Lenguajes Formales, Autómatas y Computabilidad

Clase Teórica

Segundo Cuatrimestre 2024

Bibliografía

Introduction to Automata Theory, Languages and Computation, J. Hopcroft, R. Motwani, J. Ullman, Second Edition, Addison Wesley, 2001. Capítulo 8 y 9.

Computability, Complexity, and Languages: Fundamentals of Theoretical ... By Martin Davis, Ron Sigal, Elaine J. Weyuker, Second Ediiton, Morgan Kaufmann, 1994

Tesis de Church

Hay muchos modelos de cómputo.
Está probado que tienen el mismo poder que $\mathcal{S}$

- C
- Java
- Haskell
- máquinas de Turing
- ...

Tesis de Church. Todos los algoritmos para computar en los naturales se pueden programar en $\mathcal{S}$.

Entonces, el problema de la detención dice

no hay algoritmo para decidir la verdad o falsedad de $\mathrm{HALT}(x, y)$

Universalidad

Para cada $n > 0$ definimos

$$\begin{aligned}\Phi^{(n)}(x_1, \ldots, x_n, e) &= \text{salida del programa } e \text{ con entrada } x_1, \ldots, x_n \\ &= \Psi_P^{(n)}(x_1, \ldots, x_n) \qquad \text{donde } \#(P) = e\end{aligned}$$

Teorema

Para cada $n > 0$ la función $\Phi^{(n)}$ es parcial computable.

Observar que el programa para $\Phi^{(n)}$ es un intérprete de programas.
Se trata de un programa que interpreta programas (representados por números).

Para demostrar el teorema, construimos el programa U_n que computa $\Phi^{(n)}$.

Idea de U_n

U_n es un programa que computa

$$\begin{aligned}\Phi^{(n)}(x_1,\ldots,x_n,e) &= \text{salida del programa } e \text{ con entrada } x_1,\ldots,x_n \\ &= \Psi_P^{(n)}(x_1,\ldots,x_n) \qquad \text{donde } \#(P)=e\end{aligned}$$

U_n necesita

- averiguar quién es P (decodifica e)
- llevar cuenta de los estados de P en cada paso
    - parte del estado inicial de P cuando la entrada es $x_1,\ldots,x_n$
    - codifica los estados con listas
    - por ejemplo $Y=0, X_1=2, X_2=1$ lo codifica como $[0,2,0,1]=63$

En el código de U_n

- K indica el número de instrucción de P que se está por ejecutar
- S describe el estado de P en cada momento

Inicialización

$//$ entrada $= x_1, \dots, x_n, e$

$//$ $\#(P) = e = [i_1, \dots, i_m] - 1$

$$Z \leftarrow X_{n+1} + 1$$

$//$ $Z = [i_1, \dots, i_m]$

$$S \leftarrow \prod_{j=1}^{n} (p_{2j})^{X_j}$$

$//$ $S = [0, X_1, 0, X_2, \dots, 0, X_n]$ es el estado inicial

$$K \leftarrow 1$$

$//$ la primera instrucción de P a analizar es la 1

Caso $V++$

$//$ S codifica el estado, K es el número de instrucción

$//$ $Z = [i_1, \ldots, i_m], i_K = \langle a, b \rangle + 1,$

$//$ R es el primo para la variable V que aparece en i_K

$//$ se trata de $V++$

$S \leftarrow S \cdot R$

$//$ $S =$ nuevo estado de P (suma 1 a V)

$K \leftarrow K + 1$

Caso $V--$

$//$ S codifica el estado, K es el número de instrucción

$//$ $Z = [i_1, \ldots, i_m], i_K = \langle a, b \rangle + 1$

$//$ R es el primo para la variable V que aparece en i_K

$//$ se trata de $V--$ con $V \neq 0$

$S \leftarrow S$ div P

$//$ S=nuevo estado de P (resta 1 a V)

