Lenguajes Formales, Autómatas y Computabilidad

Clase Teórica
Máquinas de Turing, funciones parcialmente computables, conjuntos computablemente enumerables

Segundo Cuatrimestre 2024

Bibliografía

Introduction to Automata Theory, Languages and Computation, J. Hopcroft, J. Ullman, First Edition, Addison Wesley, 1979. Capitulo 7, Turing Machines

Formal Languages and their relation to automata J. Hopcroft, J. Ullman, Addison Wesley, 1969 Teorema 7.3 y 7.4 (paginas 111-112)

Orígenes

Principios 1900, preocupación por fundamentos de la matemática (Hilbert)

Completitud de la aritmética

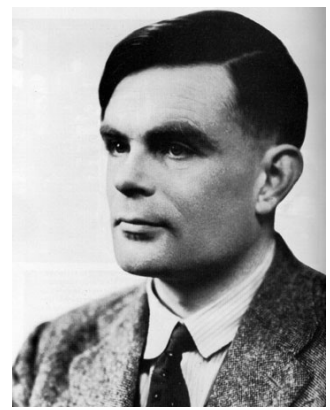
- se buscaba un sistema axiomático que capturara todas las verdades de la aritmética
- Gödel (1931): cualquier sistema axiomático suficientemente poderoso es incompleto o inconsistente

El problema de la decisión (*Entscheidungsproblem*)

- se buscaba un procedimiento efectivo para decidir si cualquier fórmula de primer orden era válida o no
- Turing (1936): no existe tal procedimiento efectivo

David Hilbert

Kurt Gödel



Alan Turing

Definición de máquina de Turing

Una máquina de Turing es una tupla $\mathcal{M} = (Q, \Sigma, \delta, q_0, q_f)$ donde

Q (finito) es el conjunto de estados

Σ es conjunto finito símbolos de la cinta, finito, $\mathbf{B} \in \Sigma$

$q_0 \in Q$ es el estado inicial

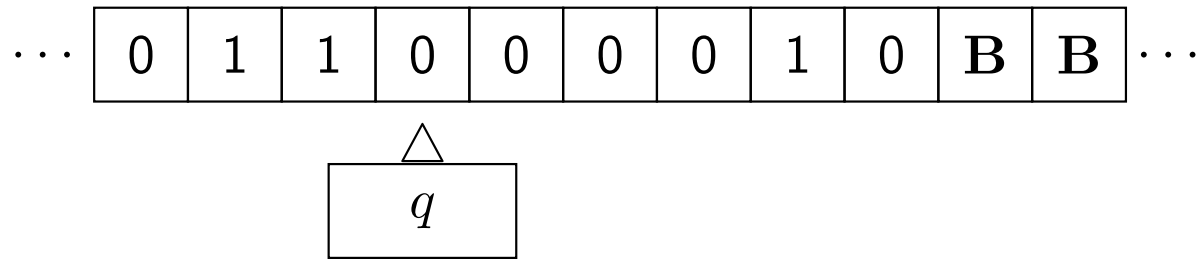
$q_f \in Q$ es el estado final

$\delta \subseteq Q \times \Sigma \times \Sigma \cup \{L, R\} \times Q$ es la tabla de instrucciones (finita)

Si no hay restricciones sobre δ decimos que $\mathcal{M}$ es una máquina de Turing no determinística

Si no hay dos instrucciones en δ que empiezan con las mismas primeras dos coordenadas, decimos que $\mathcal{M}$ es una máquina de Turing determinística
Por lo tanto, $\delta : Q \times \Sigma \to \Sigma \cup \{L, R\} \times Q$

Máquinas de Turing



Se compone de :

una cinta

- dividida en celdas
- infinita en ambas direcciones
- cada celda contiene un símbolo de un alfabeto dado Σ.
  - $\mathbf{B} \in \Sigma$, representa el blanco en una celda
  - $L, R \notin \Sigma$
    L y R son símbolos reservados (representarán acciones que puede realizar la cabeza)

una cabeza

- lee y escribe un símbolo a la vez
- se mueve una posición a la izquierda o una posición a la derecha

una función de transición es una tabla finita de instrucciones

Q es el conjunto finito de estados
Σ es el alfabeto. $L, R \notin \Sigma$, $\mathbf{B} \in \Sigma$.
$A = \Sigma \cup \{L, R\}$ es el conjunto de acciones

- un símbolo $s \in \Sigma$ se interpreta como "escribir s en la posición actual"
- L se interpreta como "mover la cabeza una posición hacia la izquierda"
- R se interpreta como "mover la cabeza una posición hacia la derecha"

Una tabla de instrucciones δ es un subconjunto (finito) de

$$Q \times \Sigma \times A \times Q$$

La tupla $(q, s, a, q') \in \delta$ se interpreta como

Si la máquina está en el estado q leyendo en la cinta el símbolo s, entonces realiza la acción a y pasa al estado q'

Definición (descripción instantánea)

$\mathcal{M}$ está en estado q con la cabeza en el símbolo más a la izquierda de α_2 Una descripción instantánea de $\mathcal{M}$ es $\alpha_1 q \alpha_2$, donde $\alpha_1, \alpha_2 \in \Sigma^*, p \in Q$.

Notar que α_1, α_2 pueden tener blancos adentro.

Definition (movimiento $\underset{\mathcal{M}}{\vdash}$ de $\mathcal{M}$)

Supongamos $X_1 \dots X_{i-1} q X_i \dots X_n$ es una descripción instantánea.
Si $(p, Y, L) \in \delta(q, X_i)$, entonces para $i > 1$,

$$X_1 \dots X_{i-1} q X_i \dots X_n \underset{\mathcal{M}}{\vdash} X_1 \dots X_{i-2} p X_{i-1} Y X_{i+1} \dots X_n$$

Si $(p, Y, R) \in \delta(q, X_i)$, entonces

$$X_1 \dots X_{i-1} q X_i \dots X_n \underset{\mathcal{M}}{\vdash} X_1 \dots X_{i-1} Y p X_{i+1} \dots X_n$$

$\mathcal{M}$ se detiene en una descripción instantánea cuando no hay un próximo movimiento, porque δ no tiene una instrucción para eso.

Definición (lenguaje aceptado por $\mathcal{M}$)

El lenguaje aceptado por una máquina de Turing $\mathcal{M} = (Q, \Sigma, \delta, q_0, q_f)$ es el conjunto de palabras $w \in (\Sigma \setminus \{\mathbf{B}\})^*$ que causan que $\mathcal{M}$ llegue a un estado final,

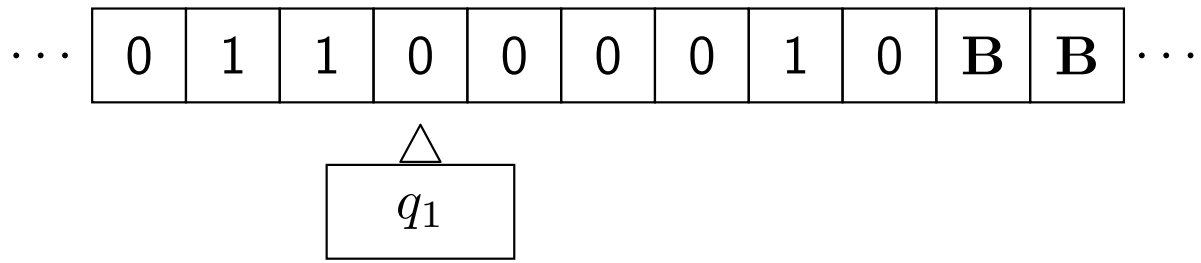
$$\mathcal{L}(\mathcal{M}) = \{w \in (\Sigma \setminus \{\mathbf{B}\})^* : q_0 w \overset{*}{\underset{\mathcal{M}}{\vdash}} \alpha q_f \beta, \text{ donde } \alpha\beta \in \Sigma^*\}$$

