Modelos de Computación y Complejidad

Grado en Ingeniería Informática. Tecnologías Informáticas

Tema 3: Programas Universales

David Orellana Martín
Mario de J. Pérez Jiménez

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. Ingeniería Informática
Universidad de Sevilla

marper@us.es

http://www.cs.us.es/~marper/

Curso 2021-2022

Objetivos del tema

- Obtener **información** acerca del **comportamiento** de los programas `GOTO`.
  - ⋆ **Codificación** de los programas `GOTO` a través de números naturales.
- Existencia de **programas universales** $\mathbf{U_n}$ (para cada $n \geq 1$):
  - ⋆ Funcionamiento de un programa universal: **ordenador convencional**.

La abstracción

Una característica esencial de la **abstracción** es la siguiente:

* **Un mismo objeto** puede representar entidades **diferentes**.

Ejemplo: Un número natural **n** puede representar:

* **Un conjunto** que consta, exactamente, de n elementos. Más aún, si $n \neq 0$ entonces $n = \{0, \ldots, n-1\}$.
* **Una tupla ordenada** de números naturales cuyo número de Gödel es **n**. Específicamente, si **n** ≥ 2 entonces **n** representa a **la tupla** $(a_1, \ldots a_k)$ tal que $a_k \neq 0$ y **n**$= [a_1, \ldots a_k]$.
* **Una instrucción** `GOTO` (lo justificaremos en este tema).
* **Un programa** `GOTO` (lo justificaremos en este tema).

Cada una de esas "representaciones" han de reportar algunas ventajas.

Codificaciones en el modelo GOTO

Vamos a representar/codificar los **actores principales** del modelo GOTO mediante **números naturales**.

Actores principales (por orden de aparición):

* **Variables**.
* **Etiquetas**.
* **Formatos**.
* **Instrucciones básicas GOTO**.
* **Programas GOTO**.
* **Estados**.
* **Configuraciones**.

Codificaciones en el modelo GOTO

Codificación de **programas** GOTO a través de números naturales (**función de Gödel**).

- ⋆ **Todo número natural** codificará **un único programa** GOTO.

Codificación de las **instrucciones** GOTO (**función par**).

- ⋆ **Todo número natural** codificará **un única instrucción** GOTO.

Codificación de:

- Las **variables**.
- Las **etiquetas**.
- Los **formatos** de las instrucciones (skip, incremento, decremento y condicional).

Codificación de variables, etiquetas y formatos

(a) Variables:

$\mathbf{Y}$	$\mathbf{X}_1$	$\mathbf{Z}_1$	$\mathbf{X}_2$	$\mathbf{Z}_2$	$\mathbf{X}_3$	$\mathbf{Z}_3$	$\mathbf{X}_4$	$\mathbf{Z}_4$	...
1	2	3	4	5	6	7	8	9	...

Los códigos de las variables son: $\#(\mathbf{Y}) = 1$ y para cada $k \geq 1$, $\#(\mathbf{X}_k) = 2k$ y $\#(\mathbf{Z}_k) = 2k + 1$.

(b) Etiquetas:

A_1	B_1	C_1	D_1	E_1	A_2	B_2	C_2	D_2	E_2	...
1	2	3	4	5	6	7	8	9	10	...

Para cada $k \geq 1$:

$$\begin{cases} \#(A_k) = 5(k-1) + 1 \\ \#(B_k) = 5(k-1) + 2 \\ \#(C_k) = 5(k-1) + 3 \\ \#(D_k) = 5(k-1) + 4 \\ \#(E_k) = 5k \end{cases}$$

(c) Formatos:

$$\begin{cases} 0 & \textit{si el formato es } V \longleftarrow V \\ 1 & \textit{si el formato es } V \longleftarrow V + 1 \\ 2 & \textit{si el formato es } V \longleftarrow V - 1 \\ \#(L) + 2 & \textit{si el formato es IF } V \neq 0 \textit{ GOTO } L \end{cases}$$

Codificación de las instrucciones GOTO

Si **I** es una instrucción GOTO, entonces se define el código de **I** así:

$$\boxed{\#(\mathbf{I}) = \langle \mathbf{a}, \langle \mathbf{b}, \mathbf{c} \rangle \rangle}$$

en donde:

$$\mathbf{a} = \begin{cases} 0 & \text{si } I \text{ no está etiquetada a la izquierda} \\ \#(L) & \text{si } I \text{ está etiquetada a la izquierda por } L \end{cases}$$

$$\mathbf{b} = \textit{Código del formato de } I$$
$$\begin{cases} 0 & \textit{si formato } V \longleftarrow V \\ 1 & \textit{si formato } V \longleftarrow V+1 \\ 2 & \textit{si formato } V \longleftarrow V-1 \\ \#(L')+2 & \textit{si formato IF } V \neq 0 \textit{ GOTO } L' \end{cases}$$

$$\mathbf{c} = \#(V) - 1 \qquad \textit{si } V \textit{ es la variable de } I.$$

Codificación de las instrucciones GOTO

Proposición 1: Todo número natural $n \in \mathbb{N}$ codifica una única instrucción GOTO.

Demostración: Sea $n \in \mathbb{N}$ un número natural. Entonces, existen unos únicos números naturales a, z tales que $n = \langle a, z \rangle$.

Ahora bien, dado $z \in \mathbb{N}$ existen unos únicos números naturales b, c tales que $z = \langle b, c \rangle$. Luego, $n = \langle a, \langle b, c \rangle \rangle$.

Entonces, existe una única instrucción GOTO, $\mathbf{I_j}$, tal que $\#(\mathbf{I_j}) = \langle \mathbf{a}, \langle \mathbf{b}, \mathbf{c} \rangle \rangle = \mathbf{n}$. ■

Ejercicio 1: Hallar el código de la instrucción $\mathbf{Y \longleftarrow Y}$.

