Tema 3: Funciones recursivas

Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Ciencias de la Computación
(4º curso, Grado en Matemáticas)
Curso 2021–22

Contenido

Introducción

Funciones primitivas recursivas

Relajación de los procedimientos de definición

Ejemplos de funciones p.r.

$\mathcal{PR}$ como modelo de computación

La función de Ackermann

Predicados primitivos recursivos

Cuantificación acotada

Codificación de sucesiones finitas

Número de Gödel

Funciones recursivas

μ-recursión

Apéndice: Conjuntos y predicados

Cuantificación

Cuestión

- ¿Cómo caracterizar la clase de las funciones computables *independientemente del modelo de computación*?
  - Es decir, ¿cómo probar que una función es computable sin explicitar el programa `GOTO` que la calcula?

¿Cómo se caracteriza una clase de funciones $\mathcal{C}$?

- **Orientación descriptiva:** $\mathcal{C}$ se define como la clase de funciones que satisface(n) ciertas propiedad(es).
  Ejemplos:
  - La clase de funciones contínuas sobre $\mathbb{R}$
  - La definición de la clase `GCOMP`.
- **Orientación generativa:** Se muestran dos elementos:
  - Un conjunto de *funciones básicas*.
  - Una serie de procedimientos de definición de funciones, a partir de otras.
  - La clase $\mathcal{C}$ **se define** como la menor clase de funciones que contiene a las básicas, y es cerrada bajo los procedimientos de definición.
- **Ejemplo:** Las funciones polinomiales sobre $\mathbb{Z}$:

  $$\mathbf{Pol}_{\mathbb{Z}} = \{f : \mathbb{Z} \to \mathbb{Z} : \quad \text{Existe } p(X) \in \mathbb{Z}[X],\ \forall a \in \mathbb{Z},\ f(a) = p(a)\}$$

  $\mathbf{Pol}_{\mathbb{Z}}$ es la menor clase de funciones que
  - Contiene las funciones constantes y la identidad.
  - Si $f, g \in \mathbf{Pol}_{\mathbb{Z}}$, entonces $f + g, f \cdot g \in \mathbf{Pol}_{\mathbb{Z}}$.

Ventaja de la segunda aproximación

- Disponemos de un método demostración **por inducción** en la clase $\mathcal{C}$:

  Supongamos que Φ *es una propiedad acerca de funciones tal que*

  - **Caso base:** *Toda función básica de la clase* $\mathcal{C}$ *satisface la propiedad* Φ.
  - **Paso de inducción:** *Si una función se obtiene aplicando a funciones que satisfacen la propiedad* Φ *(la* hipótesis de inducción*) alguno de los procedimientos de definición de la clase, entonces la nueva función también satisface* Φ.

  Entonces toda función de la clase $\mathcal{C}$ *verifica* Φ.

Cuestión

- ¿Es posible definir la clase GCOMP de forma generativa, sin hacer mención explícita al lenguaje GOTO?

  La respuesta es afirmativa
- Antes estudiaremos una clase muy importante de funciones, las **primitivas recursivas**.
  - Estas funciones son GOTO–computables.
  - Proporcionaremos una definición *generativa*.
  - Juegan un papel muy importante en la **Teoría de la demostración**.

Funciones básicas

- La función **sucesor**:

$$\mathcal{S} : \mathbb{N} \to \mathbb{N}$$

$$\mathcal{S}(x) = x + 1$$

- La función **identicamente nula**:

$$\mathcal{O} : \mathbb{N} \to \mathbb{N}$$

$$\mathcal{O}(x) = 0$$

- Las **proyecciones**: Para cada $i, n \in \mathbb{N}$ $(1 \leq i \leq n)$,

$$\Pi_i^n : \mathbb{N}^n \to \mathbb{N}$$

$$\Pi_i^n(\vec{x}) = x_i$$

Procedimientos de definición

1. Sean $g : \mathbb{N}^n \to \mathbb{N}$ y $h_1, \ldots, h_n : \mathbb{N}^m \to \mathbb{N}$. Diremos que $f : \mathbb{N}^m \to \mathbb{N}$ se obtiene de g y $h_1, \ldots, h_n$ **por composición** (notación: $f = \mathcal{C}(g; h_1, \ldots, h_n)$) si $\forall \vec{y} \in \mathbb{N}^m$
$$f(\vec{y}) = g(h_1(\vec{y}), \ldots, h_n(\vec{y})).$$
2. Dadas $g : \mathbb{N}^m \longrightarrow \mathbb{N}$, $h : \mathbb{N}^{m+2} \longrightarrow \mathbb{N}$, decimos que $f : \mathbb{N}^{m+1} \longrightarrow \mathbb{N}$ se define por **recursión primitiva** a partir de g y h (y escribimos $f = \mathcal{R}(g, h)$), si

$$\text{Para todo } \vec{x} \in \mathbb{N}^m, y \in \mathbb{N} \quad \begin{cases} f(\vec{x}, 0) = g(\vec{x}) \\ f(\vec{x}, y+1) = h(\vec{x}, y, f(\vec{x}, y)) \end{cases}$$

Observaciones

- Existe una única función f tal que $f = \mathcal{R}(g, h)$.

  Si g, h son totales, f también.
- Esta definición se extiende al caso $m = 0$:

$$\begin{cases} f(0) = k \\ f(x+1) = h(x, f(x)) \end{cases}$$

donde $k \in \mathbb{N}$. En este caso escribimos $f = \mathcal{R}(k, h)$.

