

11) 1) $A = \{x \mid \exists_x (x) \uparrow \wedge \exists_x (x) \in \mathbb{N}\}$

$$f_A(x) = \begin{cases} 1 & \text{si } \exists_x (x) \uparrow \vee \exists_x (x) \in \mathbb{N} \\ 0 & \text{c.c.} \end{cases} \quad \text{I}$$

Posteriormente se cuenta que el predicado I es un predicado tautológico ya que para todo programa representado por naturales la tabla tiene 2 opciones. O se define, o devuelva algo, necesariamente ese algo tiene que ser un natural. Posteriormente transformar f_A de la siguiente forma:

$$f_A(x) = \begin{cases} 1 & \text{si } \text{TRUE} = 1 \\ 0 & \text{c.c.} \end{cases}$$

(computable)

Al ser $f_A(x)$ la función de I, esto representa la función características de A, entonces A también es computable.

Sabemos además que A computable $\Rightarrow A \in \text{ce} \wedge A \text{co-ce}$.

Así mismo, podemos encontrar de forma muy simple una función PR que representa al conjunto A:

$$f(x) = \text{neces}(\text{ces}(x))$$

• COMPUTABLE SI

• CO-CE SI

• CE SI

• PR SI

$$3) C = \{x \mid \exists z (y \leq z, \forall y \}$$

Podemos ver que C es el conjunto de funciones que cumplen que su rango es por lo menos \geq que su dominio. Entonces, C es un conjunto de índice y podemos usar el ~~*restable de Turing*~~ TEOREMA DEL $O \leftarrow$ para

Sabemos que si C es un conjunto $\neq \emptyset$ y $\neq \mathbb{N}$ entonces A no es computable. Probemos que C es trivial:

$\exists f \in C :$

$$f(x) = 0, \text{ vale que } f(x) \leq zx \quad \forall x$$

$\exists g \notin C :$

$$g(x) = zx + 1, \text{ no vale que } g(x) \leq zx \quad \forall x$$

$\Rightarrow C$ no es computable.

Sabemos que C no computable $\Rightarrow C$ no PR.

Vemos a priori que esta función es muy parecida a TOT y por lo tanto no va a ser CE ni co-CE.

$$f_C(x) = \begin{cases} 1 & \exists y (y \leq x \quad \forall y \\ 0 & \text{CE} \end{cases} \quad | \quad \text{TOT}(x) = \begin{cases} 1 & \exists y (y \leq x \quad \forall y \\ 0 & \text{CE} \end{cases}$$

Vemos que TOT es indexado más "fácil" que f_C , ya que solo pide que $\exists x$ termine. Entonces, intentaremos reducir TOT a f_C .

Si $\exists f$ computable / $x \in \text{TOT} \Leftrightarrow f(x) \in f_C$.

\Rightarrow Si TOT no es computable
CE
co-CE $\Rightarrow f_C$ no es computable
CE
co-CE

~~Vamos al siguiente punto:~~

Sabemos que TOT no es computable ni ce ni co-ce, porque solo nos falta ver que existe en f. Para eso vamos al siguiente punto P:

$$\left[\begin{array}{l} z_1 \leftarrow \Phi_x(x_2) \\ y \leftarrow 2 \cdot z_1 + 1 \end{array} \right]$$

Sea f la función implementada, entonces podemos ver que si $\Phi_x(y) \downarrow \Rightarrow f(\Phi_x(y)) \geq y$

La vuelta es trivial ya que si $f(\Phi_x(y)) \neq y \Rightarrow \Phi_x(y) \downarrow$

- COMPUTABLE NO
- CO-CE NO
- CE NO
- PR NO

2) $B = \{(x, y) \mid \Phi_x(\Phi_y(x+y)) = y\}$

$$f_B(x, y) = \begin{cases} 1 & \text{si } \Phi_x(\Phi_y(x+y)) = y \\ 0 & \text{cc} \end{cases}$$

