Capítulo IV:
FUNCIONES RECURSIVAS

IV.6: ENUMERACIÓN Y PARAMETRIZACIÓN

Mario de J. Pérez Jiménez
Grupo de investigación en Computación Natural
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Lógica Matemática

Curso 2010-11

El predicado $STEP^{(n)}$

Proposición. Para cada $n \geq 1$ el predicado $(n+2)$–ario:

$\text{STEP}^{(n)}(x_1, \ldots, x_n, y, t) \quad \equiv \quad$ El programa de código y sobre $(x_1, \ldots, x_n)$, para en, a lo sumo, t pasos.

es primitivo recursivo.

Demostración: Sea y el código de un programa $P = (I_1, \ldots, I_j)$. Entonces $\#(I_j) = (y+1)_j = \langle a, \langle b, c \rangle \rangle$.

(a) Funciones asociadas a las instrucciones.

- *Etiqueta que precede a la instrucción*:
  $$\begin{aligned} label: \quad & \mathbb{N}^2 \longrightarrow \mathbb{N} \\ & (j, y) \rightarrow I((y+1)_j) \end{aligned}$$
  Es decir, $label(j, y) = a$

- *Variable de la instrucción:*
  $$\begin{aligned} var: \quad & \mathbb{N}^2 \longrightarrow \mathbb{N} \\ & (j, y) \rightarrow r(r((y+1)_j)) + 1 \end{aligned}$$
  Es decir, $var(j, y) = c + 1$

- *Formato de la instrucción:*
  $$\begin{aligned} instr: \quad & \mathbb{N}^2 \longrightarrow \mathbb{N} \\ & (j, y) \rightarrow I(r((y+1)_j)) \end{aligned}$$
  Es decir, $instr(j, y) = b$

- *Etiqueta a que hace referencia la instrucción:*
  $$\begin{aligned} label': \quad & \mathbb{N}^2 \longrightarrow \mathbb{N} \\ & (j, y) \rightarrow instr\ (j, y) \dot{-} 2 \end{aligned}$$
  Nota: Si I_j no es condicional, $label'(j, y) = 0$.

El predicado $STEP^{(n)}$ (II)

(b) Predicados reconocedores de instrucciones.

Sea $x = \langle j, s \rangle$ el código de una configuración de P: $\begin{cases} I(x) = j & \text{representa la instrucción a ejecutar} \\ r(x) = s & \text{es el código del estado} \end{cases}$

- Predicado *SKIP* (`sin efecto`):

$$SKIP(x,y) \equiv \begin{cases} instr(I(x),y) = 0 \wedge I(x) \leq Long(y+1) \\ \vee \\ instr(I(x),y) \geq 2 \wedge p_{var(I(x),y)} \nmid r(x) \end{cases}$$

- Predicado *INCR*.

$$INCR(x,y) \equiv instr(I(x),y) = 1$$

- Predicado *DECR* (con efecto).

$$DECR(x,y) \equiv \begin{cases} instr(I(x),y) = 2 \\ \wedge \\ p_{var(I(x),y)} | r(x) \end{cases}$$

- Predicado COND (con efecto).

$$COND(x,y) \equiv \begin{cases} instr(I(x),y) > 2 \wedge p_{var(I(x),y)} | r(x) \\ \wedge \\ \exists j \leq Long(y+1)(label(j,y) = label'(I(x),y)) \end{cases}$$

El predicado $STEP^{(n)}$ (III)

(c) Función que proporciona la configuración siguiente:

$$suc(x,y) = \begin{cases} \langle l(x)+1, r(x)\rangle & si\ SKIP(x,y) \\ \langle l(x)+1, r(x)\cdot p_{var(l(x),y)}\rangle & si\ INCR(x,y) \\ \langle l(x)+1, qt(r(x), p_{var(l(x),y)}\rangle & si\ DECR(x,y) \\ \langle \mu j \leq Long(y+1)[label(j,y) = label'(l(x),y)], r(x)\rangle & si\ COND(x,y) \\ \langle Long(y+1)+1, r(x)\rangle & e.c.o.c. \end{cases}$$