Ejemplo de un movimiento

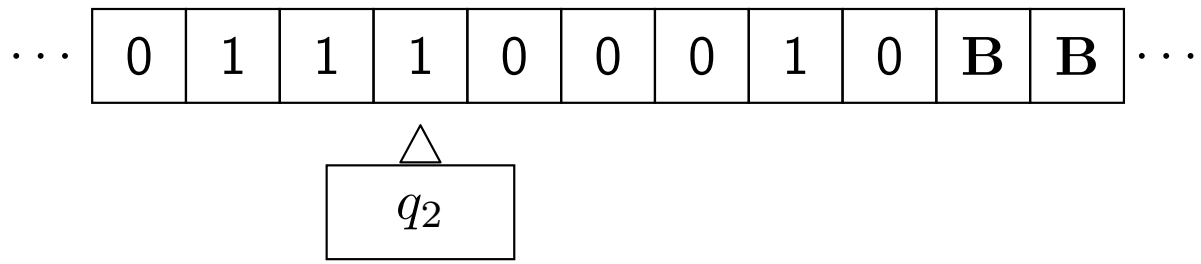
Supongamos un alfabeto $\Sigma = \{0, 1\}$.
Una máquina con esta tabla de instrucciones:

$$\{ \ (q_1, 0, 1, q_2) \quad , \quad (q_2, 1, R, q_1) \ \}$$

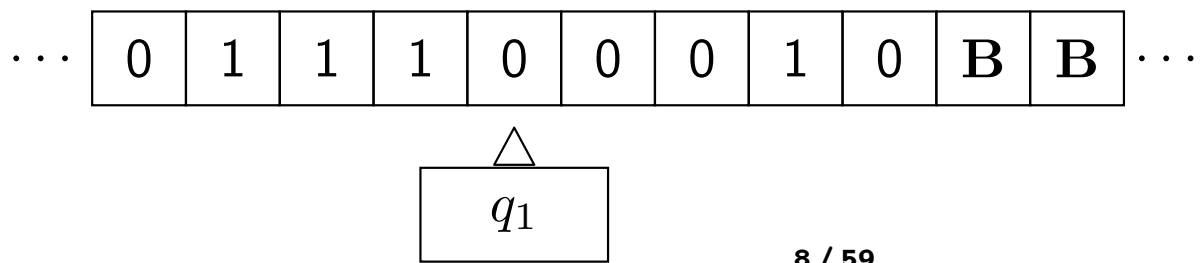
Si empieza en esta configuración



pasa a



y luego a

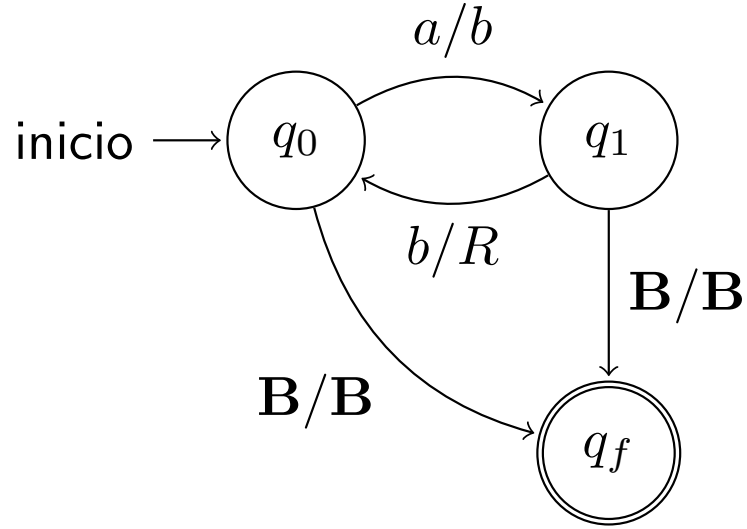


Ejemplo

Sea $\mathcal{M} = (Q, \Sigma, \delta, q_0, q_f)$ con
$\Sigma = \{\mathbf{B}, a, b\}$
$Q = \{q_0, q_1, q_f\}$

$$\delta = \begin{array}{cccc} q_0 & a & b & q_1 \\ q_1 & b & R & q_0 \\ q_0 & \mathbf{B} & \mathbf{B} & q_f \\ q_1 & \mathbf{B} & \mathbf{B} & q_f \end{array}$$

Visto como autómata



si empieza en $\underset{q_0}{\mathbf{B}}\ a\ a\ a\ a\ \mathbf{B}$ termina en $\mathbf{B}\ b\ b\ b\ b\ \underset{q_f}{\mathbf{B}}$

si empieza en $\underset{q_0}{\mathbf{B}}\ a\ a\ b\ a\ \mathbf{B}$ termina en $\mathbf{B}\ b\ b\ \underset{q_0}{b}\ a\ \mathbf{B}$

si empieza en $\underset{q_0}{\mathbf{B}}\ a\ a\ b\ a\ \mathbf{B}$ termina en $\underset{q_f}{\mathbf{B}}\ a\ a\ b\ a\ \mathbf{B}$

$$\mathcal{L}(\mathcal{M}) = a^*$$

Otros lenguajes aceptados por máquinas de Turing:

$\{ww^R : w \in \{0,1\}^*$

$\{a^n b^n c^n : n \geq 0\}$

Describir el cómputo en cada caso.

Recordemos, si MT $\mathcal{M} = (Q, \Sigma, \delta, q_0, q_f)$

$$\mathcal{L}(\mathcal{M}) = \{w \in (\Sigma \setminus \{\mathbf{B}\})^* : q_0 w \overset{*}{\underset{\mathcal{M}}{\vdash}} \alpha q_f \beta, \text{ donde } \alpha\beta \in \Sigma^*\}$$

Definición

Un lenguaje $L \subseteq (\Sigma \setminus \{\mathbf{B}\})^*$ es computablemente enumerable, que abreviamos c.e., si hay una MT $\mathcal{M}$ tal que $\mathcal{L}(M) = L$.

Teorema

Para toda MT $\mathcal{M}_1$ multicinta hay una MT $\mathcal{M}_2$ de una sola cinta tal que $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$.

Demostración.

Requiere que las funciones aritméticas sean computables

Sea $\mathcal{M}_1$ de k cintas y sea L aceptado por$\mathcal{M}_1$ Construiremos $\mathcal{M}_2$ de una sola cinta, donde codificaremos las k cintas $c_0, \ldots c_{k-1}$ de $\mathcal{M}_1$ y las k cabezas.

En la cinta de $\mathcal{M}_2$ indentificaré una posición 0 y usar la cinta a derecha.

Si la cabeza de la cinta de $\mathcal{M}_2$ está posición t,

Si $(t \mod 2k) < k$ entonces el símbolo en posición t denota el símbolo en la cinta $c_{t \mod 2k}$ de $\mathcal{M}_1$ en la posición $\lfloor t/2k \rfloor$.

Si $(t \mod 2k) \geq k$ y el símbolo en posición t si es distinto de $\mathbf{B}$ indica que la cabeza de la cinta $c_{(t-k) \mod 2k}$ está en la posición en la posición $\lfloor (t-k)/2k \rfloor$.

Cada movimiento de $\mathcal{M}_1$ es simulado por $\mathcal{M}_2$.

$\mathcal{M}_2$ acepta exactamente cuando $\mathcal{M}_1$ acepta. □

Teorema

Para toda MT $\mathcal{M}_1$ no determinística hay otra MT deterministica $\mathcal{M}_2$ tal que $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$.

Demostración.