Definición

- La clase de las **funciones primitivas recursivas** (que denotaremos por $\mathcal{PR}$) es la **menor clase de funciones** que contiene a las funciones básicas y es cerrada bajo composición y recursión primitiva .
- **Notación:** Usamos $\mathcal{PR}^{(n)}$ para denotar al conjunto de funciones primitivas recursivas n–arias.
- Toda función $\mathcal{PR}$ es total (prueba por inducción en la clase).

Una caracterización de $\mathcal{PR}$

- **Proposición:** Una función f es primitiva recursiva si y sólo si existe una sucesión de funciones

$$g_1, \ldots, g_n$$

tal que:

  - g_n=f, y
  - para cada $i \leq n$, g_i es una función básica o se obtiene por composición, o por recursión primitiva, utilizando algunas de las funciones anteriores $g_1, \ldots, g_{i-1}$.

Idea de la prueba:

  - Probar que la clase así definida contiene a las básicas y es cerrada bajo los procedimientos de definición
  - Como $\mathcal{PR}$ es la menor clase verificando lo anterior, está contenida y por tanto verifica la condición
  - Recíprocamente: toda función definida a partir de ese tipo de sucesiones es p.r.

Ejemplo 1

Supongamos que
$f : \mathbb{N}^3 \to \mathbb{N}, \quad g_1 : \mathbb{N}^2 \to \mathbb{N}, \quad g_2 : \mathbb{N}^3 \to \mathbb{N}, \quad g_3 : \mathbb{N} \to \mathbb{N}$
son funciones p.r. Veamos que $h : \mathbb{N}^2 \to \mathbb{N}$

$$h(x, y) = f(g_1(y, x), g_2(x, x, y), g_3(x))$$

es p.r.
Es necesario buscar una definición que use los esquemas de definición de $\mathcal{PR}$, y funciones primitivas recursivas:

$$\begin{aligned} h(x,y) &= f(g_1(\Pi_2^2(x,y), \Pi_1^2(x,y)), g_2(\Pi_1^2(x,y), \Pi_1^2(x,y), \Pi_2^2(x,y)), g_3(\Pi_1^2(x,y))) \\ &= f(\mathcal{C}(g_1; \Pi_2^2, \Pi_1^2)(x,y), \mathcal{C}(g_2; \Pi_1^2, \Pi_1^2, \Pi_2^2)(x,y), \mathcal{C}(g_3; \Pi_1^2)(x,y)) \\ &= \mathcal{C}(f; \mathcal{C}(g_1; \Pi_2^2, \Pi_1^2), \mathcal{C}(g_2; \Pi_1^2, \Pi_1^2, \Pi_2^2), \mathcal{C}(g_3; \Pi_1^2))(x,y) \end{aligned}$$

Es decir, $h = \mathcal{C}(f; \mathcal{C}(g_1; \Pi_2^2, \Pi_1^2), \mathcal{C}(g_2; \Pi_1^2, \Pi_1^2, \Pi_2^2), \mathcal{C}(g_3; \Pi_1^2))$
y por tanto $h \in \mathcal{PR}$.

Ejemplo 2:

Sean $g : \mathbb{N} \to \mathbb{N}$, $h : \mathbb{N}^2 \to \mathbb{N}$ funciones p.r. Veamos que es p.r. $f : \mathbb{N}^2 \to \mathbb{N}$

$$\begin{cases} f(0, y) = g(y) \\ f(x+1, y) = h(f(x, y), y) \end{cases}$$

Es p.r.

Expresión:

Ejemplos (ejercicios):

- La función identidad, $Id_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ es primitiva recursiva, pues $Id_{\mathbb{N}} = \Pi^1_1$.
- Las funciones constantes: dado $a \in \mathbb{N}$ y $n \geq 1$, se define

$$C^n_a : \mathbb{N}^n \to \mathbb{N}$$

$$C^n_a(\vec{x}) = a$$

- La función predecesor: $pr(x) = \begin{cases} 0 & \text{si } x = 0 \\ x - 1 & \text{si } x > 0 \end{cases}$

  Se define por recursión:

$$\begin{cases} pr(0) = 0 \\ pr(x+1) = x \end{cases}$$

  Expresión de *pr*:

Ejemplos (ejercicios) (II)

- La diferencia reducida:

$$x \dot{-} y \;=\; \begin{cases} 0 & \text{si } x \leq y \\ x - y & \text{en caso contrario} \end{cases}$$

- Funciones signo de Kleene:

$$\mathrm{sg}(x) \;=\; \begin{cases} 0 & \text{si } x = 0 \\ 1 & \text{si } x \neq 0 \end{cases} \qquad \text{y}$$

$$\overline{\mathrm{sg}}(x) \;=\; \begin{cases} 1 & \text{si } x = 0 \\ 0 & \text{si } x \neq 0 \end{cases}$$

Ejemplos (ejercicios) (III)

- La función $(x, y) \mapsto |x - y|$.
- Las funciones suma y producto.
- La exponencial

$$(x, y) \mapsto \begin{cases} 0 & x = 0 \\ x^y & x \neq 0 \end{cases}$$

La definición primitiva recursiva de la exponencial debe de ser total, de ahí su definición.

