Tema 5: Conjuntos recursivamente enumerables.

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

Ciencias de la Computación
(4º curso, Grado en Matemáticas)
Curso 2020–21

Motivación

1. ¿Es posible analizar un programa para saber *a priori* si para sobre un dato?
2. Si $f \in \mathcal{R}$, entonces su grafo es recursivo ¿Qué ocurre con el grafo de una función parcial?

Otras cuestiones:

- Es fácil diseñar un *procedimiento* que responde *sí* si el programa para, y no devuelve respuesta en otro caso.
  - Es un procedimiento de *semidecisión* (en contraposición con los procedimientos de *decisión* para predicados computables)
  - ¿Qué propiedades tienen estos *pseudopredicados*?
- Dado $\mathtt{p} \in \mathtt{GOTO_P}$ ¿Es el predicado

$$P_{\mathtt{p}}(\vec{x}) = \begin{cases} 1 & \text{si } [\![\mathtt{p}]\!]^{(n)}(\vec{x}) \downarrow \\ 0 & \text{e.o.c.} \end{cases}$$

  computable?
- ¿Es posible obtener un *programa* generador de los elementos del dominio de un programa $\mathtt{p}$?

Conjuntos recursivamente enumerables

Definición: Un conjunto $B \subseteq \mathbb{N}^n$ se dice **recursivamente enumerable** (r.e.) si existe $g \in \mathcal{P}$ tal que $B = \mathrm{dom}(g)$.

- Como $\mathtt{GCOMP} = \mathcal{P}$, existe $\mathtt{p} \in \mathtt{GOTO}_{\mathtt{P}}$ tal que

$$B = \mathrm{dom}(\llbracket \mathtt{p} \rrbracket^{(n)})$$

- Como los conjuntos r.e. están asociados a los dominios de programas, es muy útil la siguiente notación:

  Dados $e \in \mathbb{N}$ y $n \geq 1$, el **conjunto r.e. de índice** e

$$W_e^{(n)} = \{\vec{x} \in \mathbb{N} \ : \ \varphi_e^{(n)}(\vec{x}) \downarrow\} = \mathrm{dom}(\varphi_e^{(n)})$$

  (es decir, el dominio en $\mathbb{N}^n$ del programa $\mathtt{p}$ de código e).

- Luego para cada $B \subseteq \mathbb{N}^n$ r.e. existe e t.q. $W_e^{(n)} = B$.

Los recursivos son r.e.

Proposición: Todo conjunto recursivo es recursivamente enumerable.

- **Demostración**: Si $B \subseteq N^n$, $C_B \in \mathcal{R}$. Consideremos el programa $\mathtt{p}$

$$[\mathtt{A}] \quad \mathtt{IF}\ C_B(\vec{X}) = 0\ \mathtt{GOTO}\ \mathtt{A}$$

  es evidente que

$$\mathrm{dom}([\![\mathtt{p}]\!]^{(n)}) = \{\vec{x}\ :\ C_B(\vec{x}) = 1\} = B$$

  y por tanto B es r.e.

Teorema del complemento

$B \subseteq \mathbb{N}^n$ es recursivo si y sólo si B y $\mathbb{N}^n \setminus B$ son r.e.

- **Demostración**: Si B es recursivo, entonces $\mathbb{N}^n \setminus B$ también, pues $C_{\mathbb{N}^n \setminus B}(\vec{x}) = 1 \dot{-} C_B(\vec{x})$, luego son r.e.

  $\boxed{\Longleftarrow}$ Supongamos que p y q son tales que

  $$\mathrm{dom}(\llbracket \mathrm{p} \rrbracket^{(n)}) = B \quad \text{y} \quad \mathrm{dom}(\llbracket \mathrm{q} \rrbracket^{(n)}) = \mathbb{N}^n \setminus B$$

  Si $e_1 = \#(\mathrm{p})$ y $e_2 = \#(\mathrm{q})$, y consideramos p_B:

  $$\begin{array}{ll} [\mathtt{A}] & \mathtt{IF}\ STEP^{(n)}(\mathtt{X}_1, \ldots, \mathtt{X}_n, e_1, \mathtt{Z})\ \mathtt{GOTO\ C} \\ & \mathtt{IF}\ STEP^{(n)}(\mathtt{X}_1, \ldots, \mathtt{X}_n, e_2, \mathtt{Z})\ \mathtt{GOTO\ E} \\ & \mathtt{Z} \leftarrow \mathtt{Z} + 1 \\ & \mathtt{GOTO\ A} \\ [\mathtt{C}] & \mathtt{Y} \leftarrow 1 \end{array}$$

Se tiene que como $\mathrm{dom}(\llbracket \mathrm{p} \rrbracket^{(n)}) \cup \mathrm{dom}(\llbracket \mathrm{q} \rrbracket^{(n)}) = \mathbb{N}^n$, p_B siempre para, y

$$\llbracket \mathrm{p_B} \rrbracket^{(n)} = C_B$$

Unión e intersección de r.e.