Requiere que las funciones aritméticas sean computables Para cualquier estado y símbolo de cinta de $\mathcal{M}_1$ hay un número finito de opciones para el siguiente movimiento. Estas pueden numerarse $1, 2, \ldots$ Sea r el número máximo de opciones para cualquier par de estado-símbolo de cinta. Entonces cualquier secuencia finita de opciones puede representarse mediante una secuencia de los dígitos del 1 al r. No todas esas secuencias pueden representar opciones de movimientos, ya que puede haber menos de r opciones en algunos casos. $\mathcal{M}_2$ tendrá tres cintas. La primera contendrá la entrada. En la segunda, $\mathcal{M}_2$ generará todas las secuencias de dígitos 1 al r, en orden longitud lexicográfico.

$$(1), (2), \ldots (r), (1,1), (1,2), \ldots (r,r), (1,1,1), \ldots$$

Para cada secuencia generada en la cinta 2, $\mathcal{M}_2$ copia la entrada en la cinta 3 y luego simula $\mathcal{M}_1$ en la cinta 3 siguiendo los movimientos indicados en la cinta 2. Si $\mathcal{M}_1$ entra en un estado de aceptación, $\mathcal{M}_2$ también acepta.
Si hay una secuencia de opciones que conducen a la aceptación, eventualmente se generará en la cinta 2. Cuando se simule, $\mathcal{M}_2$ aceptará. Pero si ninguna secuencia de movimientos de $\mathcal{M}_1$ conduce a la aceptación, $\mathcal{M}_2$ no aceptará.

Tres formas de usar una MT

1. **Aceptadoras de subconjuntos c.e. en** $(\Sigma \setminus \{\mathbf{B}\})^*$

Definición (lenguaje aceptado por $\mathcal{M}$)

El lenguaje aceptado por una máquina de Turing $\mathcal{M}$ es el conjunto de palabras $w \in (\Sigma \setminus \{\mathbf{B}\})^*$ que causan que $\mathcal{M}$ llegue a un estado final,

$$\mathcal{L}(\mathcal{M}) = \{w \in (\Sigma \setminus \{\mathbf{B}\})^* : q_0 w \overset{*}{\underset{\mathcal{M}}{\vdash}} \alpha q_f \beta, \text{ donde } \alpha\beta \in \Sigma^*\}$$

2. **Funciones** $f : \mathbb{N} \to \mathbb{N}$ **Turing computables**

Lo veremos a continuación

3. **Enumeradoras de conjuntos c.e. en** $(\Sigma \setminus \{\mathbf{B}\})^*$

luego en esta msima clase

Definición (Lenguaje c.e.)

Un lenguaje $L \subseteq (\Sigma \setminus \{\mathbf{B}\})^*$ es computablemente enumerable (c.e.) si es aceptado por alguna máquina de Turing, es decir $L = \mathcal{L}(M)$.

Equivalentemente,
Si $L \subseteq (\Sigma \setminus \{\mathbf{B}\})^*$ representa el dominio de $f : \mathbb{N} \to \mathbb{N}$ parcial computable, es decir hay MT $\mathcal{M}$ que computa f.

Equivalentemente,
si $L \subseteq (\Sigma \setminus \{\mathbf{B}\})^*$ es el lenguaje generado por una MT $\mathcal{M}$, $\mathcal{G}(M) = L$.

Representación de números y tuplas

Fijamos $\Sigma = \{\mathbf{B}, 1\}$.
Representaremos a los números naturales en unario (con palotes).

- el número $x \in \mathbb{N}$ se representa como

$$\overline{x} = \underbrace{1 \ldots 1}_{x+1}$$

Representamos a las tuplas $(x_1, \ldots, x_n)$ como lista de (representaciones de) los x_i separados por blanco

- la tupla $(x_1, \ldots, x_n)$ se representa como

$$\mathbf{B}\overline{x_1}\mathbf{B}\overline{x_2}\mathbf{B} \cdots * \overline{x_n}\mathbf{B}$$

Por ejemplo,

- el número 0 se representa como 1
- el número 3 se representa como 1111
- la tupla $(1, 2)$ se representa como $\mathbf{B}11\mathbf{B}111\mathbf{B}$
- la tupla $(0, 0, 1)$ se representa como $\mathbf{B}1\mathbf{B}1\mathbf{B}11\mathbf{B}$

Funciones parciales

Una función parcial f es una función que puede estar indefinida para algunos (tal vez ninguno; tal vez todos) sus argumentos.
Siempre vamos a trabajar con funciones parciales $f : \mathbb{N}^n \to \mathbb{N}$.

- notamos $f(x_1, \ldots, x_n) \downarrow$ cuando f está definida para $x_1, \ldots, x_n$. En este caso $f(x_1, \ldots, x_n)$ es un número natural.
- notamos $f(x_1, \ldots, x_n) \uparrow$ cuando f está indefinida para $x_1, \ldots, x_n$

El conjunto de argumentos para los que f está definida se llama dominio de f, notado $\mathrm{dom}(f)$.

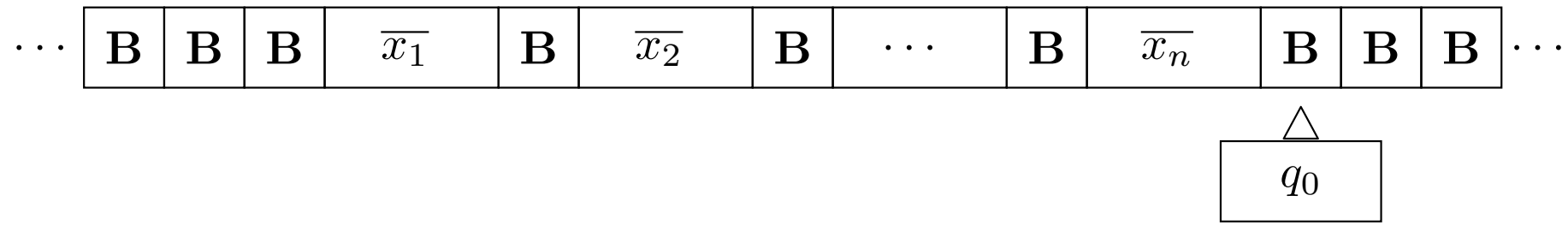
$$\mathrm{dom}(f) = \{(x_1, \ldots, x_n) : f(x_1, \ldots, x_n) \downarrow\}$$

f es total si $\mathrm{dom}(f) = \mathbb{N}^n$.

Funciones parciales Turing computables

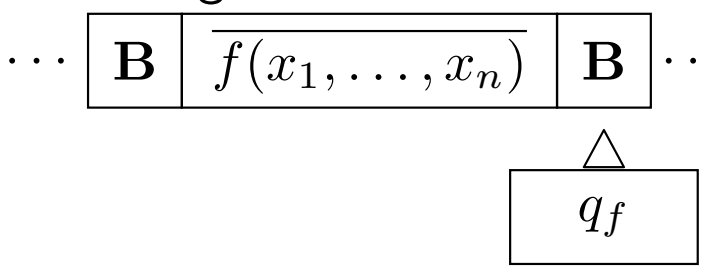
Definición

Una función parcial $f : \mathbb{N}^n \to \mathbb{N}$ es Turing computable si existe una máquina de Turing determinística $\mathcal{M} = (\Sigma, Q, \delta, q_0, q_f)$ con $\Sigma = \{\mathbf{B}, 1\}$ tal que cuando empieza en la configuración inicial



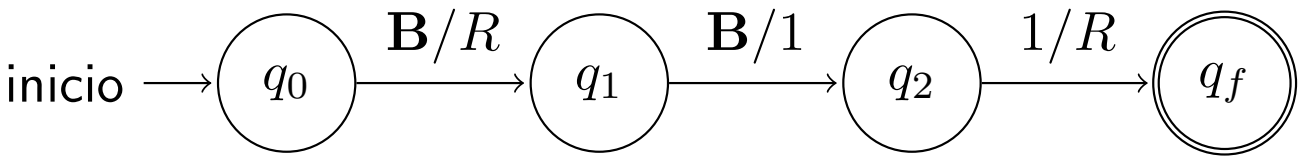
(con los enteros x_i representados en unario):

- si $f(x_1, \ldots, x_n) \downarrow$ entonces siguiendo sus instrucciones en δ llega a una configuración final de la forma

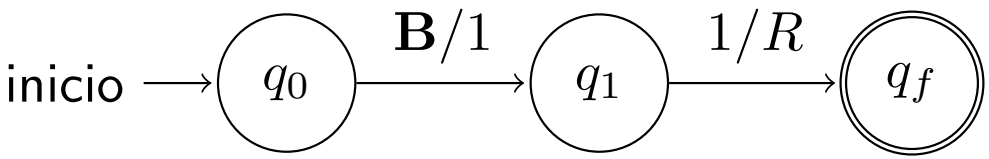
  

  (quizá algo más en la cinta)
- si $f(x_1, \ldots, x_n) \uparrow$ entonces nunca termina en el estado q_f.