Ejemplos (ejercicios) (IV)

- Si f y g son funciones primitivas recursivas, entonces $f + g$, $f \cdot g$ también lo son. Y reiterando el proceso cualquier suma finita de funciones primitivas recursivas.
- La función característica de un conjunto finito es primitiva recursiva, pues si A es finito

$$C_A(x) = \sum_{n \in A} C_{\{n\}}$$

- La función paridad, $\equiv_2$

$$x \mapsto \begin{cases} 0 & \text{si } x \text{ es par} \\ 1 & \text{si } x \text{ es impar} \end{cases}$$

$\mathcal{PR}$ como modelo de computación

- Los datos, $\mathbb{D}_{\mathtt{PR}}$ son como en `GOTO`.
- Los PR–programas son expresiones η construidas con los símbolos que denotan a las funciones básicas y procedimientos de definición permitidos en la clase $\mathcal{PR}$, junto con la aridad correspondiente.
- La función semántica $[\![\cdot]\!]^{\mathtt{PR}}$ se define como

$$[\![\eta]\!]^{\mathtt{PR}}(\vec{x}) = \eta(\vec{x})$$

- **Interpretación declarativa** (árbol de computación)

GOTO simula $\mathcal{PR}$

Teorema: Toda función primitiva recursiva es G–computable.

- La prueba es por inducción en la clase
- Caso base: Las funciones básicas son GOTO computables
- Paso de inducción (I): GCOMP es cerrado bajo composición: el siguiente programa calcula $f = \mathcal{C}(g; h_1, ..., h_n)$:

$$\begin{aligned} Z_1 &\longleftarrow h_1(X_1, ..., X_m) \\ &\vdots \\ Z_n &\longleftarrow h_n(X_1, ..., X_m) \\ Y &\longleftarrow g(Z_1, ..., Z_n) \end{aligned}$$

GOTO simula $\mathcal{PR}$ (II)

- Paso de inducción (II) GCOMP es cerrado bajo recursión primitiva: El siguiente programa calcula $f = \mathcal{R}(g, h)$:

$$
\begin{array}{ll}
 & Y \longleftarrow g(X_1, ..., X_m) \\
[A] & IF\ X_{m+1} = 0\ GOTO\ E \\
 & Y \longleftarrow h(X_1, ..., X_m, Z, Y) \\
 & Z \longleftarrow Z + 1 \\
 & X_{m+1} \longleftarrow X_{m+1} - 1 \\
 & GOTO\ A
\end{array}
$$

- Como la clase $\mathcal{PR}$ es la menor verificando lo anterior,

$$\mathcal{PR} \subseteq \texttt{GCOMP}$$

NOTA: Los modelos $\mathcal{PR}$ y GOTO no son equivalentes, pues en $\mathcal{PR}$ no existen funciones parciales

¿Es $\mathcal{PR}$ la clase de las G-computables totales?

- NO, la **función de Ackermann** es G computable, total y no p.r. La función $A(n,x) = A_n(x)$ es

$$\begin{cases} A_0(x) = x+1 \\ \\ A_{n+1}(x) = (A_n \circ \overset{(x+1)}{\cdots} \circ A_n)(1) \end{cases}$$

- Definición alternativa:

$$A(m,n) = \begin{cases} n+1 & \text{si m=0} \\ A(m-1,1) & \text{si } m>0, n=0 \\ A(m-1,A(m,n-1)) & \text{si } m>0, n>0 \end{cases}$$

  - La función de Ackermann es G-computable (lo probaremos más adelante)
  - $A(4,2)$ tiene más de 19.000 dígitos

`http://mathworld.wolfram.com/AckermannFunction.html`

La función de Ackerman no es p.r.

- **Proposición:** Para toda $f \in \mathcal{PR}$, $f : \mathbb{N}^k \to \mathbb{N}$, existe $n \in \mathbb{N}$ tal que

$$f(x_1, \ldots, x_k) \leq A_n(\text{máx}\{x_1, \ldots, x_k\})$$

(prueba por inducción en $\mathcal{PR}$)

- **Teorema:** La función de Ackermann **no** es p.r.

Demostración: Supongamos que lo fuese. Consideremos $f(x) = A(x, x) + 1$. Es evidente $f \in \mathcal{PR}$, pues

$$f = \mathcal{C}(\mathcal{S}; \mathcal{C}(A; \Pi^1_1, \Pi^1_1))$$

Por la prop. anterior existiría n_0 t.q. para todo $x \in \mathbb{N}$,

$$f(x) \leq A_{n_0}(x)$$

Evaluando la desigualdad en $x = n_0$ se llega a contradicción

A pesar de eso...

$\mathcal{PR}$ es una clase útil:

- Sólo contiene funciones totales.
- En la práctica, las funciones G-computables totales que no son p.r. son poco frecuentes.
- Podemos utilizar la clase $\mathcal{PR}$ para probar que ciertas funciones son G–computables, sin explicitar programas que las calculen: basta probar que son primitivas recursivas.
- Podemos caracterizar $\mathcal{PR}$ mediante un lenguaje de programación basado en bucles de tipo *FOR*:

Se verifica que $\mathcal{PR} = \texttt{LOOP-COMP}$

Predicados p.r.

- Diremos que un predicado $P : \mathbb{N}^k \to \{0, 1\}$ es primitivo recursivo si $P \in \mathcal{PR}$ (considerado como función $P : \mathbb{N}^k \to \mathbb{N}$).
- Un conjunto $A \subseteq \mathbb{N}^k$ se dice primitivo recursivo si la función característica de A, C_A, es primitiva recursiva.
  - Por tanto, A es p.r. sii el predicado $P(\vec{x}) \equiv \vec{x} \in A$ es p.r.