Solución: Vamos a hallar $a, b, c \in \mathbb{N}$ tales que $\#(\, Y \longleftarrow Y\,) = \langle a, \langle b, c \rangle \rangle$.

La instrucción $Y \longleftarrow Y$ verifica lo siguiente:

* **No está etiquetada** a la izquierda. Luego, $\boxed{\mathbf{a = 0}}$.
* Es una instrucción **SKIP**. Luego, $\boxed{\mathbf{b = 0}}$.
* Su variable es Y. Luego, $c = \#(Y) - 1 = 1 - 1$; es decir, $\boxed{\mathbf{c = 0}}$.

Entonces,

$\langle a, \langle b, c \rangle \rangle = \langle 0, \langle 0, 0 \rangle \rangle = \langle 0, 2^0 \cdot (2 \cdot 0 + 1) - 1 \rangle = \langle 0, 0 \rangle = 2^0 \cdot (2 \cdot 0 + 1) - 1 = 0.$

Conclusión: $\boxed{\#(\mathbf{Y \longleftarrow Y}) = \mathbf{0}}$.

Ejercicio 2: Hallar el código de la instrucción **[A] Y $\longleftarrow$ Y + 1**.

Solución: Vamos a hallar $a, b, c \in \mathbb{N}$ tales que $\#(\,[A]\ Y \longleftarrow Y + 1\,) = \langle a, \langle b, c\rangle\rangle$.

La instrucción $[A]\ Y \longleftarrow Y + 1$ verifica lo siguiente:

* Está etiquetada a la izquierda por A. Luego, $a = \#(A)$; es decir, $\boxed{\mathbf{a = 1}}$.
* Es una instrucción **incremento**. Luego, $\boxed{\mathbf{b = 1}}$.
* Su variable es Y. Luego, $c = \#(Y) - 1 = 1 - 1$; es decir, $\boxed{\mathbf{c = 0}}$.

Entonces,

$$\langle a, \langle b, c\rangle\rangle = \langle 1, \langle 1, 0\rangle\rangle = \langle 1, 2^1 \cdot (2 \cdot 0 + 1) - 1\rangle = \langle 1, 1\rangle = 2^1 \cdot (2 \cdot 1 + 1) - 1 = 5.$$

Conclusión: $\boxed{\#([\mathbf{A}]\ \mathbf{Y} \longleftarrow \mathbf{Y} + \mathbf{1}) = \mathbf{5}}$.

Ejercicio 3: Hallar **la** instrucción $\mathbf{I_j}$ cuyo código es **14**.

Solución: Vamos a hallar los números a, b, c tales que $\langle \mathbf{a}, \langle \mathbf{b}, \mathbf{c} \rangle \rangle = \mathbf{14}$.

Si $z = \langle b, c \rangle$ entonces $14 = \langle a, z \rangle = 2^a \cdot (2 \cdot z + 1) - 1$. Luego,

$15 = 2^a \cdot (2 \cdot z + 1) \Rightarrow (a = 0) \wedge (15 = 2 \cdot z + 1) \Rightarrow \boxed{\mathbf{a=0}} \wedge z = 7.$

Ahora bien:

$7 = \langle b, c \rangle = 2^b \cdot (2 \cdot c + 1) - 1 \Rightarrow 8 = 2^b \cdot (2 \cdot c + 1) \Rightarrow \boxed{\mathbf{b=3}} \wedge \boxed{\mathbf{c=0}}.$

Por tanto, se tiene que $\#(\mathbf{I_j}) = \langle 0, \langle 3, 0 \rangle \rangle$

- Al ser $\mathbf{a = 0}$ resulta que $\mathbf{I_j}$ **no está etiquetada** a la izquierda.
- Al ser $\mathbf{b = 3}$ resulta que $\mathbf{I_j}$ es una instrucción condicional cuya etiqueta referenciada, L, verifica $\#(L) + 2 = 3$; es decir, $\#(L) = 1$ y, por tanto, $L = A_1$.
- Al ser $\mathbf{c = 0}$ resulta que la variable V de $\mathbf{I_j}$ verifica $\#(V) - 1 = 0$; es decir, $\#(V) = 1$ y, por tanto, $V = Y$.

Conclusión: $\mathbf{I_j} \equiv \boxed{\mathbf{IF\ \ Y \neq 0\ \ GOTO\ A_1}}$

Codificación de los programas GOTO

Sea $P = (I_1, I_2, \ldots, I_n)$ un programa GOTO. El código de P se define así:

$$\boxed{\#(\mathbf{P}) = \left[\#(\mathbf{I_1}), \#(\mathbf{I_2}), \ldots, \#(\mathbf{I_n})\right] - \mathbf{1}}$$

Nota: Si $P_\emptyset$ es el programa vacío, entonces $\#(P_\emptyset) = \left[\varepsilon\right] - 1 = 1 - 1 = 0$.

Observaciones:

- ⋆ Puesto que $\#(Y \longleftarrow Y) = 0$ y $[a_1, ..., a_n] = [a_1, ..., a_n, 0]$, se tiene:

$$\left[\#(I_1), ..., \#(I_n)\right] = \left[\#(I_1), ..., \#(I_n), \#(Y \longleftarrow Y)\right]$$

Por tanto, si la última instrucción de un programa GOTO pudiera ser $Y \longleftarrow Y$, la codificación de programas **no sería inyectiva**.

Es decir, existirían **programas GOTO distintos** que tendrían el **mismo código**.

Codificación de los programas GOTO

Proposición 2: Todo número natural $n \in \mathbb{N}$ codifica un único programa GOTO.

Demostración: Sea $n \in \mathbb{N}$ un número natural.

* Si $\mathbf{n = 0}$ entonces el **programa vacío** es el único programa GOTO que está codificado por n.