Proposición $B, C \subseteq \mathbb{N}^n$ son r.e. $\Longrightarrow$ $B \cap C$ y $B \cup C$ son r.e.

- **Demostración**: Sean $g, h : \mathbb{N}^n - \to \mathbb{N}$ G–computables tales que $\mathrm{dom}(g) = B$ y $\mathrm{dom}(h) = C$. Consideremos

$$\mathrm{p}\begin{cases} \mathtt{Y} \leftarrow g(\vec{X}) \\ \mathtt{Y} \leftarrow h(\vec{X}) \end{cases}$$

Entonces

$$[\![\mathrm{p}]\!]^{(n)}(\vec{x})\downarrow \Longleftrightarrow g(\vec{x})\downarrow \text{ y } h(\vec{x})\downarrow \Longleftrightarrow \vec{x} \in B \cap C$$

Con respecto a la unión, tomemos e_1 y e_2 códigos de programas que calculan g y h, resp. Consideramos p:

$$\begin{array}{ll} [\mathtt{A}] & \mathtt{IF}\ STEP^{(n)}(\mathtt{X}_1, \ldots, \mathtt{X}_n, e_1, \mathtt{Z})\ \mathtt{GOTO\ E} \\ & \mathtt{IF}\ STEP^{(n)}(\mathtt{X}_1, \ldots, \mathtt{X}_n, e_2, \mathtt{Z})\ \mathtt{GOTO\ E} \\ & \mathtt{Z} \leftarrow \mathtt{Z} + 1 \\ & \mathtt{GOTOA} \end{array}$$

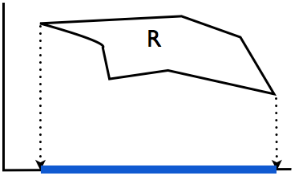
Se verifica que

$$\begin{aligned} [\![\mathrm{p}]\!]^{(n)}(\vec{x})\downarrow \quad & \Longleftrightarrow \text{Existe } z \text{ tal que } STEP^{(n)}(\vec{x}, e_1, z) \text{ ó } STEP^{(n)}(\vec{x}, e_2, z) \\ & \Longleftrightarrow g(\vec{x})\downarrow \text{ ó } h(\vec{x})\downarrow \\ & \Longleftrightarrow \vec{x} \in B \cup C \end{aligned}$$

Teorema de la proyección

Sea $B \subseteq \mathbb{N}^n$. Son equivalentes:

1. B es r.e.
2. Existe un predicado $R(\vec{x}, t)$ recursivo tal que $B = \{\vec{x} \ : \exists t R(\vec{x}, t)\}$



$B = \{\vec{x} \ : \exists t R(\vec{x}, t)\}$

Demostración: $\boxed{(1) \Longrightarrow (2)}$: Sea e tal que $B = W_e^{(n)}$. Se verifica que $B = \{\vec{x} \ : \ \exists t \ STEP^{(n)}(\vec{x}, e, t)\}$

$\boxed{(2) \Longrightarrow (1)}$: Usando $R(\vec{x}, t)$ como macro, consideremos p:

```
[A]   IF R(X_1, ..., X_n, Z) GOTO E
      Z ← Z + 1
      GOTO A
```

Se verifica que $\mathrm{dom}([\![\mathrm{p}]\!]^{(n)}) = \{\vec{x} \ : \ \exists z R(\vec{x}, z)\} = B$ (se hecho, los conjuntos r.e. son proyecciones de conjuntos p.r., pues $R(\vec{x}, t)$ es p.r.)

Teorema de enumeración

Sea $B \subseteq \mathbb{N}$ r. e. y no vacío. Entonces existe f total y recursiva tal que

$$B = \{f(0), f(1), \ldots\} = \text{rang}(f)$$

Demostración: Por el t. de la proyección, existe $R(x, z)$ p.r. tal que $\{x \ : \ \exists z R(x, z)\} = B$
Como $B \neq \emptyset$, sea $x_0 \in B$ fijo, y definamos

$$f(u) = \begin{cases} l(u) & \text{si } R(l(u), r(u)) \\ x_0 & \text{e.o.c.} \end{cases}$$

$(f \in \mathcal{PR})$. Veamos que $\text{rang}(f) = B$:

- Sea $u \in \mathbb{N}$. Si $f(u) = x_0$, entonces $f(u) \in B$. Si no, se verifica $R(l(u), r(u))$, y por tanto $f(u) = l(u) \in B$
- Sea $x \in B$. Por hipótesis, existe un $z \in N$ tal que $R(x, z)$. Luego $f(\langle x, z \rangle) = x$.

Teorema del rango

Si $f \in \mathcal{P}^{(1)}$, entonces rang(f) es r.e.