Ejemplos

- El predicado de igualdad, $P(x, y) \equiv x = y$, es p.r.

$$d(x, y) = \overline{\text{sg}}(|x - y|)$$

- El predicado $P(x, y) \equiv x \leq y$ es primitivo recursivo

$$f(x, y) = \overline{\text{sg}}(x \dot{-} y)$$

y por tanto $P = \mathcal{C}(\overline{\text{sg}}; \dot{-})$

- Análogamente, los predicado $<$ y $=$ son p.r.
- Dada $f \in \mathcal{PR}$ y $* \in \{=, \leq, \geq, <, >\}$, el predicado

$$P(\vec{x}, y) \equiv f(\vec{x}) * y$$

es p.r.

Ejemplos

- Si $P(\vec{x})$ y $Q(\vec{x})$ son n-arios p.r., entonces son p.r.

$$P \wedge Q, \qquad P \vee Q, \text{ y } \neg P$$

  - $\neg P(\vec{x}) = \overline{\text{sg}}(P(\vec{x}))$,
  - $P(\vec{x}) \wedge Q(\vec{x}) = P(\vec{x}) \cdot Q(\vec{x})$,
  - $P(\vec{x}) \vee Q(\vec{x}) = \neg(\neg P(\vec{x}) \wedge \neg Q(\vec{x}))$.

- Si $A, B \subseteq \mathbb{N}^n$ son p.r., entonces $\mathbb{N}^n \setminus A$, $A \cap B$ y $A \cup B$ son p.r.

Definición por casos

- Si $P, g, h : \mathbb{N}^k \to \mathbb{N}$ son p.r. (P es un predicado), entonces es p.r.

$$f(\vec{x}) = \begin{cases} g(\vec{x}) & \text{si } P(\vec{x}) \\ h(\vec{x}) & \text{si } \neg P(\vec{x}) \end{cases}$$

- Sea $\{A_1, \ldots A_k\}$ una partición de $\mathbb{N}^n$ por conjuntos primitivos recursivos, y $f_1, \ldots, f_k : \mathbb{N}^n \to \mathbb{N}$ funciones primitivas recursivas. Entonces es p.r.

$$g(\vec{x}) = \begin{cases} f_1(\vec{x}) & \text{si } \vec{x} \in A_1 \\ \vdots & \vdots \\ f_k(\vec{x}) & \text{si } \vec{x} \in A_k \end{cases}$$

Grafos de funciones p.r.

- $f : \mathbb{N}^k \dashrightarrow \mathbb{N}$ el grafo es

$$G(f) = \{(\vec{x}, y) \mid f(\vec{x}) = y\} \subseteq \mathbb{N}^{k+1}$$

- Si f es p.r., entonces $G(f)$ es un conjunto p.r.

$$C_{G(f)}(\vec{x}, y) = \overline{\mathrm{sg}}(|f(\vec{x}) - y|)$$

- **Cuestión**: ¿Y el recíproco?

Suma, producto y cuantificación acotada

- Sea $f : \mathbb{N}^{n+1} \to \mathbb{N}$. Se definen

$$\Sigma_f(\vec{x}, y) = \sum_{j \leq y} f(\vec{x}, j)$$

$$\Pi_f(\vec{x}, y) = \prod_{j \leq y} f(\vec{x}, j)$$

- Si $f \in \mathcal{PR}$ entonces $\Sigma_f, \Pi_f \in \mathcal{PR}$

$$\begin{cases} \Sigma_f(\vec{x}, 0) = f(\vec{x}, 0) \\ \Sigma_f(\vec{x}, y+1) = f(\vec{x}, y+1) + \Sigma_f(\vec{x}, y) \end{cases}$$

- La suma no tiene porqué comenzar en 0.
- Para el producto es análogo.

Cuantificación acotada

- Sea $A \subseteq \mathbb{N}^{n+1}$
  - $(\exists y)_{\leq z} A = \{(z, \vec{x}) \in \mathbb{N}^{n+1} \ : \ \exists y \leq z[(y, \vec{x}) \in A]\}$.
  - $(\forall y)_{\leq z} A = \{(z, \vec{x}) \in \mathbb{N}^{n+1} \ : \ \forall y \leq z[(y, \vec{x}) \in A]\}$.
- Se definen de manera análoga para los predicados, y con $<$.
- si A es p.r., entonces $(\exists y)_{\leq z} A$ y $(\forall y)_{\leq z} A$ son p.r.

  Consideremos $P(y, \vec{x}) \equiv (y, \vec{x}) \in A$
- $(\forall y)_{\leq z} P(y, \vec{x}) \Longleftrightarrow \prod_{y \leq z} P(y, \vec{x}) = 1$, luego

$$C_{(\forall y)_{\leq z} A}(z, \vec{x}) = \Pi P(z, \vec{x}) = \Pi C_A(z, \vec{x})$$

- $(\exists y)_{\leq z} P(y, \vec{x}) \Longleftrightarrow \sum_{y \leq z} P(y, \vec{x}) \neq 0$, luego

$$C_{(\exists y)_{\leq z} A}(z, \vec{x}) = \mathrm{sg}(\Sigma P(z, \vec{x})) = \mathrm{sg}(\Sigma C_A(z, \vec{x}))$$

Ejemplos

- El predicado de divisibilidad es p.r. En efecto, basta comprobar que

$$x|y \iff (\exists t)_{\leq x}(x \cdot t = y)$$

es decir

$$x|y \iff (\exists t)_{\leq x}[(x, t, y) \in G(\cdot)]$$

Como el producto es p.r., su grafo es un conjunto p.r.