* Si $\mathbf{n \geq 1}$ entonces $n + 1 \geq 2$ y, por tanto, existirán unos únicos números naturales $a_1, \ldots a_r$ tales que $a_r > 0$ y $n + 1 = p_1^{a_1} \cdot \ldots \cdot p_r^{a_r}$.

  Para cada $j, 1 \leq j \leq r$, sea $\mathbf{I_j}$ la instrucción GOTO codificada por a_j y consideremos el programa GOTO, $\mathbf{P = (I_1, \ldots, I_r)}$.

  Veamos que P es **el programa GOTO** codificado por n. En efecto:

$$\boxed{\#(P)=}\left[\#(I_1), \ldots, \#(I_r)\right] - 1 = p_1^{a_1} \cdot \ldots \cdot p_r^{a_r} - 1 = (n+1) - 1 = \boxed{n}.$$

■

Ejercicio 4: Hallar el código del siguiente programa GOTO:

$$P \equiv \boxed{\begin{array}{ll} [A] & Y \longleftarrow Y + 1 \\ & IF\ Y \neq 0\ GOTO\ A \end{array}}$$

Solución: Si $I_1 \equiv [A]\ Y \longleftarrow Y + 1$ e $I_2 \equiv IF\ Y \neq 0\ GOTO\ A$, entonces $P = (I_1, I_2)$.

Ahora bien,

* Del ejercicio 2 resulta que $\#(I_1) = \#([A]\ Y \longleftarrow Y + 1) = 5$.
* Del ejercicio 3 resulta que $\#(I_2) = \#(IF\ Y \neq 0\ GOTO\ A) = 14$.

Por tanto,

$$\boxed{\#(P)=}\left[\#(I_1), \#(I_2)\right] - 1 = \left[5, 14\right] - 1 = 2^5 \cdot 3^{14} - 1 = \boxed{153.055.007}.$$

Ejercicio 5: Hallar el programa GOTO cuyo código es 249.999.

Solución: Vamos a hallar el programa GOTO, $P = (I_1, \ldots, I_r)$, tal que $\#(P) = 249.999$.

Se ha de verificar lo siguiente: $249.999 = \#(P) = \big[\#(I_1), \ldots, \#(I_r)\big] - 1$,

Luego, $\big[\#(I_1), \ldots, \#(I_r)\big] = 250.000 = 2^4 \cdot 3^0 \cdot 5^6 = \big[4, 0, 6\big]$.

Por tanto, se tiene que $r = 3, \#(I_1) = 4, \#(I_2) = 0, \#(I_3) = 6$.

Hallemos las instrucciones I_1, I_2, I_3 cuyos códigos son los anteriomente mencionados.

* Para hallar I_1, calculemos a, b, c tales que $\langle a\langle b, c\rangle\rangle = \#(I_1) = 4$.
  Si notamos $z = \langle b, c\rangle$ entonces
  $4 = \langle a, z\rangle \Rightarrow 4 = 2^a \cdot (2 \cdot z + 1) - 1 \Rightarrow 5 = 2^a \cdot (2 \cdot z + 1) \Rightarrow (\boxed{\mathbf{a{=}0}}) \wedge (z = 2)$
  $2 = z = \langle b, c\rangle \Rightarrow 2 = 2^b \cdot (2 \cdot c + 1) - 1 \Rightarrow 3 = 2^b \cdot (2 \cdot c + 1) \Rightarrow (\boxed{\mathbf{b{=}0}}) \wedge (\boxed{\mathbf{c{=}1}})$
  Al ser $a = 0$, la instrucción I_1 **no** está etiquetada a la izquierda.
  Al ser $b = 0$, la instrucción I_1 es un **SKIP**.
  Al ser $c = 1$, la variable V de I_1 verifica $\#(V) - 1 = 1$; es decir $\#(V) = 2$ y, por tanto, la variable V es **X**.
  Por tanto, $\boxed{\mathbf{I_1 \equiv X \longleftarrow X}}$.

* La instrucción I_2, tiene código 0. Del ejercicio 1 resulta que

  $\boxed{\mathbf{I_2} \equiv \mathbf{Y} \leftarrow\!\!- \mathbf{Y}}$.

* Para hallar I_3, calculemos a, b, c tales que $\langle a\langle b, c\rangle\rangle = \#(I_3) = 6$.

  Si notamos $z = \langle b, c\rangle$ entonces
  $6 = \langle a, z\rangle \Rightarrow 6 = 2^a \cdot (2 \cdot z + 1) - 1 \Rightarrow 7 = 2^a \cdot (2 \cdot z + 1) \Rightarrow (\boxed{\mathbf{a{=}0}}) \wedge (z = 3)$

  $3 = z = \langle b, c\rangle \Rightarrow 3 = 2^b \cdot (2 \cdot c + 1) - 1 \Rightarrow 4 = 2^b \cdot (2 \cdot c + 1) \Rightarrow$
  $(\boxed{\mathbf{b{=}2}}) \wedge (\boxed{\mathbf{c{=}0}})$

  Al ser $a = 0$, la instrucción I_3 **no** está etiquetada a la izquierda.
  Al ser $b = 2$, la instrucción I_3 es un **DECREMENTO**.
  Al ser $c = 0$, la variable V de I_3 verifica $0 = \#(V) - 1$; es decir $\#(V) = 1$ y, por tanto, la variable V es **Y**.

  Por tanto, $\boxed{\mathbf{I_3} \equiv \mathbf{Y} \leftarrow\!\!- \mathbf{Y - 1}}$.

