow \vee \Phi^{(3)}_x(y, 0, 0) \leq z$

$\Phi_e^{(3)}(e,y,z)\downarrow \iff \Phi_e^{(3)}(y,0,0)\uparrow \vee \Phi_e^{(3)}(y,0,0) \leq z$
$\Phi_e^{(3)}(e,y,z)\downarrow \iff \Phi_e^{(3)}(y,0,0)\uparrow \vee \Phi_e^{(3)}(y,0,0) \leq z$
No se cerrarlo

Pruebo el caso mas simple:

$$g_3(x) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x)\downarrow \wedge \Phi_x^{(1)}(x) > x \\ 0 & \text{si } \Phi_x^{(1)}(x)\uparrow \vee \Phi_x^{(1)}(x) \leq x \end{cases}$$

Hago un programa Q3' con Q3:

```
    Q3
A   IF Y != 0 GOTO A
    Y <- X + 1
```

$$\Psi_{Q3'}(x)\downarrow \iff \Psi_{Q3}(x) = 0 \iff g_3(x) = 0 \iff \Phi_x^{(1)}(x)\uparrow \vee \Phi_x^{(1)}(x) \leq x$$

Por la cadena de sii:

$$\Psi_{Q3'}(x)\downarrow \iff \Phi_x^{(1)}(x)\uparrow \vee \Phi_x^{(1)}(x) \leq x$$

Evaluando x = e = #(Q3')

$$\Phi_e(e)\downarrow \iff \Phi_e^{(1)}(e)\uparrow \vee \Phi\_e^{(1)}(e) \leq e$$

Vemos que la salida del programa con numero e siempre es x+1, asi que $\Phi_e^{(1)}(e) \leq e$ es siempre falso.

$$\Phi_e(e) = e+1 \iff \Phi_e^{(1)}(e)\uparrow \vee \Phi_e^{(1)}(e) \leq e$$

Reducción f3

Voy a usar g3 (no es computable) para probar que f3 no es computable mostrando que g3 es una reducción de f3. Quiero ver que si f3(x,y,z) es computable entonces hay un caso especial (la reducción) g3(x) que tambien es computable, lo hago usando g3(x) = f3(x,y,z).

Supongo que f3(x,y,z) es computable, entonces existe un programa P3 que la computa. Hago un programa P3':

```
X2 <- X1
X3 <- X1
P3
```

Por inspección este programa se comporta como P3 con las 3 variables iguales a la primera:

$$\Psi_{P3'}^{(1)}(x) = \Psi_{P3}^{(3)}(x,x,x) = f_3(x,x,x)$$

Por definición de g3 sabemos que $g_3(x) = f_3(x,x,x)$ entonces:

$$\Psi_{P3'}^{(1)}(x) = g3(x)$$

Pero esto implicaría que g3 es computable, cuando sabemos que no lo es. Absurdo, no existe ningun programa Q tal que $\Psi_Q^{(1)}(x) = g3(x)$ (no es computable).
Surge de suponer que f3 es computable.

D)

$$f_4(x) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x)\downarrow \wedge \Phi_x^{(1)}(x) \neq x \\ 0 & \text{si } \Phi_x^{(1)}(x)\uparrow \vee \Phi_x^{(1)}(x) = x \end{cases}$$

Supongo f4 computable, entonces existe un programa P4 que lo computa. Llamo P4' al siguiente programa:

```
    P4
A   IF Y != 0 GOTO A
    Y <- 0
```

Por inspección vemos que el programa termina siempre en 0 y termina sii la salida de P4 es igual a 0.

$\Psi_{P3'}^{(1)}(x)\downarrow \iff \Psi_{P3}^{(1)}(x) = 0 \iff f_4(x) = 0 \iff \Phi_x^{(1)}(x)\uparrow \vee \Phi_x^{(1)}(x) = x$

Por transitividad nos queda:

$\Psi_{P3'}^{(1)}(x)\downarrow \iff \Phi_x^{(1)}(x)\uparrow \vee \Phi_x^{(1)}(x) = x$

Sea e = #(P4') y evaluando x en e:

$\Phi_e^{(1)}(e)\downarrow \iff \Phi_e^{(1)}(e)\uparrow \vee \Phi_e^{(1)}(e) = e$

La segunda parte del $\vee$ es siempre falsa ya que $\Phi_x^{(1)}(x) = 0,\ \forall x$ y como e no es el programa vacio entonces $\Phi_e^{(1)}(e) = 0 \neq e$. Finalmente:

$\Phi_e^{(1)}(e)\downarrow \iff \Phi_e^{(1)}(e)\uparrow$

Absurdo, surge de suponer que f4 es computable.

Ejercicio 2

Idea general de la reducción: Partimos con una función (f) que sabemos que no es computable, queremos probar que g tampoco lo es.

- Suponemos f computable
- Vemos como podemos escribir g en función de f
- Escribimos un programa que use el paso anterior
- Hacemos el Ψ del programa de g igual a la reducción con el Ψ del programa de f
- Reemplazamos el Ψ con f, deberia quedar la reducción que hicimos antes
- Reemplazamos la reducción con g.
- Por transitividad de igualdad nos quedó que Ψ_f es igual a g, que no es computable. Absurdo

A)

$$g_1(x,y) = \begin{cases} 1 & \text{si } \Phi_x(y)\uparrow \\ 0 & \text{si no} \end{cases}$$

Podemos hacer una reducción a f1 (ej1) : $f_1(x,y) = 1 - g_1(x,y)$

Supongo g1 computable, existe un programa Q1 que la computa. Llamo Q1' al siguiente programa:

```
    Q1
    Y <- 1 - Y
```

Por inspección:

$\Psi_{Q1'}^{(1)}(x,y) = 1 - \Psi_{Q1}^{(1)}(x,y) = 1 - g_1(x,y) = f_1(x,y)$

Por transitividad nos queda que:

$\Psi_{Q1'}^{(1)}(x,y) = f_1(x,y)$

Absurdo ya que f1 no es computable. Surge de suponer que g1 era computable.

B)

$$g_2(x,y,z,w) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(z)\downarrow \wedge\ \Phi_y^{(1)}(w)\downarrow \wedge\ \Phi_x^{(1)}(z) > \Phi_y^{(1)}(w) \\ 0 & \text{si no} \end{cases}$$

Sea k el numero del programa que computa la función identidad (sabemos que existe ya que Id es p.r.). La reducción se puede hacer con f3 del ej1:

$$f_3(x,y,z) = g_2(x,k,y,z) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y)\downarrow \wedge\ \Phi_k^{(1)}(z)\downarrow \wedge\ \Phi_x^{(1)}(y) > \Phi_k^{(1)}(z) \\ 0 & \text{si no} \end{cases}$$

Como la función identidad es total: $\Phi_k^{(1)}(z)\downarrow, \forall z$ y ademas es igual a z, entonces

$$f_3(x,y,z) = g_2(x,k,y,z) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y)\downarrow \wedge\ \Phi_x^{(1)}(y) > z \\ 0 & \text{si no} \end{cases}$$

Falta probar que es una reducción válida, para eso suponemos g2 computable y Q2 el programa que la computa, definimos el programa Q2' como:

```
X4 <- X3
X3 <- X2
X2 <- K
Q2
```

Por inspección vemos que:

$$\Psi_{Q2'}^{(3)}(x,y,z) = \Psi_{Q2}^{(4)}(x,k,y,z) = g_2(x,k,y,z) = f_3(x,y,z)$$

Pero por transitividad implica que $\Psi_{Q2'}^{(3)}(x,y,z) = f_3(x,y,z)$, lo cual es un absurdo ya que f3 no es computable.

C)

Reducción que encontré:

$$f_4(x) = g_3(x,x,x) \dot{-} x = \begin{cases} (x+1) \dot{-} x & \text{si } \Phi_x^{(1)}(x)\downarrow \wedge\ \Phi_x^{(1)}(x) \neq x \\ 0 \dot{-} x & \text{si no} \end{cases}$$

Resolviendo cada caso:

$$f_4(x) = g_3(x,x,x) \dot{-} x = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x)\downarrow \wedge\ \Phi_x^{(1)}(x) \neq x \\ 0 & \text{si no} \end{cases}$$

Falta ver que es una reducción válida:

Supongo g3 computable. Sea Q3 el programa que la computa y Q3' el siguiente programa:

```
X2 <- X1
X3 <- X1
Q3
Y  <- Y - X
```

Por inspección del programa se ve:

$$\Psi_{Q3'}^{(1)}(x) = \Psi_{Q3}^{(3)}(x,x,x) \dot{-} x = g_3(x,x,x) \dot{-} x = f_4(x)$$

Por transitividad:

$$\Psi^{(1)}_{Q3'}(x) = f_4(x)$$

Absurdo ya que f4 no es computable.

D)

Sea g la función constante 1 $g(x) = 1$ computable y sea i el numero del programa que la computa.