Cómputo de la función $f(x) = 0$



Cómputo de la función $f(x) = x + 1$



Cómputo de la función $f(x) = 2x$

Idea: por cada 1 que borro de la entrada, pongo 11 bien a la derecha. Repito esto hasta que quede solo un 1 en la entrada. Ahí pongo un 1 más a la derecha.

Ejemplo: entrada $= 2$

B	**B**	**B**	**B**	**B**	1	1	1	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**	**B**
B	**B**	**B**	**B**	**B**	**B**	1	1	**B**	1	1	**B**	**B**	**B**	**B**	**B**	**B**
B	**B**	**B**	**B**	**B**	**B**	**B**	1	**B**	1	1	1	1	**B**	**B**	**B**	**B**
B	**B**	**B**	**B**	**B**	**B**	**B**	1	**B**	1	1	1	1	1	**B**	**B**	**B**

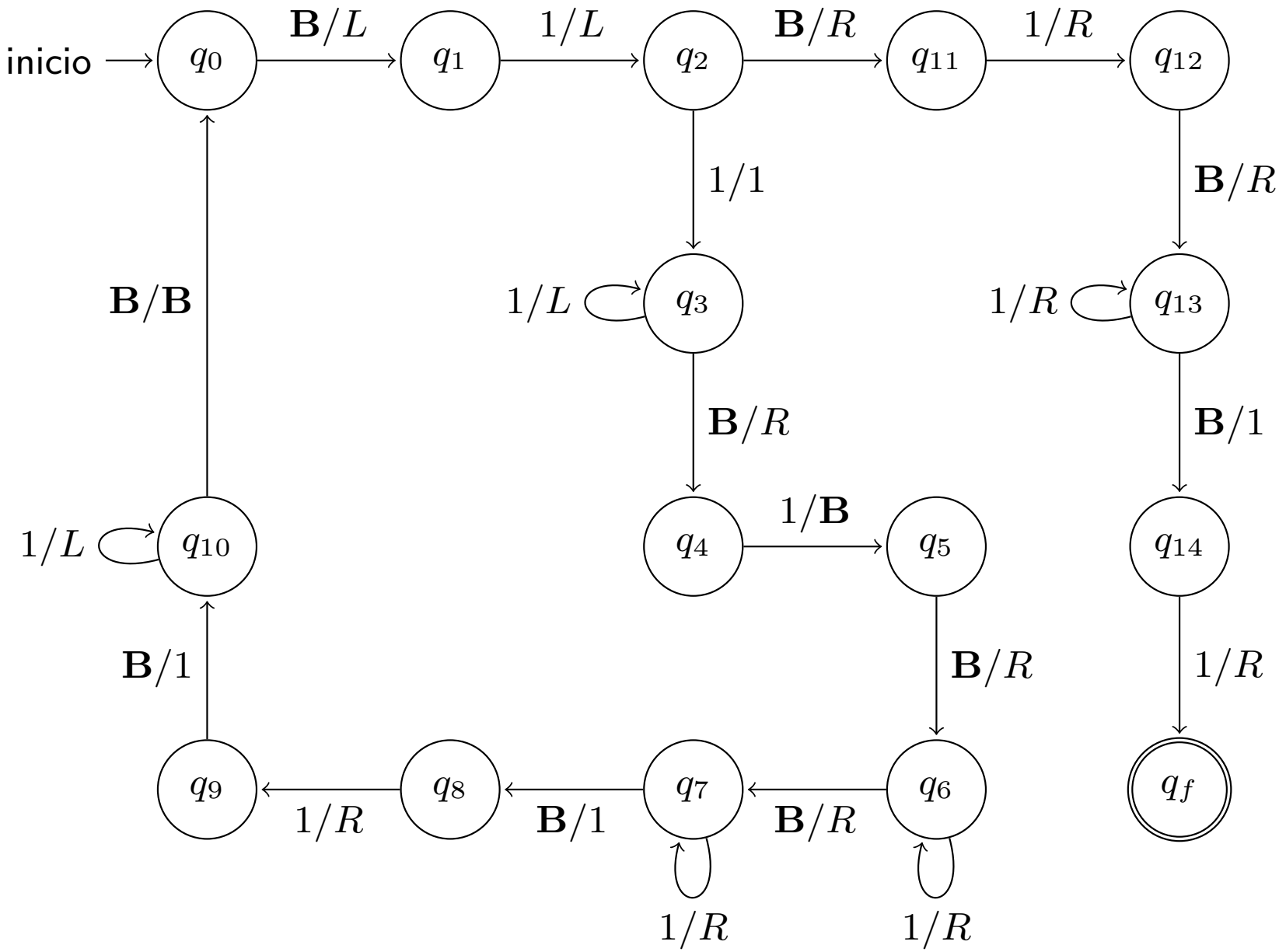
Invariante: a lo largo de cada iteración, la cinta está así:
$\mathbf{BBB}\underbrace{1\ldots1}_{n}\mathbf{B}\underbrace{1\ldots\ldots1}_{2m}\mathbf{BBB}$ para algún $n > 0, m \geq 0$, $n + m - 1 =$ entrada

Si $n = 1$ entonces pongo un 1 más a la derecha y termina en q_f con
$\mathbf{BBB}1\mathbf{B}\underbrace{1\ldots\ldots1}_{2m+1}\mathbf{BBB}$

Si $n > 1$ transformo la cinta en $\mathbf{BBB}\underbrace{1\ldots1}_{n-1}\mathbf{B}\underbrace{1\ldots\ldots1}_{2(m+1)}\mathbf{BBB}$