Conclusión: el programa `GOTO` de código 249.999 es:

$$P \equiv \boxed{\begin{array}{c} X \leftarrow\!\!- X \\ Y \leftarrow\!\!- Y \\ Y \leftarrow\!\!- Y - 1 \end{array}}$$

Codificación de estados de programas GOTO

Sea $\mathbf{s} = \{\mathbf{V_1} = \mathbf{r_1}, \ldots, \mathbf{V_n} = \mathbf{r_n}\}$ un **estado** tal que $\#(\mathbf{V_i}) = \mathbf{i}$, para cada $i \geq 1$.
El **código** de s se define como sigue:

$$\boxed{\#(\mathbf{s}) = \left[\mathbf{r_1}, \ldots, \mathbf{r_n}\right]}$$

En el número de Gödel que representa el código de un estado, los **valores** de las **variables** han de aparecer **ordenados según los códigos** de las mismas.

Teniendo presente que todo número de Gödel es un número natural mayor o igual que 1, resulta que **0 no** es el **código** de un estado.

Ejercicio 6: Hallar el código del estado $s = \{X_1 = 4, X_2 = 1\}$.

Solución: En primer lugar, se reescribe el estado a fin de ordenar las variables según sus códigos (recuérdese el orden de las variables: $Y, X_1, Z_1, X_2, Z_2, \ldots$).

Así pues, reescribimos el estado: $s = \{Y = \mathbf{0}, X_1 = \mathbf{4}, Z_1 = \mathbf{0}, X_2 = \mathbf{1}\}$.

Por tanto, $\boxed{\#(\mathbf{s}) =} \left[\mathbf{0}, \mathbf{4}, \mathbf{0}, \mathbf{1}\right] = 2^{\mathbf{0}} \cdot 3^{\mathbf{4}} \cdot 5^{\mathbf{0}} \cdot 7^{\mathbf{1}} = \boxed{\mathbf{567}}$

Codificación de estados de programas GOTO

Ejercicio 7: Hallar el código del **estado de entrada** $s = \{X_1 = a_1, \ldots, X_n = a_n\}$, siendo $a_1, \ldots, a_n$ números naturales arbitrarios.

Solución: En primer lugar, se reescribe el estado a fin de ordenar las variables según sus códigos (recuérdese el orden: $Y, X_1, Z_1, X_2, Z_2, \ldots$).

Así pues, reescribimos el estado:

$$s = \{Y = \mathbf{0}, X_1 = \mathbf{a_1}, Z_1 = \mathbf{0}, \ldots, X_n = \mathbf{a_n}, Z_n = \mathbf{0}\}$$

Por tanto,

$$\#(\mathbf{s}) = \left[\mathbf{0}, \mathbf{a_1}, \mathbf{0}, \ldots, \mathbf{a_n}, \mathbf{0}\right] = p_1^{\mathbf{0}} \cdot p_{2\cdot 1}^{\mathbf{a_1}} \cdot \ldots \cdot p_{2\cdot n}^{\mathbf{a_n}} \cdot p_{2\cdot n+1}^{\mathbf{0}} = p_{2\cdot 1}^{\mathbf{a_1}} \cdot \ldots \cdot p_{2\cdot n}^{\mathbf{a_n}}$$

Es decir:

$$\boxed{\#(\mathbf{s}) = \prod_{i=1}^{n} p_{2\cdot i}^{\mathbf{a_i}}}$$

Codificación de estados de programas GOTO

Proposición 3: Todo número natural $n \geq 1$ codifica un único estado.

Demostración: Sea $n \in \mathbb{N}$ un número natural.

* Si $\mathbf{n = 1}$ entonces $s = \{Y = 0, X_1 = 0, Z_1 = 0, X_2 = 0, Z_2 = 0, \ldots\}$ es un estado que tiene código 1 y, además, es el único que verifica la condición citada.

* Si $\mathbf{n \geq 2}$ entonces existirán unos únicos números naturales $a_1, \ldots a_r$ tales que $a_r > 0$ y $n = p_1^{a_1} \cdot \ldots \cdot p_r^{a_r}$.
  En tal situación, $s = \{Y = a_1, X_1 = a_2, Z_1 = a_3, \ldots, V_r = a_r\}$ será el único estado cuyo código es $p_1^{a_1} \cdot \ldots \cdot p_r^{a_r} = n$.

■

Codificación de configuraciones de programas GOTO

Sea $\sigma = (\mathbf{j}, \mathbf{s})$ una configuración de un programa GOTO. El código de σ se define como sigue:

$$\boxed{\#(\sigma) = \langle\, \mathbf{j},\ \#(\mathbf{s})\, \rangle}$$

Ejercicio 8: Hallar el código de la configuración inicial $\sigma = (1, s)$, siendo s el estado $\{X_1 = 4, X_2 = 1\}$.

Solución: Se tiene que:

$$\#(\sigma) = \langle\, 1,\ \#(s)\, \rangle \overset{Ej.\,6}{=} \langle\, 1,\ 567\, \rangle = 2^1 \cdot (2 \cdot 567 + 1) - 1$$

Por tanto, $\boxed{\#(\sigma) =}\ 2 \cdot 1.135 - 1 = \boxed{\mathbf{2.269}}$

Codificación de configuraciones de programas GOTO

Proposición 4: El **código** de **cualquier configuración** **siempre** es un número natural **IMPAR** **mayor o igual que 5**.

Demostración: Sea $\sigma = (j, s)$ una configuración de un programa GOTO P. Entonces se tiene que $1 \leq j \leq 1 + |P|$ y s es un estado de P.

Ahora bien, $\#(\sigma) = \langle\, \mathbf{j}, \#(\mathbf{s}) \,\rangle = 2^j \cdot (2 \cdot \#(s) + 1) - 1$.

Puesto que $j \geq 1$ y $\#(s) \geq 1$ resulta lo siguiente:

- 2^j es un número natural par mayor o igual que 2.
- $2 \cdot \#(s) + 1$ es un número natural impar mayor o igual que 3.
- $2^j \cdot (2 \cdot \#(s) + 1)$ es un número natural par mayor o igual que 6.

En consecuencia, $\#(\sigma) = \underbrace{2^j \cdot (2 \cdot \#(s) + 1)}\ -1$ será un **número** natural **IMPAR** y **mayor o igual que 5**.