La reducción que uso para g4(x,y,z) es:

$$f_1(x,y) = g_4(i,x,y) = \begin{cases} (\Phi^{(1)}_i \circ \Phi^{(1)}_x)(y) & \text{si } \Phi^{(1)}_x(y)\downarrow \wedge\ (\Phi^{(1)}_i \circ \Phi^{(1)}_x)(y)\downarrow \\ 0 & \text{si no} \end{cases}$$

Observar que

$$(\Phi^{(1)}_i \circ \Phi^{(1)}_x)(y) = \Phi^{(1)}_i(\Phi^{(1)}_x(y)) = g(\Phi^{(1)}_x(y)) = 1$$

Sii se cumple que $\Phi^{(1)}_x(y)\downarrow$. Haciendo todas las simplificaciones nuestra reducción queda como:

$$f_1(x,y) = g_4(i,x,y) = \begin{cases} 1 & \text{si } \Phi^{(1)}_x(y)\downarrow \\ 0 & \text{si no} \end{cases}$$

Supongo g4 computable, entonces existe un programa P4 que la computa. Sea P4' el siguiente programa:

```
X3 <- X2
X2 <- X1
X1 <- i
P4
```

Por inspección:

$$\Psi^{(2)}_{P4'}(x,y) = \Psi^{(3)}_{P4}(i,x,y) = g_4(i,x,y) = f_1(x,y)$$

Por transitividad nos queda el absurdo:

$$\Psi^{(2)}_{P4'}(x,y) = f_1(x,y)$$

Ya que f1 no es computable. Surge de suponer que g4 era computable.

Ejercicio 3

$$g'_3(x,y,z) = \begin{cases} z & \text{si } \Phi^{(1)}_x(y)\downarrow \wedge\ \Phi^{(1)}_x(y) \neq z \\ 0 & \text{si no} \end{cases}$$

La reducción es $f_4(x) = (g'_3(x,x,x) = x \wedge\ x > 0)$:

$$\begin{cases} 1 & \text{si } \Phi^{(1)}_x(x)\downarrow \wedge\ \Phi^{(1)}_x(x) \neq x \wedge\ x > 0 \\ 0 & \text{si no} \end{cases}$$

Vemos que en el caso de f4(0) = 0 ya que $\Phi_0(0) = 0$ por ser el programa vacío así que lo "hardcodeamos".

Sea Q el programa que computa g3', voy a escribir P usando todas funciones computables (=, $\wedge$ y $>$ son p.r.) :

```
X2 <- X1
X3 <- X1
Q
Y <- (Y = X1) && X1 > 0
```

Por inspección:

$\Psi_P^{(1)}(x) = (\Psi_Q^{(3)}(x,x,x) = x \wedge\ x > 0) = (g_3'(x,x,x) = x \wedge\ x > 0) = f_4(x)$

Por transitividad queda un absurdo $\Psi_P^{(1)}(x) = f_4(x)$ ya que f4 no es computable.

Ejercicio 4

Una función parcial computable f es *extensible* si existe g computable tal que g(x)=f(x) para todo $x \in \text{Dom}f$.

Qvq existe una función parcial computable que **no es extensible**.

Para probarlo veo:

Dada una funcion f parcial computable, es extensible sii

$\exists g : \forall x \in \text{Dom}f, g(x) = f(x)$

La f que elijo para probar esto está definida como

$$f(x) = \begin{cases} \Phi_x^{(1)}(x) & \text{si } \Phi_x^{(1)}(x)\downarrow \\ \uparrow & \text{si no} \end{cases}$$

Pruebo que es computable escribiendo un programa P que la computa:

```
Y <- Phi_X(X)
```

Por inspección se ve que termina sii $\Phi_x^{(1)}(x)$ termina su ejecución. Como pudimos escribir un programa tal que:

$$\Psi_P^{(1)}(x) = f(x)$$

Queda probado que f es parcial computable. No es total computable ya que la función no es total.

Sea h(x) una función computable (funcion total y parcial computable). Puedo definir f'(x) como su extensión usando a h(x).

f' cumple que es total y que tiene el mismo comportamiento que f para el dominio de f. Eso significa que está definida (devuelve algún valor) para la guarda en la que f se colgaba:

$$f'(x) = \begin{cases} \Phi_x^{(1)}(x) & \text{si } \Phi_x^{(1)}(x)\downarrow \\ h(x) & \text{si no} \end{cases}$$

Finalmente pruebo que para todo h(x), f' no es computable.

En realidad esto no debería servir ya que Phi puede dar igual a h

Supongo f' computable, entonces existe un programa P' que la computa. Escribo el siguiente programa Q:

```
    P
A   IF Y != h(x) GOTO A
```

Por inspección del programa Q se ve que:

$$\Psi_Q^{(1)}(x)\downarrow \iff \Psi_{P'}^{(1)}(x) \neq \Phi_x^{(1)}(x) \iff f'(x) \neq \Phi_x^{(1)}(x)$$

Por definición de f':

$$f'(x) \neq \Phi_x^{(1)}(x) \iff f'(x) = h(x) \iff \Phi_x^{(1)}(x) \uparrow$$

Finalmente $\Psi_Q^{(1)}(x) \downarrow \iff \Phi_x^{(1)}(x) \uparrow$ y evaluando x en e = #(Q):

$$\Phi_e^{(1)}(e) \downarrow \iff \Phi_e^{(1)}(e) \uparrow$$

Absurdo que surge de suponer que f' es computable.

Ejercicio 5

A)

$$g_1(x) = \begin{cases} 1 & \text{si HALT}(1337, x) \\ 0 & \text{si no} \end{cases} \quad \text{HALT}(x, y) = \begin{cases} 1 & \text{si } \Phi_y^{(1)}(x) \downarrow \\ 0 & \text{si no} \end{cases}$$

Voy a convertirlo a f1 del ej 1 y despues en otra f1(x,x) que es mas facil:

tomo e una cte que voy a definir despues. Hago el $S_1^1(u, e)\ldots$

Reduzco a

$$f_1(x, y) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y) \downarrow \\ 0 & \text{si no} \end{cases}$$

Para esto primero reescribo g1 usando la definición de HALT:

$$g_1(x) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(1337) \downarrow \\ 0 & \text{si no} \end{cases}$$

Vemos que ahora son muy parecidas. Necesitamos darle a g1 el numero de un programa que usa la X2 y X3 para ejecutar algo parecido a f1, lo construimos usando la S del teorema del párametro.

$$g_1(S_1^2(u, v, e)) = \begin{cases} 1 & \text{si } \Phi_{S_1^2(u,v,e)}^{(1)}(1337) \downarrow \\ 0 & \text{si no} \end{cases}$$

$$= \begin{cases} 1 & \text{si } \Phi_e^{(3)}(1337, u, v) \downarrow \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi_u^{(1)}(v) \downarrow \\ 0 & \text{si no} \end{cases} = f_1(u, v)$$

Si g1 es computable y Q1 es su programa entonces puedo hacer un programa P1 con numero e cuya salida es el resultado de ignorar la primera variable y usar las siguientes dos para "ejecutar" f1:

```
Z1 <- X2
Z2 <- X3
Y  <- U_n(Z1, Z2)
```

Si el numero de este programa es e, podemos construir el numero de otro programa que hardcodea X2 y X3 adentro del codigo y podemos hacer lo que queríamos

$$\Phi_e^{(3)}(x, u, v) = \Phi_{S_1^2(u,v,e)}^{(1)}(x)$$

Ademas vemos por inspección que:

$$\Phi_u^{(1)}(v)\downarrow \iff \Phi_e^{(3)}(x,u,v)\downarrow \iff \Phi^{(1)}_{S_1^2(u,v,e)}(x)\downarrow$$

g1 es una función que toma el numero de un programa y te dice si este termina o no con una entrada fija 1337. La idea es aprovechar que toma el numero de un programa y mostrar que adentro del numero de un programa podemos codificar variables extra, que nosotros decidimos sus valores (teo parametro).

De esta forma usando una función que nos dice si un programa x termina para cierta entrada, podemos ver si un programa u termina para cualquier entrada v dada. En este numero de programa vamos a querer agregar dos variables para hacer la reducción a f1 del ej1, asi que primero hacemos el programa que copia el comportamiento de $\Phi_{x2}^{(1)}(x3)$ y conseguimos su numero de programa.

Despues usamos ese numero y la función S_1^2 para transformar este programa de 3 variables a uno de 1 variable con las otras dos fijas dentro del codigo. Como S es pr y nuestro programa de recien existe, nada nos impide en darle este programa como entrada a g1.

Construyo programa P tq $\Psi_P^{(3)}(x1,x2,x3) = \Phi_{x2}^{(1)}(x3)$ (ignoramos la primera variable)

```
Y <- Phi_X2(X3)
```

Como pudimos escribir el programa, podemos saber su numero y lo llamamos e = #(P).

Por teo del parametro hay una función p.r. tq

$$\Phi_e^{(3)}(x_1,x_2,x_3) = \Phi^{(1)}_{S_1^2(x_2,x_3,e)}(x_1)$$

Por transitividad de la igualdad vemos que

$$\Phi^{(1)}_{S_1^2(x_2,x_3,e)}(x_1) = \Phi_{x2}^{(1)}(x3)$$

Ahora podemos componer esta función con g1.