- El predicado *ser número primo*, que denotaremos por $Primo(x)$, es p.r. En efecto:

$$Primo(x) \equiv \underbrace{x > 1 \wedge \underbrace{(\forall t)_{\leq x} \underbrace{[t = 1 \vee t = x \vee \neg(t|x)]}}}$$

Minimización acotada

Sea $f : \mathbb{N}^{n+1} \to \mathbb{N}$. Se define la función $f_\mu : \mathbb{N}^{m+1} \to \mathbb{N}$

$$f_\mu(\vec{x}, z) = \begin{cases} \text{mín}\{y \leq z \ : \ f(\vec{x}, y) = 0\} & \text{si existe tal mínimo} \\ z + 1 & \text{en otro caso} \end{cases}$$

También se escribe $\boxed{(\mu y)_{\leq z}(f(\vec{x}, y) = 0)}$, **teniendo en cuenta** que los argumentos son $(\vec{x}, z)$.

- Si $f \in \mathcal{PR}$ entonces $f_\mu \in \mathcal{PR}$, pues $f_\mu = \Sigma_{\mathcal{C}(\text{sg};\Pi_f)}$

  Si existe $\text{mín}\{y \leq z \ : \ f(\vec{x}, y) = 0\} = y_0$. Entonces

  $$(*) \qquad \Pi_f(\vec{x}, j) = \Pi_{k \leq j} f(\vec{x}, j) = \begin{cases} 0 & j \geq y_0 \\ \neq 0 & j < y_0 \end{cases}$$

  luego

  $\Sigma_{\mathcal{C}(\text{sg};\Pi_f)}(\vec{x}, y) = \Sigma_{j \leq z} \text{sg}(\Pi_f(\vec{x}, j)) \stackrel{*}{=} \Sigma_{j \leq y_0 - 1} 1 = y_0$

  Si no existe tal mínimo. Entonces, utilizando $*$,

  $$\Sigma_{j \leq z} \text{sg}(\Pi_f(\vec{x}, j)) = \Sigma_{j \leq z} 1 = z + 1$$

Sobre predicados. Ejemplos

Si $P(\vec{x}, y)$ es p.r., entonces la función

$$(\mu y)_{\leq z} P(\vec{x}, y) = \begin{cases} \text{mín}\{y \leq z \ : \ P(\vec{x}, y)\} & \text{si existe tal mínimo} \\ 0 & \text{e.o.c.} \end{cases}$$

es p.r.

- $\lfloor \frac{x}{y} \rfloor = \begin{cases} \text{cociente de la división de } x \text{ por } y & \text{si } y \neq 0 \\ 0 & \text{e.o.c.} \end{cases}$ es p.r.

  pues $\lfloor \frac{x}{y} \rfloor = (\mu t)_{\leq x}((t+1) \cdot y > x)$

- La función resto,

$$r(x, y) = \begin{cases} \text{resto de la división de } x \text{ por } y & \text{si } y \neq 0 \\ x & \text{en otro caso} \end{cases}$$

es p.r., pues $r(x, y) = x \dot{-} (y \cdot \lfloor \frac{x}{y} \rfloor)$

Ejemplos (II)

- La sucesión de los números primos,

$$\begin{cases} f(0) = 0 \\ f(n) = p_n = n\text{–ésimo primo} \quad n > 0 \end{cases}$$

  es p.r.
- **Teorema: (Euclides)**: $p_{n+1} \leq p_n! + 1$
- Por tanto,

$$\begin{cases} f(0) = 0 \\ f(n+1) = (\mu t)_{\leq f(n)!+1}[Primo(t) \wedge f(n) < t] \end{cases}$$

Ejemplos (III)

Sea $P(\vec{x}, y)$ un predicado. Se define la función

$$\mu_P(\vec{x}) = \begin{cases} \min\{y \ : \ P(\vec{x}, y)\} & \text{si tal conjunto es no vacío} \\ \uparrow & \text{e.o.c.} \end{cases}$$

- Si P tiene aridad $n+1$, entonces $\mu_P : \mathbb{N}^n - \to \mathbb{N}$.
- La clase $\mathcal{PR}$ **no** es cerrada bajo este procedimiento

  Si $P(x, y) \equiv x = 0$ (p.r.), entonces $\mu_P(1) \uparrow$
- Si $P(\vec{x}, y)$ es G–computable, entonces $\mu_P \in \mathcal{PR}$.

  $\mu_P = [\![\mathtt{p}]\!]^{(n)}$, donde p es

$$\begin{array}{ll} [\mathtt{A}] & \mathtt{IF}\ P(\mathtt{X}_1, \ldots \mathtt{X}_n, Y)\ \mathtt{GOTO\ E} \\ & \mathtt{Y} \leftarrow \mathtt{Y} + 1 \\ & \mathtt{GOTO\ A} \end{array}$$

Codificación en $\mathbb{N}^k$

Problema: *Encontrar una codificación*

$$(a_1, \ldots, a_n) \in \mathbb{N}^n \mapsto \langle a_1, \ldots a_n \rangle \in \mathbb{N}$$

tal que

1. *Sea inyectiva (un código sólo codifica una sucesión).*
2. *Sea primitiva recursiva.*
3. *Dado $x \in N$ en la imagen de tal función, e $i \leq n$, la función*

   $$x = \langle a_1, \ldots, a_n \rangle \in \mathbb{N} \mapsto a_i \in \mathbb{N}$$

   sea primitiva recursiva.