Programas universales

Cuestión:

Para cada $n \geq 1$, se considera la función $\mathcal{U}_n$ de aridad **n+1** que asigna a cada tupla $(a_1, \ldots, a_n, a_{n+1})$ el **resultado de la computación** $Q(a_1, \ldots, a_n)$, siendo Q **el programa** GOTO de código a_{n+1}.

¿**ES COMPUTABLE LA FUNCIÓN** $\mathcal{U}_n$?

Responderemos **afirmativamente** a esta cuestión.

- Un programa $\mathbf{U_n}$ que calcule $\mathcal{U}_n$ (de aridad **n+1**) se denomina **programa universal de orden** n.
- Un programa universal $\mathbf{U_n}$ de orden n **simula el comportamiento** de cualquier programa GOTO ejecutado sobre cualquier n-tupla.
- Es decir, para cualquier programa GOTO, Q, se verifica:

$$\mathbf{U_n}(\mathbf{a_1}, \ldots, \mathbf{a_n}, \#(\mathbf{Q})) = Q(a_1, \ldots, a_n) = \mathcal{U}_n(\mathbf{a_1}, \ldots, \mathbf{a_n}, \#(\mathbf{Q}))$$

“Aspecto” de un programa universal U_n

Programas camaleónicos

Programas universales

Teorema. Para cada $n \geq 1$, **existe** un programa GOTO, $\mathbf{U_n}$, tal que para **cualquier** programa GOTO, **Q**, y para toda n-tupla $(a_1, \ldots, a_n) \in \mathbb{N}^n$ se verifica:

$$\mathbf{U_n}\,(\mathbf{a_1}, \ldots, \mathbf{a_n}, \#(\mathbf{Q})) = \mathbf{Q}\,(\mathbf{a_1}, \ldots, \mathbf{a_n})$$

Demostración: amos a diseñar **un** programa universal $\mathbf{U_n}$ que satisfaga la condición antes citada.

- ⋆ Básicamente, un programa universal va a ser una especie de **intérprete de** GOTO.

El programa universal U_n

$$\left.\begin{array}{l} Z \longleftarrow X_{n+1} + 1 \\ S \longleftarrow \prod_{i=1}^{n} p_{2i}^{X_i} \\ K \longleftarrow 1 \end{array}\right\} (1)$$

inicialización

$$[C] \quad IF\ K = Long(Z) + 1\ GOTO\ F\ \Big\} (2)$$

test de parada

$$\left.\begin{array}{l} U \longleftarrow r((Z)_k) \\ P \longleftarrow p_{r(U)+1} \\ IF\ l(U) = 0\ GOTO\ N \\ IF\ l(U) = 1\ GOTO\ A \\ IF\ P \nmid S\ GOTO\ N \\ IF\ l(U) = 2\ GOTO\ M \\ K \longleftarrow \mu j \leq Long(Z)[l((Z)_j) + 2 = l(U)] \\ GOTO\ C \end{array}\right\} (3)$$

paso de computación de X_{n+1}

$$\left.\begin{array}{ll} [M] & S \longleftarrow qt(S, P) \\ & GOTO\ N \\ [A] & S \longleftarrow S \cdot P \\ [N] & K \longleftarrow K + 1 \\ & GOTO\ C \\ [F] & Y \longleftarrow (S)_1 \end{array}\right\} (4)$$

salida

Descripción del programa universal U_n

* **BLOQUE 1**: **Inicialización** del programa U_n que va a **simular la computación** $Q(X_1, \ldots, X_n)$, para cada programa `GOTO`, Q.

$$\begin{array}{l} Z \longleftarrow X_{n+1} + 1 \\ S \longleftarrow \prod_{i=1}^{n} p_{2i}^{X_i} \\ K \longleftarrow 1 \end{array}$$

- En Z se almacena información acerca del programa $Q = (I_1, \ldots, I_r)$ tal que $X_{n+1} = \#(Q) = \left[\#(I_1), \ldots, \#(I_r)\right] - 1$.
  * Obsérvese que la componente $(Z)_j = (X_{n+1} + 1)_j$ representa el **código** de la instrucción I_j de Q.
- En S se almacena el **código** del **estado actual** de la computación $Q(X_1, \ldots, X_n)$. Al inicio, $S = p_{2\cdot 1}^{X_1} \ldots p_{2\cdot n}^{X_n} = \left[0, X_1, 0, X_2, \ldots, 0, X_n\right]$ representa el estado inicial (dato de entrada) de esa computación.
- En K se almacena el índice de la "siguiente" instrucción a ejecutar en la computación $Q(X_1, \ldots, X_n)$. En el caso $K = Long(Z) + 1$, esa computación que se "simula" deberá **parar**.

(Si $K = j$ y $S = s$, el par (j, s) codifica una configuración de Q)

Descripción del programa universal U_n

* **BLOQUE 2**: **Test de parada**.

$$[C] \quad IF\ K = Long(Z) + 1\ GOTO\ F$$

K almacena el índice de la "siguiente" instrucción a ejecutar en la computación $Q(X_1, \ldots, X_n)$.

A lo largo de la ejecución, el valor de K se va aumentando en una unidad, o bien se obtiene por minimización acotada de un predicado.

El valor $K = Long(Z) + 1$ se alcanza cuando la computación "simulada" $Q(X_1, \ldots, X_n)$ es de parada:

  * Puede suceder que se haya ejecutado un condicional que referencia a una etiqueta que no aparece a la izquierda etiquetando a ninguna instrucción.