$K \leftarrow K + 1$

Caso **while**

Caso **pass**

$//$ $\quad S$ codifica el estado, K es el número de instrucción

$//$ $\quad Z = [i_1, \ldots, i_m], i_K = 0$

$//$ $\quad$ la instrucción no cambia el estado

$K \leftarrow K + 1$

Devolución del resultado

// S codifica el estado final de P

// sale del ciclo principal

$Y \leftarrow S[1]$

// $Y =$ el valor que toma la variable Y de P al terminar

$\Phi(x_1, \ldots, x_n, e) = \Psi_P(x_1, \ldots x_n)$, donde $e = \#(P)$

$Z \leftarrow X_{n+1} + 1$

$S \leftarrow \prod_{i=1}^{n} (primo_{2i})^{X_i}$

$K \leftarrow 1$

while $(0 \leq K \leq |Z|)$**do** $\{$

$I \leftarrow Z[K]$

if $(I \neq 0)\{$

if $(r(I) = 0)\{S \leftarrow S primo(\ell(I))\}$ suma

if $(r(I) = 1)\{S \leftarrow \max(1, S/primo(\ell(I)))\ \}$ resta

if $r(I) \geq 2\{$

while $(S[\ell(I)] \neq 0)$**do** $\{S \leftarrow \Phi(S, g(r(I), n))\}$

$g(x, n)$ es el codigo de programa x pero adaptado para recibir una secuencia de 2n+1 variables y devolver una secuencia de 2n+1 variables

$K++$

$\}$

$Y \leftarrow S[1]$

Step Counter

Definimos

$\mathsf{STP}^{(n)}(x_1, \ldots, x_n, e, t)$	sii	el programa e termina en t o menos pasos con entrada $x_1, \ldots, x_n$
	sii	hay un cómputo del programa e de longitud $\leq t+1$, comenzando con la entrada $x_1, \ldots, x_n$

Teorema

Para cada $n > 0$, el predicado $\mathsf{STP}^{(n)}(x_1, \ldots, x_n, e, t)$ *es r.p.*

Snapshot

Definimos

$$\mathsf{SNAP}^{(n)}(x_1, \ldots, x_n, e, t) \quad = \quad \text{representación de la configuración instantánea del programa } e \text{ con entrada } x_1, \ldots, x_n \text{ en el paso } t$$

La configuración instantánea se representa como

$$\langle \text{número de instrucción, lista representando el estado} \rangle$$

Teorema

Para cada $n > 0$, la función $\mathsf{SNAP}^{(n)}(x_1, \ldots, x_n, e, t)$ *es r.p.*

Teorema de la Forma Normal

Teorema

Sea $f : \mathbb{N}^n \to \mathbb{N}$ una función parcial computable. Entonces existe un predicado r.p. $R : \mathbb{N}^{n+1} \to \mathbb{N}$ tal que

$$f(x_1, \ldots, x_n) = l\left(\min_z R(x_1, \ldots, x_n, z)\right)$$

Demostración.

Sea e el número de algún programa para $f(x_1, \ldots, x_n)$.
Recordar que la configuración instantánea se representa como

$$\langle \text{número de instrucción}, \text{lista representando el estado} \rangle$$

El siguiente predicado $R(x_1, \ldots, x_n, z)$ es el buscado:

$$\begin{array}{rl} & \mathsf{STP}^{(n)}(x_1, \ldots, x_n, e, r(z)) \quad \wedge \\ l(z) = & \underbrace{\underbrace{r\left(\mathsf{SNAP}^{(n)}(x_1, \ldots, x_n, e, r(z))\right)}_{\text{estado final de } e \text{ con entrada } x_1, \ldots, x_n} [1]}_{\text{valor de la variable } Y \text{ en ese estado final}} \end{array}$$

□

Teorema del Parámetro

Hay un programa P_{x_2} para la función $f_{x_2}(x_1) = f(x_1, x_2)$

La transformación $(x_2, \#(P)) \mapsto \#(P_{x_2})$ es r.p., es decir, existe una función $S : \mathbb{N}^2 \to \mathbb{N}$ r.p. tal que dado x_2 e $y = \#(P)$ calcula $\#(P_{x_2})$:

$$S(x_2, y) = \left(2^{109} \cdot 3^{110} \cdot \prod_{j=1}^{x_2} p_{j+2}^{26} \cdot \prod_{j=1}^{|y+1|} p_{j+x_2+2}{}^{(y+1)[j]} \right) - 1$$