y vuelvo al paso 1

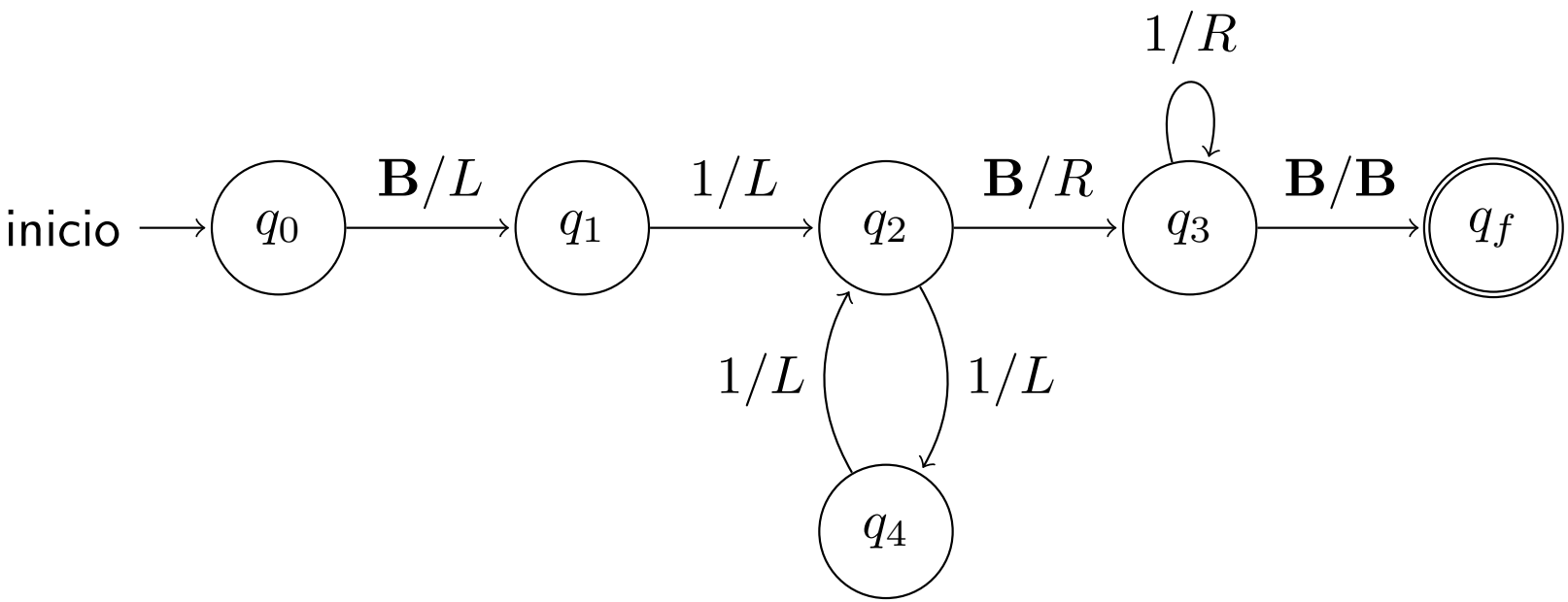
Cómputo de la función $f(x) = 2x$



Cómputo de una función parcial

Supongamos

$$f(x) = \begin{cases} x & \text{si } x \text{ es par} \\ \uparrow & \text{si no} \end{cases}$$



Iniciales, composición y recursión primitiva

Definición

Las siguientes funciones se llaman iniciales:
$s(x) = x + 1$, $n(x) = 0$, proyecciones: $u_i^n(x_1, \ldots, x_n) = x_i$ para $i \in \{1, \ldots, n\}$

Definición

Sea $f : \mathbb{N}^k \to \mathbb{N}$ y $g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$. $h : \mathbb{N}^n \to \mathbb{N}$ se obtiene a partir de f y $g_1, \ldots, g_k$ por composición si

$$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n))$$

Definición

$h : \mathbb{N}^{n+1} \to \mathbb{N}$ se obtiene a partir de $g : \mathbb{N}^{n+2} \to \mathbb{N}$ y $f : \mathbb{N}^n \to \mathbb{N}$ por recursión primitiva si

$$\begin{aligned} h(x_1, \ldots, x_n, 0) &= f(x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n, t+1) &= g(h(x_1, \ldots, x_n, t), x_1, \ldots, x_n, t) \end{aligned}$$

En este contexto, una función 0-aria es una constante k.
Si h es 1-aria y $t = 0$, entonces $h(t) = k = s^{(k)}(n(t))$.

Definition (funciones recursivas primitivas)

Una función es recursiva primitiva (p.r.), $f : \mathbb{N}^k \to \mathbb{N}$, si se define mediante las funciones iniciales y un número finito de aplicaciones de composición y recursión primitiva.

Ejemplos de función p.r.

La función $suma(x, y) = x + y$ es p.r., porque

$$\begin{aligned} suma(x, 0) &= u_1^1(x) \\ suma(x, y+1) &= g(suma(x, y), x, y) \end{aligned}$$

donde $g(x_1, x_2, x_3) = s(u_1^3(x_1, x_2, x_3))$

- $x \cdot y$
- $x!$
- x^y
- $x \dot{-} y = \begin{cases} x - y & \text{si } x \geq y \\ 0 & \text{si no} \end{cases}$
- $\alpha(x) = \begin{cases} 1 & \text{si } x = 0 \\ 0 & \text{si no} \end{cases}$
- y muchas más.

Clases PRC

Una clase $\mathcal{C}$ de funciones totales es PRC (primitive recursive closed) si

1. las funciones iniciales están en $\mathcal{C}$
2. si una función f se obtiene a partir de otras pertenecientes a $\mathcal{C}$ por medio de composición o recursión primitiva, entonces f también está en $\mathcal{C}$

Teorema

Una función es p.r. sii pertenece a toda clase PRC.

Demostración.

($\Leftarrow$) La clase de funciones p.r. es una clase PRC. Luego, si f pertenece a toda clase PRC, en particular f es p.r.

($\Rightarrow$) Sea f p.r. y sea $\mathcal{C}$ una clase PRC. Como f es p.r., hay una lista

$$f_1, f_2, \ldots, f_n$$

tal que

- $f = f_n$
- f_i es inicial (luego está en $\mathcal{C}$) o se obtiene por composición o recursión primitiva a partir de funciones f_j, $j < i$ (luego también está en $\mathcal{C}$).

Entonces todas las funciones de la lista están en $\mathcal{C}$ (por inducción). En particular, $f_n \in \mathcal{C}$. $\square$

clase de funciones p.r. es la clase PRC más chica

Teorema

La clase de funciones totales Turing computables es una clase PRC.

Corolario

Toda función p.r. es total y Turing computable.

Demostración.

Sabemos que la clase de funciones totales Turing computables es PRC. Por el teorema anterior, si f es p.r., entonces f pertenece a la clase de funciones Turing computables. □

Â¿toda función total Turing computable es p.r.?

Veremos que no

La función de Ackermann (1928)

$$A(x,y,z) = \begin{cases} y+z & \text{si } x = 0 \\ 0 & \text{si } x = 1 \text{ y } z = 0 \\ 1 & \text{si } x = 2 \text{ y } z = 0 \\ y & \text{si } x > 2 \text{ y } z = 0 \\ A(x-1, y, A(x, y, z-1)) & \text{si } x, z > 0 \end{cases}$$

- $A_0(y,z) = A(0,y,z) = y + z = y \underbrace{+1 + \cdots + 1}_{z \text{ veces}}$
- $A_1(y,z) = A(1,y,z) = y \cdot z = \underbrace{y + \cdots + y}_{z \text{ veces}}$
- $A_2(y,z) = A(2,y,z) = y \uparrow z = y^z = \underbrace{y \cdot \cdots \cdot y}_{z \text{ veces}}$
- $A_3(y,z) = A(3,y,z) = y \uparrow\uparrow z = \underbrace{y^{y^{\cdot^{\cdot^{\cdot^{y}}}}}}_{z \text{ veces}}$
- $\ldots$

Para cada i, $A_i : \mathbb{N}^2 \to \mathbb{N}$ es p.r. pero $A : \mathbb{N}^3 \to \mathbb{N}$ no es p.r.

Versión de Robinson & Peter (1948)

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

- $B_0(n) = B(0,n) = n+1$
- $B_1(n) = B(1,n) = 2 + (n+3) - 3$
- $B_2(n) = B(2,n) = 2 \cdot (n+3) - 3$
- $B_3(n) = B(3,n) = 2 \uparrow (n+3) - 3$
- $B_4(n) = B(4,n) = 2 \uparrow\uparrow (n+3) - 3$
- $\ldots$

Para cada i, $B_i : \mathbb{N} \to \mathbb{N}$ es p.r. pero $B : \mathbb{N}^2 \to \mathbb{N}$ no es p.r.

- $B(4,2) \simeq 2 \times 10^{19728}$

$B'(x) = B(x,x)$ crece más rápido que cualquier función p.r.

$$(\forall f \text{ p.r.})(\exists n)(\forall x > n)\ B'(x) > f(x)$$

Teorema

La función de Ackermann es total Turing computable y no es recursiva primitiva.