Supongo g1 computable y Q1 el programa que la computa, entonces puedo pasarle el numero del programa que acabo de construir.

$$g_1(S_1^2(u,v,e)) = \begin{cases} 1 & \text{si } \Phi^{(1)}_{S_1^2(u,v,e)}(1337)\downarrow \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi_u^{(1)}(v)\downarrow \\ 0 & \text{si no} \end{cases} = f_1(u,v)$$

Pero si

- e es un numero de programa valido (existe xq lo escribí)
- S es pr $\implies$ S es computable
- g1 es computable

Esto implica que la composición $g_1 \circ S$ es computable, lo cual es un absurdo ya que $g_1 \circ S = f_1$ que **no es computable**.

B)

$$g_2(x,y,z) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(z)\downarrow \wedge\ \Phi_y^{(1)}(z)\downarrow \wedge\ \Phi_x^{(1)}(z) > \Phi_y^{(1)}(z) \\ 0 & \text{si no} \end{cases}$$

Primero evalúo y en i = #(Id)

$$g_2(x,i,z) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(z)\downarrow \wedge\ \Phi_x^{(1)}(z) > z \\ 0 & \text{si no} \end{cases}$$

Ahora defino un programa que recibe tres variables (x1,x2,x3) (ignora la primera) y computa $\Phi^{(1)}_{x_2}(x_3)$. Llamo P a este programa y e a su número.

```
Y <- Phi_X2(X3)
```

Por el teorema del parámetro $\Phi^{(3)}_e(z,x,y) = \Phi^1_{S^2_1(x,y,e)}(z)$ que por inspección del programa y transitividad nos queda:

$$\Phi^1_{S^2_1(x,y,e)}(z) = \Phi^{(1)}_x(y)$$

Finalmente hago la reducción usando esta S:

Supongo g2 computable.

$$g_2(S^2_1(x,y,e),i,z) = \begin{cases} 1 & \text{si } \Phi^{(1)}_{S^2_1(x,y,e)}(z)\downarrow \wedge\ \Phi^{(1)}_{S^2_1(x,y,e)}(z) > x \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi^{(1)}_x(y)\downarrow \wedge\ \Phi^{(1)}_x(y) > z \\ 0 & \text{si no} \end{cases} = f_3(x,y,z)$$

Entonces la composición g2 y S es computable.

Absurdo ya que la composición da una f que no es computable.

C)

$$g_3(x) = \begin{cases} 13 & \text{si } \Phi^{(1)}_x \text{ es la constante 7} \\ 0 & \text{si no} \end{cases}$$

Hago un programa que cumple que es la constante 7 pero que solo termina sii $\Phi^{(1)}_{x_2}(x_2)\downarrow$, lo llamo P3.

```
Z1 <- Phi_X2(X2)
Y  <- 7
```

Por inspección de P3: $\Psi^{(2)}_{P3}(x_1,x_2)$ es cte 7 $\iff \Phi^{(1)}_{x_2}(x_2)\downarrow$

Sea e = #(P3), puedo usar el teorema del parametro para fijar x2 y tener el numero de un programa nuevo en función de x2.

$$\Phi^{(2)}_e(x_1,x_2) = \Psi^{(1)}_{S^1_1(x_2,e)}(x_1)$$

Supongo g3 computable. Sabiendo que S^1_1 es pr (entonces computable) puedo componerlas para formar algo computable:

$$g_3(S^1_1(x_2,e)) = \begin{cases} 13 & \text{si } \Phi^{(1)}_{S^1_1(x_2,e)} \text{ es la constante 7} \\ 0 & \text{si no} \end{cases} = \begin{cases} 13 & \text{si } \Phi^{(2)}_e \text{ es la constante 7} \\ 0 & \text{si no} \end{cases} = \begin{cases} 13 & \text{si } \Phi^{(1)}_{x_2}(x_2)\downarrow \\ 0 & \text{si no} \end{cases}$$

Absurdo, ya que la ultima función es una que sabemos que no es computable.

D)

$$g_4(x,y) = \begin{cases} 1 & \text{si } \Phi^{(1)}_x(y)\downarrow \wedge\ \Phi^{(1)}_y(x)\downarrow \wedge\ \Phi^{(1)}_x(y) \neq \Phi^{(1)}_y(x) \\ 0 & \text{si no} \end{cases}$$

Voy a reducir g4 para llegar a f4 del ejercicio 1.

Quiero meter en x el numero de un programa que toma otra variable (una mas) y que computa $\Phi^{(1)}_{x_2}(x_2)$. Defino el programa P4 como:

```
Y <- Phi_X2(X2)
```

Sea e = #(P4) el numero de este programa. Por el teorema del parametro existe una S_1^1:

$$\Phi_e^{(2)}(x,y) = \Phi^{(1)}_{S_1^1(y,e)}(x)$$

Y como nuestro programa e computa $\Phi^{(1)}_{x_2}(x_2)$ nos queda que:

$$\Phi_y^{(1)}(y) = \Phi^{(1)}_{S_1^1(y,e)}(x)$$

Evaluando la x de g4 en $S_1^1(x,e)$:

$$g_4(S_1^1(x,e),y) = \begin{cases} 1 & \text{si } \Phi^{(1)}_{S_1^1(x,e)}(y)\downarrow \wedge\ \Phi_y^{(1)}(S_1^1(x,e))\downarrow \wedge\ \Phi^{(1)}_{S_1^1(x,e)}(y) \neq \Phi_y^{(1)}(S_1^1(x,e))\downarrow \\ 0 & \text{si no} \end{cases}$$

Reemplazando $\Phi^{(1)}_{S_1^1(x,e)}(y)$ por $\Phi_x^{(1)}(x)$

$$g_4(S_1^1(x,e),y) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x)\downarrow \wedge\ \Phi_y^{(1)}(S_1^1(x,e))\downarrow \wedge\ \Phi_x^{(1)}(x) \neq \Phi_y^{(1)}(S_1^1(x,e))\downarrow \\ 0 & \text{si no} \end{cases}$$

Ahora tengo que ver que programa darle a y para que se cumpla $\Phi_y^{(1)}(S_1^1(x,e)) = x$, lo hago definiendo un programa que ignora la primera variable y devuelve siempre la segunda.

Defino un programa con numero k que cumple $\Phi_k^{(2)}(x_1,x_2) = x_2$

```
Y <- X2
```

Ahora uso el teorema del parametro para encontrar una funcion p.r. $S_1^{1'}$ tal que $\Phi_k^{(2)}(x,y) = \Phi^{(1)}_{S_1^{1'}(y,k)}(x) = y$

$$g_4(S_1^1(x,e),S_1^{1'}(x,k)) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x)\downarrow \wedge\ \Phi^{(1)}_{S_1^{1'}(x,k)}(S_1^1(x,e))\downarrow \wedge\ \Phi_x^{(1)}(x) \neq \Phi^{(1)}_{S_1^{1'}(x,k)}(S_1^1(x,e)) \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x)\downarrow \wedge\ \Phi_x^{(1)}(x) \neq \\ 0 & \text{si no} \end{cases}$$

Para cerrar

- Tomo la primera variable como $S_1^1(x,e)$, ya que $\Phi^{(1)}_{S_1^1(x,e)}(y) = \Phi_x^{(1)}(x)$.
- Tomo la segunda variable como $S_1^{1'}(x,k)$, ya que $\Phi^{(1)}_{S_1^{1'}(x,k)}(y) = x$.

Supongo g4 computable, como las dos S son pr ($\implies$ computables) y la composicion es computable entonces la siguiente función es computable:

$$\begin{cases} 1 & \text{si } \Phi_x^{(1)}(x)\downarrow \wedge\ \Phi_x^{(1)}(x) \neq x \\ 0 & \text{si no} \end{cases}$$

Pero es la función f4 que ya probamos que no es computable en el ejercicio 1. Faltaría ver que la reducción es computable.

Ejercicio 6

Demostrar que existe un programa P (con numero e) tal que $\Psi_P^{(1)}(x)\downarrow \iff x = e$

Defino una función g : $\mathbb{N}^2 \to \mathbb{N}$ como:

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

Es trivial ver que es parcial computable. Sabiendo esto podemos usar el teorema de la recursion sobre g:

Por el teo sabemos que existe un programa con numero e tal que $\Phi_e^{(1)}(y) = g(e,y)$, finalmente:

$$\Phi_e^{(1)}(x) = \begin{cases} 42 & \text{si } x = e \\ \uparrow & \text{si no} \end{cases}$$

Por la definición de Φ se ve que $\Phi_e^{(1)}(x)\downarrow \iff x = e$ como queriamos probar.

Ejercicio 7

Probar que no son computables usando Teorema de la Recursión.

h1

$$h_1(x) = \begin{cases} 1 & \text{si } x \in \text{Im } \Phi_x^1 \\ 0 & \text{si no} \end{cases}$$

Supongo h1 computable y defino la siguiente función:

$$g_1(x,y) = \begin{cases} \uparrow & \text{si } h_1(x) \\ x & \text{si no} \end{cases}$$

Observar que es igual a:

$$g_1(x,y) = \begin{cases} \uparrow & \text{si } x \in \text{Im } \Phi_x^1 \\ x & \text{si no} \end{cases}$$

Como g1 está definida por composición de funciones computables (h1 incluida) y es parcial, es parcial computable. Puedo usar el teorema de la recursión.

