Lógica y Computabilidad

Gianfranco Zambonni

14 de octubre de 2019

Índice

Parte I
Computabilidad

1. Máquina de Turing

Una máquina de Turing está compuesta por:

- Una **cinta infinita** dividida en celdas que contienen un símbolo de un alfabeto Σ. En esta materia, Σ siempre contiene al símbolo $*$, que representa el blanco, y nunca contiene a L ni a R, que son las etiquetas usadas para indicar al cabezal hacia que lado debe moverse.
- El **cabezal** lee un símbolo y, dependiendo del estado de la máquina, puede escribir uno nuevo o moverse una posición a la derecha o una a la izquierda. Cuando completa una acción, cambia el estado de la máquina.
- Una **tabla finita de instrucciones** que, dado un estado y el símbolo que lee el cabezal, indica que acción debe ser tomada y cual es el estado al que se tiene que pasar.

1.1. Tabla de instrucciones

Cada instrucción es una tupla $(q, s, a, q') \in Q \times \Sigma \times (\Sigma \cup \{L, R\}) \times Q$ tal que:

- $q, q' \in Q$ son estados de la máquina de turing. q es el estado necesario para ejecutar la instrucción y q' el estado en el que queda la máquina después de ejecutarla.
- $s \in \Sigma$ es el símbolo que se tiene que leer cuando la máquina está en q para que la instrucción sea ejecutada.
- $a \in \Sigma \cup \{L, R\}$ es la acción a realizar. Puede ser escribir un símbolo del álfabeto Σ o mover el cabezal hacia alguno de los lados.

Entonces, la tupla (q, s, a, q') 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'*".

1.2. Definición mátematica

Una máquina de Turing $\mathcal{M}$ es una tupla (Σ, Q, T, q_0, q_f) donde:

- Σ es un conjunto finito de símbolos ($L, R \in \Sigma$ y $* \in \Sigma$)
- Q es un conjunto finito de **estados**, de los cuales dos son:
  - el **estado inicial** q_0
  - y el **estado final** q_f.
- $T \subseteq Q \times \Sigma \times \Sigma$ es la **tabla de instrucciones**.

Hay dos tipos de máquinas de turing:

- **Deterministicas:** Es cuando no hay dos instrucciones que tengan el mismo estado inicial y necesiten leer el mismo símbolo, es decir:

$$\nexists(q_1, s_1, a_1, q_1'), (q_2, s_2, a_2, q_2') \in T \text{ tal que } q_1 = q_2 \ \wedge \ s_1 = s_2$$

- **No deterministicas:** Cuando no es deterministica, osea que cabe la posibilidad que se ejecuten más de una instrucción en un paso y la máquina este en dos o más estados simultáneamente.

$$\exists(q_1, s_1, a_1, q_1'), (q_2, s_2, a_2, q_2') \in T \text{ tal que } q_1 = q_2 \ \wedge \ s_1 = s_2$$

1.3. Representación de números y tuplas

1.3.1. Números naturales

Sea $\sigma = \{*, 1\}$, representaremos a los números naturales en unario (como si usasemos palitos). La representación $\bar{x}$ de $x \in \mathbb{N}$, consta de $x + 1$ palitos.

$$\bar{x} = \underbrace{1...1}_{x+1}$$

Ejemplo: El 0 es "1", el 1 es "11", el 2 es "111", etc.

1.3.2. Tuplas

Las tuplas $(x_1, \ldots, x_n)$ las representamos como una lista de (representaciones de) x_i separados por un blanco $(*)$.

$$*\bar{x_1} * \bar{x_2} * \cdots * \bar{x_n} *$$

Ejemplo: La tupla (1,2) sería $*11 * 111*$

1.4. Funciones parciales

Sea $f : \mathbb{N}^n \to \mathbb{N}$, f es una función parcial si está definida para algunos (tal vez ninguno; tal vez todos) sus argumentos.

Notación:

- $f(x_1, \ldots, x_n) \downarrow$: f está definida para la tupla $(x_1, \ldots, x_n)$ y, en este caso, $f(x_1, \ldots, x_n)$ es un natural.
- $f(x_1, \ldots, x_n) \uparrow$: f se indefine para la tupla $(x_1, \ldots, x_n)$.

Dominio: Conjunto de argumentos para los que f está definida y se nota $\text{dom}(f)$.

