Modelos de Computación y de Complejidad

Grado en Ingeniería Informática. Tecnologías Informáticas (Curso 2021–2022)

ETS Ingeniería Informática - Universidad de Sevilla

SOLUCIONES DE EJERCICIOS DEL TEMA 3

Ejercicio 1: Sea $f : \mathbb{N} \longrightarrow \mathbb{N}$ una función total. Se define la *función historia* de f, así:

$$\widehat{f}(n) = [f(0), \ldots, f(n)]$$

Probar que $\widehat{f}$ es GOTO-computable sí y sólo si f es GOTO–computable.

Solución:

$\boxed{\Longrightarrow}$ Supongamos que la función f es GOTO-computable.

De acuerdo con la definición de $\widehat{f}$, para cada número natural n se verifica:

$$\widehat{f}(n+1) = [f(0), \ldots, f(n), f(n+1)] = [f(0), \ldots, f(n)] \cdot p_{n+2}^{f(n+1)} = \widehat{f}(n) \cdot p_{n+2}^{f(n+1)}$$

Sea $h : \mathbb{N}^2 \longrightarrow \mathbb{N}$ definida por $h(n,t) = t \cdot p_{n+2}^{f(n+1)}$. Entonces, h es GOTO-computable por ser la composición de dos funciones GOTO-computables (la función producto y la función $p_{n+1}^{f(n)}$). Además, se tiene que

$$\begin{cases} \widehat{f}(0) & = & [\, f(0)\,] = 2^{f(0)} \\ \widehat{f}(n+1) & = & \widehat{f}(n) \cdot p_{n+2}^{f(n+1)} = h(n, \widehat{f}(n)) \end{cases}$$

Por tanto, $\widehat{f} = \mathcal{R}(2^{f(0)}, h)$; es decir, la función historia $\widehat{f}$ está definida por recursión, a partir de la constante $2^{f(0)}$ y de la función h. Puesto que h es una función GOTO-computable, concluimos que también lo es la función historia $\widehat{f}$.

$\boxed{\Longleftarrow}$ Supongamos que la función $\widehat{f}$ es GOTO-computable.

Teniendo presente que $f(n) = (\widehat{f}(n))_{n+1}$ se deduce que f es GOTO-computable, ya que es la composición de dos funciones GOTO-computable: la función "componente $n+1$" y la función $\widehat{f}$.

Ejercicio 2: Probar que las siguientes funciones f y g son GOTO–computables:

$$\begin{cases} f(0) & = & 0 \\ f(n+1) & = & 1 + g(n) \end{cases} \qquad \begin{cases} g(0) & = & 0 \\ g(n+1) & = & 2 + f(n) \end{cases}$$

Solución: Consideremos la función $\psi : \mathbb{N} \to \mathbb{N}$ definida por $\psi(n) = [f(n), g(n)]$, para cada $n \in \mathbb{N}$. Se tiene que $\psi(0) = [\, f(0), g(0)\,] = 2^0 \cdot 3^0 = 1$ y, además, para cada $n \in \mathbb{N}$:

$\psi(n+1) = [\, f(n+1), g(n+1)\,] = [\, 1 + g(n), 2 + f(n)\,] = 2^1 \cdot 2^{g(n)} \cdot 3^2 \cdot 3^{f(n)} = 18 \cdot 2^{g(n)} \cdot 3^{f(n)}$

Sea $h : \mathbb{N}^2 \longrightarrow \mathbb{N}$ definida por $h(n,t) = 18 \cdot 2^{(t)_2} \cdot 3^{(t)_1}$. Entonces, h es GOTO-computable por ser el producto de tres funciones GOTO-computables (la función constante igual a 18, la función exponencial $2^{(t)_2}$ y la función exponencial $3^{(t)_1}$. Además, se tiene que

$$\begin{cases} \psi(0) & = & 1 \\ \psi(n+1) & = & 18 \cdot 2^{g(n)} \cdot 3^{f(n)} = 18 \cdot 2^{(\psi(n))_2} \cdot 3^{(\psi(n))_1} = h(n, \psi(n)) \end{cases}$$

Por tanto, $\psi = \mathcal{R}(1, h)$; es decir, la función ψ está definida por recursión, a partir de la constante 1 y de la función h. Puesto que h es GOTO-computable, concluimos que también lo es la función ψ.

Ahora bien, $f(n) = (\psi(n))_1$; es decir, f es la composición de dos funciones GOTO-computables (la función "componente 1" y la función ψ) y, en consecuencia, f será GOTO-computable. De manera similar se prueba la GOTO-computabilidad de la función g, observando que $g(n) = (\psi(n))_2$.

Ejercicio 4: Sea p_k el k-ésimo número primo ($k \geq 1$). Determinar para qué valores de k el número natural $4 \cdot p_k + 1$ es el código de una configuración inicial de un programa `GOTO`.

Indicación: Téngase presente que en una configuración inicial los valores de cualquier variable que **no** sea de entrada, ha de valer 0.