Descripción del programa universal U_n

* **BLOQUE 3**: **Simulación** de **un paso** de **la computación** $Q(X_1, \ldots, X_n)$. **Selección** de la **instrucción** I_K de Q.

$$
\begin{array}{l}
U \longleftarrow r((Z)_k) \\
P \longleftarrow p_{r(U)+1} \\
IF\ I(U) = 0\ GOTO\ N \\
IF\ I(U) = 1\ GOTO\ A \\
IF\ P \nmid S\ GOTO\ N \\
IF\ I(U) = 2\ GOTO\ M \\
K \longleftarrow \mu j \leq Long(Z)[I((Z)_j) + 2 = I(U)] \\
GOTO\ C
\end{array}
$$

- ⋆ Al ejecutar la instrucción $U \longleftarrow r((Z)_k))$, se tiene $1 \leq K \leq r$: I_K es una instrucción de Q.
- ⋆ $(Z)_K = \#(I_K) = \langle a, \langle b, c \rangle \rangle$ y $U = r(Z)_K = \langle b, c \rangle$ codifica el formato y la variable de I_K.
- ⋆ $r(U) + 1 = c + 1$ es el código de la **variable V** de I_K, y **I(U)** $= b$ es el **formato** de I_K.
- ⋆ $P = p_{r(U)+1}$ es el número primo cuyo exponente representa **el valor** de la **variable V** de I_K.
- ⋆ $P \nmid S$ significa que **el valor de la variable** de I_K en el estado S **es 0**.
- ⋆ Si $I(U) = 0$, la instrucción I_K es un **skip**. Si $I(U) = 1$, la instrucción I_K es un **incremento**.
- ⋆ Al ejecutar $IF\ I(U) = 2\ GOTO\ M$ se tiene que I_K es un **decremento** y $\mathbf{V} \neq 0$.
- ⋆ Al ejecutar $K \longleftarrow \mu j \leq Long(Z)[I((Z)_j) + 2 = I(U)]$ se tiene que I_K es un **condicional** y $\mathbf{V} \neq 0$.

Descripción del programa universal U_n

* **BLOQUE 4**: **Simulación** de **un paso** de **la computación** $Q(X_1, \ldots, X_n)$. **Ejecución** de la **instrucción** I_K de Q.

$$
\begin{array}{ll}
 & K \longleftarrow \mu j \leq Long(Z)[I((Z)_j) + 2 = I(U)] \\
 & GOTO\ C \\
[M] & S \longleftarrow qt(S, P) \\
 & GOTO\ N \\
[A] & S \longleftarrow S \cdot P \\
[N] & K \longleftarrow K + 1 \\
 & GOTO\ C
\end{array}
$$

- ★ $K \longleftarrow \mu j \leq Long(Z)[I((Z)_j) + 2 = I(U)]$ se ejecuta cuando I_K es un **condicional** y $\mathbf{V} \neq 0$.
- ★ Al ejecutar $GOTO\ C$ se vuelve al test de parada.
- ★ $[M]\ S \longleftarrow qt(S, P)$ se ejecuta cuando I_K es un **decremento con V** $\neq 0$: disminuir en 1 ese valor equivale a dividir el estado S por el primo P cuyo exponente codifica el valor de **V**. A continuación, se ejecuta $GOTO\ N$, aumentando K en una unidad y volviendo al test de parada.
- ★ $[A]\ S \longleftarrow S \cdot P$ se ejecuta cuando I_K es un **incremento**: aumentar en 1 el valor de **V** equivale a multiplicar el estado S por el primo cuyo exponente codifica el valor de **V**. Seguidamente, se aumenta K en una unidad y se vuelve al test de parada.
- ★ $[N]\ K \longleftarrow K + 1$ se ejecuta cuando I_K es un **skip, incremento, decremento** ó un **condicional** cuya variable vale **0**.

Descripción del programa universal U_n

* **BLOQUE 5**: **Fase de salida**.

$$[F] \quad Y \longleftarrow (S)_1$$

  - $[F]\ Y \longleftarrow (S)_1$ se ejecuta si se verifica la condición $K = Long(Z) + 1$ del test de parada; es decir, cuando la ejecución del programa ha de finalizar.
  - En ese caso, ha de devolver como resultado el valor de la variable de salida Y en el estado actual S.

Programas universales: Simuladores de programas

Recordemos que para cada $n \geq 1$, el programa universal $\mathbf{U_n}$ verifica lo siguiente:

Para **cualquier** programa GOTO, Q, y para toda n-tupla $(a_1, \ldots, a_n) \in \mathbb{N}^n$, con $n \geq 1$:

$$\mathbf{U_n}(\mathbf{a_1}, \ldots, \mathbf{a_n}, \#(\mathbf{Q})) = \mathbf{Q}(\mathbf{a_1}, \ldots, \mathbf{a_n})$$

En tal situación, diremos que $\mathbf{U_n}$ **simula** al programa Q para **n**-tuplas (o, más brevemente, $\mathbf{U_n}$ **simula** a Q).

Un mundo camaleónico

Mundo virtual: los únicos individuos que existen son los programas `GOTO`.

* Cada uno tiene un **nombre**: $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n, \ldots,$
* Cada uno tiene un **apellido** (**código**): $\#(\mathbf{P}_k) = k$.
* Cada uno tiene su propio **disfraz**: $\mathbf{D_k}$.

Algunos de esos individuos son especiales $\mathbf{U_1}, \mathbf{U_2}, \ldots, \mathbf{U_n} \ldots$ (programas universales: "**camaleones**")

* Tienen la habilidad de **simular** el comportamiento de otros individuos.
* Nombre de $\mathbf{U_k}$: $\mathbf{P}_{c_k}$.
* Apellido de $\mathbf{U_k}$: c_k.
* Disfraz de $\mathbf{U_k}$: $\mathbf{D_{c_k}}$.

Un mundo camaleónico

Cada camaleón tiene un armario que contiene:

* Los disfraces de cada uno de los programas GOTO: $\mathbf{D_0}, \mathbf{D_1}, \ldots \mathbf{D_k}, \ldots$.