Por TDR existe un programa con numero e tal que $\Phi_e^{(1)}(y) = g(e,y)$, veamos que implica:

$$\Phi_e^{(1)}(y) = \begin{cases} \uparrow & \text{si } e \in \text{Im } \Phi_e^1 \\ e & \text{si no} \end{cases}$$

Acá llegamos a un absurdo:

Supongo $e \in \text{Im } \Phi_e^1$, entonces:

$e \in \text{Im } \Phi_e^1 \iff \Phi_e^{(1)}(y)\uparrow \ \forall y \iff \text{Im } \Phi_e^1 = \emptyset \implies e \notin \text{Im } \Phi_e^1$

Supongo $e \notin \text{Im } \Phi_e^1$, entonces:

$e \notin \text{Im } \Phi_e^1 \iff \Phi_e^{(1)}(y) = e \ \forall y \implies e \in \text{Im } \Phi_e^1$

Comentario de la función a elegir

No necesitaba que el programa se cuelgue, solo necesitaba que implique que e no esta en la imagen para todo y.

Tampoco necesitaba definir la segunda guarda como e, necesitaba que haya algun valor que devuelve e. Otra opción podría haber sido

$$g_1(x,y) = \begin{cases} 0 & \text{si } x \in \text{Im } \Phi_x^1 \\ y & \text{si no} \end{cases}$$

Que reemplazando nos queda:

$$\Phi_e^{(1)}(y) = \begin{cases} 0 & \text{si } e \in \text{Im } \Phi_e^1 \\ y & \text{si no} \end{cases}$$

Entonces el absurdo surge de que:

- En la primera guarda la imagen es 0, y el programa e no es vacio.
- En la segunda guarda se implica que existe un y = e tq Phi(e)=e entonces e SI pertenece a la imagen.

h2

$$h_2(x,y) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y) \downarrow \wedge\ \Phi_x^{(1)}(y) > x \\ 0 & \text{si no} \end{cases}$$

Lo voy a llevar a un programa de una sola entrada (que es y) y quiero que

- el programa se cuelgue sii la nueva función no se cuelga (absurdo)
- el programa termina sii la nueva función se cuelga

La negación de la guarda es $\Phi_x^{(1)}(y) \uparrow \vee\ \Phi_x^{(1)}(y) \leq x$ asi que si el programa termina tambien quiero que contradiga la otra parte del or. Devuelvo una salida mayor a x. Finalmente defino:

$$g_2(x,y) = \begin{cases} \uparrow & \text{si } h_2(x,y) \\ x+1 & \text{si no} \end{cases}$$

Como g2 es una función parcial y composición de funciones computables (y una parcial computable) sabemos que es parcial computable.

Viendo la función h2 podemos ver que g2 es igual a:

$$g_2(x,y) = \begin{cases} \uparrow & \text{si } \Phi_x^{(1)}(y) \downarrow \wedge\ \Phi_x^{(1)}(y) > x \\ x+1 & \text{si no} \end{cases}$$

Como g2 es parcial computable podemos usar el TDR sobre ella: existe un programa con numero e tal que $\Phi_e^{(1)}(y) = g_2(e,y)$, entonces

$$\Phi_e^{(1)}(y) = \begin{cases} \uparrow & \text{si } \Phi_e^{(1)}(y) \downarrow \wedge\ \Phi_e^{(1)}(y) > e \\ e+1 & \text{si no} \end{cases}$$

Divido en dos casos distintos:

Supongo la primera guarda y veo que implica:

$\Phi_e^{(1)}(y) \downarrow \wedge\ \Phi_e^{(1)}(y) > e \iff \Phi_e^{(1)}(y) \uparrow \forall y$

Supongo la segunda guarda y veo que implica:

$\Phi_e^{(1)}(y) \uparrow \vee\ \Phi_e^{(1)}(y) \leq e \iff \Phi_e^{(1)}(y) = e+1 \implies \Phi_e^{(1)}(y) \downarrow \wedge\ \Phi_e^{(1)}(y) > e$

Los dos casos dan absurdo que surge de suponer h2 computable.

h3

$$h_3(x) = \begin{cases} 1 & \text{si Im}\Phi_x^{(1)}\text{es infinita} \\ 0 & \text{si no} \end{cases}$$

Supongo h3 computable, defino la siguiente función g3:

$$g_3(x,y) = \begin{cases} 0 & \text{si } h_3(x) \\ y & \text{si no} \end{cases} = \begin{cases} 0 & \text{si Im}\Phi_x^{(1)}\text{es infinita} \\ y & \text{si no} \end{cases}$$

Como g3 es parcial computable, por TDR $\Phi_e^{(1)}(y) = g_3(e,y)$

$$\Phi_e^{(1)}(y) = \begin{cases} 0 & \text{si Im}\Phi_e^{(1)}\text{es infinita} \\ y & \text{si no} \end{cases}$$

Supongo primera guarda y veo que implica absurdo:

$\text{Im}\Phi_x^{(1)}\text{es infinita} \iff \Phi_e^{(1)}(y) = 0\ \forall y \implies \text{Im}\Phi_x^{(1)}\text{es finita}$

Supongo segunda guarda y veo que implica absurdo:

$\text{Im}\Phi_x^{(1)}\text{es finita} \iff \Phi_e^{(1)}(y) = y\ \forall y \implies \text{Im}\Phi_x^{(1)}\text{es infinita}$

Los dos absurdos

h4

$$h_4(x) = \begin{cases} 1 & \text{si } |\text{Dom}\Phi_x^{(1)}| = x \\ 0 & \text{si no} \end{cases}$$

Supongo h4 computable. Defino g4 en función de esta de forma tal que g4 sea parcial computable.

$$g_4(x,y) = \begin{cases} \uparrow & \text{si } h_4(x) \\ p(y) & \text{si no} \end{cases} = \begin{cases} \uparrow & \text{si } |\text{Dom}\Phi_x^{(1)}| = x \\ p(x,y) & \text{si no} \end{cases}$$

Y defino p(x, y) de forma tal que | Dom(p) | = x :

$$p(x,y) = \begin{cases} 1 & \text{si } y < x \\ \uparrow & \text{si no} \end{cases}$$

Finalmente, por TDR se existe un e tal que $\Phi_e^{(1)}(y) = g_4(e,y)$ entonces

$$\Phi_e^{(1)}(y) = \begin{cases} \uparrow & \text{si } |\text{Dom}\Phi_e^{(1)}| = e \\ p(e,y) & \text{si no} \end{cases}$$

Analizo cada guarda por separado:

Supongo la primera guarda y veo que implica absurdo:

$|\text{Dom}\Phi_e^{(1)}| = e \iff \Phi_e^{(1)}(y) \uparrow \forall y \iff |\text{Dom}\Phi_e^{(1)}| = 0 \implies |\text{Dom}\Phi_e^{(1)}| \neq e$

Absurdo ya que e != 0 (el programa es no nulo)

Supongo la segunda guarda y veo que implica absurdo:

- $|\text{Dom}\Phi_e^{(1)}| \neq e \iff \Phi_e^{(1)}(y) = p(e,y)\ \forall y \implies |\text{Dom}\Phi_e^{(1)}| = e$

Absurdo

Ejercicio 8

Sean $C_1, \ldots, C_k$ conjuntos de indices de programas y sea $C = C_1 \cap \cdots \cap C_k$.

A

Probar que C es un conjunto de indices de programas.

Como C1, ..., Ck son conjuntos de indices de programas se cumple que:

$$\forall i \in [1,k],\ C_i = \{x : \Phi_x \in \mathcal{C}_i\} (\mathcal{C}_i \text{ siendo una clase de funciones})$$

Puedo reescribir a C usando la definición de conjuntos de indices de programas:

$$C = \{x : \Phi_x \in \mathcal{C}_1 \wedge\ \cdots \wedge\ \Phi_x \in \mathcal{C}_k\} = \{x : \Phi_x \in (\mathcal{C}_1 \cap \cdots \cap \mathcal{C}_k)\} = \{x : \Phi_x \in \mathcal{C}\}$$

Donde $\mathcal{C}$ es $(\mathcal{C}_1 \cap \cdots \cap \mathcal{C}_k)$ (creo que vale).

B

Proponer un conjunto que no sea un conjunto de indices de programas y no sea computable.

Un ejemplo facil es $K = \{x : \Phi_x(x) \downarrow\}$.

No es un conjunto de indices de programas (prueba):

Supongo que K es un cip:

Sea p el numero de un programa tal que $\Phi_p(x) \downarrow \iff x = p$ (se que existe por el ejercicio 6). Entonces pertenece a K ya que

Por definición de K: $p \in D \iff \Phi_p(p) \downarrow$

Defino otro programa con numero q tal que $\Phi_p = \Phi_q$:

```
A   IF X != p GOTO A
```

Usando la equivalencia del 9, como K es un cip entonces se cumple que

$$p \in D \wedge\ \Phi_p = \Phi_q \implies q \in D$$

Pero por definición de q y K: $(\Phi_q(x) \downarrow \iff x = p) \implies \Phi_q(q) \uparrow \implies q \notin K$

Absurdo por suponer que K era un cip.