- **Demostración** $e = \#(\text{q})$ con $[\![\text{q}]\!] = f$ y $B =$rang(f)

$$g(x) = \begin{cases} 0 & \text{si } x \in B \\ \uparrow & \text{e.o.c.} \end{cases}$$

es computada por el siguiente programa:

$$\text{p} \left\{ \begin{array}{ll} [\texttt{A}] & \texttt{IF } \neg STEP^{(1)}(\texttt{Z}, e, \texttt{Z}_2) \texttt{ GOTO B} \\ & \texttt{V} \leftarrow f(\texttt{Z}) \\ & \texttt{IF V} = \texttt{X GOTO E} \\ [\texttt{B}] & \texttt{Z} \leftarrow \texttt{Z} + 1 \\ & \texttt{IF Z} \leq \texttt{Z}_2 \texttt{ GOTO A} \\ & \texttt{Z}_2 \leftarrow \texttt{Z}_2 + 1 \\ & \texttt{Z} \leftarrow 0 \\ & \texttt{GOTO A} \end{array} \right.$$

Idea: $\texttt{Z}_2$ es cota de la búsqueda de x en el rango de f, para el nº de pasos de q, y para los argumentos de f:

- Para cada valor z_2 de $\texttt{Z}_2$, generamos todos los valores $\{f(0), \ldots, f(z_2)\}$ calculables en menos de z_2 pasos de q
- Nos preguntamos si alguno es x. Si es así, devolvemos 0 (GOTO E). Si no, se incrementa z_2.

$$x \in \text{rang}(f) \iff \exists z, z_2 \in \mathbb{N}[f(z) = x \wedge STEP^{(1)}(z, e, z_2)]$$

- Luego $[\![\text{p}]\!]^{(1)} = g$ (mismo dominio y los mismos valores).

Teorema del grafo (I): primer intento

Teorema: Si f es recursiva, entonces $G(f)$ es recursivamente enumerable

Demostración: Sea $f \in \mathcal{P}$. La función $g(\vec{x}, y) = (\mu z)(|f(\vec{x}) - y| = 0)$ es

$$g(\vec{x}, y) = \begin{cases} 0 & \text{si } f(\vec{x}) = y \\ \uparrow & \text{e.o.c.} \end{cases}$$

y por tanto $\mathrm{dom}(g) = G(f)$

¿Y el recíproco? La definición

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

no lo justifica, pues ignoramos si $C_{G(f)}$ es recursiva

Teorema del grafo (II): Caracterización

Lema Un conjunto B es r.e. sii su función característica parcial, C_B^*, es recursiva.

Demostración Basta tener en cuenta que $\mathrm{dom}(C_B^*) = B$, y que si $B = \mathrm{dom}(g)$, entonces $C_B^* = \mathcal{C}_1(g(x))$

Teorema del grafo: Sea $f : \mathbb{N}^n - \to \mathbb{N}$.

$$f \Longleftrightarrow G(f) \text{ es r.e.}$$

Demostración La implicación directa ya la tenemos
Supongamos que $G(f)$ es r.e. Sea $R(\vec{x}, y, z)$ p.r. tal que

$$(\vec{x}, y) \in G(f) \Longleftrightarrow \exists z \; R(\vec{x}, y, z)$$

Entonces se verifica que $f(\vec{x}) = l((\mu u)R(\vec{x}, l(u), r(u)))$

Utilidad del teorema del grafo

Un ejemplo: Probaremos que la función

$$f(x) = \begin{cases} \varphi_x^{(1)}(x) & \text{si } \varphi_x^{(1)}(x)\downarrow \\ \uparrow & \text{e.o.c.} \end{cases}$$

es recursiva.

(Nótese que el razonamiento usual para la definición por casos no se puede usar, pues no sabemos si es recursivo el predicado $P(x) \equiv \varphi_x^{(1)}(x)\downarrow$

Sin embargo, si describimos el grafo de la función vemos que es r.e., pues es proyección de un conjunto p.r.:

$$\begin{aligned} (x,y) \in G(f) &\iff \varphi_x^{(1)}(x) = y \\ &\iff \exists z\,[\mathcal{T}_1(x,x,z) \wedge l(z) = y] \end{aligned}$$

El problema de la parada

Informalmente es

$$\begin{aligned} \mathit{Parada}(x, \mathrm{p}) &\iff \text{el programa } \mathrm{p} \text{ para sobre } x \\ &\iff [\![\mathrm{p}]\!](x)\downarrow \end{aligned}$$

Es decir,

$$\begin{aligned} \mathrm{HALT}(x, y) &\iff \text{el programa de código } y \text{ para sobre } x \\ &\iff \varphi_y(x)\downarrow \end{aligned}$$