Solución: Sea $\sigma = (j, s)$ la configuración inicial asociada a un dato de entrada $(a_1, a_2, \ldots, a_r)$. Entonces, $j = 1$ y, además, $s = \{Y = 0, X_1 = a_1, Z_1 = 0, X_2 = a_2, Z_2 = 0, \cdots X_r = a_r, Z_r = 0\}$. Luego, $\#(s) = [0, a_1, 0, a_2, 0, \ldots, a_r] = p_{2\cdot 1}^{a_1} \cdot p_{2\cdot 2}^{a_2} \cdot \cdots \cdot p_{2\cdot r}^{a_r}$

Si p_k representa el k-ésimo número primo ($k \geq 1$), veamos para qué valores de k, se tiene que $\#(s) = 4 \cdot p_k + 1$. Entonces, se ha de verificar que $4 \cdot p_k + 1 = \#(\sigma) = \langle 1, \#(s)\rangle = 2^1 \cdot (2 \cdot \#(s) + 1) - 1$; es decir, $4 \cdot p_k + 2 = 2^1 \cdot (2 \cdot \#(s) + 1)$. Ahora bien,

$$4 \cdot p_k + 1 = \#(\sigma) = \langle 1, \#(s)\rangle = 2^1 \cdot (2 \cdot \#(s) + 1) - 1 \implies 4 \cdot p_k + 2 = 2^1 \cdot (2 \cdot \#(s) + 1) \implies$$
$$\implies 2 \cdot p_k + 1 = 2 \cdot \#(s) + 1 \implies p_k = \#(s) = p_{2\cdot 1}^{a_1} \cdot p_{2\cdot 2}^{a_2} \cdot \cdots \cdot p_{2\cdot r}^{a_r} \implies \exists j \ (1 \leq j \leq r \wedge k = 2j \wedge \alpha_j = 1)$$

Por tanto, k ha de ser un número par ($k = 2j$) y, además, un único valor del dato de entrada (concretamente, el de la variable X_j) de esa configuración inicial, será distinto de cero (y valdrá 1).

Ejercicio 10: Se considera la computación del siguiente programa con dato de entrada $(2, 1)$:

$$P \equiv \left\{ \begin{array}{lll} I_1 \equiv & [A] & Z_2 \longleftarrow Z_2 - 1 \\ \ldots & \ldots & \ldots \\ I_6 \equiv & & X \longleftarrow X + 1 \\ I_7 \equiv & & IF\ X_2 \neq 0\ GOTO\ A \\ \ldots & \ldots & \ldots \end{array} \right.$$

Al cabo de $k - 1$ pasos se sabe que la configuración σ_k, es tal que $\#(\sigma_k) = 6623295$. Se pide: a) Obtener $\#(\sigma_{k+1}), \#(\sigma_{k+2})$ y $\#(\sigma_{k+3})$. b) Obtener el valor de la variable Y en el estado asociado a la configuración σ_{k+3}.

Solución: Sea $\sigma_k = (j, s)$. Entonces $6623295 = \#\sigma_k = \langle j, \#(s)\rangle = 2^j \cdot (2 \cdot \#(s) + 1) - 1$. Luego,

$$2^j \cdot (2 \cdot \#(s) + 1) = 6623296 = 2^6 \cdot 103489 \implies j = 6 \text{ y } 2 \cdot \#(s) + 1 = 103489$$

De donde resulta que $\#(s) = 51744 = 2^5 \cdot 3 \cdot 7^2 \cdot 11 = [5, 1, 0, 2, 1]$. Por tanto, $\sigma_k = (6, s)$

Cálculo de $\#(\sigma_{k+1})$: Puesto que $\sigma_k = (6, [5, 1, 0, 2, 1])$, la configuración σ_{k+1} se obtiene ejecutando la instrucción $I_6 \equiv X \longleftarrow X + 1$ al estado $[5, 1, 0, 2, 1]$. Por tratarse de una instrucción incremento, habrá que aumentar en 1 el valor de la primera componente e incrementar en 1 el valor de X. Luego, $\sigma_{k+1} = (7, [5, \mathbf{2}, 0, 2, 1]) = (7, 51744 \cdot 3) = (7, 155232)$. Por tanto,

$$\#(\sigma_{k+1}) = \langle 7, 155232\rangle = 2^7 \cdot (2 \cdot 155232 + 1) - 1 = 39739520 - 1 = 39739519$$

Cálculo de $\#(\sigma_{k+2})$: Puesto que $\sigma_{k+1} = (7, [5, 2, 0, 2, 1])$, la configuración σ_{k+2} se obtiene ejecutando la instrucción $I_7 \equiv IF\ X_2 \neq 0\ GOTO\ A$ al estado $[5, 2, 0, 2, 1]$). Teniendo presente que, en ese estado, el valor de X_2 es distinto de cero (concretamente, su valor es 2), el control del programa se transfiere a la primera instrucción etiquetada a la izquierda por A, es decir, a la instrucción I_1. Luego, $\sigma_{k+2} = (1, [5, 2, 0, 2, 1])$ y, por tanto,

$$\#(\sigma_{k+2}) = \langle 1, 155232\rangle = 2^1 \cdot (2 \cdot 155232 + 1) - 1 = 620930 - 1 = 620929$$

Cálculo de $\#(\sigma_{k+3})$: Puesto que $\sigma_{k+2} = (1, [5, 2, 0, 2, 1])$, la configuración σ_{k+3} se obtiene ejecutando la instrucción $I_1 \equiv Z_2 \longleftarrow Z_2 - 1$ al estado $[5, 2, 0, 2, 1]$). Luego, $\sigma_{k+3} = (2, [5, 2, 0, 2, \mathbf{0}])$ y, por tanto, $\#(\sigma_{k+3}) = \langle 2, \frac{155232}{11}\rangle = 2^2 \cdot (2 \cdot 14112 + 1) - 1 = 112900 - 1 = 112899$.

Nota importante: Obsérvese que sólo se dispone de una información parcial del programa P. Pues bien, teniendo presente que $\sigma_{k+3} = (2, [5, 2, 0, 2, 0])$ y que se desconoce la instrucción I_2, con los datos del enunciado **no** se podría calcular su configuración siguiente σ_{k+4}.

Ejercicio 13: Sea $\mathbf{U_2}$ un programa universal asociado al número natural 2. Hallar las computaciones $\mathbf{U_2}(4,5,e)$ y $\mathbf{U_2}(7,0,e)$, siendo $e = 2^{105} \cdot 3^2 \cdot 5^{110} - 1$.

