

Lenguajes Formales, Autómatas y Computabilidad

Clase Teórica

Máquinas de Turing, funciones parciales computables, conjuntos computablemente enumerables

Segundo Cuatrimestre 2025

Bibliografía

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

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**

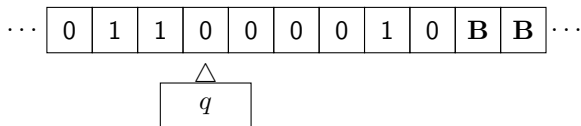
$\delta \subseteq Q \times \Sigma \times \Sigma \cup \{L, R\} \times Q$ es la **tabla de instrucciones** (finita)

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, $\delta : Q \times \Sigma \rightarrow \Sigma \cup \{L, R\} \times Q$

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 .
 - ▶ $B \in \Sigma$, 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. $L, R \notin \Sigma$, $\mathbf{B} \in \Sigma$.

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'

es decir, la tabla de instrucciones δ es un “programa”:

q_0 asignacion de símbolo e ir a q_1

q_1 asignacion de símbolo e ir a q_2

q_2 if condicion mover e ir a q_3

q_2 if otra condicion mover una poosición ir a q_f

q_3 asignacion de símbolo e ir q_2

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 $\vdash_{\mathcal{M}}$ de \mathcal{M})

Supongamos $X_1 \dots X_{i-1} q X_i \dots X_n$ es una descripción instantánea.

Si $(q, X_i, L, p) \in \delta$, entonces para $i > 1$,

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

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

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

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

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

\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 al estado final,

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

Definición (función computada por \mathcal{M})

La función $f : (\Sigma \setminus \{\mathbf{B}\})^* \rightarrow (\Sigma \setminus \{\mathbf{B}\})^*$ computada por una máquina de Turing determinística $\mathcal{M} = (Q, \Sigma, \delta, q_0, q_f)$ es la función tal que para cada $w \in (\Sigma \setminus \{\mathbf{B}\})^*$ $f(w) = y$ definido por

$$q_0 w \stackrel{*}{\vdash}_{\mathcal{M}} \alpha q_f \beta$$

donde $\alpha = \gamma \mathbf{B} y$, $\alpha\beta\gamma \in \Sigma^*$

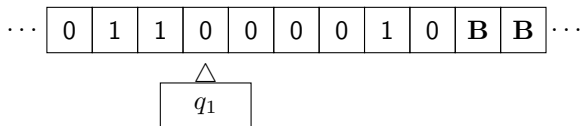
Ejemplo de un movimiento

Supongamos un alfabeto $\Sigma = \{0, 1\}$.

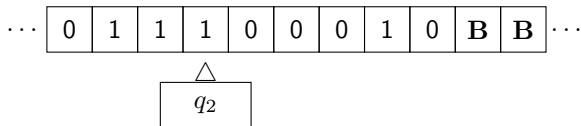
Una máquina con esta tabla de instrucciones:

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

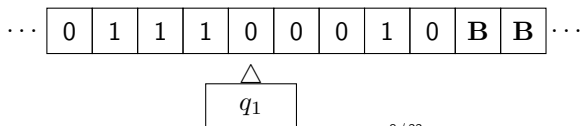
Si empieza en esta configuración



pasa a



y luego a



Ejemplo

Sea $\mathcal{M} = (Q, \Sigma, \delta, q_0, q_f)$ con

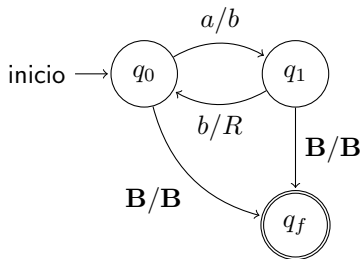
$\Sigma = \{\mathbf{B}, a, b\}$

$Q = \{q_0, q_1, q_f\}$

$\delta =$

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

Visto como autómata



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

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

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

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

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.

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, \dots, c_{k-1} de \mathcal{M}_1 y las k cabezas.

En la cinta de \mathcal{M}_2 indentificaré una posición 0 y usaré la cinta a derecha. Cuando la cabeza de la cinta de \mathcal{M}_2 está posición t ,

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

si $(t \bmod 2k) \geq k$ y el símbolo en posición t es distinto de **B** indica que la cabeza de la cinta $c_{(t-k) \bmod 2k}$ está 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 determinística \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 $1, 2, \dots$. 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. Sea \mathcal{M}_2 con 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.

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

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.



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

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

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

- ▶ la tupla (x_1, \dots, x_n) se representa como

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

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 parcial computables y totalmente computables

Trabajaremos con funciones $f : \mathbb{N}^n \rightarrow \mathbb{N}$.