Predicados primitivos recursivos

Los predicados son simplemente funciones que toman valores en $\{0, 1\}$.

- 1 se interpreta como verdadero
- 0 se interpreta como falso

Los predicados p.r. son aquellos representados por funciones p.r. en $\{0, 1\}$.

Por ejemplo, el predicado $x \leq y$ es p.r. ya que $\alpha(x \dot{-} y)$.

Operadores lógicos

Teorema

Sea $\mathcal{C}$ una clase PRC. Si p y q son predicados en $\mathcal{C}$ entonces $\neg p$, $p \wedge q$ y $p \vee q$ están en $\mathcal{C}$.

Demostración.

- $\neg p$ se define como $\alpha(p)$
- $p \wedge q$ se define como $p \cdot q$
- $p \vee q$ se define como $\neg(\neg p \wedge \neg q)$

□

Corolario

Si p y q son predicados p.r., entonces también lo son los predicados $\neg p$, $p \vee q$ y $p \wedge q$

Corolario

Si p y q son predicados totales Turing computables entonces también lo son los predicados $\neg p$, $p \vee q$ y $p \wedge q$

Definición por casos (2)

Teorema

Sea $\mathcal{C}$ una clase PRC. Sean $h, g : \mathbb{N}^n \to \mathbb{N}$ funciones en $\mathcal{C}$ y sea $p : \mathbb{N}^n \to \{0, 1\}$ un predicado en $\mathcal{C}$. La función

$$f(x_1, \ldots, x_n) = \begin{cases} g(x_1, \ldots, x_n) & \text{si } p(x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n) & \text{si no} \end{cases}$$

está en $\mathcal{C}$.

Demostración.

$f(x_1, \ldots, x_n) =$
$g(x_1, \ldots, x_n) \cdot p(x_1, \ldots, x_n) + h(x_1, \ldots, x_n) \cdot \alpha(p(x_1, \ldots, x_n))$ □

Definición por casos $(m+1)$

Teorema

Sea $\mathcal{C}$ una clase PRC. Sean $g_1, \ldots, g_m, h : \mathbb{N}^n \to \mathbb{N}$ funciones en $\mathcal{C}$ y sean $p_1, \ldots, p_m : \mathbb{N}^n \to \{0,1\}$ predicados mutuamente excluyentes en $\mathcal{C}$. La función

$$f(x_1, \ldots, x_n) = \begin{cases} g_1(x_1, \ldots, x_n) & \text{si } p_1(x_1, \ldots, x_n) \\ \vdots & \\ g_m(x_1, \ldots, x_n) & \text{si } p_m(x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n) & \text{si no} \end{cases}$$

está en $\mathcal{C}$.

Sumatorias y productorias (desde 0)

Teorema

Sea $\mathcal{C}$ una clase PRC. Si $f : \mathbb{N}^{n+1} \to \mathbb{N}$ está en $\mathcal{C}$ entonces también están las funciones

$$g(y, x_1, \ldots, x_n) = \sum_{t=0}^{y} f(t, x_1, \ldots, x_n)$$

$$h(y, x_1, \ldots, x_n) = \prod_{t=0}^{y} f(t, x_1, \ldots, x_n)$$

Demostración.

$$\begin{aligned} g(0, x_1, \ldots, x_n) &= f(0, x_1, \ldots, x_n) \\ g(t+1, x_1, \ldots, x_n) &= g(t, x_1, \ldots, x_n) + f(t+1, x_1, \ldots, x_n) \end{aligned}$$

Idem para h con $\cdot$ en lugar de $+$. □

Observar que no importa la variable en la que se hace la recursión: podemos definir $g'(x, t)$ como la clase pasada y luego
$g(t, x) = g'(u_2^2(t, x), u_1^2(t, x)) = g'(x, t)$.

Sumatorias y productorias (desde 1)

Teorema

Sea $\mathcal{C}$ una clase PRC. Si $f : \mathbb{N}^{n+1} \to \mathbb{N}$ está en $\mathcal{C}$ entonces también están las funciones

$$g(y, x_1, \ldots, x_n) = \sum_{t=1}^{y} f(t, x_1, \ldots, x_n)$$

$$h(y, x_1, \ldots, x_n) = \prod_{t=1}^{y} f(t, x_1, \ldots, x_n)$$

(como siempre, sumatoria vacía = 0, productoria vacía = 1)

Demostración.

$$\begin{aligned} g(0, x_1, \ldots, x_n) &= 0 \\ g(t+1, x_1, \ldots, x_n) &= g(t, x_1, \ldots, x_n) + f(t+1, x_1, \ldots, x_n) \end{aligned}$$

Idem para h con $\cdot$ en lugar de $+$ y 1 en lugar de 0 en el caso base. □

Cuantificadores acotados

Sea $p : \mathbb{N}^{n+1} \to \{0, 1\}$ un predicado.

$(\forall t)_{\leq y}\ p(t, x_1, \ldots, x_n)$ es verdadero sii

- $p(0, x_1, \ldots, x_n)$ es verdadero y

$$\vdots$$

- $p(y, x_1, \ldots, x_n)$ es verdadero

$(\exists t)_{\leq y}\ p(t, x_1, \ldots, x_n)$ es verdadero sii

- $p(0, x_1, \ldots, x_n)$ es verdadero o

$$\vdots$$

- $p(y, x_1, \ldots, x_n)$ es verdadero

Lo mismo se puede definir con $< y$ en lugar de $\leq y$.

$$(\exists t)_{<y}\ p(t, x_1, \ldots, x_n) \quad \text{y} \quad (\forall t)_{<y}\ p(t, x_1, \ldots, x_n)$$

Cuantificadores acotados (con $\leq$)

Teorema

Sea $p : \mathbb{N}^{n+1} \to \{0, 1\}$ un predicado perteneciente a una clase PRC $\mathcal{C}$. Los siguientes predicados también están en $\mathcal{C}$:

$$(\forall t)_{\leq y}\ p(t, x_1, \ldots, x_n)$$

$$(\exists t)_{\leq y}\ p(t, x_1, \ldots, x_n)$$

Demostración.

$$(\forall t)_{\leq y}\ p(t, x_1, \ldots, x_n) \text{ sii } \prod_{t=0}^{y} p(t, x_1, \ldots, x_n) = 1$$

$$(\exists t)_{\leq y}\ p(t, x_1, \ldots, x_n) \text{ sii } \sum_{t=0}^{y} p(t, x_1, \ldots, x_n) \neq 0$$

- la sumatoria y productoria están en $\mathcal{C}$
- la comparación por $=$ está en $\mathcal{C}$

Cuantificadores acotados (con $<$)

Teorema

Sea $p : \mathbb{N}^{n+1} \to \{0, 1\}$ *un predicado perteneciente a una clase PRC* $\mathcal{C}$*. Los siguientes predicados también están en* $\mathcal{C}$*:*

$$(\forall t)_{<y}\ p(t, x_1, \ldots, x_n)$$

$$(\exists t)_{<y}\ p(t, x_1, \ldots, x_n)$$

Demostración.

$(\forall t)_{<y}\ p(t, x_1, \ldots, x_n)$ sii $(\forall t)_{\leq y}\ (t = y \vee p(t, x_1, \ldots, x_n))$
$(\exists t)_{<y}\ p(t, x_1, \ldots, x_n)$ sii $(\exists t)_{\leq y}\ (t \neq y \wedge p(t, x_1, \ldots, x_n))$ □

Más ejemplos de funciones recursivas primitivas

- $y|x$ sii y divide a x. Se define como

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

  Notar que con esta definición $0|0$.
- $primo(x)$ sii x es primo.