Solución: Vamos a hallar, con todo detalle, la computación $\mathbf{U_2}(4,5,e)$, siendo $e = 2^{105} \cdot 3^2 \cdot 5^{110} - 1$. De manera similar se puede hallar la computación $\mathbf{U_2}(7,0,e)$.

Teniendo presente que $\mathbf{U_2}(4,5,e) = Q(4,5)$, siendo Q el programa de código e, para hallar la computación $\mathbf{U_2}(4,5,e)$ se puede proceder de dos maneras: **(1)** calculando, explícitamente, el programa Q de código e y realizando la ejecución de dicho programa con dato de entrada $(4,5)$; y **(2)** hallando, directamente, una traza de dicha computación, a partir de la descripción del programa universal $\mathbf{U_2}$.

(1) A continuación, vamos a resolver el problema calculando, explícitamente, el programa Q de código $e = 2^{105} \cdot 3^2 \cdot 5^{110} - 1$ y, posteriormente, hallando la computación $Q(4,5)$.

Sea $Q = (I_1, \ldots, I_r)$ tal que $\#(Q) = e = 2^{105} \cdot 3^2 \cdot 5^{110} - 1$. Se verifica:

$$\#(Q) = [\#(I_1), \ldots, \#(I_r)] - 1 = 2^{105} \cdot 3^2 \cdot 5^{110} - 1 \Longrightarrow [\#(I_1), \ldots, \#(I_r)] = 2^{105} \cdot 3^2 \cdot 5^{110}$$

Luego, $r = 3$, $\#(I_1) = 105, \#(I_2) = 2, \#(I_3) = 110$.

Razonando de la manera habitual, se obtiene las instrucciones que componen el programa Q:

$$\left\{ \begin{array}{lcll} I_1 & \equiv & [A_1] & X_7 \longleftarrow X_7 \\ I_2 & \equiv & & Y \longleftarrow Y + 1 \\ I_3 & \equiv & & IF\ X_2 \neq 0\ GOTO\ A_1 \end{array} \right.$$

Seguidamente, vamos a hallar la computación $Q(4,5)$:

- ⋆ Configuración inicial: $\sigma_1 = (1, \{Y = 0, X_1 = 4, X_2 = 5\})$
- ⋆ $\sigma_2 = (2, \{Y = 0, X_1 = 4, X_2 = 5\})$
- ⋆ $\sigma_3 = (3, \{Y = 1, X_1 = 4, X_2 = 5\})$
- ⋆ $\sigma_4 = (1, \{Y = 1, X_1 = 4, X_2 = 5\})$
- ⋆ $\sigma_5 = (2, \{Y = 1, X_1 = 4, X_2 = 5\})$
- ⋆ $\sigma_6 = (3, \{Y = 2, X_1 = 4, X_2 = 5\})$
- ⋆ $\sigma_7 = (1, \{Y = 2, X_1 = 4, X_2 = 5\})$
- ⋆ $\sigma_8 = (2, \{Y = 2, X_1 = 4, X_2 = 5\})$
- ⋆ $\sigma_9 = (3, \{Y = 3, X_1 = 4, X_2 = 5\})$
- ⋆ $\sigma_5 = (1, \{Y = 3, X_1 = 4, X_2 = 5\})$
- ⋆ Y así sucesivamente ... **no** hay **configuración de parada** (ya que el valor de X_2 siempre vale 5 y, por tanto, siempre es distinto de cero).

Conclusión: $Q(4,5)\uparrow$.

(2) Vamos a realizar una traza de la computación $\mathbf{U_2}(4,5,e)$. Para ello, comenzamos con la fase de inicialización del programa $\mathbf{U_2}$. Puesto que el dato de entrada de $\mathbf{U_2}$ es (X_1, X_2, X_3), siendo $X_1 = 4, X_2 = 5$ y $X_3 = e = 2^{105} \cdot 3^2 \cdot 5^{110} - 1$, inicializamos las variables de trabajo Z, S y K como sigue:

- ⋆ $Z = X_{n+1} + 1 \overset{n=2}{=} X_3 + 1 = 2^{105} \cdot 3^2 \cdot 5^{110} = [\,105,\ 2,\ 110\,]$
- ⋆ $S = \prod_{i=1}^{n} p_{2\cdot i}^{X_i} \overset{n=2}{=} p_{2\cdot 1}^{X_1} \cdot p_{2\cdot 2}^{X_2} = p_2^4 \cdot p_4^5 = [\,0,\ 4,\ 0,\ 5\,]$ (que codifica $\{Y = 0, X_1 = 4, Z_1 = 0, X_2 = 5\}$)
- ⋆ $K = 1$

Seguidamente, se pasa a ejecutar la subrutina etiquetada a la izquierda por C; es decir, volvemos al test de parada. El condicional de dicho test $(K = Long(Z) + 1)$ **no** se verifica ya que $K = 1$ y $Long(Z) + 1 = 3 + 1 = 4$. Por tanto, el control del programa pasa a la siguiente instrucción.

- $\star$ $U = r((Z)_K) = r((Z)_1) = r(105) = r(\langle 1, \langle 0, 13\rangle\rangle) = \langle 0, 13\rangle$
- $\star$ $P = p_{r(U)+1} = p_{13+1} = p_{14}$ (la variable de la instrucción cuya ejecución se simula, es X_7)

Puesto que $l(U) = 0$ (es decir, la instrucción cuya ejecución se simula, es $X_7 \longleftarrow X_7$), el control del programa pasa a la subrutina etiquetada a la izquierda por N, con lo cual el nuevo valor de K es 2 y se vuelve a ejecutar el test de parada $(GOTO\ C)$.

