CODIFICACIÓN DE PROGRAMAS, FUNCIONES NO COMPUTABLES Y PROGRAMAS UNIVERSALES

1. Codificación de Programas

Tenemos como objetivo asignarle un número natural a cada programa en lenguaje S, es decir definir $\sharp\mathcal{P} \in \mathbb{N}$. Además queremos que se cumplan las siguientes restricciones:

1. Si dos programas $\mathcal{P}$ y $\mathcal{P}'$ son distintos, entonces queremos que $\sharp\mathcal{P} \neq \sharp\mathcal{P}'$.
2. Dado cualquier $n \in \mathbb{N}$, queremos que exista un programa en lenguaje S tal que $\sharp\mathcal{P} = n$.

Es decir, lo que queremos definir es una función biyectiva:

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

Por lo tanto para lograr esta codificación de programas, primero vamos a codificar las instrucciones.

1.1. Codificación de Instrucciones. .

Enumeramos las variables en el siguiente orden:

$$Y,\ X_1,\ Z_1,\ X_2,\ Z_2,\ X_3,\ Z_3,\ \cdots$$

es decir, la variable Y esta en la posición 1 de la lista, la variable X_1 está en la posición 2 de la lista, etc.

Enumeramos las etiquetas en el siguiente orden:

$$A_1,\ B_1,\ C_1,\ D_1,\ E_1,\ A_2,\ B_2,\ C_2,\ \cdots$$

es decir, la etiqueta A_1 esta en la posición 1 de la lista, la etiqueta A_2 está en la posición 6 de la lista, etc.

Si bien se dispone de las siguientes 3 instrucciones en el lenguaje S:

$$V \leftarrow V + 1$$
$$V \leftarrow V - 1$$
$$IF\ V \neq 0\ \ GOTO\ L$$

vamos a agregar una instrucción que no hace nada:

$$V \leftarrow V$$

Definimos

$$\sharp : \{Instrucciones\} \to \mathbb{N}$$

de la siguiente forma:

$$\sharp I = \langle a, \langle b, c \rangle \rangle$$

Veamos qué representan **a**, **b** y **c**:

a: Está asociado a la etiqueta de la instrucción.

$$a = \begin{cases} 0 & si\ I\ no\ tiene\ etiqueta \\ \sharp L & si\ I\ tiene\ adelante\ la\ etiqueta\ L \end{cases}$$

c: Está asociado a la variable que aparece en la instrucción.

$C = \sharp V - 1$, siendo V la variable que aparece en la instrucción I.

b: Está asociado al tipo de instrucción.

$$b = \begin{cases} 0 & si\ I\ es\ V \leftarrow V \\ 1 & si\ I\ es\ V \leftarrow V + 1 \\ 2 & si\ I\ es\ V \leftarrow V - 1 \\ \sharp L + 2 & si\ I\ es\ IF\ V \neq 0\ \ GOTO\ L \end{cases}$$

Proposición 1.1. *La función recién definida que asigna un número natural a cada instrucción es biyectiva.*

Ejercicio 1. .

1. Hallar la instrucción de código 6.
2. Hallar el código de la instrucción:

$$[B_1]\ IF\ Z_3 \neq 0\ \ GOTO\ A_1$$

1.2. Codificación de programas. .

Dado un programa $\mathcal{P}$ que tiene k instrucciones: I_1, I_2, $\cdots$, I_k, definimos

$$\sharp : \{Programas\} \to \mathbb{N}$$

de la siguiente manera:

$$\sharp \mathcal{P} = [(\sharp I_1,\ \cdots,\ \sharp I_k)] - 1$$

Ejercicio 2. Hallar el código del siguiente programa:

$$[B_1]\ X_1 \leftarrow X_1 - 1$$
$$IF\ X_1 \neq 0\ \ GOTO\ B_1$$

Ejercicio 3. Hallar el código del siguiente programa:

$$[B_1]\ X_1 \leftarrow X_1 - 1$$
$$IF\ X_1 \neq 0\ \ GOTO\ B_1$$
$$Y \leftarrow Y$$

Luego de resolver los dos ejercicios anteriores, detectamos que la función recién definida no es inyectiva. Para resolver este problema vamos a agregar una restricción a los programas escritos en lenguaje S:

La última instrucción no puede ser $Y \leftarrow Y$, a menos que sea la única instrucción.

Proposición 1.2. *La función recién definida que asigna un número natural a cada programa es biyectiva.*

Ejercicio 4. Hallar el programa de código 71.