Minimización acotada

Sea $p : \mathbb{N}^{n+1} \to \{0, 1\}$ un predicado de una clase PRC $\mathcal{C}$.

$$g(y, x_1, \ldots, x_n) = \sum_{u=0}^{y} \prod_{t=0}^{u} \alpha(p(t, x_1, \ldots, x_n))$$

Â¿Qué hace g?

- supongamos que existe un $t \leq y$ tal que $p(t, x_1, \ldots, x_n)$ es verdadero
  - sea t_0 el mínimo tal t
  - $p(t, x_1, \ldots, x_n) = 0$ para todo $t < t_0$
  - $p(t_0, x_1, \ldots, x_n) = 1$
  - $\prod_{t=0}^{u} \alpha(p(t, x_1, \ldots, x_n)) = \begin{cases} 1 & \text{si } u < t_0 \\ 0 & \text{si no} \end{cases}$
  - $g(y, x_1, \ldots, x_n) = \overbrace{\underbrace{1 + 1 + \cdots + 1}_{t_0 \text{ veces}} + 0 + 0 + \cdots + 0}^{y+1 \text{ veces}} = t_0$
  - entonces $g(y, x_1, \ldots, x_n)$ es el mínimo $t \leq y$ tal que $p(t, x_1, \ldots, x_n)$ es verdadero
- si no existe tal t, $g(y, x_1, \ldots, x_n) = y + 1$

Minimización acotada

Notamos

$$\min_{t \leq y} p(t, x_1, \ldots, x_n) = \begin{cases} \text{mínimo } t \leq y \text{ tal que} \\ p(t, x_1, \ldots, x_n) \text{ es verdadero} & \text{si existe tal } t \\ 0 & \text{si no} \end{cases}$$

Teorema

Sea $p : \mathbb{N}^{n+1} \to \{0, 1\}$ *un predicado de una clase PRC* $\mathcal{C}$*. La función*

$$\min_{t \leq y} p(t, x_1, \ldots, x_n)$$

también está en $\mathcal{C}$*.*

Más ejemplos de funciones recursivas primitivas

- x div y es la división entera de x por y

$$\min_{t \leq x}((t+1) \cdot y > x)$$

  Notar que con esta definición 0 div 0 es 0.

- $x \mod y$ es el resto de dividir a x por y
- p_n es el n-ésimo primo ($n > 0$). Se define $p_0 = 0, p_1 = 2, p_2 = 3, p_3 = 5, \ldots$

$$\begin{aligned} p_0 &= 0 \\ p_{n+1} &= \min_{t \leq K(n)} (primo(t) \wedge t > p_n) \end{aligned}$$

  Necesitamos una cota $K(n)$ que sea buena, es decir
  - suficientemente grande y
  - recursiva primitiva

  $K(n) = p_n! + 1$ funciona (ver que $p_{n+1} \leq p_n! + 1$).

Codificación de pares

Definimos la función recursiva primitiva

$$\langle x, y \rangle = 2^x(2 \cdot y + 1) \dot{-} 1$$

Notar que $2^x(2 \cdot y + 1) \neq 0$.

Proposición

Hay una única solución (x, y) *a la ecuación* $\langle x, y \rangle = z$.

Demostración.

- x es el máximo número tal que $2^x | (z + 1)$
- $y = ((z + 1)/2^x - 1)/2$

□

Observadores de pares

Los observadores del par $z = \langle x, y \rangle$ son

- $\ell(z) = x$
- $r(z) = y$

Proposición

Los observadores de pares son recursivas primitivas.

Demostración.

Como $x, y < z + 1$ tenemos que

- $\ell(z) = \min_{x \leq z} ((\exists y)_{\leq z}\ z = \langle x, y \rangle)$
- $r(z) = \min_{y \leq z} ((\exists x)_{\leq z}\ z = \langle x, y \rangle)$

□

Por ejemplo,

- $\langle 2, 5 \rangle = 2^2(2 \cdot 5 + 1) \dot{-} 1 = 43$
- $\ell(43) = 2$
- $r(43) = 5$

Codificación de secuencias

El número de Gödel de la secuencia

$$a_1, \ldots, a_n$$

es el número

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

donde p_i es el i-ésimo primo ($i \geq 1$).

Por ejemplo el número de Gödel de la secuencia

$$1, 3, 3, 2, 2$$

es

$$[1, 3, 3, 2, 2] = 2^1 \cdot 3^3 \cdot 5^3 \cdot 7^2 \cdot 11^2 = 40020750.$$

Propiedades de la codificación de secuencias

Teorema

Si $[a_1, \ldots, a_n] = [b_1, \ldots, b_n]$ *entonces* $a_i = b_i$ *para todo* $i \in \{1, \ldots, n\}$.

Demostración.

Por la factorización única en primos. □

Observar que

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

pero

$$[a_1, \ldots, a_n] \neq [0, a_1, \ldots, a_n]$$

Observadores de secuencias

Los observadores de la secuencia $x = [a_1, \ldots, a_n]$ son

- $x[i] = a_i$
- $|x| =$ longitud de x

Proposición

Los observadores de secuencias son recursivas primitivas.

Demostración.

- $x[i] = \min_{t \leq x}(\neg p_i^{t+1} | x)$
- $|x| = \min_{i \leq x} (x[i] \neq 0 \wedge (\forall j)_{\leq x}(j \leq i \vee x[j] = 0))$

Por ejemplo, □

- $[1,3,3,2,2][2] = 3 = 40020750[2]$
- $[1,3,3,2,2][6] = 0 = 40020750[6]$
- $|[1,3,3,2,2]| = 5 = |40020750|$
- $|[1,3,3,2,2,0]| = |[1,3,3,2,2,0,0]| = 5 = |40020750|$
- $x[0] = 0$ para todo x
- $0[i] = 0$ para todo i

En resumen: codificación y decodificación de pares y secuencias

Teorema (Codificación de pares)

- $\ell(\langle x, y\rangle) = x, r(\langle x, y\rangle) = y$
- $z = \langle \ell(z), r(z)\rangle$
- $\ell(z), r(z) \leq z$
- *la codificación y observadores de pares son p.r.*

Teorema (Codificación de secuencias)

- $[a_1, \ldots, a_n][i] = \begin{cases} a_i & \textit{si } 1 \leq i \leq n \\ 0 & \textit{si no} \end{cases}$
- *si* $n \geq |x|$ *entonces* $[x[1], \ldots, x[n]] = x$
- *la codificación y observadores de secuencias son p.r.*

Minimización no acotada

Definimos la minimización no acotada

$$\min_t p(t, x_1, \ldots, x_n) = \begin{cases} \text{mínimo } t \text{ tal que} \\ p(t, x_1, \ldots, x_n) \text{ es verdadero} & \text{si existe tal } t \\ \\ \uparrow & \text{si no} \end{cases}$$

Funciones computables

Definición

Una función parcial es parcialmente computable si se puede obtener a partir de las funciones iniciales por un número finito de aplicaciones de

- composición,
- recursión primitiva y
- minimización no acotada

Definición

Una función total es computable si se puede obtener a partir de las funciones iniciales por un número finito de aplicaciones de

- composición,
- recursión primitiva y
- minimización propia
  (del tipo $\min_t q(x_1, \ldots, x_n, t)$ donde siempre existe al menos un t tal que $q(x_1, \ldots, x_n, t)$ es verdadero)

Damos un lenguaje de programación que permite definir todas las funciones computables en base a las funciones iniciales, composicion, recursión primitiva , minimización acotada y no acotada.

Teorema

Una función parcial $f : \mathbb{N}^m \to \mathbb{N}$ *es parcialmente computable equivalentemte existe un programa* P *en este lenguaje de programación tal que*

$$f(r_1, \ldots, r_m) = \Psi_P^{(m)}(r_1, \ldots, r_m)$$

para todo $(r_1, \ldots, r_m) \in \mathbb{N}^m$.