Cada camaleón puede coger un disfraz de su armario y "colocárselo".

* Si $\mathbf{U_n}$ se coloca el disfraz $\mathbf{D_k}$ entonces trata de "parecerse" a $\mathbf{P}_k$.

Ejercicio 9: ¿Qué se puede asegurar del programa universal $\mathbf{U_n}$ cuando se simula a sí mismo? ($\mathbf{U_n}$ se pone el disfraz $\mathbf{D_{c_n}}$)

Solución: Decir que $\mathbf{U_n}$ se simula a sí mismo significa que para cada n-tupla $(a_1, \ldots, a_n)$ se verifica $\mathbf{U_n}\ (a_1, \ldots, a_n, \#(\mathbf{U_n})\ ) = \mathbf{U_n}\ (a_1, \ldots, a_n)$. Ahora bien,

$$\mathbf{U_n}\ (a_1, \ldots, a_n) = \mathbf{U_n}\ (a_1, \ldots, a_n, 0) = P_\emptyset\ (a_1, \ldots, a_n) = 0$$

Cuando $\mathbf{U_n}$ se simula a sí mismo, entonces $\mathbf{U_n}$ se "**transforma**" en el **programa vacío**.

Un mundo camaleónico

Ejercicio 10: ¿Qué se puede asegurar del programa universal $\mathbf{U_n}$ cuando simula al programa universal $\mathbf{U_{n+1}}$? ($\mathbf{U_n}$ se pone el disfraz $\mathbf{D_{c_{n+1}}}$)

Solución: Decir que $\mathbf{U_n}$ simula a $\mathbf{U_{n+1}}$ significa que para cada n-tupla de entrada $(a_1, \ldots, a_n)$ se verifica $\mathbf{U_n}\ (a_1, \ldots, a_n, \#(\mathbf{U_{n+1}})\ ) = \mathbf{U_{n+1}}\ (a_1, \ldots, a_n)$. Ahora bien,

$$\mathbf{U_{n+1}}\ (a_1, \ldots, a_n) = \mathbf{U_{n+1}}\ (a_1, \ldots, a_n, 0, 0) = P_\emptyset\ (a_1, \ldots, a_n)$$

Luego, $\boxed{\mathbf{U_n}\ (a_1, \ldots, a_n, \#(\mathbf{U_{n+1}})\ ) = P_\emptyset\ (a_1, \ldots, a_n, 0) = 0}$

Cuando $\mathbf{U_n}$ simula a $\mathbf{U_{n+1}}$, entonces $\mathbf{U_n}$ se "**transforma**" en el **programa vacío**.

Un mundo camaleónico

Ejercicio 11: ¿Qué se puede asegurar del programa universal $\mathbf{U_{n+1}}$ cuando simula al programa universal $\mathbf{U_n}$? ($\mathbf{U_{n+1}}$ se pone el disfraz $\mathbf{D_{c_n}}$)

Solución: Decir que $\mathbf{U_{n+1}}$ simula a $\mathbf{U_n}$ significa que para cada $(n+1)$-tupla de entrada $(a_1, \ldots, a_n, a_{n+1})$ se tiene $\mathbf{U_{n+1}}\ (a_1, \ldots, a_n, a_{n+1}\ \#(\mathbf{U_n})\ ) = \mathbf{U_n}\ (a_1, \ldots, a_n, a_{n+1})$.

Ahora bien,

$\mathbf{U_n}\ (a_1, \ldots, a_n, a_{n+1}) = P_{a_{n+1}}\ (a_1, \ldots, a_n)$, siendo $P_{a_{n+1}}$ el programa de código a_{n+1}

Luego, $\boxed{\mathbf{U_{n+1}}\ (a_1, \ldots, a_n, a_{n+1}, \#(\mathbf{U_n})\ ) = P_{a_{n+1}}\ (a_1, \ldots, a_n)}$

Cuando $\mathbf{U_{n+1}}$ simula a $\mathbf{U_n}$, entonces $\mathbf{U_{n+1}}$ se "**transforma**" en el **programa de código la última componente** de la $(n+1)$-tupla de entrada de $\mathbf{U_n}$.

Funciones índice

Para cada $n \geq 1$, $\mathcal{U}_{\mathbf{n}}$ es la función de aridad $(n+1)$ calculada por el programa universal $\mathbf{U_n}$; es decir, $\mathcal{U}_{\mathbf{n}}(a_1, \ldots, a_n, \#(P)) = P(a_1, \ldots, a_n)$.

* Para cada $n \geq 1$ y cada $e \in \mathbb{N}$, se define la **función GOTO–computable n-aria de índice e**, así:

$$\varphi_{\mathbf{e}}^{(\mathbf{n})}(a_1, \ldots, a_n) = \mathcal{U}_{\mathbf{n}}(a_1, \ldots, a_n, e) = \mathbf{P_e}(a_1, \ldots, a_n, e)$$

siendo P_e el programa GOTO de código e.

$\mathbf{P_0}$	$\mathbf{P_1}$	$\mathbf{P_2}$	.....	$\mathbf{P_n}$	.....
$\varphi_0^{(1)}$	$\varphi_1^{(1)}$	$\varphi_2^{(1)}$	.....	$\varphi_n^{(1)}$	.....
$\varphi_0^{(2)}$	$\varphi_1^{(2)}$	$\varphi_2^{(2)}$	.....	$\varphi_n^{(2)}$	.....
$\varphi_0^{(3)}$	$\varphi_1^{(3)}$	$\varphi_2^{(3)}$	.....	$\varphi_n^{(3)}$	.....
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\varphi_0^{(k)}$	$\varphi_1^{(k)}$	$\varphi_2^{(k)}$	.....	$\varphi_n^{(k)}$	.....
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

El predicado $\mathrm{STEP}^{(n)}$

Consideremos el siguiente predicado $(n+2)$-ario sobre $\mathbb{N}$:

$$\left[\ \mathrm{STEP}^{(n)}(a_1, \ldots, a_n, y, t) = 1\ \right] \Longleftrightarrow \left[\begin{array}{l}\text{El programa de código } y \text{ con dato de entrada} \\ (a_1, \ldots, a_n) \text{ para en, a lo sumo, } t \text{ pasos}\end{array}\right]$$

De acuerdo con la definición anterior, es posible expresar la **parada de un** cierto **programa** en, **exactamente**, un determinado paso de computación.

