Codificación de programas

TEOREMA 1

Sea $\langle , \rangle : \mathbb{N}^2 \to \mathbb{N} / \langle x, y \rangle = 2^x(2y+1)-1$ la función par, una función biyectiva.

DEMOSTRACIÓN

Inyectiva:

$\langle x,y \rangle = \langle a,b \rangle \Leftrightarrow 2^x(2y+1)-1 = 2^a(2b+1)-1 \Leftrightarrow 2^x(2y+1) = 2^a(2b+1)$

Notemos que 2^x y 2^a son par y $(2y+1)$ y $(2b+1)$ son impares $\Rightarrow$ por TFA, $x=a$ y $(2y+1)=(2b+1)$
$\Rightarrow x=a \wedge y=b \Rightarrow \langle x,y \rangle = \langle a,b \rangle$

Obs: $V_2(z+1)$ = potencia de 2 en la descomposición por primos

Ej. $V_2(36) = V_2(9.4) = V_2(2^2.3^2) = 2$

Ej. $V_3(15) = V_3(3^1.5^1) = 1$

Sobreyectiva:

Sea $z \in \mathbb{N}$, qvq $\exists\ (x,y) \in \mathbb{N}^2 / \langle x,y \rangle = z \Leftrightarrow 2^x(2y+1)-1 = z \Leftrightarrow 2^x(2y+1) = z+1$.
Luego, $x = V_2(z+1) \in \mathbb{N}$ e $y = \left(\frac{z+1}{2^x} - 1\right) \cdot \frac{1}{2} \in \mathbb{N}$ pues $\frac{z+1}{2x} = 2y+1 =$ impar.

$\Rightarrow \langle , \rangle$ es biyectiva

EJEMPLO

Hallar $(x,y) \in \mathbb{N}^2 / \langle x,y \rangle = 11 \Leftrightarrow 2^x(2y+1)-1 = 11 \Leftrightarrow 2^x(2y+1) = 12 = 2^2.3 \Leftrightarrow x=2$ e $y=1 \Rightarrow \langle 2,1 \rangle = 11$

TEOREMA 2

Sea $\ell : \mathbb{N} \to \mathbb{N} / \ell(z) = x$ c/ $\langle x,y \rangle = z$
Sea $r : \mathbb{N} \to \mathbb{N} / r(z) = y$ c/ $\langle x,y \rangle = z$
bien definidas pues $\exists! x\ \exists! y / \langle x,y \rangle = z$

$\Rightarrow \ell, r$ y $\langle , \rangle$ son RP

DEMOSTRACIÓN

Veamos que $\langle , \rangle$ es RP:

Obs: $POT(x,y) = (x+1)^y$

$\langle x,y \rangle = POT(1,x).SUC(PROD(2,y)) - 1$ notemos que $POT(1,x).SUC(PROD(2,y)) \geq 1 \Rightarrow$ la resta es eq a la resta truncada $\Rightarrow \langle x,y \rangle = \dot{-} \circ (PROD \circ (POT \circ (h_1 \times \pi_1) \times SUC \circ PROD \circ (h_2 \times \pi_2)) \times h_1) \Rightarrow \langle , \rangle$ es RP por ser composición de funciones RP.

Veamos que ℓ es RP:

$\ell(z) = \min_{x \leq z}\ \exists_{y \leq z}\ \langle x,y \rangle = z \Rightarrow \ell(z) = \min_{x \leq z}\ \exists_{y \leq z}\ EQ \circ (\langle x,y \rangle \times z) \Rightarrow \ell$ es RP por ser mínimo acotado de un existencial acotado aplicado a un predicado P RP (pues C_P es composición de funciones RP)

Obs: $2^x > x$ se prueba por inducción

Sup que $x \geq z+1 \Rightarrow 2^x > x \geq z+1 \Rightarrow 2^x > z+1 \Rightarrow$ como $(2y+1) \geq 1$, $2^x(2y+1) \geq 2^x > z+1 \Rightarrow 2^x(2y+1)-1 > z$
$\Rightarrow z > z$ ABS! pues $2^x(2y+1)-1 = z \Rightarrow x \leq z$

Sup $y \geq z+1 \Rightarrow (2y+1) > y \geq z+1 \Rightarrow 2y+1 > z+1$ y $2^x \geq 1 \Rightarrow 2^x(2y+1) \geq 2y+1 > z+1 \Rightarrow 2^x(2y+1)-1 > z \Rightarrow z > z$ ABS!
$\Rightarrow y \leq z$

Veamos que r es RP:

De manera análoga se prueba que $y \leq z$ y $x \leq z$.