(d) Función de iniciación:

$$inic^{(n)}(x_1,\ldots,x_n) = \langle 1,\ \textstyle\prod_{j=1}^{n}(p_{2j})^{x_i}\rangle$$

(e) Predicado de finalización:

$$TERM(x,y) \equiv l(x) > Long(y+1)$$

(f) Función que proporciona la configuración k–ésima:

$$\begin{cases} di^{(n)}(x_1,\ldots,x_n,y,0) & = inic^{(n)}(x_1,\ldots x_n) \\ di^{(n)}(x_1,\ldots,x_n,y,k+1) & = suc(di^{(n)}(x_1,\ldots,x_n,y,k),y) \end{cases}$$

Finalmente: $STEP^{(n)}(x_1,\ldots,x_n,y,t) \equiv TERM(di^{(n)}(x_1,\ldots,x_n,y,t),y)$.

Por tanto, el predicado $(n+2)$–ario $STEP^{(n)}$ es primitivo recursivo.

Teorema de la forma normal de Kleene

Enunciado: Para cada $n \geq 1$, existe un predicado primitivo recursivo $(n+2)$–ario, $\mathcal{T}_n(x_1, \ldots, x_n, e, z)$, que verifica:

- Para cada función GOTO–computable y n-aria, f, existe $e \in \mathbb{N}$ tal que para cada $(n+1)$–tupla $(\vec{x}, e, z)$ se tiene:
  - $f(\vec{x}) \downarrow \Leftrightarrow \exists z \; \mathcal{T}_n(\vec{x}, e, z)$
  - $f(\vec{x}) = l(\mu z(\mathcal{T}_n(\vec{x}, e, z))$

Demostración: Se considera el predicado $(n+2)$–ario:
$\mathcal{T}_n(\vec{x}, e, z) \equiv$ El programa de código e con entrada $\vec{x}$ para en, a lo sumo, $r(z)$ pasos y devuelve $l(z)$
Entonces $\mathcal{T}_n(\vec{x}, e, z)$ es **PR** ya que $\mathcal{T}_n(\vec{x}, e, z) \equiv STEP^{(n)}(\vec{x}, e, r(z)) \wedge (r(di^{(n)}(\vec{x}, e, r(z))))_1 = l(z)$.

Sea f una función GOTO–computable y n-aria, P un programa GOTO tal que $[\![P]\!]^{(n)} = f$ y $e = \#(P)$.

Entonces: $f(\vec{x}) \downarrow \Leftrightarrow [\![P]\!]^{(n)}(\vec{x}) \downarrow \Leftrightarrow \exists t STEP^{(n)}(\vec{x}, e, t) \overset{(\star)}{\Leftrightarrow} \exists z \mathcal{T}_n(\vec{x}, e, z)$.

Además:

- Si $f(\vec{x}) \uparrow$ entonces $\neg \exists z \mathcal{T}_n(\vec{x}, e, z)$. Luego $\mu z(\mathcal{T}_n(\vec{x}, e, z)) \uparrow$
- Si $f(\vec{x}) \downarrow$ entonces $f(\vec{x}) = [\![P]\!]^{(n)}(\vec{x}) = l(\mu z(\mathcal{T}_n(\vec{x}, e, z)))$.

Consecuencia: Toda función GOTO–computable es recursiva.

En efecto: si f es una función n–aria y GOTO–computable, entonces existe $e \in \mathbb{N}$ tal que $f(\vec{x}) = l(\mu z(\mathcal{T}_n(\vec{x}, e, z)))$.

Enumeración efectiva de las funciones recursivas

Para $n \geq 1$ y $e \in \mathbb{N}$, se define la función recursiva n-aria de índice e, $\varphi_e^{(n)}$, así:

$$\varphi_e^{(n)}(\vec{x}) = \mathcal{U}_n(\vec{x}, e)$$

Consideraciones inmediatas:

- $\varphi_{\#(p)}^{(n)} = [\![p]\!]^{(n)}$.
- $\mathcal{U}_n(\vec{x}, e) = I(\mu z \mathcal{T}_n(\vec{x}, e, z))$.
- Si $k > n$, $\varphi_e^{(n)}(x_1, ..., x_n) = \varphi_e^{(k)}(x_1, ..., x_n, 0, ..., 0)$.