Definition

Una función $f : \mathbb{N}^n \rightarrow \mathbb{N}$ es **parcial computable** es una función que puede estar indefinida para algunos (tal vez ninguno; tal vez todos) sus argumentos. Para los argumentos en que está definida, el cómputo lo realiza una máquina de Turing.

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

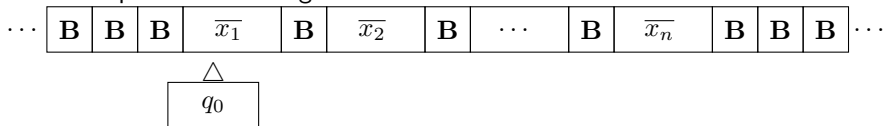
El conjunto de argumentos para los que f está definida se llama **dominio** de f ,

$$\text{dom}(f) = \{(x_1, \dots, x_n) : f(x_1, \dots, x_n) \downarrow\}$$

Decimos que f es **totalmente computable** si $\text{dom}(f) = \mathbb{N}^n$.

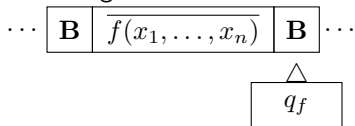
Funciones parciales computables

Una función $f : \mathbb{N}^n \rightarrow \mathbb{N}$ es **parcial 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):

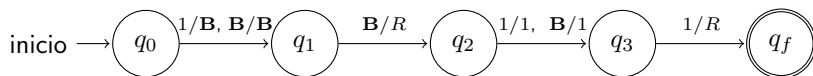
- ▶ si $f(x_1, \dots, x_n) \downarrow$ entonces siguiendo sus instrucciones en δ llega a una configuración final de la forma



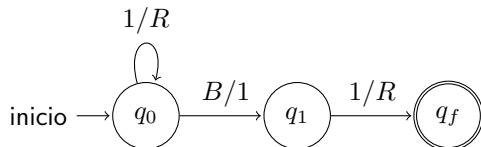
(quizá algo más en la cinta)

- ▶ si $f(x_1, \dots, x_n) \uparrow$ entonces nunca termina en el estado q_f .

Cómputo de la función $f(x) = 0$



Cómputo de la función $f(x) = x + 1$



Cálculo de la función $f(x) = 2x$

Recordar que para representar x escribimos $x + 1$ 1s.

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:

$\text{BBB} \underbrace{1 \dots 1}_n \text{B} \underbrace{1 \dots 1}_{2m} \text{BBB}$ para algún $n > 0, m \geq 0, n + m - 1 = \text{entrada}$

Si $n = 1$ pongo un 1 más a la derecha y termina en q_f con

$\text{BBB} \text{B} \underbrace{1 \dots 1}_{2m+1} \text{BBB}$

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

y vuelvo al paso 1

Caracterización de funciones computables

Teorema (Turing 1936)

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

- ▶ composición,
- ▶ recursión primitiva y
- ▶ minimización no acotada

Teorema (Turing 1936)

Una función totalmente 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 $\min_t q(x_1, \dots, x_n, t)$ donde siempre existe al menos un t tal que $q(x_1, \dots, x_n, t)$ es verdadero)

A continuación definimos las funciones iniciales las funciones obtenidas

Funciones Iniciales

sucesor: $s(x) = x + 1,$

inicialización: $n(x) = 0,$

proyecciones: $u_i^n(x_1, \dots, x_n) = x_i$ para $i \in \{1, \dots, n\}$

Composición

Sean $f : \mathbb{N}^k \rightarrow \mathbb{N}$ y $g_1, \dots, g_k : \mathbb{N}^n \rightarrow \mathbb{N}$. La función $h : \mathbb{N}^n \rightarrow \mathbb{N}$ se obtiene a partir de f y g_1, \dots, g_k por **composición** si

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

Ejemplos:

- ▶ Funciones constantes, $f : \mathbb{N} \rightarrow \mathbb{N}$, $f(x) = s(\dots(s(s(x))))$.
- ▶ Función *swap* : $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \times \mathbb{N}$, $swap(x, y) = (u_2^2(x, y), u_1^2(x, y))$
- ▶ Función que da la suma de una constante con alguno de sus argumentos.

Recursión primitiva

La función $h : \mathbb{N} \rightarrow \mathbb{N}$ se obtiene de $g : \mathbb{N}^2 \rightarrow \mathbb{N}$ por **recursión primitiva** si

$$\begin{aligned}h(0) &= k \\h(t+1) &= g(t, h(t))\end{aligned}$$

Otro orden de argumentos también funciona: Se puede definir $g(t, y)$ como lo dimos y luego $g'(t, y) = g(u_2^2(t, y), u_1^2(t, y))$.