El condicional de dicho test sigue sin verificarse ya que $K = 2$ y $Long(Z)+1 = 4$. Por tanto, el control del programa pasa a la siguiente instrucción.

- $\star$ $U = r((Z)_K) = r((Z)_2) = r(2) = r(\langle 0, \langle 1, 0\rangle\rangle) = \langle 1, 0\rangle$
- $\star$ $P = p_{r(U)+1} = p_{0+1} = p_1$ (la variable de la instrucción cuya ejecución se simula, es Y)

Puesto que $l(U) = 1$ (es decir, la instrucción cuya ejecución se simula, es $Y \longleftarrow Y + 1$), el control del programa pasa a la subrutina etiquetada a la izquierda por A, con lo cual el nuevo valor de S será $S \cdot P$ (es decir, $S = p_2^4 \cdot p_4^5 \cdot p_1 = [\,\mathbf{1},\, 4,\, 0,\, 5\,]$) (es decir, el valor de la variable Y se ha aumentado en 1), y el nuevo valor de K será 3, volviéndose a ejecutar el test de parada $(GOTO\ C)$.

El condicional de dicho test sigue sin verificarse ya que $K = 3$ y $Long(Z)+1 = 4$. Por tanto, el control del programa pasa a la siguiente instrucción.

- $\star$ $U = r((Z)_K) = r((Z)_3) = r(110) = r(\langle 0, \langle 3, 3\rangle\rangle) = \langle 3, 3\rangle$
- $\star$ $P = p_{r(U)+1} = p_{3+1} = p_4$ (la variable de la instrucción cuya ejecución se simula, es X_2)
- $\star$ La condición P **no divide** a S es **falsa** ya que $P = p_4$ divide a $S = p_1 \cdot p_2^4 \cdot p_4^5$.

Puesto que la relación $l(U) = 2$ **es falsa** ya que, en realidad, $l(U) = 3$ (es decir, la instrucción cuya ejecución se simula, es $IF\ X_2 \neq 0\ GOTO\ A$, ya que $\#(A) = l(U) - 2 = 1$), se pasa a ejecutar la instrucción $K \longleftarrow \mu j \leq Long(Z)[l((Z)_j) + 2 = l(U)]$. Es decir, el nuevo valor de K será el menor $j \leq Long(Z)$ tal que $l((Z)_j) = l(U) - 2 = 3 - 2 = 1$.

Teniendo presente que

- $\star$ $l((Z)_1) = l(105) = l(\langle 1, \langle 0, 13\rangle\rangle = 1$
- $\star$ $l((Z)_2) = l(2) = l(\langle 0, \langle 1, 0\rangle\rangle = 0$
- $\star$ $l((Z)_3) = l(110) = l(\langle 0, \langle 3, 3\rangle\rangle = 0$

se deduce que el nuevo valor de K es 1 y, a continuación, se ejecuta la instrucción $GOTO\ C$; es decir, se vuelve al test de parada. Dicho con otras palabras, a partir de ahora, se vuelve a ejecutar la primera instrucción pero, en esta ocasión, aplicada al estado $S = [\,\mathbf{1},\, 4,\, 0,\, 5\,]$. Es decir, podemos pensar que **se ha dado una vuelta de un cierto "bucle"**, habiéndose modificado el estado, específicamente, se ha pasado del estado $S = [\,0,\, 4,\, 0,\, 5\,]$ al estado $S = [\,\mathbf{1},\, 4,\, 0,\, 5\,]$.

Si se repite el proceso y se da una nueva "vuelta del bucle", entonces tendríamos que $K = 1$ y $S = [\,\mathbf{2},\, 4,\, 0,\, 5\,]$. En la siguiente vuelta, se tendría que $K = 1$ y $S = [\,\mathbf{3},\, 4,\, 0,\, 5\,]$. Este proceso se puede reiterar indefinidamente y, en consecuencia, la computación $\mathbf{U_2}(4, 5, e)$ **no es de parada**. Es decir, $\mathbf{U_2}(4, 5, e) \uparrow$ y, por tanto, $\mathbf{U_2}(4, 5, e) = Q(4, 5)$.

Ejercicio 14: Probar que los siguientes conjuntos son GOTO–computables:

- $A = \{x \in \mathbb{N}:$ el programa GOTO de código x carece de instrucciones del tipo CONDICIONAL$\}$.
- $B = \{x \in \mathbb{N}:$ el programa GOTO de código $x+8$ posee, a lo sumo, 8 instrucciones$\}$.
- $C = \{x \in \mathbb{N}:$ el programa GOTO de código x posee 10 instrucciones y ninguna de ellas es SKIP$\}$.
- $D = \{x \in \mathbb{N}:$ el programa GOTO de código $2x+3$ posee, a lo sumo, 7 instrucciones y alguna de ellas es un DECREMENTO$\}$.
- $E = \{x \in \mathbb{N}:$ el programa GOTO de código $7x+7$ posee un número par de instrucciones y todas ellas son de tipo CONDICIONAL $\}$.
- $F = \{x \in \mathbb{N}:$ x es el código de un programa GOTO tal que todas sus variables son de entrada$\}$.
- $G = \{x \in \mathbb{N}:$ el programa GOTO de código x posee alguna instrucción etiquetada a la izquierda$\}$.
- $H = \{x \in \mathbb{N}:$ el programa GOTO de código x^2 **para** sobre $x+4$ en, **exactamente**, x pasos $\}$.
- $I = \{x \in \mathbb{N}:$ el programa GOTO de código $3x$ para sobre $4x$ en, **exactamente**, $5x$ pasos$\}$.

Solución: Antes de comenzar la resolución, hagamos varias observaciones.