Proposición: Sea $n \geq 1$. La sucesión $\{\varphi_e^{(n)} : e \in \mathbb{N}\}$ es una enumeración efectiva de las funciones recursivas n–arias. Es decir:

1) $\forall e \in \mathbb{N},\ \varphi_e^{(n)} \in \mathcal{P}$
2) Si $f \in \mathcal{P}^{(n)}$, existe $e \in \mathbb{N}$, $f = \varphi_e^{(n)}(\vec{x})$.
3) Existe $F_n \in \mathcal{P}^{(n+1)}$ tal que $(F_n(\vec{x}, e) = \varphi_e^{(n)}(\vec{x}))$, para cada $\vec{x} \in \mathbb{N}^n, e \in \mathbb{N}$.

Basta tener presente que $\varphi_e^{(n)}(\vec{x}) = \mathcal{U}_n(\vec{x}, e)$ y que $\{\mathcal{U}_n(\vec{x}, k) : k \in \mathbb{N}\}$ constituye una enumeración de las funciones n-arias G-computables. En cuanto a 3) tómese $F_n = \mathcal{U}_n$

El teorema s-n-m

Enunciado: Para cada $n, m \geq$ existe una función $(n+1)$–aria primitiva recursiva e inyectiva, S_m^n, que verifica:

$$\forall n \in \mathbb{N}\ \forall m \in \mathbb{N}\ \forall e \in \mathbb{N}\ \forall \vec{x} \in \mathbb{N}^m\ \forall \vec{a} \in \mathbb{N}^n\ (\varphi^{(m)}_{S_m^n(e,\vec{a})}(\vec{x}) = \varphi^{(m+n)}_e(\vec{x}, \vec{a}))$$

Se prueba por inducción sobre n.

- Caso base: $n = 1$.
  Sean $m, e, a \in \mathbb{N}$ tales que $m \geq 1$. Sea P el programa de código e y sea e' el código del programa

$$\begin{cases} X_{m+1} \leftarrow X_{m+1} + 1 \\ \overbrace{\cdots\cdots\cdots}^{(a \text{ veces})} \\ X_{m+1} \leftarrow X_{m+1} + 1 \\ P \end{cases}$$

  Entonces

$$\varphi^{(m)}_{S_m^1(e,a)}(x_1, \ldots, x_m) = \varphi^{(m)}_{e'}(x_1, \ldots, x_m) = \varphi^{(m+1)}_e(x_1, \ldots, x_m, a)$$

- Paso inductivo: supongamos cierto el resultado para n.
  Suponiendo la existencia de S_m^n verificando las condiciones del enunciado, se define

$$S_m^{n+1}(e, a_1, \ldots, a_n, a_{n+1}) = S_m^n(S_{m+n}^1(e, a_{n+1}), a_1, \ldots, a_n))$$

Corolario: Existe una función binaria primitiva recursiva, g, tal que:

$$\forall e_1, e_2 \in \mathbb{N} \ \ (\ \varphi^{(1)}_{e_1} \circ \varphi^{(1)}_{e_2} = \varphi^{(1)}_{g(e_1,e_2)} \ )$$

Basta considerar la función recursiva $h(z, e_1, e_2) = \varphi^{(1)}_{e_1}(\varphi^{(1)}_{e_2}(z))$ y aplicar el teorema $s - n - m$ para $n = 2$ y $m = 1$, a la función h.

Sea $e \in \mathbb{N}$ tal que $\varphi^{(3)}_e = h$. Definamos $g(u, v) = S^2_1(e, u, v)$. Entonces

$$\varphi^{(1)}_{g(e_1,e_2)}(z) = \quad \varphi^{(1)}_{S^2_1(e,e_1,e_2)}(z) \overset{s-n-m}{=} \varphi^{(3)}_e(z, e_1, e_2) =$$

$$h(z, e_1, e_2) = \varphi^{(1)}_{e_1}(\varphi^{(1)}_{e_2}(z)) = (\varphi^{(1)}_{e_1} \circ \varphi^{(1)}_{e_2})(z)$$