En efecto:

$$\mathrm{STEP}^{(n)}(a_1, \ldots, a_n, y, t) \wedge \forall z < t\,(\neg \mathrm{STEP}^{(n)}(a_1, \ldots, a_n, y, z))$$

expresa que el programa de código y con dato de entrada $(a_1, \ldots, a_n)$ **para en, a lo sumo, t** pasos pero que, en cambio, **no para en menos** de t pasos. t.

En consecuencia, la expresión anterior indica que el programa de código y con dato de entrada $(a_1, \ldots, a_n)$ **para en, EXACTAMENTE t** pasos.

Proposición 5: Para cada $n \geq 1$, el predicado $\mathtt{STEP}^{(n)}$ es GOTO-computable.

Demostración: Consideremos el siguiente programa GOTO, que notaremos $\mathbf{H_n}$:

$$
\begin{array}{ll}
 & Z \leftarrow X_{n+1} + 1 \\
 & S \leftarrow \prod_{i=1}^{n} p_{2i}^{X_i} \\
 & K \leftarrow 1 \\
[\mathbf{C}] & \mathbf{IF\ T > X_{n+2}\ GOTO\ E} \\
 & \mathbf{T \leftarrow T + 1} \\
 & IF\ K = Long(Z) + 1\ GOTO\ F \\
 & U \leftarrow r((Z)_k) \\
 & P \leftarrow p_{r(U)+1} \\
 & IF\ I(U) = 0\ GOTO\ N \\
 & IF\ I(U) = 1\ GOTO\ A \\
 & IF\ P \nmid S\ GOTO\ N \\
 & IF\ I(U) = 2\ GOTO\ M \\
 & K \leftarrow \mu i \leq Long(Z)[I((Z)_i) + 2 = I(U)] \\
 & GOTO\ C \\
[M] & S \leftarrow qt(S, P) \\
 & GOTO\ N \\
[A] & S \leftarrow S \cdot P \\
[N] & K \leftarrow K + 1 \\
 & GOTO\ C \\
[\mathbf{F}] & \mathbf{Y \leftarrow 1}
\end{array}
$$

(Recuento) (4)

Veamos que el programa $\mathbf{H_n}$, calcula el predicado $\mathtt{STEP}^{(n)}$.

El programa $\mathbf{H_n}$ se obtiene del programa universal $\mathbf{U_n}$ añadiendo las "instrucciones":

[C] IF $\mathbf{T} > \mathbf{X_{n+2}}$ GOTO E $\mathbf{T} \longleftarrow \mathbf{T}+\mathbf{1}$	[F] $\mathbf{Y} \longleftarrow \mathbf{1}$

Obsérvese que en el programa $\mathbf{U_n}$ el test de parada aparecía como sigue:

$$[\,\mathbf{C}\,] \quad \mathbf{IF\ K} = \mathbf{Long(Z)} + \mathbf{1} \quad \mathbf{GOTO\ \ F}$$

Pues bien, en el programa $\mathbf{H_n}$ esa macro es sustituida por lo siguiente:

$$\begin{array}{ll} [\mathbf{C}] & \mathbf{IF\ T} > \mathbf{X_{n+2}} \ \ \mathbf{GOTO\ \ E} \\ & \mathbf{T} \longleftarrow \mathbf{T}+\mathbf{1} \\ & \mathbf{IF\ K} = \mathbf{Long(Z)}+\mathbf{1} \ \ \mathbf{GOTO\ \ F} \end{array}$$

La nueva variable **T** contabiliza el número de pasos que lleva realizada la **computación simulada** del programa de código X_{n+1} con dato de entrada $(X_1, \ldots, X_n)$.

En el programa $\mathbf{U_n}$ se transfería el control al test de parada (mediante **GOTO C**), a fin de saber si la computación simulada debía finalizar o no.

En el programa $\mathbf{H_n}$, cuando se transfiere el control a la instrucción etiquetada a la izquierda por **[C]** se va directamente a la siguiente "subrutina":

$$\begin{array}{ll} \textbf{[C]} & \textbf{IF T} > \mathbf{X_{n+2}} \ \textbf{GOTO E} \\ & \mathbf{T} \longleftarrow \mathbf{T+1} \\ & \textbf{IF K} = \textbf{Long(Z)} + \mathbf{1} \ \textbf{GOTO F} \end{array}$$

Se pueden presentar dos casos:

(a) La computación simulada ha realizado un número de pasos **estrictamente mayor** que X_{n+2},

Entonces el programa $\mathbf{H_n}$ ha de finalizar y devolver 0.

Ello se consigue ejecutando la macro **IF T** $>$ $\mathbf{X_{n+2}}$ **GOTO E**.

(b) La computación simulada ha realizado un número de pasos **menor o igual** que X_{n+2}.

Entonces $\mathbf{H_n}$ ha de continuar ejecutando esa computación, contabilizando un paso más.

Ello se consigue ejecutando la instrucción $\mathbf{T \longleftarrow T + 1}$ y yendo después al test de parada.

Cuando se verifique la condición del test, el control de $\mathbf{H_n}$ se transfiere a la instrucción etiquetada a la izquierda por **[F]**; es decir, a la última instrucción $\mathbf{[F]\ Y \longleftarrow 1}$. Por tanto, en este caso, el programa $\mathbf{H_n}$ para y devuelve 1.

■