- Para establecer que un conjunto es GOTO-computable, basta probar que la correspondiente relación de pertenencia a dicho conjunto, es un predicado GOTO-computable.
- Si P **es un programa de código** x entonces:
  - El **número** de **instrucciones** de P es $Long(x+1)$.
  - El **código** de la **instrucción** j-ésima de P es $(x+1)_j$.
  - El **código** del **formato** de la instrucción j-ésima de P es $l(r((x+1)_j))$.
  - Una **instrucción** de P **no** es un **CONDICIONAL** si y sólo si $l(r((x+1)_j))) \leq 2$.
  - Una **instrucción** de P **no** es un **SKIP** si y sólo si $l(r((x+1)_j))) \geq 1$.
  - Una **instrucción** de P es un **DECREMENTO** si y sólo si $l(r((x+1)_j))) = 2$.
  - El **código** de la **variable** de la instrucción j-ésima de P es $r(r((x+1)_j))+1$.
  - Una **instrucción está etiquetada a la izquierda** si y sólo si $l((x+1)_j)) \geq 1$.
  - El **código** de cualquier **variable de entrada** es un número par.

GOTO-computabilidad del conjunto A

$A = \{x \in \mathbb{N}:$ **el programa de código** x **carece de instrucciones del tipo CONDICIONAL**$\}$

Para cada número natural $x \in \mathbb{N}$ se verifica:

$$\begin{array}{rcl} x \in A & \iff & \text{toda instrucción del programa de código } x \text{ es del tipo skip, incremento o decremento} \\ & \iff & \forall j \leq Long(x+1)\ (j \geq 1 \longrightarrow l(r((x+1)_j)) \leq 2) \end{array}$$

Este predicado es GOTO-computable ya que se trata de una cuantificación universal acotada (en donde aparece la función "longitud" de una función "suma", que son GOTO-computables) de un predicado implicación, en donde el antecedente es un predicado "$\geq$" y el consecuente es un predicado "$\leq$", en donde aparecen las funciones "parte izquierda" (l), "parte derecha" (r) y la función "componente j-ésima" (todas ellas GOTO-computables).

GOTO-computabilidad del conjunto B

$B = \{x \in \mathbb{N} :$ **el programa de código $x+8$ posee, a lo sumo, 8 instrucciones**$\}$

Para cada número natural $x \in \mathbb{N}$ se verifica:

$$x \in B \iff Long[(x+8)+1] \leq 8 \Longleftrightarrow Long(x+9) \leq 8$$

Este predicado es GOTO-computable ya que se trata de un predicado "$\leq$", en donde aparecen la función "longitud" y la función "suma" (ambas GOTO-computables).

GOTO-computabilidad del conjunto C

$C = \{x \in \mathbb{N} :$ **el programa de código x posee 10 instrucciones y ninguna de ellas es SKIP**$\}$

Para cada número natural $x \in \mathbb{N}$ se verifica:

$$x \in C \iff Long(x+1) = 10 \ \wedge \ \forall j \leq Long(x+1) \ (j \geq 1 \longrightarrow l(r((x+1)_j)) \geq 1)$$

Este predicado es GOTO-computable ya que se trata de la conjunción de dos predicados GOTO-computables: el primero es un predicado de igualdad asociado a la función "longitud" de una función "suma", que son GOTO-computables, y el segundo es una cuantificación universal acotada (en donde aparece la función "longitud" de una función "suma") de un predicado implicación, en donde el antecedente es un predicado "$\geq$" y el consecuente es otro predicado "$\geq$", en donde aparecen las funciones "parte izquierda" (l), "parte derecha" (r) y la función "componente j-ésima" (todas ellas GOTO-computables).

GOTO-computabilidad del conjunto D

$D = \{x \in \mathbb{N} :$ **el programa de código $2x+3$ posee, a lo sumo, 7 instrucciones y alguna de ellas es un DECREMENTO**$\}$

Para cada número natural $x \in \mathbb{N}$ se verifica:

$$x \in D \iff Long(2x+4) \leq 7 \ \wedge \ \exists j \leq Long(2x+4) \ (j \geq 1 \wedge l(r((2x+4)_j)) = 2)$$

Este predicado es GOTO-computable ya que se trata de la conjunción de dos predicados GOTO-computables: el primero es un predicado "$\leq$" asociado a la función "longitud" de una función "suma", que son GOTO-computables, y el segundo es una cuantificación existencial acotada (en donde vuelve a aparecer la función "longitud" de una función "suma") de un predicado implicación, en donde el antecedente es un predicado "$\geq$" y el consecuente es un predicado "$=$", en donde aparecen las funciones "parte izquierda" (l), "parte derecha" (r) y la función "componente j-ésima" (todas ellas GOTO-computables).

GOTO-computabilidad del conjunto E

$E = \{x \in \mathbb{N} :$ **el programa de código $7x+7$ posee un número par de instrucciones y todas ellas son de tipo CONDICIONAL** $\}$

Para cada número natural $x \in \mathbb{N}$ se verifica:

$$x \in E \iff Long(7x+8) \text{ es un número par } \wedge \ \forall j \leq Long(7x+8) \ (j \geq 1 \longrightarrow l(r((7x+8)_j)) \geq 3)$$

Este predicado es GOTO-computable ya que se trata de la conjunción de dos predicados GOTO-computables: el primero es un predicado del tipo "ser número par", asociado a la función "longitud" de una función "suma", que son GOTO-computables, y el segundo es una cuantificación universal acotada (en donde vuelve a aparecer la función "longitud" de una función "suma") de un predicado implicación, en donde el antecedente es un predicado "$\geq$" y el consecuente es otro predicado "$\geq$", en donde aparecen las funciones "parte izquierda" (l), "parte derecha" (r) y la función "componente j-ésima" (todas ellas GOTO-computables).