K no es computable ya que su indicadora no es computable

Ejercicio 9

Probar que son equivalentes:

- D es un conjunto de indices de programas.
- Para todo par de programas P y Q, si #P ∈ D y $\Psi_P = \Psi_Q$ entonces #Q ∈ D.

I $\implies$ II

Supongo que D es un conjunto de indices de programas.

Quiero ver que para todo par de programas P y Q con numeros p y q, si p $\in$ D y $\Psi_P = \Psi_Q$ entonces q $\in$ D.

Si D es un CIP por definición: $D = \{x : \Phi_x \in C\}$. Como p pertenece entonces se cumple que $\Phi_p \in C$.

Por hipotesis sabemos que $\Phi_p = \Psi_P = \Psi_Q = \Phi_q$, finalmente

$$\Phi_p \in C \land \Phi_q = \Phi_p \implies \Phi_q \in C \implies q \in D$$

II $\implies$ I

Supongo que para todo para de programas P y Q con numeros p y q, se cumple que $(p \in D \land \Phi_q = \Phi_p \implies q \in D)$.

Quiero ver que D es un CIP.

Supongo que D es no vacío y sean P y Q programas tal que $p \in D \land \Phi_q = \Phi_p$.

Puedo definir una clase de funciones $F = \{\Phi_p\}$

Se que lo estoy encarando mal, no encuentro cuales teoremas usar

Contrareciproca de Rice? por el absurdo Probar algo -> computable -> contra de rice a un absurdo

Ejercicio 10

g1

$$g_1(x) = \begin{cases} 1 & \text{si } \mathrm{Dom}\Phi_x^{(1)} = \emptyset \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y) \uparrow \forall y \\ 0 & \text{si no} \end{cases}$$

Pruebo que es un CIP (conjunto de indices de programa) usando el predicado del ejercicio 9.

$$D1 = \{x : \mathrm{Dom}\Phi_x^{(1)} = \emptyset\}$$

Sean p y q numeros de programas tal que $p \in D1$ y $\Phi_q = \Phi_p$

$$p \in D1 \implies \mathrm{Dom}\Phi_p^{(1)} = \emptyset \implies \mathrm{Dom}\Phi_q^{(1)} = \emptyset \implies q \in D1$$

Esto es equivalente a decir que D1 es un CIP.

Ahora veo que es no trivial (distinto de $\mathbb{N}$ y $\emptyset$). Para esto busco un programa que pertenezca y uno que no.

Es trivial definir un programa que siempre se cuelga: $\Phi_e(x) = \uparrow$ y definir una funcion constante $f(x) = 1$ (pr -> computable).

Por el Teorema de Rice, como D1 es un conjunto de indices de programas no trivial entonces es no computable.

g2

$$g_2(x,y) = \begin{cases} 1 & \text{si } \mathrm{Dom}\Phi_x^{(1)} \cup \mathrm{Dom}\Phi_y(1) = \mathbb{N} \\ 0 & \text{si no} \end{cases}$$

Ahora evalúo y = x para encontrar una reducción posible.

$$g_2(x,x) = \begin{cases} 1 & \text{si } \mathrm{Dom}\Phi_x^{(1)} = \mathbb{N} \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi_x^{(1)} \text{ es total} \\ 0 & \text{si no} \end{cases}$$

Esta función es la indicadora de Tot, voy a probar que no es computable.

$$D2 = \{x : \Phi_x^{(1)} \text{ es total } \} = \text{Tot}$$

Pruebo que es un conjunto de indices de programas:

Sean p y q numeros de programas tal que $p \in D2 \land \ \Phi_p = \Phi_q$.

Notar que $\Phi_p = \Phi_q \implies \text{Dom}(\Phi_p) = \text{Dom}(\Phi_q)$

$$p \in D2 \iff \text{Dom}(\Phi_p) = \mathbb{N} \implies \text{Dom}(\Phi_q) = \mathbb{N} \iff q \in D2$$

Como pudimos probar que ($p \in D2 \land \ \Phi_p = \Phi_q \implies q \in D2$) concluimos que D2 es un CIP. Falta ver que es no trivial, podemos usar los mismos programas del item anterior para probarlo.

Por el Teorema de Rice, como D2 es un conjunto de indices de programas no trivial entonces es no computable.

g3

$$g_3(x,y) = \begin{cases} 1 & \text{si } y \in \text{Dom}\Phi_x^{(1)} \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y) \downarrow \\ 0 & \text{si no} \end{cases}$$

Hago la reducción a un CIP fijando y en 1:

$$g_3(x,1) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(1) \downarrow \\ 0 & \text{si no} \end{cases}$$

Tenemos que g3(x,1) es la función indicadora del conjunto D3 definido como

$$D3 = \{x : \Phi_x^{(1)}(1) \downarrow\}$$

Pruebo que es un conjunto de indices de programas:

Sean p y q numeros de programas tal que $p \in D3 \land \ \Phi_p = \Phi_q$.

Notar que $\Phi_p = \Phi_q \implies (\Phi_p^{(1)}(1) \downarrow \iff \Phi_q^{(1)}(1) \downarrow)$

$$p \in D3 \iff \Phi_p^{(1)}(1) \downarrow \implies \Phi_q^{(1)}(1) \downarrow \iff q \in D3$$

Como pudimos probar que ($p \in D3 \land \ \Phi_p = \Phi_q \implies q \in D3$) concluimos que D3 es un CIP. Falta ver que es no trivial.

Podemos definir un programa que se cuelga para 1 y uno que no se cuelga bastante facil.

Por el Teorema de Rice, como D3 es un conjunto de indices de programas no trivial entonces es no computable.

g4

$$g_4(x,y) = \begin{cases} \Phi_x^{(1)}(\Phi_y^{(1)}(72)) & \text{si } \Phi_x^{(1)} \circ \Phi_y^{(1)} \text{ es total} \\ 73 & \text{si no} \end{cases}$$

Para la reducción hacemos que Φ_x devuelva siempre 74, pero por composición sabemos que como f(x)=74 es total, la composición se indefine sii su Φ_y se indefine. Llamo e al numero del programa constante 74.

$$g_4(e,y) - 73 = \begin{cases} 74 - 73 & \text{si } \Phi_y^{(1)} \text{ es total} \\ 73 - 73 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi_y^{(1)} \text{ es total} \\ 0 & \text{si no} \end{cases}$$

Pudimos hacer otra reducción a Tot, que ya probamos no computable en g2.

Ejercicio 11

Demostrar o refutar cada una de las siguientes afirmaciones.

A

Si B es computable entonces es c.e.

Por definición de conjunto computable, B es computable sii su función caracteristica es computable:

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

Quiero ver que B es computablemente enumerable. Lo es sii existe una funcion parcial computable $g : \mathbb{N} \to \mathbb{N}$ tal que

$$B = \{x : g(x) \downarrow\} = \text{Dom } g$$

Defino una g de la siguiente forma:

$$g(x) = \begin{cases} 1 & \text{si } B(x) \\ \uparrow & \text{si no} \end{cases}$$

Se ve que es trivial hacer un programa que computa esto, ya que B(x) es computable. En conclusión nuestra g es parcial computable ya que no es una función total. Finalmente pudimos encontrar g tal que

$$B(x) = \text{Dom } g$$

B

Si B es c.e. entonces B es computable o su complemento lo es.

Busco un contraejemplo que sea solamente c.e. (no co-c.e.). Tomo una variante parcial computable de halt:

$$B(x) = \{x : \Phi_x(x) \downarrow\} = K \qquad g(x) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x) \downarrow \\ \uparrow & \text{si no} \end{cases}$$

Vemos que g es parcial computable y se cumple B = Dom(g) entonces B es c.e. Como la indicadora de B es una variante de halt, sabemos que no es computable. Falta ver que el complemento de B no es computable.

$$\bar{B} = \{x : \Phi_x(x) \uparrow\} = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x) \uparrow \\ 0 & \text{si no} \end{cases}$$

Como podemos hacer una reducción a la indicadora de B con 1-f(x) se ve que la indicadora de $\bar{B}$ es no computable.

Tambien podría haber usado que B es c.e. y no es co-c.e. entonces no es computable. El mismo argumento para su complemento, B c.e. y no co-c.e. implica que $\bar{B}$ es co-c.e. y no c.e. entonces no es computable. Si alguno de los dos fuera computable su complemento tambien lo sería. Solo trate de tomar el camino de encontrar un contra ejemplo.

C

Si B es c.e. entonces su complemento es c.e.

Tomo el mismo contra ejemplo K

K es c.e. ya que existe g tal que K = Dom g

$$K = \{x : \Phi_x(x) \downarrow\} \qquad g(x) = \begin{cases} 1 & \text{si } \Phi_x(x) \downarrow \\ \uparrow & \text{si no} \end{cases}$$

Pero el complemento no es c.e: Supongamos que lo es, entonces K sería co-c.e. Como K sería c.e. y co-c.e. entonces tambien sería computable, lo cual es un absurdo.

