Límites de la computabilidad

No computabilidad y cómo demostrarla

Departamento de Computación
FCEyN, UBA

6 de noviembre de 2024

¿Todo es computable?

- No todas los lenguajes son computables.
- Hoy veremos técnicas para demostrar que lenguaje no computable no es computable.

Diagonalización

- Técnica para demostrar por absurdo que lenguaje no es computable (computable a secas y/o parcialmente computable).
- **Idea**: asumir que es computable y generar una función computable que falle sobre una entrada particular.

Ejemplo: HALT no es computable

Sea $\mathrm{HALT} = \{\langle \#(M), \omega \rangle \mid M(\omega) \downarrow\}$.

Teorema: HALT no es computable.

Demostrémoslo por absurdo usando diagonalización.

Asumamos que HALT es computable.

Consideramos el programa P:

while $\mathrm{HALT}(X, X) \neq 0$ **do { pass }**

Ejemplo: HALT no es computable

La función computada por P es total, ya que HALT es computable:

$$\Psi_P^{(n)} = \begin{cases} \uparrow & \text{si } \mathrm{HALT}(x,x) = 1 \\ 0 & \text{si no} \end{cases}$$

Supongamos $\#(P) = k$

¿Qué pasa con $\langle k, k \rangle$? ¿$\langle k, k \rangle \in \mathrm{HALT}$?

Ejemplo: HALT no es computable

$\langle k, k \rangle \in \mathrm{HALT}$

$\iff \Phi_k^{(1)}(k) \downarrow$ (por definición de HALT)

$\iff \mathrm{HALT}(k, k) = 1$ (por definición de función característica de HALT)

$\iff \Phi_k^{(1)}(k) \uparrow$ (por definición de P)

$\iff \mathrm{HALT}(k, k) = 0$ (por definición de función característica de HALT)

Otro ejemplo de diagonalización

Consideremos $X \subseteq \Sigma^*$,

$$X = \{\langle \#(M), \omega \rangle \mid M(\omega) \downarrow \text{y } M(\omega) = \omega^r\}$$

Lema: X no es computable.

Demostrémoslo por absurdo usando diagonalización.

Supongamos que X es computable, sea P un programa tal que:

- $P(x) \downarrow \forall x$
  (cada x codifica una tupla $\langle \#(M), \omega \rangle$)
- $P(x) = 1$ si $x \in X$ (esto es, si $M(\omega) = \omega^r$)
- $P(x) = 0$ si $x \notin X$

Otro ejemplo de diagonalización

Consideremos el programa D:

if $P(\langle X_1, X_1 \rangle) = 1$ **then** {
  Y = $(X_1 + 1)^r$
} **else** {
  Y = $(X_1)^r$
}

¿Qué pasa con $D(\#(D))$?

Otro ejemplo de diagonalización

$D(\#(D)) = (\#(D))^r$

$\iff P(\langle \#(D), \#(D) \rangle) = 0$ (por definición de D)

$\iff \langle \#(D), \#(D) \rangle \notin X$ (P es un programa que computa la función característica de X)

$\iff D(\#(D)) \neq (\#(D))^r$ (porque $D(\#D)\downarrow$)

¡Absurdo!

Reducciones

Técnica para demostrar que un lenguaje no es computable, utilizando otro lenguaje que sepamos que no es computable.

Reducciones: Sea $A, B \subseteq \Sigma^*$. Decimos que A se reduce a B sii existe una función $f : \Sigma^* \to \Sigma^*$ total computable tal que $\forall \omega \in \Sigma^*, \omega \in A \iff f(\omega) \in B$.

Notación: $A \leq B$

Lenguajes computables, c.e. y co-c.e.

Consideremos un alfabeto Σ, y un lenguaje $\mathcal{L} \subseteq \Sigma^*$. Decimos que $\mathcal{L}$ es un **lenguaje computable** si, considerando una codificación de cadenas $\Sigma^* \to \mathbb{N}$ (por ejemplo, la función ρ_Σ definida en el ejercicio 13 de la práctica 7), su predicado característico

$$p_{\mathcal{L}}(x) = \begin{cases} 1 & \text{si la cadena codificada por } x \in \mathcal{L} \\ 0 & \text{si no} \end{cases}$$

es computable.

Lenguajes computables, c.e. y co-c.e.

Si su predicado característico es parcialmente computable, decimos que $\mathcal{L}$ es un **lenguaje computablemente enumerable (c.e.)**.

Si el predicado característico del complemento del lenguaje $\mathcal{L}$, notado $\mathcal{L}^c$, es parcialmente computable, entonces decimos que $\mathcal{L}$ es **co-c.e.**

Reducciones y computabilidad

Si $A \leq B$, entonces

- B es computable $\Rightarrow$ A es computable
- B es c.e. $\Rightarrow$ A es c.e.

Corolario: Si $A \leq B$:

- A no computable $\Rightarrow$ B no es computable
- A no c.e. $\Rightarrow$ B no es c.e.

Pasos para usar reducibilidad

Para mostrar que B no es computable usando reducción:

1. Elegir un A no computable conveniente
2. Mostrar que existe una función f que efectivamente reduce A en B
3. Mostrar que dicha f es computable
3. Explicitar qué lema o corolario se usa para llegar al absurdo.

Ejemplo: TOT no es computable

Sea $\mathrm{TOT} = \{\#(M) \mid \forall \omega \in \Sigma^*, M(\omega) \downarrow\}$.

Lema: El lenguaje TOT no es computable.

Lo vamos a demostrar reduciendo HALT a TOT.