$r(z) = \min_{y \leq z}\ \exists_{x \leq z}\ \langle x,y \rangle = z \Rightarrow C_P(x,y,z) = EQ \circ (\langle , \rangle \circ (\pi_1 \times \pi_2) \times \pi_3) \Rightarrow C_P$ es RP por ser composición de funciones RP $\Rightarrow r$ es RP por ser mínimo acotado de un existencial acotado aplicado a un predicado RP

DEFINICIÓN 1: Funciones de Gödel

$\Rightarrow$ numeración de Gödel

Sea $k \in \mathbb{N}$ / para cada k se define $[,] : \mathbb{N}^{k+1} \to \mathbb{N} / [(a_0, \dots, a_k)] = P_0^{a_0} P_1^{a_1} \dots P_k^{a_k}$ dnd P_i es el i-ésimo primo
i.e. $P_0 = 2$, $P_1 = 3$, $P_2 = 5 \dots$

Nombre: $[(a_0, \dots, a_k)]$ se llama número de Gödel $(a_0, \dots, a_k)$

EJEMPLOS

1. $[(3,1,0,4)] = 2^3.3^1.5^0.7^4 \Rightarrow Dom([\,]) = \mathbb{N}^4$
2. $[(3,1,0,4,0)] = 2^3.3^1.5^0.7^4.11^0 \Rightarrow Dom([\,]) = \mathbb{N}^5$

TEOREMA 3

Las funciones de Gödel son inyectivas y RP PERO no son sobreyectivas.

DEMOSTRACIÓN

Inyectiva:

$[(a_0, \dots, a_k)] = [(b_0, \dots, b_k)] \Leftrightarrow P_0^{a_0} \dots P_k^{a_k} = P_0^{b_0} \dots P_k^{b_k} \Rightarrow$ por TFA, $a_j = b_j$, $0 \leq j \leq k \Rightarrow (a_0, \dots, a_k) = (b_0, \dots, b_k)$

No sobreyectiva: $P_{k+1} \notin Im([\,]_k)$ Obs: 0 tmp pertenece a $Im([\,]_k)$

$f : \mathbb{N} \to \mathbb{N} / f(n) = P_n$ siendo P_n el $(n+1)$-ésimo primo $\Rightarrow f$ RP (Práctica)

Función RP:

$[(a_0, \dots, a_k)] = P_0^{a_0} \dots P_k^{a_k} = POT \circ (f(0) \dot{-} 1, a_0) \cdot \ldots \cdot POT \circ (f(k) \dot{-} 1, a_k)$ dnd $f(j) \geq 1$, $0 \leq j \leq k \Rightarrow [\,]$ es RP por ser composición de funciones RP (PROD, POT, f, π_j, h_i, $\dot{-}$)

DEFINICIÓN 2: Indicadores de número de Gödel

1. $\cdot[\cdot] : \mathbb{N}^2 \to \mathbb{N} / x[i] = V_{P_i}(x)$
2. $|\cdot| : \mathbb{N} \to \mathbb{N} / |n| = \begin{cases} long(n) & \text{si } n \neq 0 \\ 0 & \text{si } n = 0 \end{cases}$

Obs: si $n \neq 0$, $n = P_0^{\alpha_0} P_1^{\alpha_1} \dots P_k^{\alpha_k}$ c/ $\alpha_k \neq 0 \Rightarrow long(n) = long([(\alpha_0, \dots, \alpha_k)]) = k+1$

Obs: $long([2,0,0,0,1,0]) = 5$

Ej. $long(44) = long(2^2.11) = long([2,0,0,0,1]) = 5$

TEOREMA 4

Las funciones long y Vp son RP

DEMOSTRACION

Veamos que $\cdot[\cdot]$ es RP:

$x[i] = \min_{t \leq x} \; p_i^{t+1} \nmid x = \max_{t \leq x} \; p_i^{t} \mid x$

Notemos que $t \leq x$ porque $2^x > x \Rightarrow 2^x \nmid x$ (para cualquier otro primo pasa lo mismo).