La codificación de $\mathbb{N}^k$

- La **función par** $\langle . \rangle : \mathbb{N}^2 \to \mathbb{N}$ def. por $\langle x, y \rangle = 2^x(2y+1) \dot{-} 1$ es una **biyección** p.r.
- Existe una función inversa de $\langle . \rangle$

$$\begin{aligned} \mathbb{N} &\to \mathbb{N}^2 \\ z &\mapsto (l(z), r(z)) \end{aligned}$$

$$l(z) = x \iff \text{existe } y \in \mathbb{N} \text{ tal que } \langle x, y \rangle = z$$

$$r(z) = y \iff \text{existe } x \in \mathbb{N} \text{ tal que } \langle x, y \rangle = z$$

- $l(\langle x, y \rangle) = x$, y $r(\langle x, y \rangle) = y$
- $\langle . \rangle \in \mathcal{PR}^{(2)}$, y $l, r \in \mathcal{PR}^{(1)}$
- Fijado $n \geq 2$, componiendo $\langle . \rangle$ podemos codificar n–uplas (de manera biyectiva). Por ej. $n = 3$:

$$(x, y, z) \mapsto \langle x, \langle y, z \rangle \rangle$$

Caso general: longitud arbitraria

$$\mathbb{N}^* = \{(a_1, \cdots, a_n) \ : \ n \in N, \ a_1, \ldots, a_n \in \mathbb{N}\}$$

- Sea $(a_1, \ldots, a_n) \in \mathbb{N}^*$. El **número de Gödel** de $(a_1, \ldots a_n)$ es

$$[a_1, \ldots a_n] = \prod_{i=1}^{n} p_i^{a_i}$$

  donde $\{p_i \ : \ i \in \mathbb{N}\}$ es la sucesión de números primos

  $[.] : \mathbb{N}^* \to \mathbb{N}$ está bien definida
- **Proposición:** Consideremos $[.] \restriction_{\mathbb{N}^n} : \mathbb{N}^n \to \mathbb{N}$. Se verifica que
  - $[.] \restriction_{\mathbb{N}^n} \in \mathcal{PR}^{(n)}$.
  - $[.] \restriction_{\mathbb{N}^n}$ es inyectiva.

Propiedades del número de Gödel

1. La función $[.] : \mathbb{N}^* \to \mathbb{N}$ **no** es inyectiva, pues

$$[a_1, \ldots a_n] = [a_1, \ldots a_n, 0] = [a_1, \ldots a_n, 0, 0] = \cdots$$

2. **Sí** es sobreyectiva sobre $\mathbb{N} \setminus \{0\}$
3. Todo número x codifica infinitas sucesiones finitas, pero todas estas sucesiones son nulas a partir de un lugar (relacionado con el mayor primo que divide a x).
4. Sin embargo, $[0, a_1, \ldots a_n] \neq [a_1, \ldots a_n]$, en general.
5. El número 1 es el número de Gödel de la sucesión vacía (por la definición que hemos tomado del producto).

Trabajando con codificaciones

- La **función componente:**

$$(.)_. : \mathbb{N}^2 \to \mathbb{N}$$
$$(x, i) \mapsto (x)_i = a_i$$

donde $(x)_i = a_i \iff x = [a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots a_n]$

Es p.r., pues $(x)_i = (\mu t)_{\leq x}[\neg(p_i^{t+1}|x)]$

Se verifica que $(x)_0 = 0$ y $(0)_i = 0$ para todo i.

- La **función longitud** (términos no nulos):

$$\mathrm{Lt} : \mathbb{N} \to \mathbb{N}$$
$$\mathrm{Lt}(x) = (\mu i)_x \left[(x)_i \neq 0 \wedge (\forall i)_{\leq x}(j \leq i \vee (x)_j = 0)\right]$$

Por ejemplo, 360 codifica la sucesión $[3, 2, 1]$, pues $360 = 2^3 \cdot 3^2 \cdot 5^1$ y, por tanto
$\mathrm{Lt}(360) = 3,\ (360)_1 = 3,\ (360)_2 = 2,$ y $(360)_3 = 1$

Propiedades de Lt

1. Si $x > 1$ y $\mathrm{Lt}(x) = n$, entonces ningún primo mayor que p_n divide a x.
2. $\mathrm{Lt}([a_1, \ldots, a_n]) = n \Longleftrightarrow a_n \neq 0$.
3. $([a_1, \ldots a_n])_i = \begin{cases} a_i & \text{si } 1 \leq i \leq n \\ 0 & e.o.c. \end{cases}$
4. Si $\mathrm{Lt}(x) \leq n$, entonces $x = [(x)_1, \ldots, (x)_n]$.