Por el corolario anterior, resulta TOT no computable.

Ejemplo: TOT no es computable

Veamos que $\mathrm{HALT} \leq \mathrm{TOT}$.

Construyamos $f : \Sigma^* \to \Sigma^*$ tal que para todo M, ω,
$\langle \#(M), \omega \rangle \in \mathrm{HALT} \iff f(\langle \#(M), \omega \rangle) \in \mathrm{TOT}$

Proponemos

$$f(\#(M), \omega) = \#(M_\omega),$$

donde $M_\omega(x) = \#M_\omega$ (la función constante que devuelve la codificación de $\#(M_\omega)$).

f es computable porque vimos que codificar un programa es computable.

Ejemplo: TOT no es computable

Veamos que $\langle \#(M), \omega \rangle \in \mathrm{HALT} \iff f(\langle \#(M), \omega \rangle) \in \mathrm{TOT}$.

$\langle \#(M), \omega \rangle \in \mathrm{HALT}$

$\iff M(\omega) \downarrow$

$\iff M_\omega(x) \downarrow \forall x \in \Sigma^*$

$\iff \#(M_\omega) \in \mathrm{TOT}$

Ejemplo: EQ no es computable

$$\mathrm{EQ} = \{\langle \#(M), \#(N) \rangle \mid \forall \omega \in \Sigma^*, M(\omega)\downarrow \Leftrightarrow N(\omega)\downarrow\}.$$

Lema: EQ no es computable.

Vamos a demostrarlo reduciendo TOT a EQ. Ya vimos que TOT no es computable.

Proponemos la siguiente $f : \Sigma^* \to \Sigma^*$

$f(\#(M)) = \langle \#(M), \#(\mathrm{id}) \rangle.$

Notar que un programa que computa id es:

$$Y = X_1$$

Ejemplo: EQ no es computable

f es computable ya que es composición de funciones computables, tupla y codificación de programas.

Veamos que $\langle \#(M), \omega \rangle \in \mathrm{HALT} \iff f(\#(M)) \in \mathrm{TOT}$.

$$\#(M) \in \mathrm{TOT}$$

$$\iff M(x) \downarrow \forall x \in \Sigma^*$$

$$\iff (M(x) \downarrow \Leftrightarrow \mathrm{id}(x) \downarrow) \forall x \in \Sigma^* \text{ (pues } \mathrm{id}(x) \downarrow \forall x\text{)}$$

$$\iff \langle \#(M), \#(\mathrm{id}) \rangle \in \mathrm{EQ}$$

Aún hay más

Aún hay más

Observación:

$$\mathrm{HALT} \leq \mathrm{TOT} \leq \mathrm{EQ}$$

Lema: EQ no es c.e.

Idea de demo: Sé que el complemento de HALT no es c.e., voy a demostrar que EQ tampoco es c.e. reduciendo el complemento de HALT a EQ.

Aún hay más

Tomando $f : \Sigma^* \to \Sigma^*$

$f(M, \omega) = \langle \#(M_\omega), \varsigma \rangle = \langle \#(M_\omega), \varsigma \rangle$, donde M_ω es el programa que siempre devuelve $M(\omega)$ y ς es un programa que se cuelga para toda entrada.

M_ω: Y = $M(\omega)$

ς: **while** true **do** {
pass
}

Aún hay más

f es computable total porque es composición de tupla y codificación de programas.

Veamos que esta función efectivamente reduce HALT^c a EQ.

$\langle \#(M), \omega \rangle \in \mathrm{HALT}^c$

$\iff M(\omega) \uparrow$

$\iff M_\omega(x) \uparrow \forall x \in \Sigma^*$ (por definición de M_ω)

$\iff (M_\omega(x) \uparrow \Leftrightarrow \varsigma(x) \uparrow) \forall x \in \Sigma^*$

(ya que $\varsigma \uparrow (x) \forall x$)

Aún hay más

$$\iff (M_\omega(x)\downarrow \Leftrightarrow \varsigma(x)\downarrow)\forall x \in \Sigma^*$$

$$\iff \langle \#(M_\omega), \varsigma\rangle \in \mathrm{EQ}$$

$$\iff f(M, \omega) \in \mathrm{EQ}$$ (por definición de f)

Conseguimos demostrar que $\langle \#(M), \omega\rangle$ pertenece a $\mathrm{HALT}^c \iff f(\#(M), \omega) \in \mathrm{EQ}$.

Propiedades

- A es computable $\iff A \leq \{\lambda\}$

  $\Rightarrow$) Tomar la función partida:

$$f(\omega) = \begin{cases} \lambda & \omega \notin A \\ a & \text{si no} \end{cases}$$

  $\Leftarrow$) Para ver si $\omega \in A$, veo si $f(\omega)$ = λ

- Sea Z computable, no trivial ($Z \neq \varnothing, Z \neq \Sigma^*$), entonces $\forall A$, A es computable $\iff A \leq Z$

Propiedades

- La computabilidad es cerrada por complemento. Es decir, si un lenguaje es computable, también lo es su complemento.

$$A \text{ computable} \iff A^c \text{ computable}$$

¡Ojo! La condición de ser computablemente enumerable no es cerrada por complemento.

Por ejemplo, HALT es computablemente enumerable, pero HALT^c no lo es.

¿Por qué?

Más propiedades

- A computable $\iff$ A es c.e. y A^c es c.e.

  $\Rightarrow$) Podemos usar la función característica y su negación.

  $\Leftarrow$) Construyendo un programa que ejecute los programas de las funciones características intercalando instrucciones de A y A^c.