GOTO-computabilidad del conjunto F

$F = \{x \in \mathbb{N} :$ **x es el código de un programa tal que todas sus variables son de entrada**$\}$

Para cada número natural $x \in \mathbb{N}$ se verifica:

$$\begin{array}{rcl} x \in F & \iff & \forall j \leq Long(x+1)\ [\, j \geq 1 \longrightarrow r(r((x+1)_j)) + 1 \text{ es par}\,] \\ & \iff & \forall j \leq Long(x+1)\ [\, j \geq 1 \longrightarrow r(r((x+1)_j)) \text{ es impar}\,] \end{array}$$

Este predicado es GOTO-computable ya que se trata de una cuantificación universal acotada (en donde aparece la función "longitud" de una función "suma", que son GOTO-computables) de un predicado que es una implicación, en donde el antecedente es un predicado "$\geq$" y el consecuente es un predicado del tipo "ser número impar", en donde aparecen las funciones "parte derecha" (r) y la función "componente j-ésima" (ambas GOTO-computables).

GOTO-computabilidad del conjunto G

$G = \{x \in \mathbb{N} :$ **el programa de código x posee alguna instrucción etiquetada a la izquierda**$\}$

Para cada número natural $x \in \mathbb{N}$ se verifica:

$$x \in G \iff \exists j \leq Long(x+1)\ (j \geq 1 \longrightarrow l((x+1)_j) \geq 1)$$

Este predicado es GOTO-computable ya que se trata de una cuantificación existencial acotada (en donde aparece la función "longitud" de una función "suma", que son GOTO-computables) de un predicado implicación, cuyo antecedente es un predicado "$\geq$" y el consecuente es otro predicado "$\geq$", en donde aparecen las funciones "parte izquierda" (l) y la función "componente j-ésima" (ambas GOTO-computables).

GOTO-computabilidad del conjunto H

$H = \{x \in \mathbb{N} :$ **el programa de código x^2 para sobre $x+4$ en, exactamente, x pasos** $\}$

Para cada número natural $x \in \mathbb{N}$ se verifica:

$$x \in H \iff \mathtt{STEP}^{(1)}(x+4, x^2, x)\ \wedge\ \forall t < x\ (\neg \mathtt{STEP}^{(1)}(x+4, x^2, t))$$

Este predicado es GOTO-computable ya que se trata de la conjunción de dos predicados que son GOTO-computables: el primero de ellos es un predicado STEP y el segundo es una cuantificación universal acotada de la negación de un predicado STEP.

GOTO-computabilidad del conjunto I

$I = \{x \in \mathbb{N} :$ **el programa GOTO de código $3x$ para sobre $4x$ en, exactamente, $5x$ pasos**$\}$

Para cada número natural $x \in \mathbb{N}$ se verifica:

$$x \in I \iff \mathtt{STEP}^{(1)}(4x, 3x, 5x)\ \wedge\ \forall t < 5x\ (\neg \mathtt{STEP}^{(1)}(x+4, x^2, t))$$

Este predicado es GOTO-computable ya que se trata de la conjunción de dos predicados que son GOTO-computables: el primero de ellos es un predicado STEP y el segundo es una cuantificación universal acotada de la negación de un predicado STEP.

Ejercicio 15: Justificar la veracidad o falsedad de las siguientes afirmaciones.

(a) Dos funciones GOTO–computables pueden tener asociadas el **mismo** índice.

(b) El conjunto $A = \{\#(P) : P$ es un programa GOTO cuya última instrucción es $Y \longleftarrow Y + 1\}$ es GOTO–computable.

(c) La configuración cuyo código sea 5293 es una configuración inicial

(d) El código de la configuración de parada de un programa GOTO **nunca** puede ser un número **impar**.

(e) La configuración de código 31755 es una **configuración inicial** del programa GOTO de código 3968 y el resultado de la ejecución del programa con esa configuración inicial es 1.

(f) Una configuración $\sigma = (j, s)$ de un programa $P = (I_1, \ldots, I_r)$ es **inicial** si el valor de j es 1; es decir, si la primera instrucción a ejecutar es I_1. Además, la configuración σ es de **parada** si $j = r + 1$.

(g) Los códigos de las configuraciones iniciales y las configuraciones de parada de un programa GOTO siempre son números impares.

(h) Sean P y Q programas GOTO cuyos códigos son m y n, respectivamente. Si $m + 1$ y $n + 1$ son números naturales con el **mismo número** de factores primos, entonces dichos programas pueden tener **distinta longitud**

(i) Un programa universal es capaz de reproducir el comportamiento de cualquier programa GOTO (es decir, **todas** las ejecuciones de cualquier programa P).

(j) El programa universal $\mathtt{U}_2$ tiene la capacidad de simular **todas** las computaciones de **cualquier** programa GOTO con dato de entrada un par ordenado (a_1, a_2) **arbitrario**.

(k) Si $\mathtt{U}_2$ es un programa universal (correspondiente al número natural 2), entonces se verifica que $[\![\mathtt{U}_2]\!]^{(3)}(1, 2, 3) = 4$.

(l) Sean e, e' números naturales tales que $\varphi_e^{(1)} = \varphi_{e'}^{(1)}$. Entonces se tiene que $\varphi_e^{(2)} = \varphi_{e'}^{(2)}$.

Solución:

(a) **Dos funciones GOTO–computables pueden tener asociadas el mismo índice.**

Respuesta: Cierto.

Justificación de la respuesta: Para responder afirmativamente, bastará presentar dos funciones GOTO–computables que sean distintas y, a la vez, tengan el mismo índice.

Recordemos que todo programa GOTO calcula infinitas funciones: una para cada aridad. Por tanto, **sí** es posible que existan dos funciones distintas (con diferentes aridades) de tal manera que sean calculadas por un **mismo** programa y, en consecuencia, dichas funciones tendrán el **mismo índice**.