Ejercicio 12

Decidir cuáles de los siguientes conjuntos son p.r., cuáles son computables, cuáles son c.e., cuáles son co-c.e. y demostrar en cada caso

C1

$$C_1 = \{x : \Phi_x^{(1)}(x) = 2 * x\}$$

Puedo definir su función indicadora como:

$$C_1(x) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x) = 2x \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x) = 2x \\ 0 & \text{si } \Phi_x^{(1)}(x) \neq 2x \vee \Phi_x^{(1)}(x) \uparrow \end{cases}$$

Escribo el siguiente programa para probar que C1 es c.e.

```
    Z <- Phi_X(X)
A   IF Z != 2X GOTO A
    Y <- 1
```

Sea e el numero de este programa, podemos definir su comportamiento por inspección:

$$\Phi_e^{(1)}(x) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(x) = 2x \\ \uparrow & \text{si no} \end{cases}$$

Finalmente podemos ver que existe la función $\Phi_e^{(1)}(x)$ parcial computable tal que $C_1 = \text{Dom } \Phi_e^{(1)}$, entonces C1 es c.e.

Puedo probar que la indicadora no es computable usando el teorema de la recursión y definiendo la función g(x,y)

$$g(x, y) = \begin{cases} 0 & \text{si } \Phi_x^{(1)}(x) = 2x \\ 2e & \text{si no} \end{cases}$$

Se llega al absurdo bastante directo.

En conclusión:

- Si C1 fuera pr entonces sería computable. C1 no es pr
- C1 es c.e.
- C1 no es computable
- Si C1 fuera co-c.e. entonces sería computable (por ser c.e.). C1 no es co-c.e.

C2

$$C_2 = \{x : 1 \in \text{Dom } \Phi_x^{(1)}\}$$

Podemos definir su indicadora como

$$C_2(x) = \begin{cases} 1 & \text{si } 1 \in \text{Dom } \Phi_x^{(1)} \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(1) \downarrow \\ 0 & \text{si no} \end{cases}$$

Es un conjunto de indices de programa no trivial, asi que por Rice sabemos que no es computable. Es c.e. (creo) asi que lo pruebo definiendo un programa tal que C2 = dom g

```
Z <- Phi_x(1)
Y <- 1
```

Sea e el numero de este programa, defino su comportamiento por inspección

$$\Phi_e^{(1)}(x) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(1) \downarrow \\ \uparrow & \text{si no} \end{cases}$$

Como la función es parcial y el programa e la computa entonces es parcial computable. Esto implica que el conjunto C2 es c.e.

En conclusión:

- Si C2 fuera pr entonces sería computable. C2 no es pr
- C2 es c.e.
- C2 no es computable
- Si C2 fuera co-c.e. entonces sería computable (por ser c.e.). C2 no es co-c.e.

C3

$$C_3 = \{x : \text{Dom } \Phi_x^{(1)} \subseteq \{0, \ldots, x\}\}$$

No es un conjunto de indices de programas asi que no puedo usar Rice, escribo su indicadora para ver que puedo hacer.

$$C_3(x) = \begin{cases} 1 & \text{si Dom } \Phi_x^{(1)} \subseteq \{0, \ldots, x\} \\ \uparrow & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y) \downarrow \implies 0 \leq y \leq x \\ \uparrow & \text{si no} \end{cases}$$

Para ver que Dom $\Phi_x^{(1)} \subseteq \{0, \ldots, x\}$ habría que probar que Φ se define solamente para elementos entre 0 y x, pero para eso habría que asegurarse que se indefine para todos los mayores a x (claramente no computable, hay que recorrer todo N).

Para ver lo contrario: Dom $\Phi_x^{(1)} \not\subseteq \{0, \ldots, x\}$ hay que ver que existe un elemento para el cual se define Φ y que es mayor a x. Podemos verlo encontrando el primer (o alguno) que cumpla (parcial computable!). Lo escribo en un programa:

```
    Z1 <- 0     ; tupla
A   Z2 <- STP( l(Z1)+X+1, X, r(Z1) )
    Z1 <- Z1 + 1
    IF Z2 = 0 GOTO A
    Y  <- 1
```

Observar que la tupla mapea a todos los naturales para sus dos parametros, como le sumo X+1 al primero entonces me mapea a todos los naturales $\geq X + 1$. De esta forma estoy recorriendo todas las posibles entradas con todas las posibles cantidades de pasos. Si el programa X termina para alguna entrada en alguna cantidad de pasos el programa debería terminar.

Sea e el numero del programa, por inspección:

$$\Phi_e^{(1)}(x) = \begin{cases} 1 & \text{si } \exists_y : (\Phi_x^{(1)}(y) \downarrow \wedge\ y > x) \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si Dom } \Phi_x^{(1)} \not\subseteq \{0, \ldots, x\} \\ 0 & \text{si no} \end{cases}$$

Como existe una función parcial computable tal que su dominio es $\bar{C_3}$ entonces C3 es co-c.e.

$$C_3(x) = \begin{cases} 1 & \text{si Dom } \Phi_x^{(1)} \subseteq \{0, \ldots, x\} \\ 0 & \text{si no} \end{cases}$$

Usando teorema de la recursión y una g(x,y) puedo probar que no es computable:

$$g_3(x, y) = \begin{cases} 256 & \text{si } C_3(x) \\ \uparrow & \text{si no} \end{cases} = \begin{cases} 256 & \text{si Dom } \Phi_x^{(1)} \subseteq \{0, \ldots, x\} \\ \uparrow & \text{si no} \end{cases}$$

En conclusión:

- Si C3 fuera pr entonces sería computable. C3 no es pr
- Si C3 fuera c.e. entonces sería computable (por ser co-c.e.). C3 no es co-c.e.
- C3 no es computable
- C3 es co-c.e.

C4

$$C_4 = \{\langle x, y\rangle : (\forall z \in (\text{Dom } \Phi_x^{(1)} \cap \text{Dom } \Phi_y^{(1)}))(\Phi_x^{(1)}(z) < \Phi_y^{(1)}(z))\}$$

Reescribo la indicadora de una forma que resulta mas obvio ver si es co-c.e.:

$$C_4(\langle x, y\rangle) = \begin{cases} 1 & \text{si } (\forall z \in (\text{Dom } \Phi_x^{(1)} \cap \text{Dom } \Phi_y^{(1)}))(\Phi_x^{(1)}(z) < \Phi_y^{(1)}(z)) \\ 0 & \text{si } (\exists z \in (\text{Dom } \Phi_x^{(1)} \cap \text{Dom } \Phi_y^{(1)}))(\Phi_x^{(1)}(z) \geq \Phi_y^{(1)}(z)) \end{cases}$$

Para computar el $\forall$ habría que iterar por todos los naturales (nunca terminamos), pero para el $\exists$ hay que encontrar el primero que cumpla. Trato de probar que es co-c.e. definiendo el siguiente programa con numero e.

```
    Z <- 0
C   IF STP(l(Z), l(X), r(Z)) = 0 GOTO A
    IF STP(l(Z), r(X), r(Z)) = 1 GOTO E
A   Z <- Z + 1
    GOTO C
E   IF Phi_{l(X)}(l(Z)) >= Phi_{r(X)}(l(Z)) GOTO A
    Y <- 1
```

Vemos el comportamiento por inspección:

$$\Phi_e^{(1)}(\langle x, y\rangle) = \begin{cases} 1 & \text{si } (\exists x \in (\text{Dom } \Phi_x^{(1)} \cap \text{Dom } \Phi_y^{(1)}))(\Phi_x^{(1)}(z) \geq \Phi_y^{(1)}(z)) \\ \uparrow & \text{si no} \end{cases}$$

Como $\Phi_e^{(1)}$ es parcial computable, entonces C4 es co-c.e. por definición.

Para ver que no es c.e. reduzco C4 a Tot usando el Teorema del Parametro. Evalúo x = i el numero del programa que computa la función identidad para llevarlo a algo parecido a h2 del ej7 (antes probé y=Id pero conviene esto):

$$C_4(\langle i, y\rangle) = \begin{cases} 1 & \text{si } (\forall z \in \text{Dom } \Phi_y^{(1)})(\Phi_y^{(1)}(z) > z) \\ 0 & \text{si no} \end{cases}$$

Me gustaría pasarle el numero de un programa que usa dos variables $\Phi_e(x,y)$ y reducirlo para llegar a otro programa tal que $\Phi_?(x) > x \iff \Phi_y(x)\downarrow$ (medio choto explicado). Escribo el programa con numero e:

```
Z <- Phi_X2(X1)
Y <- X1 + 1
```

Por inspección:

$$\Phi_e^{(2)}(x,y) = \begin{cases} x+1 & \text{si } \Phi_y^{(1)}(x)\downarrow \\ \uparrow & \text{si no} \end{cases}$$

Viendo la función se nota que $\Phi_e^{(2)}(x,y) > x \iff \Phi_y^{(1)}(x)\downarrow$.