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

Función Total: f es total si está definida para todos sus posibles argumentos:

$$\text{dom}(f) = \mathbb{N}^n$$

1.4.1. Cómputo de funciones parciales en máquinas de Turing

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

- Si $f(x_1, \ldots, x_n) \downarrow$ entonces, siguiendo sus instrucciones en T, llega al estado q_f,
- Si $f(x_1, \ldots, x_n) \uparrow$ nunca termina en el estado q_f

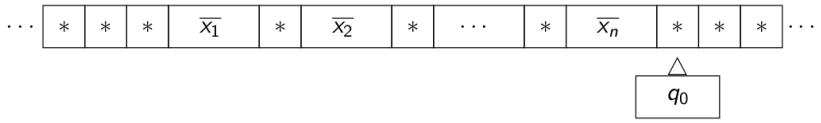

Figura 1: Estado inicial de la máquina de turring

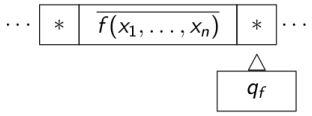

Figura 2: Estado final de la máquina de turing

1.4.2. Poder de cómputo

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

1. f es computable en Java
2. f es computable en C
3. f es computable en Haskell
4. f es Turing computable.

2. Funciones primitivas recursivas y clases PRC

2.1. Funciones iniciales

Función calculable de manera efectiva: Son funciones que podemos escribir combinando, de alguna manera, funciones más simples que sabemos que ya sabemos que son efectivas.

La idea es que, dado un **conjunto inicial** de funciones, podamos combinarlas de ciertas formas para conseguir nuevas funciones que permitan realizar cálculos más complejos.

Funciones iniciales:

- $s(x) = x + 1$
- $n(x) = 0$
- **Proyecciones:** $u_i^n(x_1, \ldots, x_n) = x_i$ para $i \in \{1, \ldots, n\}$

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

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

En este contexto, una constante k puede ser definida como:

$$h(t) = s^{(k)}(n(t))$$

Recursión primitiva: $h : \mathbb{N}^{n+1} \to \mathbb{N}$ 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:

$$h(x_1, \ldots, x_n, 0) = f(x_1, \ldots, x_n)$$
$$h(x_1, \ldots, x_n, t+1) = g(h(x_1, \ldots, x_n, t), x_1, \ldots, x_n, t)$$

2.2. Clases Primitive Recursive Closed (PRC)

Función primitiva recursiva: Función que se puede obtener a partir de las funciones iniciales por un número finito de aplicaciones de composición y recursión primitiva.

Clase PRC: Una clase $\mathcal{C}$ de funciones totales es **PRC** si

- Las funciones iniciales están en $\mathcal{C}$.
- 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 $\mathcal{C}$.

Corolario: La clase de funciones primitivas recursivas es una clase *PRC*.

Teorema: La clase de funciones totales Turing computables es una clase *PRC*.

Teorema: Una función es p.r. si y solo si pertenece a toda clase *PRC*.

DEMOSTRACION

$\Leftarrow$) Si una función f pertenece a todas las clases PRC, en particular, pertenece a la clase de funciones primitiva recursivas. Por lo tanto f es primitiva recursiva.

$\Rightarrow$) Sea f una función primitiva recursiva y sea $\mathcal{C}$ una clase PRC. Como f es p.r, hay una lista $f_1, \ldots, f_n$ 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 con $j < i$ (luego tambien está en $\mathcal{C}$)

Entonces todas las funcionesde la lista están en $\mathcal{C}$. En particular $f_n \in \mathcal{C}$.

■

Corolario: La clase de funciones primitivas recursivas es la clase PRC más chica.

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

DEMOSTRACION

Sabemos que la clase de funciones totales Turing computables es PRC. Por el teorema, anterior, si f es p.r, entonces f pertence a todas las clases PRC, en particular a la clase de funciones totales Turing computables.

■

2.3. Funciones primitivas recursivas básicas:

- Todas las constantes k están en todas las clases PRC.

DEMOSTRACION

$$k = k(t) = s^{(k)}(n(t))$$

■

- $suma(x, y) = x + y$

DEMOSTRACION

$$suma(x,0) = u_1^1(x)$$
$$suma(x,y+1) = g(suma(x,y),x,y) = s(u_1^3(suma(x,y),x,y))$$

■

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

Para las demostraciones faltantes ir a

2.3.1. 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 primitivos recursivos son aquellos representados por funciones primitivas recursivas en $\{0,1\}$.

2.3.2. 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 $\mathcal{C}$

DEMOSTRACION

- $\neg p = \alpha(p)$
- $p \wedge q = p \cdot q$
- $p \vee q = \neg(\neg p \wedge \neg q)$

■

Corolario: Si p y q son predicados primitivos recursivos también lo son los predicados $\neg p$, $p \vee q$ y $p \wedge q$.