⇒ DIV devuelve 1 si p_i^{t+1} divide x y 0 sino

Ahora, $C_P = \alpha(\text{DIV}(p_i^{t+1}, x)) = \alpha \circ \text{DIV} \circ (\text{POT} \circ (f(i) \dot- 1, \text{SUC}(t)), x) \Rightarrow$ P es RP pues C_P es composicion de funciones RP (α, DIV, f, π_j, h_1, $\dot-$, SUC) $\Rightarrow \cdot[\cdot]$ es RP por ser minimo acotado aplicado a un predicado RP.

Veamos que $|\cdot|$ es RP:

$|n| = \max_{t \leq n} \; p_{t+1} \nmid n \wedge p_t \mid n$

$C_P(t,n) = \text{AND} \circ (\alpha(\text{DIV}(f(\text{SUC}(t)), n)) \times \text{DIV}(f(t), n)) \Rightarrow$ P es RP pues C_P es composicion de funciones RP (π_j, f, AND, α, DIV, SUC) $\Rightarrow |\cdot|$ es RP por ser max acotado aplicado a un predicado RP.

DEFINICION 3: Codificacion de programas

Variables: $Y, X_1, Z_1, X_2, Z_2, \dots$ i.e. Y en posicion 1, X_1 en posicion 2, etc.

Etiquetas: $A_1, B_1, C_1, D_1, E_1, A_2, B_2, C_2, \dots$ i.e. A_1 en posicion 1, A_2 en posicion 6, etc.

Objetivo: asignar un num natural a cada programa en lenguaje S i.e. $\#P \in \mathbb{N}$.

1. Si $P \neq P' \Rightarrow \#P \neq \#P'$
2. Dado cualquier $n \in \mathbb{N}$, queremos que $\exists$ P programa en lenguaje S tal que $\#P = n$

⇒ queremos definir la funcion biyectiva

$\boxed{\# : \{\text{Programas en lenguaje S}\} \to \mathbb{N}}$

Instrucciones:

1. $V \leftarrow V+1$
2. $V \leftarrow V-1$
3. IF $V \neq 0$ GOTO L
4. $V \leftarrow V$

⇒ cada instruccion queda determinada por tres numeros i.e. $\# : \{\text{Instrucciones}\} \to \mathbb{N} \,/\, \#I = \langle a, \langle b, c \rangle\rangle$

- a esta asociado a la etiqueta de I $\Rightarrow a = \begin{cases} 0 & \text{si I no tiene etiqueta} \\ \#L & \text{si I tiene adelante la etiqueta L} \end{cases}$

⇒ #L = posicion de la etiqueta L en la lista de etiquetas

- c esta asociado a la variable que aparece en la instruccion $\Rightarrow c = \#V - 1$ and V es la variable que aparece en la instruccion I y #V es la posicion de V en la lista de variables

Obs: le resto uno porque quiero que abarque todos los numeros naturales

- b esta asociada al tipo de instruccion $\Rightarrow b = \begin{cases} 0 & \text{si I es } V \leftarrow V \\ 1 & \text{si I es } V \leftarrow V+1 \\ 2 & \text{si I es } V \leftarrow V-1 \\ \#L+2 & \text{si I es IF } V \neq 0 \text{ GOTO L} \end{cases}$

EJEMPLO

Hallar el codigo de: [B_1] IF $Z_3 \neq 0$ GOTO A_1 $\Rightarrow I = \langle a, \langle b, c\rangle\rangle = \langle 2, \langle 3, 6\rangle\rangle$

$\Rightarrow \langle 3, 6\rangle = 2^3(2.6+1) - 1 = 103 \Rightarrow I = \langle 2, 103\rangle = 2^2(2.103+1) - 1 = 827 \Rightarrow$ el codigo es 827

TEOREMA 5

La funcion $\# : \{\text{Instrucciones}\} \to \mathbb{N} \,/\, \#I = \langle a, \langle b, c\rangle\rangle$ es biyectiva.

DEMOSTRACION

Inyectividad:

$\langle a, \langle b, c\rangle\rangle = \langle x, \langle y, z\rangle\rangle \Leftrightarrow a = x \wedge \langle b, c\rangle = \langle y, z\rangle \Leftrightarrow a = x \wedge b = y \wedge c = z \Rightarrow I(a,b,c) = I(x,y,z)$

Obs: la funcion par es biyectiva

Sobreyectividad:

Sea $z \in \mathbb{N}$, como $\langle , \rangle$ es biyectiva $\Rightarrow \exists\, a, x \in \mathbb{N} \,/\, \langle a, x\rangle = z$. Nuevamente, como $\langle , \rangle$ es sobreyectiva, $\exists\, b, c \in \mathbb{N}$ tal que $\langle b, c\rangle = x \Rightarrow \exists\, a, b, c \in \mathbb{N} \,/\, \langle a, \langle b, c\rangle\rangle = z$.