μ-recursión

- $\mathcal{PR} \subsetneq \mathtt{GCOMP}$, ni siquiera contiene a todas las computables totales
- Introducimos un nuevo procedimiento de definición, para obtener **todas** las computables
- Sea $f : \mathbb{N}^{n+1} \dashrightarrow \mathbb{N}$. La función obtenida de f por μ-recursión es:

$$(\mu y) f(\vec{x}, y) \simeq \begin{cases} \inf(\{y : \ f(\vec{x}, y) = 0 \ \wedge \ \forall z < y (f(\vec{x}, z) \downarrow)\}) \\ \uparrow \quad \text{si no existe tal ínfimo} \end{cases}$$

- La condición

$$\forall z < y (f(\vec{x}, z) \downarrow)$$

permite calcular de manera efectiva el valor (si existe)

Observaciones

- Las funciones construidas por μ–recursión pueden ser parciales (incluso si f es total): $(\mu y)C_1^{(2)}(x, y)$
- Que $(\mu y)f(\vec{x}, y) \uparrow$ puede ser debido a:
  - No existe $y \in \mathbb{N}$ tal que $f(\vec{x}, y) = 0$.
  - Existe el mínimo y, pero existe $z < y$ tal que $f(\vec{x}, z) \uparrow$.
- Si f es **total**, la μ–recursión es **minimización** sobre

$$\{y \ : \ f(\vec{x}, y) = 0\}$$

- Relajaremos la notación, permitiendo minimización en cualquier variable y escribiremos

$$(\mu y)[f(\vec{x}, y) = 0]$$

- Es una función n–aria con argumentos son $x_1, \ldots x_n$.

Funciones recursivas

- La colección de las **funciones recursivas** (que denotaremos por $\mathcal{P}$) es la menor clase de funciones que
  - contiene a las funciones básicas y
  - es cerrada bajo composición, recursión primitiva y μ–recursión.
- Notaremos por $\mathcal{R}$ la colección de las funciones recursivas totales.
- Notaremos por $\mathcal{P}^{(k)}$ y $\mathcal{R}^{(k)}$ a las funciones k–arias recursivas y recursivas totales, respectivamente.
- De manera natural, se definen los **conjuntos recursivos** (resp. los **predicados recursivos**) como aquellos que su función característica es recursiva (resp. aquellos que, como funciones booleanas, son recursivas).
- Se verifica que

$$\mathcal{PR} \subsetneq \mathcal{R} \subsetneq \mathcal{P}$$

μ–recursión para predicados

Si $P(\vec{x}, y)$ es un predicado de aridad $n+1$, se le asocia una función n–aria aplicando μ–recursión como sigue

$$(\mu y)P(\vec{x}, y) = \begin{cases} \text{mín}\{y \ : \ P(\vec{x}, y)\} & \text{si existe} \\ \uparrow & \text{e.o.c.} \end{cases}$$

- **Lema:** $P \in \mathcal{R} \Longrightarrow (\mu y)P(\vec{x}, y) \in \mathcal{P}$

  Se verifica que

$$(\mu y)P(\vec{x}, y) = (\mu y)[|P(\vec{x}, y) - 1| = 0]$$

Propiedades de las funciones y predicados recursivos

- **Todos los procedimientos descritos para $\mathcal{PR}$ son válidos para $\mathcal{P}$**
- Sea $f : \mathbb{N}^k \to \mathbb{N}$ **total**. Entonces

$$f \in \mathcal{R} \Longleftrightarrow G(f) \text{ es recursivo}$$

Demostración: Si $f \in \mathcal{R}$, entonces la función característica del grafo es

$$C_{G(f)}(\vec{x}, y) = \overline{\text{sg}}(|f(\vec{x}) - y|)$$

Recíprocamente, si $G(f)$ es recursivo, entonces

$$f(\vec{x}) = (\mu y)(|C_{G(f)}(\vec{x}, y) - 1| = 0)$$

La clase $\mathcal{P}$ como modelo de computación

La clase $\mathcal{P}$ es susceptible de ser interpretada como un modelo de computación, que llamaremos, REC.

- La definición matemática de la μ–recursión proporciona un método de cálculo
- Supongamos, por ejemplo, que deseamos calcular

$$(\mu y)(|x - \mathcal{S}(y)|)(3)$$

La interpretación como un procedimiento: ir calculando los valores de la función sobre

$$(3, 0), (3, 1), \ldots$$

hasta encontrar un valor y tal que $|3 - \mathcal{S}(y)| = 0$.

GOTO simula REC

- **Teorema:** Toda función recursiva es G–computable.
- **Demostración**: por inducción en la clase $\mathcal{P}$

  Toda función básica es p.r., luego es computable.

  Sólo necesitamos probar que si $f : \mathbb{N}^{n+1} \dashrightarrow \mathbb{N}$ es G–computable, entonces $(\mu y) f(\vec{x}, y)$ también

$$(\mu y) f(\vec{x}, y) = [\![\mathtt{p}]\!]^{(n)}(\vec{x})$$

  donde p es el G–programa

$$\begin{array}{ll} [\mathtt{A}] & \mathtt{IF}\ f(\mathtt{X}_1, \ldots, \mathtt{X}_n, \mathtt{Y}) = 0\ \mathtt{GOTO\ E} \\ & \mathtt{Y} \leftarrow \mathtt{Y} + 1 \\ & \mathtt{GOTO\ A} \end{array}$$

  La macro $\mathtt{IF}\ f(\mathtt{X}_1, \ldots, \mathtt{X}_n, \mathtt{Y}) = 0\ \mathtt{GOTO\ E}$ estaría indefinida (su ejecución no para) si f no está definida

  Por tanto, el programa presentado realmente calcula $(\mu y) f(\vec{x}, y)$.

Apéndice: Conjuntos y Predicados

Definición: $(n \geq 1)$ Un predicado n–ario sobre A es una aplicación $\theta : A^n \to \{0, 1\}$.