Por ejemplo, consideremos las funciones $f : \mathbb{N} \to \mathbb{N}$ y $g : \mathbb{N}^2 \to \mathbb{N}$ definidas como sigue: $f(x) = 0$, para cada $x \in \mathbb{N}$, y $g(x_1, x_2) = 0$, para cada $(x_1, x_2) \in \mathbb{N}^2$. Es decir, se trata de dos funciones "idénticamente nulas", pero la primera es de aridad 1 y la segunda es de aridad 2; por tanto, dichas funciones serán distintas (téngase presente que $f = \{(x, 0) \mid x \in \mathbb{N}\}$ y $g = \{(x_1, x_2, 0) \mid x_1 \in \mathbb{N} \wedge x_2 \in \mathbb{N}\}$).

Se piede probar muy fácilmente que **el programa vacío** calcula ambas funciones. Por tanto, f y g serán funciones distintas que tienen el mismo índice, que será el código del programa vacío; es decir, el número 0.

(b) **El conjunto $A = \{\#(P) : P$ es un programa GOTO cuya última instrucción es $Y \longleftarrow Y + 1\}$ es GOTO–computable.**

Respuesta: Cierto.

Justificación de la respuesta: Basta demostrar que la relación de pertenencia a dicho conjunto puede ser descrita como sigue:

$$\begin{array}{rcl} x \in A & \Longleftrightarrow & \text{la última instrucción del programa de código } x \text{ es } Y \longleftarrow Y+1 \\ & \Longleftrightarrow & \text{el código de la última instrucción del programa de código } x \text{ es } 2 \\ & \Longleftrightarrow & (x+1)_{Long(x+1)} = 2 \end{array}$$

(téngase presente que el código de la instrucción $Y \longleftarrow Y + 1$ es 2):

El predicado de pertenencia antes descrito es GOTO-computable ya que se trata de un predicado de igualdad "=", en donde aparecen las funciones "longitud" (asociada a una función "suma") y la "componente $Long(x + 1)$-ésima", todas ellas son GOTO-computables.

(c) **La configuración cuyo código sea 5293 es una configuración inicial.**

Respuesta: Cierto.

Justificación de la respuesta: Sea $\sigma = (j, s)$ la configuración cuyo código es 5293. Entonces, teniendo presente que $\#(\sigma) = \langle j, \#(s)\rangle = 2^j \cdot (2 \cdot \#(s) + 1) - 1$, se tiene que:

$$2^j \cdot (2 \cdot \#(s) + 1) = 5294 = 2 \cdot 2647$$

Luego, **j=1** y $2 \cdot \#(s) + 1 = 2647$; es decir, $\#(s) = 1323$.

Teniendo presente que $1323 = 3^3 \cdot 7^2$ se deduce que $\#(s) = [\,0,\, 3,\, 0,\, 2\,]$

Por tanto, la primera componente de dicha configuración es 1 y, además, el estado s asociado a la misma es $\{Y = 0, X_1 = 3, Z_1 = 0, X_2 = 2\}$. En consecuencia, la configuración σ **sí** es una configuración inicial ya que es 0 el valor de cualquier variable que no sea de entrada.

(d) **El código de una configuración de parada de un programa GOTO nunca puede ser un número impar.**

Respuesta: Falso.

Justificación de la respuesta: Vamos a probar no sólo que existen configuraciones de parada de programas GOTO cuyo código sea un número impar, sino algo mucho más general: veamos que **cualquier configuración de un programa GOTO siempre es un un número impar**.

En efecto: sea $\sigma = (j, s)$ una configuración arbitraria de un programa GOTO, P. Entonces, $1 \leq j \leq 1 + |P|$. Además, $\#(\sigma) = \langle j, \#(s)\rangle = 2^j \cdot (2 \cdot \#(s) + 1) - 1$. Puesto que $j \geq 1$ se deduce que $2^j \cdot (2 \cdot \#(s) + 1)$ es un número par, mayor que 1, y, por tanto, $\#(\sigma)$ es un número impar.

(e) **La configuración de código 31755 es una configuración inicial del programa GOTO de código 3968 y, además, el resultado de la ejecución del programa con esa configuración inicial es 1.**

Respuesta: Falso.

Justificación de la respuesta: En este ejercicio se pregunta si son verdaderas dos afirmaciones. Vamos a probar que la primera de ellas ("la configuración de código 31755 es una configuración inicial del programa GOTO de código 3968") es falsa, lo cuál será suficiente para justificar nuestra respuesta negativa. Más concretamente, vamos a probar que esa configuración no es inicial de **ningún** programa GOTO.

En efecto: sea $\sigma = (j, s)$ la configuración de código 31755. Entonces se verifica lo siguiente:

$$\#(\sigma) = \langle j, \#(s)\rangle = 2^j \cdot (2 \cdot \#(s) + 1) - 1 = 31755 \Longrightarrow 2^j \cdot (2 \cdot \#(s) + 1) = 31756 = 2^2 \cdot 7939$$

Por tanto, $j = 2$ y, en consecuencia, σ **no** es una configuración inicial.

(f) **Una configuración $\sigma = (j, s)$ de un programa $P = (I_1, \ldots, I_r)$ es inicial si el valor de j es 1; es decir, si la primera instrucción a ejecutar es I_1. Además, la configuración σ es de parada si $j = r + 1$.**

Respuesta: Falso.

Justificación de la respuesta: En este aserto se afirma dos cosas, de las cuáles la primera (*una configuración $\sigma = (j, s)$ es inicial si $j = 1$*) es **falsa** y la segunda (*una configuración $\sigma = (j, s)$ de un programa de longitud r es de parada si $j = r + 1$*) es **verdadera**.