La igualdad (del meta-lenguaje) es verdadera si

- *los dos lados están definidos y tienen el mismo valor o*
- *los dos lados están indefinidos*

Las funciones parcialemte computables forman una clase PRC

Teorema

La clase de funciones parcialmente ecomputables es una clase PRC.

Teorema

Sea $f : \mathbb{N}^m \to \mathbb{N}$ una función parcial. Son equivalentes:

1. *f es parcialmente computable en Python*
2. *f es parcialmente computable en C*
3. *f es parcialmente computable en Haskell*
4. *f es parcialmente Turing computable*

No es importante

- qué base usamos para representar a los números
  - usamos representación unaria ($\Sigma = \{\mathbf{B}, 1\}$)
  - pero podríamos haber elegido la binaria ($\Sigma = \{\mathbf{B}, 0, 1\}$)
  - o base 10 ($\Sigma = \{\mathbf{B}, 0, 1, 2, \ldots, 9\}$)
- si permitimos que al terminar la cinta tenga otras cosas escritas además de la salida o solo contenga la salida
- si usamos esta variante de arquitectura:
  - una cinta de entrada (solo de lectura)
  - una cinta de salida (solo de escritura)
  - una o varias cintas de trabajo, de lectura/escritura

Â¡Siempre computamos la misma clase de funciones!

Máquinas de Turing como generadoras

Recordemos que L es c.e. si hay MT $\mathcal{M}$ tal que $\mathcal{L}(\mathcal{M}) = L$.

Consideremos MT con dos cintas, una de ellas de salida, que no puede borrar lo que escribe y escribe palabrsa separadas por $\#$.

Sea $\mathcal{G}(\mathcal{M})$ el lenguaje generado por $\mathcal{M}$. El conjunto $\mathcal{G}(+M)$ puede ser finot o infinito. Si es infinito $\mathcal{M}$ debe computar para siempre. El orden de los elementos no se piuede elegir, ni se puede exigir que sea sin repetición.

Teorema

L es c.e exactamente cuando hay MT $\mathcal{M}$ tal que $\mathcal{G}(\mathcal{M}) = L$.

Lemma

Si MT $\mathcal{M}$ entonces $\mathcal{G}(\mathcal{M})$ es un conjunto c.e.

Sea MT $\mathcal{M}'$ con input x, que simula $\mathcal{M}$. Cada vez que $\mathcal{M}$ escribe un elemento en la salida, $\mathcal{M}'$ lo compara con x. Si $x \in \mathcal{G}(\mathcal{M})$ entoces $\in \mathcal{L}(\mathcal{M})$.

Lemma

Si L es c.e. entonces existe TM $\mathcal{M}'$ tal que $\mathcal{G}(\mathcal{M}') = L$.

Consideremos $w_1, w_2 \ldots$ la enumeración de todas las palabras de $(\Sigma \setminus \{\mathbf{B}\})^*$. Sea $\mathcal{M}'$ con una cinta de salida más que $\mathcal{M}$. La máquina $\mathcal{M}'$ hace : Usando la función de pares $\mathcal{M}$ considera todos los $i, j \geq 1$, dea uno por vez. Computa $\mathcal{M}$ con entrada w_i por j pasos, Si $\mathcal{M}$ acepta w_i, $\mathcal{M}'$ copia w_i en la cinta de salida de $\mathcal{M}'$ y pone un $\#$.

Un lenguaje L es de tipo 0 si es $L = \mathcal{L}(G)$, con $G = (V_N, V_T, P, S)$ de tipo 0, es decir, sin restricciones en P.

Teorema

Para toda gramática tipo 0 hay una MT tal que $\mathcal{L}(G) = \mathcal{L}(\mathcal{M})$.

Proof. Let $G = (V_N, V_T, P, S)$ be a type 0 grammar, with $L = L(G)$. We informally describe a Turing machine T accepting L. T will be nondeterministic. Let

$$T = (K, V_T, \Gamma, \delta, q_0, F), \qquad \text{where } \Gamma = V_N \cup V_T \cup \{B, \#, X\}.$$

The last three symbols are assumed not to be in V_N or V_T. We do not enumerate all the states in K, but designate some of them as it becomes necessary. We allow T, informally, to print the blank B, if necessary.

To begin, T has an input w in V_T^* on its tape. T inserts $\#$ before w, shifting the symbols of w to the right one cell and following it by $\#S\#$. The contents of the tape are now $\#w\#S\#$.

Now T will nondeterministically simulate a derivation in G starting with S. Each sentential form in the derivation will appear in turn between the last two $\#$'s. If some choice of moves leads to a string of terminals there, that string is compared with w. If they are the same, T accepts.

Formally, let T have $\#w\#A_1A_2 \ldots A_k\#$ on its tape. T moves its head through $A_1A_2 \ldots A_k$, nondeterministically choosing a position i and a constant r between 1 and the maximum length of the left side of any production in P. Then T examines the substring $A_iA_{i+1} \ldots A_{i+r-1}$. If $A_iA_{i+1} \ldots A_{i+r-1}$ is the left-hand side of some production in P, it may be replaced by the right-hand side. T may have to shift $A_{i+r}A_{i+r+1} \ldots A_k\#$ either to the left or right to make room or fill up space, should the right side of the production used have a length other than r.†

From this simple simulation of derivations in G, it should be clear that T will print on its tape a string of the form $\#w\#\alpha\#$, α in V^* exactly when $S \xrightarrow[G]{*} \alpha$. Also, if $\alpha = w$, T accepts.

Teorema

Para toda TM $\mathcal{M}$ hay una gramática tipo 0 tal que $\mathcal{L}(\mathcal{M}) = \mathcal{L}(G)$.

Proof. Let L be accepted by $T = (K, \Sigma, \Gamma, \delta, q_0, F)$. We construct a grammar G, which nondeterministically generates two copies of a representation of some word in Σ^* and then simulates the action of T on one copy. If T accepts the word, then G converts the second copy to a terminal string. If T does not accept, the derivation never results in a terminal string. Again we assume without loss of generality that for each q in F and a in Σ, $\delta(q, a)$ is undefined.

Formally, let

$$G = (V_N, \Sigma, P, A_1), \qquad \text{where } V_N = ([\Sigma \cup \{\epsilon\}] \times \Gamma) \cup K \cup \{A_1, A_2, A_3\}$$

and the productions in P are:

1. $A_1 \to q_0 A_2$.
2. $A_2 \to [a, a]A_2$ for each a in Σ.
3. $A_2 \to A_3$.
4. $A_3 \to [\epsilon, B]A_3$.
5. $A_3 \to \epsilon$.
6. $q[a, C] \to [a, D]p$ for each a in $\Sigma \cup \{\epsilon\}$ and each q in K and C in Γ, such that $\delta(q, C) = (p, D, R)$.
7. $[b, E]q[a, C] \to p[b, E][a, D]$ for each C, D, and E in Γ, a and b in $\Sigma \cup \{\epsilon\}$, and q in K, such that $\delta(q, C) = (p, D, L)$.
8. $[a, C]q \to qaq$, $q[a, C] \to qaq$, and $q \to \epsilon$ for each a in $\Sigma \cup \{\epsilon\}$, C in Γ, and q in F.

Using Rules 1 and 2,

$$A_1 \overset{*}{\Rightarrow} q_0[a_1, a_1][a_2, a_2] \ldots [a_n, a_n]A_2,$$

where a_i is in Σ for each i. Suppose that T accepts the string $a_1 a_2 \ldots a_n$. Then for some m, T uses no more than m cells to the right of its input. Using Rule 3, then Rule 4 m times, and finally Rule 5, we have

$$A_1 \overset{*}{\Rightarrow} q_0[a_1, a_1][a_2, a_2] \ldots [a_n, a_n][\epsilon, B]^m$$