Teorema

Hay una función r.p. $S : \mathbb{N}^2 \to \mathbb{N}$ *tal que*

$$\Phi_y^{(2)}(x_1, x_2) = \Phi_{S(x_2,y)}^{(1)}(x_1).$$

Teorema

Para cada $n, m > 0$ *hay una función r.p. inyectiva* $S_m^n : \mathbb{N}^{n+1} \to \mathbb{N}$ *tal que*

$$\Phi_y^{(n+m)}(x_1, \ldots, x_m, u_1, \ldots, u_n) = \Phi_{S_m^n(u_1,\ldots,u_n,y)}^{(m)}(x_1, \ldots, x_m)$$

Programas autoreferentes

- en la demostración del Halting Problem construimos un programa P que, cuando se ejecuta con su mismo número de programa (es decir, $\#(P)$), evidencia una contradicción
- en general, los programas pueden dar por supuesto que conocen su mismo número de programa
- pero si un programa P conoce su número de programa, podría, por ejemplo, devolver su mismo número, es decir, $\#(P)$

Teorema de la Recursión

Teorema

Si $g : \mathbb{N}^{n+1} \to \mathbb{N}$ *es parcial computable, existe un* e *tal que*

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Demostración.

Sea S_n^1 la función del Teorema del Parámetro:

$$\Phi_y^{(n+1)}(x_1, \ldots, x_n, u) = \Phi_{S_n^1(u,y)}^{(n)}(x_1, \ldots, x_n).$$

La función $(x_1, \ldots, x_n, v) \mapsto g(S_n^1(v, v), x_1, \ldots, x_n)$ es parcial computable, de modo que existe d tal que

$$\begin{aligned} g(S_n^1(v, v), x_1, \ldots, x_n) &= \Phi_d^{(n+1)}(x_1, \ldots, x_n, v) \\ &= \Phi_{S_n^1(v,d)}^{(n)}(x_1, \ldots, x_n) \end{aligned}$$

d está fijo; v es variable. Elegimos $v = d$ y $e = S_n^1(d, d)$. $\square$

Teorema de la Recursión

Corolario

Si $g : \mathbb{N}^{n+1} \to \mathbb{N}$ *es parcial computable, existen infinitos* e *tal que*

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Demostración.

En la demostración del teorema anterior, existen infinitos d tal que

$$\Phi_d^{(n+1)} = g(S_n^1(v, v), x_1, \ldots, x_n).$$

$v \mapsto S_n^1(v, v)$ es inyectiva de modo que existen infinitos

$$e = S_n^1(d, d).$$

$\square$

Quines

Un quine es un programa que cuando se ejecuta, devuelve como salida el mismo programa.

Por ejemplo:

```
char*f="char*f=%c%s%c;main()
{printf(f,34,f,34,10);}%c";
main(){printf(f,34,f,34,10);}
```

Quines

¿Existe e tal que $\Phi_e(x) = e$?

Sí, el programa vacío tiene numero 0 y computa la función constante 0, es decir, $\Phi_0(x) = 0$.

Proposición

Hay infinitos e tal que $\Phi_e(x) = e$.

Demostración.

Considerar la función $g : \mathbb{N}^2 \to \mathbb{N}, g(z, x) = z$.
Aplicando el Teorema de la Recursión, existen infinitos e tal que

$$\Phi_e(x) = g(e, x) = e.$$

$\square$

Quines

No hay nada especial con que la salida del programa sea su propio número en el resultado anterior. Funciona para cualquier h parcial computable.

¿Existe e tal que $\Phi_e(x) = h(e)$?

Proposición

Hay infinitos e tal que $\Phi_e(x) = h(e)$.

Demostración.

Considerar la función $g : \mathbb{N}^2 \to \mathbb{N}, g(z, x) = h(z)$.
Aplicando el Teorema de la Recursión, existen infinitos e tal que

$$\Phi_e(x) = g(e, x) = h(e).$$

$\square$

Teorema del Punto Fijo

Teorema

Si $f : \mathbb{N} \to \mathbb{N}$ *es computable, existe un* e *tal que* $\Phi_{f(e)} = \Phi_e$.

Demostración.