Por teorema del parametro existe S pr tal que $\Phi_e^{(2)}(x,y) = \Phi^{(1)}_{S_1^1(y,e)}(x)$.

Finalmente encontre ese programa con una variable que buscaba: $\Phi^{(1)}_{S_1^1(y,e)}(x) > x \iff \Phi_y^{(1)}(x)\downarrow$

Supongo C4 computable, entonces puedo componerla por funciones computables para hacer otra computable, notar que Dom $\Phi^{(1)}_{S_1^1(y,e)}$ = Dom $\Phi_y^{(1)}$:

$$C_4(\langle i, S_1^1(y,e)\rangle) = \begin{cases} 1 & \text{si } (\forall z \in \text{Dom } \Phi_y^{(1)})(\Phi^{(1)}_{S_1^1(y,e)}(z) > z) \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } (\forall z \in \text{Dom } \Phi_y^{(1)})(\Phi_y^{(1)}(z)\downarrow) \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi_y^{(1)}(z) \text{ es total} \\ 0 & \text{si no} \end{cases} =$$

Absurdo ya que Tot no es computable.

C5

$$C_5 = \{\langle x,y,t,i\rangle : \exists\sigma.\text{SNAP}^{(1)}(x,y,t) = \langle i,\sigma\rangle\}$$

Ver que no es necesario usar el existe. Una 4upla pertenece al conjunto si el programa numero y con entrada x está en la linea i despues de ejecutar t pasos. Podemos reescribir su indicadora asi:

$$C_5(\langle x,y,t,i\rangle) = \begin{cases} 1 & \text{si } l\big(\text{SNAP}^{(1)}(x,y,t)\big) = i \\ 0 & \text{si no} \end{cases}$$

Al ser composición de funciones p.r. (SNAP, observadores, =, etc) podemos afirmar que C5 es p.r.

C6

$$C_6 = \{\langle x,y,i\rangle : \exists\sigma,t.\text{SNAP}^{(1)}(x,y,t) = \langle i,\sigma\rangle\}$$

Reescribo la indicadora:

$$C_6(\langle x,y,i\rangle) = \begin{cases} 1 & \text{si } \exists t : l\big(\text{SNAP}^{(1)}(x,y,t)\big) = i \\ 0 & \text{si } \forall t : l\big(\text{SNAP}^{(1)}(x,y,t)\big) \neq i \end{cases}$$

Tambien podemos ver que es puede reescribir usando C5:

$$C_6(\langle x,y,i\rangle) = \begin{cases} 1 & \text{si } \exists t : C_5(\langle x,y,t,i\rangle) \\ 0 & \text{si } \forall t : \neg C_5(\langle x,y,t,i\rangle) \end{cases}$$

Hacer el $\forall$ no es computable, pero si el $\exists$. Para probar que es c.e. uso la definición. Hago un programa con numero e que la computa usando C5 el programa que computa C5.

```
    Z <- 0
A   IF C5(<X1, X2, Z, X3>) = 1 GOTO E
    Z <- Z + 1
    GOTO A
E   Y <- 1
```

El programa computa la siguiente función:

$$\Phi_e^{(3)}(x,y,i) = \begin{cases} 1 & \text{si } \exists t : C_5(\langle x,y,t,i\rangle) \\ \uparrow & \text{si no} \end{cases}$$

Entonces puedo definir la siguiente función $g : \mathbb{N} \to \mathbb{N}$

$$g(T) = g(\langle x,y,i\rangle) = \Phi_e^{(3)}(T[0],T[1],T[2]) = \Phi_e^{(3)}(x,y,i)$$

Como es composición de parciales computables y pr entonces es parcial computable. Finalmente existe g parcial computable tal que Dom $g = C_6$ entonces C6 es c.e.

Para ver que no es computable voy a reducir C6 a una versión de Halt.

Notar que existe la función pr que devuelve la cantidad de instrucciones de un programa, dado su numero. Voy a pasarle la longitud + 1 a C6, entonces me va a decir si el programa llego a su final para cierta entrada o no. Se dice que **la configuración instantánea es terminal** si pasa eso.

$$C_6(\langle x,y,|y|+1\rangle) = \begin{cases} 1 & \text{si } \exists t : l\big(\text{SNAP}^{(1)}(x,y,t)\big) = |y|+1 \\ 0 & \text{si } \forall t : l\big(\text{SNAP}^{(1)}(x,y,t)\big) \neq |y|+1 \end{cases}$$

Una configuración instantanea es terminal sii el programa termina para la entrada dada, entonces:

$$C_6(\langle x,y,|y|+1\rangle) = \begin{cases} 1 & \text{si } \Phi_y^{(1)}(x)\downarrow \\ 0 & \text{si } \Phi_y^{(1)}(x)\uparrow \end{cases} = \text{HALT}(x,y)$$

Si suponiamos C6 computable llegabamos a un absurdo ya que HALT no es computable. Entonces C6 es no computable.

Ejercicio 13

Demostrar o refutar cada una de las siguientes afirmaciones.

A

Si $B_1,\ldots,B_k$ son c.e. entonces $\bigcup_{n\in\{1,\ldots,k\}} B_n$ es c.e.

Si $B_1,\ldots,B_k$ son c.e. por definición se cumple que existen funciones parciales computables $g_1,\ldots,g_k$ tales que $B_i = \text{Dom } g_i$.

Podemos definir el siguiente programa que usa los programas de las $g_1,\ldots,g_k$ cuyos numeros de programa son $e_1,\ldots,e_k$.

```
A   IF STP(X, e1, T) = 1 GOTO E
    IF STP(X, e2, T) = 1 GOTO E
    ...
    IF STP(X, ek, T) = 1 GOTO E
    T <- T + 1
    GOTO A
```

Este programa llamado P con numero e cumple que:

$$\Phi_e^{(1)}(x) = \begin{cases} 1 & \text{si } g_1(x)\downarrow \vee \cdots \vee g_k(x)\downarrow \\ \uparrow & \text{si no} \end{cases}$$

Como cada funcion se define solamente si x pertenece al conjunto correspondiente entonces:

$$\Phi_e^{(1)}(x) = \begin{cases} 1 & \text{si } x \in B_1 \vee \cdots \vee x \in B_k \\ \uparrow & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } x \in B_1 \cup \cdots \cup B_k \\ \uparrow & \text{si no} \end{cases}$$

Finalmente $\bigcup_{n\in\{1,\ldots,k\}} B_n$ es c.e.

B

Es falso, voy a mostrarlo haciendo la union de infinitos conjuntos finitos (entonces c.e.) que dan como resultado un conjunto no c.e. (Tot).

Defino un conjunto para cada natural, si el natural n está en Tot el n-esimo conjunto es vacío y si no está en Tot es el conjunto {n}:

$$(B_n)_{n\in\mathbb{N}} = \begin{cases} \emptyset & \text{si } n \notin \text{Tot} \\ \{n\} & \text{si no} \end{cases}$$

Como son todos conjuntos finitos entonces son c.e., la union da como resultado:

$$\bigcup_{k\in\mathbb{N}} B_k = \text{T}ot$$

C

Si $B_1, \ldots, B_k$ son c.e. entonces $\bigcap_{n\in\{1,\ldots,k\}} B_n$ es c.e.

Si $B_1, \ldots, B_k$ son c.e. por definición se cumple que existen funciones parciales computables $g_1, \ldots, g_k$ tales que $B_i = \text{Dom } g_i$.

Podemos definir el siguiente programa que usa los programas de las $g_1, \ldots, g_k$ llamados $P_1, \ldots, P_k$.

```
Z <- P1
...
Z <- Pk
Y <- 1
```

Este programa llamado P con numero e cumple que:

$$\Phi_e^{(1)}(x) = \begin{cases} 1 & \text{si } g_1(x)\downarrow \wedge \cdots \wedge g_k(x)\downarrow \\ \uparrow & \text{si no} \end{cases}$$

Como cada funcion se define solamente si x pertenece al conjunto correspondiente entonces:

$$\Phi_e^{(1)}(x) = \begin{cases} 1 & \text{si } x \in B_1 \wedge \cdots \wedge x \in B_k \\ \uparrow & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } x \in B_1 \cap \cdots \cap B_k \\ \uparrow & \text{si no} \end{cases}$$

Finalmente $\bigcap_{n\in\{1,\ldots,k\}} B_n$ es c.e.

completar D

Hacer lo mismo que B pero con

$$B_n = \begin{cases} \mathbb{N} - \{n\} & \text{si } n \in \text{T}ot \\ \mathbb{N} & \text{si no} \end{cases}$$

Ejercicio 14

Sea B un conjunto *infinito*

A

($\Longleftarrow$)

Es directo ya que:

$\exists f$ inyectiva y computable : Im $f = B \implies \exists f$ computable : Im $f = B \implies \exists f$ parcial computable : Im $f = B$

Que es la definición de c.e

($\Longrightarrow$)

Por equivalencias de c.e:

Si B es c.e entonces B es el rango de una f computable. Uso esta f y minimización no acotada (p.c.) para construir una función parcial computable que hace esto:

$$g(x) = f\Big(\min_k\{k \geq x \wedge \forall_{y<x}(f(k) \neq f(y))\}\Big)$$

Finalmente esta g es parcial computable e inyectiva. Veamos que al ser B un conjunto infinito, dado un x siempre existe uno mas grande que tambien pertenece. Entonces la minimización no acotada nunca se cuelga. Esto implica que nuestra composición es total y por ende computable.