En efecto, el primero de los asertos es falso ya que, por ejemplo, la configuración $\sigma = (1, s)$, en donde $s = \{Y = 3, X_1 = 2\}$ **no** es inicial (pues el valor de Y no es 0). Por otra parte, el segundo de los asertos es verdadero ya que, por **definición**, las configuraciones de parada de un programa de longitud r son aquellas cuya primera componente, j, es igual a $r + 1$.

(g) **Los códigos de las configuraciones iniciales y las configuraciones de parada de un programa GOTO siempre son números impares.**

Respuesta: Cierto.

Justificación de la respuesta: En el apartado (d) se ha demostrado que el código de **cualquier** configuración de un programa GOTO, **siempre** es un número impar. Por tanto, la afirmación que se hace en este apartado, es verdadera.

(h) **Sean P y Q programas GOTO cuyos códigos son m y n, respectivamente. Si $m + 1$ y $n + 1$ son números naturales con el mismo número de factores primos, entonces dichos programas pueden tener distinta longitud.**

Respuesta: Cierto.

Justificación de la respuesta: Vamos a proporcionar un ejemplo "adecuado". Sean P y Q programas GOTO tales que $\#(P) = m+1$ y $\#(Q) = n+1$, siendo $m+1 = 2^4 \cdot 5^1$ y $n+1 = 3^8 \cdot 7^2$. Entonces, en la descomposición en factores primos, los códigos de P y Q tienen el mismo número de primos (en este caso, 2). Además, se verifica:

$$\#(P) = m + 1 = 2^4 \cdot 5^1 = [\,4,\, 0,\, 1\,] \quad \text{y} \quad \#(Q) = n + 1 = 3^8 \cdot 7^2 = [\,0,\, 8,\, 0,\, 2\,]$$

Es decir, los programas P y Q satisfacen las condiciones del aserto y, en cambio, tienen distinta longitud: P tiene longitud 3 y, en cambio, Q tiene longitud 4.

(i) **Un programa universal es capaz de reproducir el comportamiento de cualquier programa GOTO (es decir, todas las ejecuciones de cualquier programa P).**

Respuesta: Falso.

Justificación de la respuesta: Si $\mathbf{U_n}$ es un programa universal asociado a $n \geq 1$, entonces $\mathbf{U_n}$ tiene la "habilidad" de simular todas las computaciones de cualquier programa GOTO cuyo dato de entrada sea una tupla arbitraria pero con, exactamente, n componentes. Es decir, $\mathbf{U_n}$ es capaz de "reproducir" $P(x_1, \ldots, x_n)$, para cualquier programa P y cualquier n-tupla $(x_1, \ldots, x_n)$. Ahora bien, **no sería capaz** de reproducir, por ejemplo, ninguna computación de P con dato de entrada una $(n + 1)$-tupla $(x_1, \ldots, x_n, x_{n+1})$.

(j) **El programa universal $\mathbf{U_2}$ tiene la capacidad de simular todas las computaciones de cualquier programa GOTO con dato de entrada un par ordenado (a_1, a_2) arbitrario.**

Respuesta: Cierto.

Justificación de la respuesta: Eso es, precisamente, lo que afirma el teorema de existencia de programas universales. Es decir, como se ha indicado anteriormente, para cada número natural $n \geq 1$, el programa universal $\mathbf{U_n}$ tiene la "habilidad" de simular todas las computaciones de cualquier programa GOTO cuyo dato de entrada sea una tupla arbitraria pero con, exactamente, n componentes. Aplíquese este resultado al caso $n = 2$.

(k) **Si $\mathbf{U_2}$ es un programa universal asociado al número natural 2, entonces se verifica que $\mathbf{U_2}(1, 2, 3) = 4$.**

Respuesta: Falso.

Justificación de la respuesta: En primer lugar, observemos que, del teorema de existencia de programas universales, se deduce que $\mathbf{U_2}(1, 2, 3) = P_3(1, 2)$, siendo P_3 el programa GOTO de código 3.

Pues bien, vamos a hallar el programa P_3. Se tiene que $\#(P_3) = [\#(I_1), \ldots, \#(I_r)] - 1 = 3$. Luego, $[\#(I_1), \ldots, \#(I_r)] = 4 = 2^2 = [2]$. Así pues, $r = 1$ y $\#(I_1) = 2$. Ya se ha probado en otro ejercicio que la instrucción de código 2 es $Y \longleftarrow Y + 1$. Por tanto, el programa P_3 contiene una única instrucción, que es $Y \longleftarrow Y + 1$. Por tanto, $\mathbf{U_2}(1, 2, 3) = P_3(1, 2) = 1$; es decir, $\mathbf{U_2}(1, 2, 3) = P_3(1, 2) \neq 4$.

(l) **Sean e, e' números naturales tales que $\varphi_e^{(1)} = \varphi_{e'}^{(1)}$. Entonces se tiene que $\varphi_e^{(2)} = \varphi_{e'}^{(2)}$.**

Respuesta: Falso.

Justificación de la respuesta: Vamos a proporcionar un contraejemplo adecuado. Sea P_e el programa vacío y sea $P_{e'}$ el programa cuya única instrucción es $[A]$ IF $X_2 \neq 0$ $GOTO$ A. Entonces se verifica:

- ⋆ Para cada $x \in \mathbb{N}$ se tiene que $P_e(x) = 0$ y $P_{e'}(x) = 0$. Luego, $\varphi_e^{(1)} = \varphi_{e'}^{(1)}$.
- ⋆ $P_e(2, 3) = 0$ y $P_{e'}(2, 3) \uparrow$. Por tanto, se tiene que $\varphi_e^{(2)} \neq \varphi_{e'}^{(2)}$.