Más general:

Sean $f : \mathbb{N}^n \rightarrow \mathbb{N}$ y $g : \mathbb{N}^{n+2} \rightarrow \mathbb{N}$. La función $h : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ se obtiene a partir de f y g por **recursión primitiva** si

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

En este contexto, una función 0-aria es una constante k .

Si h es 1-aria y $t = 0$, entonces $h(t) = k = s^{(k)}(n(t))$.

Ejemplos

$$\begin{aligned} suma(x, 0) &= u_1^1(x) \\ suma(x, y + 1) &= g(y, suma(x, y), x), \\ g(z, v, w) &= s(u_3^2(z, v, w)) \end{aligned}$$

$$\begin{aligned} mult(x, 0) &= n(x) \quad \text{escribiremos } x \cdot y \\ mult(x, y + 1) &= g'(y, mult(x, y), x) \\ g'(z, v, w) &= suma(u_3^1(z, v, w), u_3^2(z, v, w)) \end{aligned}$$

$$\begin{aligned} predecesor(0) &= n(x) \\ predecesor(x + 1) &= u_2^1(x, predecesor(x)) \end{aligned}$$

$$\begin{aligned} menos(x, 0) &= u_1^1(x), \\ menos(x, y + 1) &= g''(y, menos(x, y), x) \\ g''(z, v, w) &= predecesor(u_3^2(z, v, w)) \end{aligned}$$

Notar que si $x < y$ en tonces $menos(x, y)$ es 0

Otras: $x!$, x^y , etc

Predicados y conectivos

Los **predicados** son funciones $\mathbb{N} \rightarrow \mathbb{N}$ que toman valores en $\{0, 1\}$.

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

Definimos el predicado $\alpha(0) = 1$, y $\alpha(x) = 0$ para $x > 0$ así:

$$\alpha(0) = s(n(0)), \quad \alpha(x+1) = n(x)$$

Definimos

$x \leq y$ se define como $\alpha(x \dot{-} y)$.

$\neg p$ se define como $\alpha(p)$

$p \wedge q$ se define como $p \cdot q$

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

Definición por casos

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

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

Sumatorias y productorias

Si $f : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ está definida por iniciales, composicion, recursion primitiva y minimización acotada, entonces estas otras también:

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

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

ya que

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

Lo mismo para h con \cdot en lugar de $+$.

Una definición muy parecida se usa para la sumatoria y la productoria desde 1.

Cuantificadores acotados

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

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

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

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

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

Más ejemplos

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

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

Notar que con esta definición $0|0$.

- ▶ $\text{primo}(x)$ sii x es primo.

Minimización acotada

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

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

¿Qué hace g ?

Supongamos que existe un $t \leq y$ tal que $p(t, x_1, \dots, x_n)$ es verdadero

► sea t_0 el mínimo tal t

► $p(t, x_1, \dots, x_n) = 0$ para todo $t < t_0$

► $p(t_0, x_1, \dots, x_n) = 1$

► $\prod_{t=0}^u \alpha(p(t, x_1, \dots, x_n)) = \begin{cases} 1 & \text{si } u < t_0 \\ 0 & \text{si no} \end{cases}$

► $g(y, x_1, \dots, x_n) = \underbrace{1 + 1 + \dots + 1}_{t_0 \text{ veces}} + \overbrace{0 + 0 + \dots + 0}^{y+1 \text{ veces}} = t_0$

► entonces $g(y, x_1, \dots, x_n)$ es el mínimo $t \leq y$ tal que $p(t, x_1, \dots, x_n)$ es verdadero

Si no existe tal t , $g(y, x_1, \dots, x_n) = y + 1$

Minimización acotada

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

Ejemplo:

$x \text{ div } y$ es la división entera de x por y

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

Notar que con esta definición $0 \text{ div } 0$ es 0 .

Minimización no acotada

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

Si para todo (x_1, \dots, x_n) existe t tal que $p(t, x_1, \dots, x_n)$ la minimización se llama **propia**.

Ejemplos de minimización no acotada

- Primer primo gemelo mayor que n
(dos números primos son gemelos si son dos impares sucesivos).
- El k -ésimo número $t_k = (k)(k+1)/2$ que tiene al menos n factores primos distintos
- El menor factorial divisible por n primos distintos
- La cantidad de pasos que una máquina de Turing determinística con entrada dada n necesita para llegar al estado final.

El lenguaje \mathcal{S}^{++}

Damos un lenguaje de programación \mathcal{S}^{++} que permite definir todas las funciones computables en base a las funciones iniciales, composicion, recursión primitiva , minimización acotada y no acotada.