Sea e la función nula $\tilde{f}(x) = f_B(x, 0)$:

$$\tilde{f}(x) = \begin{cases} 1 & \text{si } \Phi_x(\Phi_e(x+e)) = e \\ 0 & \text{cc} \end{cases} \Rightarrow \tilde{f}(x) = \begin{cases} 1 & \text{si } \Phi_x(0) = 0 \\ 0 & \text{cc} \end{cases}$$

Podemos ver que \tilde{f} es la función característica del conjunto de los indices cuyas funciones son iguales a 0 cuando se lo evalúa en 0.

Vemos que es no trivial:

(4)

congents de \tilde{f}
 $f \in C$:

$$f(x) = 0$$

$g \notin C$:

$$g(x) = 1$$

Entonces \tilde{f} no es computable. Llamé \tilde{f} origen de una transformación no computable de f_B y \tilde{f} no es computable $\Rightarrow f_B$ no es computable $\Rightarrow f_B$ no es PR.

Vemos que B es CE:

Son P:

$z_1 \leftarrow \exists_{x_2} (x_1 + x_2)$ $z_2 \leftarrow \exists_x (z_1)$ [A] if ($z_2 \neq x_2$) GOTO A $y \leftarrow 1$	$\left(\begin{array}{l} \text{en realidad lo computa con la tijera, pero la transformación} \\ \text{es trivial con } l(\cdot) \text{ y } r(\cdot). \end{array} \right)$ Pádron ver que este programa computa: $f_B(x_1, x_2) = \begin{cases} 1 & \text{si } \exists_x (\exists_y (x+y) = y) \\ & \uparrow \text{CC} \end{cases}$ Ents no origina que f_B es CE. \square
---	--

\therefore Llamé f_B no es computable y es CE $\Rightarrow f_B$ no es co-CE.

- COMPUTABLE NO
- CO-CE NO
- CE SI
- PR NO

4) $D = \{x \mid (\exists_{S4321}(x) \wedge x < 10) \vee g(x) = f\}$ g computable

Vemos que en CE:

$$\left[\begin{array}{l} z_1 \leftarrow \exists_{S4321}(x_1) \\ \text{if } (z_1 = 1) \text{ goto F} \\ [\text{A}] \text{ if } (x \geq 10) \text{ goto A} \\ z_2 \leftarrow \exists_{S4321}(x_1) \\ y \leftarrow 1 \end{array} \right] \quad \begin{array}{l} \text{Podemos ver que este programa computa:} \\ f(x) = \begin{cases} 1 & \forall (x \in S4321 \wedge x < 10) \vee g(x) = f \\ 0 & \text{cc} \end{cases} \\ \therefore D \text{ es ce} \end{array}$$

Q.D. Vemos computable:

$$\text{Vemos que } f_D(x) = \begin{cases} 1 & (\exists_{S4321}(x) \wedge x < 10) \vee g(x) = f \\ 0 & \text{cc} \end{cases}$$

$$\text{en igual d. } f_D(x) = \begin{cases} 1 & g(x) = f \\ 1 & \exists_{S4321}(x) \wedge x < 10 \quad \text{(II)} \\ 0 & \text{cc} \end{cases}$$

El predicado $g(x) = f$ no molesta porque g es computable.

Podemos ver solamente que en resibiendo el predicado (II) termina

molesta porque en rebte un conjunto de x finito (10). *ni lo hace en una orilla*

Entonces f_D es computable $\Rightarrow D$ es computable $\Rightarrow D$ es CE y co-CE.

D es PR porque para comprobar $g(x) = f$ podemos usar STEP y SNAP

con una minimización acotada suficiente grande. Existe $t / STP = 1$ por computable.

Para ~~reducir~~ el predicado (II) reparamos en 10 caso y que se arregle el que lo quiere usar.

- COMPUTABLE SI
- CORRECT SI
- CE SI
- PR SI