Considerar la función $g : \mathbb{N}^2 \to \mathbb{N}$,

$$g(z, x) = \Phi_{f(z)}(x).$$

Aplicando el Teorema de la Recursión, existe un e tal que para todo x,

$$\Phi_e(x) = g(e, x) = \Phi_{f(e)}(x)$$

□

Conjuntos en teoría de la computabilidad

Cuando hablamos de un conjunto de naturales A pensamos siempre en la función característica de ese conjunto.

$$A(x) = \begin{cases} 1 & \text{si } x \in A \\ 0 & \text{si no} \end{cases}$$

Así, un conjunto puede ser:

- computable
- primitivo recursivo

Teorema

Sean A, B conjuntos de una clase PRC $\mathcal{C}$. Entonces $A \cup B$, $A \cap B$ y $\overline{A}$ están en $\mathcal{C}$.

Conjuntos computablemente enumerables

Igual que con las funciones

- hay conjuntos computables, por ejemplo

$$\emptyset \quad , \quad \mathbb{N} \quad , \quad \{p : p \text{ es primo}\}$$

- hay conjuntos no computables, por ejemplo

$$\{\langle x, y\rangle : \mathsf{HALT}(x, y)\} \quad , \quad \{\langle x, \langle y, z\rangle\rangle : \Phi_x(y) = z\}$$

Definición

Un conjunto A es computablemente enumerable (c.e.) cuando existe una función parcial computable $g : \mathbb{N} \to \mathbb{N}$ tal que

$$A = \{x : g(x)\downarrow\} = \text{dom } g$$

- podemos decidir algoritmicamente si un elemento sí pertenece a A, pero para elementos que no pertenecen a A, el algoritmo se indefine
- se llaman algoritmos de semi-decisión: resuelven una aproximación al problema de decidir la pertenencia de un elemento al conjunto A

Propiedades de los conjuntos c.e.

Un conjunto A es co-c.e. si $\overline{A}$ es c.e.

Teorema

Si A es computable entonces A es c.e.

Demostración.

Sea P_A un programa para [la función característica de] A. Consideremos el siguiente programa P:

$$[C] \qquad \text{IF } P_A(X) = 0 \text{ GOTO } C$$

Tenemos

$$\Psi_P(x) = \begin{cases} 0 & \text{si } x \in A \\ \uparrow & \text{si no} \end{cases}$$

y por lo tanto

$$A = \{x : \Psi_P(x) \downarrow\}$$

□

Propiedades de los conjuntos c.e.

Teorema

Si A y B son c.e. entonces $A \cup B$ y $A \cap B$ también son c.e.

Demostración.

Sean $\quad A = \{x : \Phi_p(x)\downarrow\} \quad , \quad B = \{x : \Phi_q(x)\downarrow\}$

$(A \cap B)$ El siguiente programa R tiene como dominio a $A \cap B$:

$$Y \leftarrow \Phi_p(x)$$
$$Y \leftarrow \Phi_q(x)$$

En efecto, $\Psi_R(x)\downarrow$ sii $\Phi_p(x)\downarrow$ y $\Phi_q(x)\downarrow$.

$(A \cup B)$ El siguiente programa R' tiene como dominio a $A \cup B$:

$$
\begin{array}{ll}
[C] & \text{IF STP}^{(1)}(X,p,T) = 1 \text{ GOTO } E \\
 & \text{IF STP}^{(1)}(X,q,T) = 1 \text{ GOTO } E \\
 & T \leftarrow T + 1 \\
 & \text{GOTO } C
\end{array}
$$

En efecto, $\Psi_{R'}(x)\downarrow$ sii $\Phi_p(x)\downarrow$ o $\Phi_q(x)\downarrow$. $\square$

Propiedades de los conjuntos c.e.

Teorema

A es computable sii A y $\overline{A}$ son c.e.

Demostración.

$(\Rightarrow)$ si A es computable entonces $\overline{A}$ es computable

$(\Leftarrow)$ supongamos que A y $\overline{A}$ son c.e.

$$A = \{x : \Phi_p(x) \downarrow\} \qquad , \qquad \overline{A} = \{x : \Phi_q(x) \downarrow\}$$

Consideremos P:

$$\begin{array}{ll}
[C] & \text{IF STP}^{(1)}(X, p, T) = 1 \text{ GOTO } F \\
 & \text{IF STP}^{(1)}(X, q, T) = 1 \text{ GOTO } E \\
 & T \leftarrow T + 1 \\
 & \text{GOTO } C \\
[F] & Y \leftarrow 1
\end{array}$$

Para cada x, $x \in A$ o bien $x \in \overline{A}$. Entonces Ψ_P computa A. $\square$

Teorema de la enumeración

Definimos

$$W_n = \{x : \Phi_n(x)\downarrow\} = \text{dominio del } n\text{-ésimo programa}$$

Teorema

Un conjunto A es c.e. sii existe un n tal que $A = W_n$.