EJEMPLO

Hallar la instruccion de codigo 6.

⚠ Unica instruccion de codigo 6

$6 = 2.3 \Rightarrow 2^0(2.3+1) - 1 \Rightarrow I = \langle 0, 3\rangle$ and $3 = 2^2(2.0+1) - 1 \Rightarrow I = \langle 0, \langle 2, 0\rangle\rangle \Rightarrow I = Y \leftarrow Y - 1$

DEFINICION 4

Sea P un programa con $I_1, I_2, \dots, I_k$ instrucciones.

Defino $\# : \{\text{Programas en lenguaje S}\} \to \mathbb{N} \,/\, \#P_{I_1, I_2, \dots, I_k} = [\#I_1, \dots, \#I_k] - 1$

EJEMPLO

Hallar #P / P: [B_1] $X_1 \leftarrow X_1 - 1$ (I_1)

IF $X_1 \neq 0$ GOTO B_1 (I_2)

- $\#I_1 = \langle 2, \langle 2, 1\rangle\rangle = \langle 2, 11\rangle = 2^2(2.11+1) - 1 = 91$
- $\#I_2 = \langle 0, \langle 4, 1\rangle\rangle = \langle 0, 47\rangle = 2^0(2.47+1) - 1 = 94$

$\Rightarrow \#P_{I_1, I_2} = [\#I_1, \#I_2] - 1 = 2^{91} . 3^{94} - 1$

Problema: sea Q el programa [B_1] $X_1 \leftarrow X_1 - 1$

IF $X_1 \neq 0$ GOTO B_1

$Y \leftarrow Y$

$\Rightarrow \#Q = 2^{91} . 3^{94} . 5^0 - 1 = 2^{91} . 3^{94} - 1 \Rightarrow$ mismo codigo que P NO PUEDEN HABER 2 PROGRAMAS CON MISMO CODIGO

Obs: se agrega la instruccion $Y \leftarrow Y$ para que exista una instruccion de codigo cero

RESTRICCION

Un programa en lenguaje S NO puede terminar con la instruccion $Y \leftarrow Y$, salvo que sea la unica instruccion pues sino la funcion no seria biyectiva.

TEOREMA 6

La funcion recien definida es biyectiva.

DEMOSTRACION

Sobreyectividad:

1. $0 \in Im(\#)$ pues $P: Y \leftarrow Y$ tiene codigo $\#P = 2^0 - 1 = 0$ pues $\#I = \langle 0, \langle 0, 0\rangle\rangle = 0$

2. Sea $n \in \mathbb{N} \Rightarrow n+1 = p_0^{\alpha_0} \dots p_k^{\alpha_k} = [\alpha_0, \dots, \alpha_k]$

⇒ $n+1 \geq 1$ pues $n \in \mathbb{N}$ y por TFA, lo puedo descomponer en primos

Como $\#: \{Instrucciones\} \to \mathbb{N}$ es sobrey ⇒ $\exists\ I_j / \#I_j = \alpha_j,\ 0 \leq j \leq k$ ⇒ $n+1 = [\#I_0, \dots, \#I_k]$ ⇒ $n = [\#I_0, \dots, \#I_k] - 1$

⇒ $\#P_{I_0, \dots, I_k} = n$ i.e. $\exists$ un programa P de instrucciones $(I_0, \dots, I_k)$ c/ codigo n

Inyectividad:

$\#P_{I_0, \dots, I_k} = \#P_{\tilde{I}_0, \dots, \tilde{I}_t} \Rightarrow [\#I_0, \dots, \#I_k] - 1 = [\#\tilde{I}_0, \dots, \#\tilde{I}_t] - 1$

1. Si $k = t$ ⇒ $p_0^{\#I_0} \dots p_k^{\#I_k} = p_0^{\#\tilde{I}_0} \dots p_k^{\#\tilde{I}_k}$ ⇒ por TFA, $\#I_j = \#\tilde{I}_j$, $0 \leq j \leq k = t$

Luego, como $\#: \{Instrucciones\} \to \mathbb{N}$ es iny ⇒ $I_j = \tilde{I}_j$, $0 \leq j \leq k$ ⇒ $P_{I_0, \dots, I_k} = P_{\tilde{I}_0, \dots, \tilde{I}_k}$