- Los predicados nos permiten identificar con funciones los subconjuntos de A y, en general, las relaciones entre elementos de A.
- Dado $(x_1, \ldots, x_n) \in A^n$, si $\theta(x_1, ..., x_n) = 1$ diremos que el predicado θ se verifica (o que es cierto) para $(x_1, ..., x_n)$. Escribiremos $\theta(\vec{x})$ en vez de $\theta(\vec{x}) = 1$.
- Si $\theta(x_1, ..., x_n) = 0$ diremos que el predicado no se verifica (o que es falso) para $(x_1, ..., x_n)$.
- Podemos identificar un predicado n–ario sobre A, $\theta : A^n \to \{0, 1\}$, con un subconjunto de A^n:

$$S_\theta = \{\vec{x} \in A^n : \ \theta(\vec{x}) = 1\}$$

Predicados y conjuntos

Definición: Sea $B \subseteq A^n$. Llamaremos **función característica** del subconjunto B, al predicado C_B definido sobre A^n como sigue:

$$C_B(\vec{x}) = \begin{cases} 1 & si\ \vec{x} \in B \\ 0 & si\ \vec{x} \notin B \end{cases}$$

- La función característica de $B \subseteq A^n$, nos permite identificar el conjunto B con un predicado (precisamente, C_B).
- Si θ es un predicado n–ario sobre A y $B = S_\theta$ entonces $C_B = \theta$.

Operaciones con predicados

Definición: Dados los predicados θ y θ', definimos los predicados $\neg\theta$, $\theta \vee \theta'$, $\theta \wedge \theta'$, $\theta \to \theta'$ y $\theta \leftrightarrow \theta'$, así:

- $(\neg\theta)(\vec{x}) \equiv \neg\theta(\vec{x}) = 1 - \theta(\vec{x})$.
- $(\theta \vee \theta')(\vec{x}) \equiv \theta(\vec{x}) \vee \theta'(\vec{x}) = \max(\theta(\vec{x}), \theta'(\vec{x}))$
- $(\theta \wedge \theta')(\vec{x}) \equiv \theta(\vec{x}) \wedge \theta'(\vec{x}) = \min(\theta(\vec{x}), \theta'(\vec{x}))$.
- $\theta \to \theta' = (\neg\theta) \vee \theta'$.
- $\theta \leftrightarrow \theta' = (\theta \to \theta') \wedge (\theta' \to \theta)$.

Estas operaciones reflejan las operaciones entre conjuntos del siguiente modo. Sean θ_1 y θ_2 predicados n–arios sobre A^n. Entonces

- $S_{\theta_1 \vee \theta_2} = S_{\theta_1} \cup S_{\theta_2}$.
- $S_{\theta_1 \wedge \theta_2} = S_{\theta_1} \cap S_{\theta_2}$.
- $S_{\neg\theta_1} = A^n - S_{\theta_1}$.

Cuantificación acotada

Sea $\theta(x_1, ..., x_n, y)$ un predicado $(n+1)$–ario sobre $\mathbb{N}$.
El predicado obtenido a partir de θ por **cuantificación existencial acotada** es el predicado $(n+1)$–ario sobre $\mathbb{N}$ que denotaremos por $(\exists z)_{\leq y}\theta$ y se define mediante:

$$(\exists z)_{\leq y}\theta(\vec{x}, z) = \begin{cases} 1 & \text{si existe } z_0 \leq y \text{ tal que } \theta(\vec{x}, z_0). \\ 0 & \text{en caso contrario} \end{cases}$$

El predicado obtenido a partir de θ por **cuantificación universal acotada** es el predicado $(n+1)$–ario sobre $\mathbb{N}$ que denotaremos por $(\forall z)_{\leq y}\theta$ y se define mediante:

$$(\forall z)_{\leq y}\theta(\vec{x}, z) = \begin{cases} 1 & \text{si para todo } z_0 \leq y \text{ se tiene } \theta(\vec{x}, z_0). \\ 0 & \text{en caso contrario} \end{cases}$$

- $(\exists z)_{\leq y}\theta(\vec{x}, z) = \theta(\vec{x}, 0) \vee \theta(\vec{x}, 1) \vee ... \vee \theta(\vec{x}, y)$
- $(\forall z)_{\leq y}\ \theta(\vec{x}, z) = \theta(\vec{x}, 0) \wedge \theta(\vec{x}, 1) \wedge ... \wedge \theta(\vec{x}, y)$

Cuantificación no acotada

Sea $\theta(x_1, ..., x_n, y)$ un predicado $(n+1)$–ario sobre $\mathbb{N}$.
El predicado obtenido a partir de θ por **cuantificación existencial** es el predicado n–ario sobre $\mathbb{N}$ que denotaremos por $(\exists z)\,\theta$ y se define mediante:

$$(\exists z)\,\theta(\vec{x}, z) = \begin{cases} 1 & \text{si existe } z_0 \text{ tal que } \theta(\vec{x}, z_0). \\ 0 & \text{en caso contrario.} \end{cases}$$

El predicado obtenido a partir de θ por **cuantificación universal** es el predicado n–ario sobre $\mathbb{N}$ que denotaremos por $(\forall z)\,\theta$ y se define mediante:

$$(\forall z)\,\theta(\vec{x}, z) = \begin{cases} 1 & \text{si para todo } z_0 \text{ se tiene } \theta(\vec{x}, z_0). \\ 0 & \text{en caso contrario.} \end{cases}$$