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

2.3.3. Definición por casos

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 $\mathcal{C}$

DEMOSTRACION

$$\begin{aligned} f(x_1, \ldots, x_n) = & g_1(x_1, \ldots, x_n) \cdot p_1(x_1, \ldots, x_n) + \ldots \\ & + g_m(x_1, \ldots, x_n) \cdot p_m(x_1, \ldots, x_n) \\ & + h(x_1, \ldots, x_n) \cdot (\neg p_1(x_1, \ldots, x_n) \wedge \cdots \wedge \neg p_m(x_1, \ldots, x_n)) \end{aligned}$$

Si algún p_i es verdadero, valdrá 1 y el resto 0 porque todos los predicados son mutuamente excluyentes. Al evaluar f, obtendremos

$$f(x_1, \ldots, x_n) = g_i(x_1, \ldots, x_n) * p_i(x_1, \ldots, x_n) = g_i(x_1, \ldots, x_n) \cdot 1 = g_i(x_1, \ldots, x_n)$$

v321v
Si todos los predicados son falsos, entonces todas las g_i se multiplican por cero y, además vale la condición *si no* que es la conjuncion de sus negaciones. Y h es multiplicado por 1.

■

2.3.4. Sumatorias, productorias

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:

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

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

DEMOSTRACION

$$g(0, x_1, \ldots, x_n) = f(0, x_1, \ldots, x_n)$$
$$g(t+1, x_1, \ldots, x_n) = g(t, x_1, \ldots, x_n) + f(t+1, x_1, \ldots, x_n)$$

Para h hay que hacer lo mismo pero multiplicando en vez de sumar.

■

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:

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

La demostración es la misma que la anterios, pero los resultados de los casos bases valen 0 y 1, respectivamente.

2.3.5. Cuantificadores acotados

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

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

- $p(0, x_1, \ldots, x_n)$ 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 si y solo si:

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

  $\vdots$

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

De la misma manera se pueden definir el existencial y el para todo con $< y$ en vez de $\leq y$.

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

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

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

DEMOSTRACION

$(\forall t)_{\leq y} p(t, x_1, \ldots, x_n)$ si y solo si $\prod_{t=1}^{y} f(t, x_1, \ldots, x_n)) = 1$

$(\exists t)_{\leq y} p(t, x_1, \ldots, x_n)$ si y solo si $\sum_{t=1}^{y} f(t, x_1, \ldots, x_n)) \neq 0$

Definimos los cuantificadores en función de la sumatoria, productoria y la comparación de igualdad que están en $\mathcal{C}$ (Ver Seccion 2.3.4 y apéndice A.1). Entonces el para todo y el existencial acotados por $\leq$ pertenece a $\mathcal{C}$.

■

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

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

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

DEMOSTRACION

$(\forall t)_{<y} p(t, x_1, \ldots, x_n)$ si y solo si $(\forall t)_{\leq y}(t = y \vee p(t, x_1, \ldots, x_n)$

$(\exists t)_{<y} p(t, x_1, \ldots, x_n)$ si y solo si $(\forall t)_{\leq y}(t \neq y \wedge p(t, x_1, \ldots, x_n)$

■

2.3.6. Minimización acotada

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

Teorema: Sean $p : \mathbb{N}^{n+1} \to \{0, 1\}$ un predicado de una clase PRC $\mathcal{C}$. Entonces la función $\min_{t \leq y} p(t, x_1, \ldots, x_n)$ también están en $\mathcal{C}$

DEMOSTRACION

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

■

2.3.7. Codificación de tuplas

Definimos la función primitiva recursiva

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

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

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

Observadores: Sea $z = \langle x, y\rangle$:

- $l(z) = x$
- $r(z) = y$

Proposición: Los observadores de pares son primitivas recursivas

DEMOSTRACION

Como x e y son menores a $z + 1$ tenemos que:

- $l(z) = \min\limits_{x\leq z}\Big((\exists y)_{\leq z}\ \ z = \langle x, y\rangle\Big)$
- $r(z) = \min\limits_{y\leq z}\Big((\exists x)_{\leq z}\ \ z = \langle x, y\rangle\Big)$

■

2.3.8. 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 $(i \geq 1)$

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

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

Observadores: Sea $x = [a_1, \ldots, a_n]$:

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

Proposición: Los observadores de secuencias son primitivas recursivas

DEMOSTRACION

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

■

3. Funciones S-Computables

4. Funciones no computables y conjuntos c.e.

Parte II

Lógica

5. Lógica Proposicional

6. Sistemas deductivos para lógica propisicional

7. Lógica de primer orden

8. Completitud y compacidad para lógica de primer orden

A. Operaciones primitivas recursivas

A.1. Igualdad

$x = y$

DEMOSTRACION

$$igual(x, y) = \alpha(x \dot{-} y)$$

■

A.2. Divisor

$y|x$ si y solo y divide a x

DEMOSTRACION

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

■