Obs: si $\#I_k \neq 0$, $\exists$ un num primo que divide $[\#I_0, \dots, \#I_k]$ y no a $[\#\tilde{I}_0, \dots, \#\tilde{I}_k]$ ⇒ $\#P_{I_0, \dots, I_k} \neq \#P_{\tilde{I}_0, \dots, \tilde{I}_k}$

2. Si $k > t$ ⇒ $\#P_{I_0, \dots, I_k} = \#P_{\tilde{I}_0, \dots, \tilde{I}_t}$ ⇒ $p_0^{\#I_0} \dots p_k^{\#I_k} = p_0^{\#\tilde{I}_0} \dots p_t^{\#\tilde{I}_t}$ ⇒ $p_k^{\#I_k} \mid p_0^{\#\tilde{I}_0} \dots p_t^{\#\tilde{I}_t}$ ⇒ como $k > t$, $\#I_k = 0$

ABS! pues $k \geq 1$ y la ultima instruccion no puede ser $Y \leftarrow Y$ ⇒ debe ser $k = t$

3. Si $k < t$ ⇒ se prueba de manera analoga al caso 2

Hallar el programa de codigo 71.

⇒ $[\#I_1, \dots, \#I_k] - 1 = 71 \Rightarrow [\#I_1, \dots, \#I_k] = 72 = 2^3 \cdot 3^2 \Rightarrow \#I_1 = 3$ y $\#I_k = 2$

Luego, $\#I_1 = 3 = \langle 2, 0\rangle = \langle 2, \langle 0, 0\rangle\rangle$ y $\#I_2 = 2 = \langle 0, 1\rangle = \langle 0, \langle 1, 0\rangle\rangle$.

⇒ $I_1 = [B_1]\ Y \leftarrow Y$ y $I_2 = Y \leftarrow Y+1$ ⇒

```
[B1] Y <- Y
     Y <- Y+1
```

TEOREMA 7

Existen funciones NO computables.

DEMOSTRACION

Notemos que $\#\{Programas\} = \aleph_0$ ($\exists$ fun biyectiva $\#: \{Programas\} \to \mathbb{N}$).

Por otro lado, $\#\{f: \mathbb{N} \to \mathbb{N} / f \text{ funcion}\} = \aleph_0^{\aleph_0} = c$ ⇒ $\exists$ mas funciones que programas para implementarlas pues $c > \aleph_0$.

DEFINICION 5: Problema de la parada

Recordemos que si P es un programa y $u_1, u_2, \dots, u_m$ numeros dados y σ_1 un estado inicial, si hay un computo $d_1 = (1, \sigma_1), d_2, \dots, d_k$, $\Psi_P^{(m)}$ devuelve el valor de y en d_k. Si no hay computo, $\Psi_P^{(m)} = \uparrow$.

La funcion $HALT: \mathbb{N}^2 \to \mathbb{N} / HALT(x,y) = \begin{cases} 1 & \text{si el programa de codigo } y \text{ se detiene ante la entrada } x \\ 0 & \text{sino} \end{cases}$ i.e. $\Psi_P(x) = \uparrow / \#P = y$

EJEMPLOS

EJEMPLO 1 Sea $f: \mathbb{N} \to \mathbb{N} / f(x) = HALT(x, 0) = 1$ pues $\#P = 0 \Leftrightarrow P = Y \leftarrow Y$ ⇒ $f = SUC \circ CERO$

EJEMPLO 2 $HALT(x, 71) = 1$ pues $P: [B_1]\ Y \leftarrow Y$ ⇒ ante toda entrada x, devuelve 1
$\qquad Y \leftarrow Y+1$

EJEMPLO 3 $P: [A_1]$ IF $X_1 \neq 0$ GOTO A_1 ⇒ $HALT(x, \#P) = \begin{cases} 1 & \text{si } x = 0 \\ 0 & \text{si } x \neq 0 \text{ pues } \Psi_P(x) = \uparrow \end{cases}$

TEOREMA 8

La funcion HALT no es computable.

DEMOSTRACION

Sup que HALT es computable ⇒ $f: \mathbb{N} \to \mathbb{N} / f(x) = HALT(x, x)$ es computable pues f es composicion de funciones computables ($f = HALT \circ (\pi_1 \times \pi_1)$) ⇒ $\exists$ un programa que computa f y puedo usar una macro.