2. Funciones no computables

Tesis 2.1 (de Church)**.** *Todos los algortimos para computar funciones $f : A \subset \mathbb{N}^k \to \mathbb{N}$ se pueden programar en lenguaje S.*

Teorema 2.1. *Existen funciones no computables.*

Teorema 2.2. *La función $Halt : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ definida de la siguiente manera:*

$$Halt(x, y) = \begin{cases} 1 & \text{si el programa de código } y \text{ ante la entrada } x \text{ se detiene} \\ 0 & \text{sino} \end{cases}$$

no es computable.

3. Programas universales

Definimos para cada $n \in \mathbb{N}_{>0}$ la siguiente función:

$$\phi^n(x_1, \cdots, x_n, e) = \psi^n_{\mathcal{P}}(x_1, \cdots, x_n)$$

siendo $\sharp\mathcal{P} = e$.
Notemos que esta función solo está definida si el programa $\mathcal{P}$ termina ante la entrada $(x_1, \cdots, x_n)$ y en ese caso devuelve la salida del programa.

Teorema 3.1. *La función ϕ^n es parcialmente computable para cada $n \in \mathbb{N}_{>0}$.*

Demostración: El siguiente programa computa la función universal Φ^n,
tal que $\Phi(x_1, \cdots, x_n, e) = \psi^n_P(x_1, \cdots, x_n)$ siendo $P = \#e$:

1) $Z \leftarrow X_{n+1} + 1$
2) $S = \prod_{i=1}^{n} p_{2i}^{X_i}$
3) $K \leftarrow 1$
4) $[C]$ IF $K =\mid Z \mid +1 \vee K = 0$ GOTO F
5) $U \leftarrow r(Z[k])$
6) $P \leftarrow p_{r(U)+1}$
7) IF $l(U) = 0$ GOTO N
8) IF $l(U) = 1$ GOTO S
9) IF $\neg(DIV(P, S))$ GOTO N
10) IF $l(U) = 2$ GOTO R
11) $K \leftarrow \text{mín}_{i \leq |Z|}(l(Z[i]) + 2 = l(U))$
12) GOTO C
13) $[R]$ $S \leftarrow$ coc(P,S)
14) GOTO N
15) $[S]$ $S \leftarrow S \cdot P$
16) GOTO N
17) $[N]$ $K \leftarrow K + 1$
18) GOTO C
19) $[F]$ $Y \leftarrow S[1]$

Analicemos un poco el programa...

En la primer instrucción se guarda en la variable $Z = [\#I_1, \cdots, \#I_n]$, dado que $X_{n+1} = e = [\#I_1, \cdots, \#I_n] - 1$.

En el segundo renglón se guarda en la variable S el estado inicial del programa de código e, de la siguiente manera: $S = [0, X_1, 0, X_2, 0, \cdots, 0, X_n]$

En el tercer renglón se guarda en la variable K la instrucción a la que se apunta del programa de código e.

En el renglón 4, si la instrucción a la que se apunta en el programa de código e es la 0 (que no existe) o la siguiente a la última (que no existe) se direcciona a la etiqueda F, sino quiere decir que apunta a una instrucción correcta del programa de código e que hay que decodificar para ejecutarla y para ello se va al siguiente renglón.

En el renglón 5: Recordemos que en $Z[K]$ está el código de la $K-$ésima instrucción del programa de código e, es decir $Z[K] =< a, < b, c >>$, entonces se guarda en la variable $U =< b, c >$.

En el renglón 6: La variable que aparece en la $K-$ésima instrucción del programa de código e es la (c+1)-ésima, entonces se guarda en P el $(c+1)$-ésimo primo, notemos que $c = r(U)$, ya que $U =< b, c >$.

En el renglón 7: Si $b = 0$, es una instrucción del tipo $V \leftarrow V$ y direcciona a la etiqueta N.

En el renglón 8: Si $b = 1$, es una instrucción del tipo $V \leftarrow V + 1$ y direcciona a la etiqueta S.

En el renglón 9: Si P no divide a S, es decir si la variable V tiene el valor 0, va a la etiqueta N,

En el renglón 10: Si $b = 2$, es una instrucción del tipo $V \leftarrow V - 1$, entonces direcciona a la etiqueta R.

En el renglón 11: Si se llegó hasta acá es porque $b \geq 2$, es decir es una instrucción del tipo IF. Por lo tanto el estado no cambia (S), y hay que determinar la próxima instrucción a ejecutar (a decodificar) del programa de código e y guardar su número en la variable K. Como en$l(Z[j])$ está guaradado el número a vinculado a la instrucción j, busca la primer instrucción cuya etiqueta tengo el número tal que si le sumo 2 es b.

En el renglón 13: Ejecuta la instrucción $V \leftarrow V - 1$ es decir divide S por P.

En el renglón 15: Ejecuta la instrucción $V \leftarrow V + 1$ es decir multiplica S por P.

□

Teorema 3.2. *Existen funciones que son computables y no son recursivas primitivas.*