B

($\Longrightarrow$)

B computable implica B es c.e. y usando el item A implica que B es el rango de una f computable e inyectiva. Como f es total e inyectiva (ademas de ser igual a B) entonces la imagen es infinita.

Podemos definir una función que siempre termina y que devuelve una salida mayor a todas sus salidas anteriores:

$$g(x) = f\Big(\min_k\{k \geq x \wedge \forall_{y<x}(f(k) > f(y))\}\Big)$$

Es una minimización propia (sabemos que siempre termina por el argumento anterior) entonces es computable.

($\Longleftarrow$)

Sea f una funcion computable y estrictamente creciente tal que Im(f)=B.

Como es total y estrictamente creciente entonces puedo chequear facilmente si un numero n pertenece o no pertenece viendo la imagen de f con las entradas desde 0 hasta n (podría hacerlo de forma optima pero bue). Este argumento sirve para ver que es c.e. y co-c.e. o tambien para ver que es computable directamente:

$$B(x) = \exists_{y \leq x} : (f(y) = x)$$

Al ser una composición de funciones computables y minimización acotada entonces es computable.

Ejercicio 15

Tot no es co-c.e.

Para ver esto pruebo que $\overline{\text{Tot}}$ no es c.e.

Supongo $\overline{\text{Tot}}$ c.e. para llegar a un absurdo

Por definición esto implica que existe una función h parcial computable tal que Dom $h = \overline{\text{Tot}}$

$$h(x) = \begin{cases} 1 & \text{si } \Phi_x \text{ es parcial} \\ \uparrow & \text{si no} \end{cases}$$

Ahora defino otra función g parcial computable usando h tal que g(x,y)=h(x). Por el teorema de la recursion existe un programa con numero e tal que $\Phi_e^{(1)}(y) = g(e, y)$ entonces:

$$\Phi_e^{(1)}(y) = \begin{cases} 1 & \text{si } \Phi_e^{(1)} \text{ es parcial} \\ \uparrow & \text{si no} \end{cases}$$

Veo que cada guarda lleva a un absurdo:

$\Phi_e^{(1)}$ es parcial $\implies \Phi_e^{(1)}(y) = 1, \forall_y \implies \Phi_e^{(1)}$ es total

$\Phi_e^{(1)}$ es total $\implies \Phi_e^{(1)}(y) \uparrow, \forall_y \implies \Phi_e^{(1)}$ es parcial

Tot no es c.e.

Supongo Tot c.e. entonces por equivalencia existe una función f computable tal que el rango es igual a Tot. Usando a p, el numero de programa que computa a f, defino al siguiente programa con numero e:

```
Z <- P
Y <- Phi_Z(X)
Y <- Y + 1
```

Por inspección vemos que el programa computa la siguiente función

$$\Phi_e^{(1)}(x) = \Phi_{f(x)}^{(1)}(x) + 1$$

Como la función es total, el numero e pertenece a Tot, entonces existe un k tal que $f(k) = e$, evaluo x = k

$$\Phi_e^{(1)}(k) = \Phi_{f(k)}^{(1)}(k) + 1 = \Phi_e^{(1)}(k) + 1$$

Absurdo que surge de suponer que Tot es computablemente enumerable.

Ejercicio 16

ID

$$\text{ID} = \{x : \Phi_x^{(1)} \text{ es la función identidad}\}$$

Otra forma de reescribir la indicadora de ID es:

$$ID(x) = \begin{cases} 1 & \text{si } (\forall_z : \Phi_x^{(1)}(z) = z) \\ 0 & \text{si no} \end{cases}$$

Defino un programa con numero e tal que $\Phi_e^{(2)}(x,y) = x \iff \Phi_y^{(1)}(x)\downarrow$. Observar que necesito que dependan de la misma variable (es la que usa el $\forall$ de la definición de Id y de Tot).

```
Z <- Phi_X2(X1)
Y <- X1
```

Por inspección:

$$\Phi_e^{(2)}(x,y) = \begin{cases} x & \text{si } \Phi_y^{(1)}(x)\downarrow \\ \uparrow & \text{si no} \end{cases}$$

Por el teorema del parametro existe una función pr S tal que

$$\Phi_e^{(2)}(x,y) = \Phi^{(1)}_{S_1^1(y,e)}(x)$$

Como son iguales ya se que $\Phi^{(1)}_{S_1^1(y,e)}(x) = x \iff \Phi_y^{(1)}(x)\downarrow$

Uso S en ID:

$$ID(S_1^1(x,e)) = \begin{cases} 1 & \text{si } (\forall_z : \Phi^{(1)}_{S_1^1(y,e)}(z) = z) \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } (\forall_z : \Phi_x^{(1)}(z)\downarrow) \\ 0 & \text{si no} \end{cases}$$

Quedó igual a Tot pero no se como cerrar que no es c.e. ni co-c.e.

Voy a usar esto:

$$\text{B es c.e.} \iff \text{Existe una función f pr/pc/c tal que Im f} = B$$

Supongo ID c.e. (vale lo mismo para co-c.e.), entonces es la imagen de una función p.r. Como S es p.r. entonces el conjunto que surge de componer ambas funciones tambien es pr. Usando todo esto la conclusión es que Tot es p.r. lo cual es un absurdo.

Supongo $\overline{ID}$ c.e. , entonces es la imagen de una función p.r. Como S es p.r. entonces el conjunto que surge de componer ambas funciones tambien es pr. Usando todo esto la conclusión es que $\overline{Tot}$ es p.r. lo cual es un absurdo.

S

$$S = \{x : \text{Im } \Phi_x^{(1)} = \mathbb{N}\}$$

$$S(x) = \begin{cases} 1 & \text{si Im } \Phi_x^{(1)} = \mathbb{N} \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \forall_y \exists_z : \Phi_x^{(1)}(z) = y \\ 0 & \text{si no} \end{cases}$$

Defino un programa con numero e que computa la función Identidad si termina.

```
Z <- Phi_X2(X1)
Y <- X1
```

Por inspección vemos que

$$\Phi_e^{(2)}(x,y) = \begin{cases} x & \text{si } \Phi_y^{(1)}(x)\downarrow \\ \uparrow & \text{si no} \end{cases}$$

Notar que la imagen de e es N sii Phi termina para todas las entradas (es inyectiva).

$$\text{Im } \Phi_e^{(2)} = \mathbb{N} \iff \Phi_y^{(1)}(x)\downarrow \forall x$$

Por teorema del parametro existe una S p.r. tal que

$$\Phi^{(1)}_{S^1_1(y,e)}(x) = \Phi^{(2)}_e(x,y)$$

Falta hacer la reducción:

$$S(S^1_1(y,e)) = \begin{cases} 1 & \text{si Im } \Phi^{(1)}_{S^1_1(y,e)} = \mathbb{N} \\ 0 & \text{si no} \end{cases} = \begin{cases} 1 & \text{si } \Phi^{(1)}_y \text{ es total} \\ 0 & \text{si no} \end{cases} = \text{Tot}(y)$$

Mismo argumento que antes.

Observación

Podria haber hecho que mi programa copie cualquier función biyectiva ya que:

Si se cuelga en una entrada especifica, la función (en este caso Id) no puede devolver lo mismo que hubiese devuelto si Phi no se colgó (es inyectiva).

Si nunca se cuelga entonces devuelve todos los naturales, ya que la función (en este caso Id) es sobreyectiva.

$$\Phi^{(2)}_e(x,y) = \begin{cases} h(x) & \text{si } \Phi^{(1)}_y(x) \downarrow \\ \uparrow & \text{si no} \end{cases}$$

Ejercicio 17 (borrador)

Dar un conjunto C no vacío y contenido en $\mathbb{N}$ tal que

$$\forall f \text{ función p.r. }, \exists k : \Big(k \in (C \cup \text{Im } f) \wedge k \notin (C \cap \text{Im } f)\Big)$$

Es lo mismo que pedir un o excluyente:

$$\forall f \text{ función p.r. }, \exists k : \Big(k \in C \ \veebar \ k \in \text{Im } f\Big)$$

un conjunto es el rango de una función f sii:

$$\forall k \in \text{Im } f : k \in C \iff \forall x : f(x) \in C$$

Si lo niego:

$$\exists k \in \text{Im } f : k \notin C \iff \exists x : f(x) \notin C$$

Como Tot no es ce por la equivalencia se cumple que para todas las funciones p.r. Tot no es el rango de ninguna. Se puede reescribir como que existe al menos un elemento de la imagen de todas las f p.r. que no está en C.

Si hubiera alguna f pr tal que todos sus elementos pertenecen a C, entonces C sería c.e.