Defino el siguiente programa:

$[A_1]$ IF $f(x) \neq 0$ GOTO A_1 ⇒ este programa tiene un codigo asociado ⇒ $\#P = n_0$

Sup que $f(n_0) = 1 \Leftrightarrow HALT(n_0, n_0) = 1 \Leftrightarrow$ el programa de codigo n_0 termina ante la entrada n_0 ⇔ mirando el programa, $f(n_0) = 0$ ABS! pues la unica manera de que el programa termine ante la entrada n_0 es que $f(n_0)$ valga 0 ⇒ HALT no debe ser computable.

COROLARIO

$f: \mathbb{N} \to \mathbb{N} / f(x) = HALT(x, x)$ NO es computable

EJEMPLO

Decidir si f es computable.

1. $f: \mathbb{N}^2 \to \mathbb{N} / f(x,y) = HALT(x,y)^2 + xy$

Obs: $HALT(x,x) + x^2 \geq x^2$ ⇒ $\dot{-}$ eq a $-$. Se puede probar por induccion que $HALT(x,y)^n = HALT(x,y)$

Sup f computable ⇒ defino $g: \mathbb{N} \to \mathbb{N} / g(x) = f(x,x) \dot{-} x^2 = HALT(x,x)^2 + x^2 \dot{-} x^2 = HALT(x,x)^2 + x^2 - x^2$

⇒ $g(x) = HALT(x,x)^2 = HALT(x,x)$ pues $HALT(x,x) = 1$ ⇒ $HALT(x,x)^2 = 1$ y $HALT(x,x) = 0$ ⇒ $HALT(x,x)^2 = 0$

Notemos que g computable pues composicion de funciones computables (f, $\dot{-}$, PROD, π_1)

Obs: $\dot{-}$, PROD y π_1 son RP ⇒ son computables

⇒ HALT es computable ABS!

Conclusion: lo abs vino de sup f computable ⇒ f no es computable

2. $f: \mathbb{N}^2 \to \mathbb{N} / f(x,y) = \langle HALT(x,y), xy \rangle$

Sup f computable ⇒ defino $g: \mathbb{N}^2 \to \mathbb{N} / g(x,y) = \ell(f(x,y)) = HALT(x,y)$ ⇒ g computable por composicion de computables ⇒ HALT computable ABS! ⇒ f no es computable.

3. $f: \mathbb{N}^2 \to \mathbb{N} / f(x,y) = HALT(x,y)^2 \dot{-} HALT(x,y)$

Vimos que $HALT(x,y)^n = HALT(x,y)$, en particular $HALT(x,y)^2 = HALT(x,y)$ ⇒ $f(x,y) = HALT(x,y) \dot{-} HALT(x,y) = 0$.

Conclusion: $f = CERO \circ \pi_1 \Rightarrow f$ es computable por composicion de computables (CERO y π_1 son R.P)

⚠⚠ Una funcion puede ser composicion de NO computables y aun asi ser computable.

TESIS DE CHURCH

Todos los algoritmos para computar funciones $f : A \subseteq \mathbb{N}^k \to \mathbb{N}$ se pueden programar en lenguaje S i.e. si no se puede programar en lenguaje S, en ningun otro lenguaje se puede.

TESIS DE TURING

Cualquier funcion que sea computable se puede implementar en una maquina de Turing i.e. lo que no se puede resolver con una maquina de Turing no se puede con otra maquina.

⇒ Tesis de Turing ≡ Tesis de Church

Obs: todo lo que se puede programar c/ lenguaje S, se puede implementar c/ una maquina de Turing y vice versa

DEFINICION 5: Programas universales

Para cada $n > 0$ se define:

$\Phi : \mathbb{N}^{n+1} \to \mathbb{N} \,/\, \Phi^n(x_1, \dots, x_n, e) = \Psi_P^n(x_1, \dots, x_n)$ siendo $\#P = e$

$$\Rightarrow \Phi^n(x_1, \dots, x_n) = \begin{cases} y & \text{la salida del programa P si ante la entrada } (x_1, \dots, x_n) \text{ el programa se detiene} \\ \uparrow & \text{si el programa P no termina ante la entrada } (x_1, \dots, x_n) \end{cases}$$

TEOREMA 9

La funcion Φ^n es parcialmente computable $\forall n > 0$.

⇒ Ver demostracion en el apunte