Existe una enumeración de todos los conjuntos c.e.

$$W_0, W_1, W_2, \ldots$$

Problema de la detención (visto como conjunto)

Recordar que

$$W_n = \{x : \Phi_n(x)\downarrow\}$$

Definimos

$$K = \{n : n \in W_n\}$$

Observar que

$$n \in W_n \quad \text{sii} \quad \Phi_n(n)\downarrow \quad \text{sii} \quad \text{HALT}(n, n)$$

Teorema

K es c.e. pero no computable.

Demostración.

- la función $n \mapsto \Phi(n, n)$ es parcial computable, de modo que K es c.e.
- supongamos que K fuera computable. Entonces $\overline{K}$ también lo sería. Luego existe un e tal que $\overline{K} = W_e$. Por lo tanto

$$e \in K \quad \text{sii} \quad e \in W_e \quad \text{sii} \quad e \in \overline{K}$$

□

Más propiedades de los conjuntos c.e.

Teorema

Si A es c.e., existe un predicado r.p. $R : \mathbb{N}^2 \to \mathbb{N}$ tal que

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

Demostración.

Sea $A = W_e$. Es decir,

$$A = \{x : \Phi_e(x)\downarrow\}.$$

Entonces $x \in A$ cuando en algún tiempo t, el programa e con entrada x termina, es decir,

$$A = \{x : (\exists t)\ \underbrace{\mathsf{STP}^{(1)}(x,e,t)}_{R(x,t)}\}$$

$\square$

Más propiedades de los conjuntos c.e.

Teorema

Si $A \neq \emptyset$ *es c.e., existe una función r.p.* $f : \mathbb{N} \to \mathbb{N}$ *tal que*

$$A = \{f(0), f(1), f(2), \dots\}$$

Demostración.

Por el teorema anterior, existe P r.p. tal que

$$A = \{x : (\exists t)\ P(x,t)\}.$$

Sea $a \in A$ y definamos

$$f(u) = \begin{cases} l(u) & \text{si } P(l(u), r(u)) \\ a & \text{si no} \end{cases}$$

- $x \in A \Rightarrow$ existe t tal que $P(x,t) \Rightarrow f(\langle x,t \rangle) = x$
- sea x tal que $f(u) = x$ para algún u. Entonces $x = a$ o bien u es de la forma $u = \langle x,t \rangle$, con $P(x,t)$. Luego $x \in A$. $\square$

Más propiedades de los conjuntos c.e.

Teorema

Si $f : \mathbb{N} \to \mathbb{N}$ *es parcial computable,* $A = \{f(x) : f(x)\downarrow\}$ *es c.e.*

Demostración.

Sea $\Phi_p = f$. Definamos el programa Q

$$
\begin{array}{ll}
[A] & \text{IF STP}^{(1)}(Z,p,T) = 0 \text{ GOTO } B \\
 & \text{IF } \Phi_p(Z) = X \text{ GOTO } E \\
[B] & Z \leftarrow Z + 1 \\
 & \text{IF } Z \leq T \text{ GOTO } A \\
 & T \leftarrow T + 1 \\
 & Z \leftarrow 0 \\
 & \text{GOTO } A
\end{array}
$$

Notar que $\Psi_Q(X)\downarrow$ si existen Z, T tal que

- $Z \leq T$
- $\text{STP}^{(1)}(Z,p,T)$ es verdadero (es decir, el programa para f termina en T o menos pasos con entrada Z)
- $X = f(Z)$

$$\Psi_Q(x) = \begin{cases} 0 & \text{si } x \in A \\ \uparrow & \text{si no} \end{cases}$$

Luego A es c.e. $\square$

Caracterizaciones de los conjuntos c.e.