Teorema (el problema de la parada):
$\mathrm{HALT}(x, y)$ es recursivamente enumerable y **no es recursivo.**

Demostración (I)

- El predicado es r.e.:

$$\mathrm{HALT}(x, y) \Longleftrightarrow \exists z\ \mathcal{T}_1(x, y, z)$$

  y por tanto, por el teorema de la proyección, es r.e.
- Utilizamos el **método diagonal**: si suponemos que dicho predicado es recursivo, construimos una función computable que difiere de cada función computable en algún valor, llegando a contradicción
- Supongamos pues que es computable. Tomemos la función

$$f(x) = \begin{cases} \uparrow & \text{si HALT}(x, x) \\ 0 & \text{e.o.c.} \end{cases}$$

Demostración (I)

- f es computable por

$$\mathtt{p} \equiv \quad \mathtt{[A]} \quad \mathtt{IF}\ \mathsf{HALT}(x,x)\ \mathtt{GOTO\ A}$$

  (como suponemos que HALT(x, y) es computable, la podemos usar como macro).

- Sea $e = \#(\mathtt{p})$. Como $\varphi_e = [\![\mathtt{p}]\!]^{(1)}$, entonces para todo $x \in \mathbb{N}$

$$\mathsf{HALT}(x,e) \Longleftrightarrow [\![\mathtt{p}]\!]^{(1)}(x)\downarrow \Longleftrightarrow f(x)\downarrow \Longleftrightarrow \neg\mathsf{HALT}(x,x)$$

  pero tomando $x = e$ se llega a una contradicción.

Idea del método diagonal (I)

Representando con una *tabla infinita* los valores:

	0	1	2	3	...
φ_0	$\boxed{\varphi_0(0)}$	$\varphi_0(1)$	$\varphi_0(2)$	$\varphi_0(3)$	...
φ_1	$\varphi_1(0)$	$\boxed{\varphi_1(1)}$	$\varphi_1(2)$	$\varphi_1(3)$	...
φ_2	$\varphi_2(0)$	$\varphi_2(1)$	$\boxed{\varphi_2(2)}$	$\varphi_2(3)$	...
φ_3	$\varphi_3(0)$	$\varphi_3(1)$	$\varphi_3(2)$	$\boxed{\varphi_3(3)}$	...
$\vdots$					$\boxed{\ddots}$

(escribiendo $\uparrow$ si $\varphi_i(j)\uparrow$)

Idea del método diagonal (II)

Si HALT(x, x) es computable sabemos si $\varphi_x(x)\downarrow$. Definimos la función computable que difiere de cada una en la diagonal:

	0	1	2	3	...
φ_0	$\boxed{\varphi_0(0)}$	$\varphi_0(1)$	$\varphi_0(2)$	$\varphi_0(3)$	...
φ_1	$\varphi_1(0)$	$\boxed{\varphi_1(1)}$	$\varphi_1(2)$	$\varphi_1(3)$	...
φ_2	$\varphi_2(0)$	$\varphi_2(1)$	$\boxed{\varphi_2(2)}$	$\varphi_2(3)$	...
φ_3	$\varphi_3(0)$	$\varphi_3(1)$	$\varphi_3(2)$	$\boxed{\varphi_3(3)}$	...
$\vdots$					$\boxed{\cdot}$
f	δ_0	δ_1	δ_2	δ_3	...

Idea del método diagonal (III)

Por tanto
$$f(i) = \delta_i = \begin{cases} \uparrow & \text{si } \varphi_i(i)\downarrow \\ 0 & \text{si } \varphi_i(i)\uparrow \end{cases}$$
que es computable (su grafo es r.e.), y como difiere de cada φ_i en el valor (indefinido o no) en i no podría ser computable

Notas sobre el teorema

1. Es el primer problema, íntimamente relacionado con la computación, del que hemos demostrado su indecidibilidad.
2. Hemos demostrado que **el problema de decidir si un programa para, o no, sobre un dato de entrada no es resoluble mecánicamente**:

   No existe un programa $\mathtt{p}$ *que resuelva el problema:*

   - Entrada: $\mathtt{q} \in \mathtt{GOTO}_{\mathtt{P}}$, $x \in \mathbb{N}$.
   - Salida: 1 si $[\![\mathtt{q}]\!](x) \downarrow$; 0 en otro caso

3. El método diagonal, es útil para demostrar resultados de este tipo.

Notas sobre el teorema

1. En la prueba del resultado, se ha demostrado que el conjunto
$$\mathcal{K} = \{x \ : \ \varphi_x(x) \downarrow\}$$
es **recursivamente enumerable y no recursivo**.
2. Es más, $\mathbb{N} \setminus \mathcal{K}$ **no es r.e.** (si lo fuese, al ser K y $\mathbb{N} \setminus \mathcal{K}$ r.e., por el teorema del complemento, $\mathcal{K}$ sería recursivo).