Teorema

Si $A \neq \emptyset$, son equivalentes:

1. A *es c.e.*
2. A *es el rango de una función primitiva recursiva*
3. A *es el rango de una función computable*
4. A *es el rango de una función parcial computable*

Demostración.

$(1 \Rightarrow 2)$ *Teorema de hoja 31*

$(2 \Rightarrow 3)$ *Trivial*

$(3 \Rightarrow 4)$ *Trivial*

$(4 \Rightarrow 1)$ *Teorema de hoja 32*

$\square$

Teorema de Rice

$A \subseteq \mathbb{N}$ es un conjunto de índices si existe una clase de funciones $\mathbb{N} \to \mathbb{N}$ parciales computables $\mathcal{C}$ tal que $A = \{x : \Phi_x \in \mathcal{C}\}$

Teorema

Si A es un conjunto de índices tal que $\emptyset \neq A \neq \mathbb{N}$, A no es computable.

Demostración.

Supongamos $\mathcal{C}$ tal que $A = \{x : \Phi_x \in \mathcal{C}\}$ computable. Sean $f \in \mathcal{C}$ y $g \notin \mathcal{C}$ funciones parciales computables.
Sea $h : \mathbb{N}^2 \to \mathbb{N}$ la siguiente función parcial computable:

$$h(t,x) = \begin{cases} g(x) & \text{si } t \in A \\ f(x) & \text{si no} \end{cases}$$

Por el Teorema de la Recursión, existe e tal que $\Phi_e(x) = h(e,x)$.

- $e \in A \Rightarrow \Phi_e = g \Rightarrow \Phi_e \notin \mathcal{C} \Rightarrow e \notin A$
- $e \notin A \Rightarrow \Phi_e = f \Rightarrow \Phi_e \in \mathcal{C} \Rightarrow e \in A$ □

Aplicaciones del Teorema de Rice

El teorema da una fuente de conjuntos no computables:

- $\{x : \Phi_x \text{ es total}\}$
- $\{x : \Phi_x \text{ es creciente}\}$
- $\{x : \Phi_x \text{ tiene dominio infinito}\}$
- $\{x : \Phi_x \text{ es primitiva recursiva}\}$

Todos son no computables porque todos son conjuntos de índices no triviales!

Ejemplos de conjuntos no c.e.

- $\overline{K} = \{x : \Phi_x(x) \uparrow\}$ no es c.e.
  - K es c.e. de modo que si $\overline{K}$ lo fuera, K sería computable
- $Tot = \{x : \Phi_x \text{ es total}\}$ no es c.e.:
  - es una diagonalización simple. Supongamos que Tot es c.e.
  - existe f computable tal que $Tot = \{f(0), f(1), f(2), \dots\}$
  - entonces existe e tal que $\Phi_e(x) = \Phi_{f(x)}(x) + 1$
  - como Φ_e es total, $e \in Tot$. De modo que existe u tal que $f(u) = e$
  - $\Phi_{f(u)}(x) = \Phi_{f(x)}(x) + 1$. Absurdo para $x = u$.
- $\overline{Tot} = \{x : \Phi_x \text{ no es total}\}$ no es c.e.
  - parecido a lo que hicimos la clase pasada. Supongamos $\overline{Tot}$ c.e.
  - existe d tal que $\overline{Tot} = \text{dom } \Phi_d$
  - definimos el siguiente programa P:

$$
\begin{array}{ll}
[C] & \text{IF STP}^{(1)}(X, d, T) = 1 \text{ GOTO } E \\
 & T \leftarrow T + 1 \\
 & \text{GOTO } C
\end{array}
$$

  - sigue igual a lo que vimos la clase pasada

$$\Psi_P^{(2)}(x, y) = g(x, y) = \begin{cases} \uparrow & \Phi_x \text{ es total} \\ 0 & \text{si no} \end{cases}$$

Conjuntos más difíciles que el halting problem

- $K = \{x : \Phi_x(x) \downarrow\}$ no es computable
  - pero K es c.e.
- $Tot = \{x : \Phi_x \text{ es total}\}$ no es computable
  - Tot no es c.e.
  - $\overline{Tot}$ no es c.e.
- de alguna forma, Tot es más difícil que K
  - esto se formaliza dentro de la teoría
  - no hay tiempo para verlo en esta materia