

# Lenguajes Formales, Autómatas y Computabilidad

## Funciones primitivas recursivas

Sabrina Silvero

Departamento de Computación,  
Facultad de Ciencias Exactas y Naturales,  
Universidad de Buenos Aires

Segundo cuatrimestre 2024

# Empecemos por el principio

Una manera de formalizar la idea de función calculable de manera efectiva<sup>1</sup>:

- Empezar por funciones muy simples, efectivas intuitivamente
- Si mezclamos de alguna manera efectiva dos o más funciones que ya eran efectivas, entonces obtenemos una función calculable de manera efectiva

---

<sup>1</sup>Llamaremos función efectiva cuando podamos programarla en una computadora.

# Empecemos por el principio

Una manera de formalizar la idea de función calculable de manera efectiva<sup>1</sup>:

- Empezar por funciones muy simples, efectivas intuitivamente
- Si mezclamos de alguna manera efectiva dos o más funciones que ya eran efectivas, entonces obtenemos una función calculable de manera efectiva

## Funciones iniciales

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

Hay infinitas funciones  $u_i^n$ . Por lo tanto, hay infinitas funciones iniciales.

---

<sup>1</sup>Llamaremos función efectiva cuando podamos programarla en una computadora.

# Composición

## Definición

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

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

A aplicar los conceptos aprendidos nomas :)



## Ejercicio 1

Determinar si cada una de las siguientes funciones se pueden obtener por composición a partir de funciones iniciales:

a.  $uno : \mathbb{N} \rightarrow \mathbb{N}$ ,  $uno(x) = 1$

## Ejercicio 1

Determinar si cada una de las siguientes funciones se pueden obtener por composición a partir de funciones iniciales:

a.  $uno : \mathbb{N} \rightarrow \mathbb{N}$ ,  $uno(x) = 1$

Tomando en el esquema de composición:

- $k = n = 1$
- $f = s$
- $g_1 = n$

tenemos que

$$uno(x) = 1 = s(n(x))$$

para todo  $x \in \mathbb{N}$ .

## Ejercicio 1

Determinar si cada una de las siguientes funciones se pueden obtener por composición a partir de funciones iniciales:

- b.  $id : \mathbb{N} \rightarrow \mathbb{N}$ ,  $id(x) = x$

## Ejercicio 1

Determinar si cada una de las siguientes funciones se pueden obtener por composición a partir de funciones iniciales:

b.  $id : \mathbb{N} \rightarrow \mathbb{N}$ ,  $id(x) = x$

$id$  es simplemente la proyección de una componente, es decir,

$$id(x) = x = u_1^1(x)$$

$$x \in \mathbb{N}.$$

## Ejercicio 1

Determinar si cada una de las siguientes funciones se pueden obtener por composición a partir de funciones iniciales:

## Ejercicio 1

Determinar si cada una de las siguientes funciones se pueden obtener por composición a partir de funciones iniciales:

- c.  $f_1 : \mathbb{N}^2 \rightarrow \mathbb{N}$ ,  $f_1(x, y) = f(y, x)$  donde  $f : \mathbb{N}^2 \rightarrow \mathbb{N}$  se puede obtener a partir de funciones iniciales.

## Ejercicio 1

Determinar si cada una de las siguientes funciones se pueden obtener por composición a partir de funciones iniciales:

- c.  $f_1 : \mathbb{N}^2 \rightarrow \mathbb{N}$ ,  $f_1(x, y) = f(y, x)$  donde  $f : \mathbb{N}^2 \rightarrow \mathbb{N}$  se puede obtener a partir de funciones iniciales.

Tomando en el esquema de composición:

- $k = n = 2$
- $g_1 = u_2^2$
- $g_2 = u_1^2$

tenemos que,

$$f_1(x, y) = f(y, x) = f(g_1(x, y), g_2(x, y))$$

$$\forall (x, y) \in \mathbb{N}^2.$$

# Recursión primitiva

## Definición

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

$$h(x_1, \dots, x_n, 0) = f(x_1, \dots, x_n)$$

$$h(x_1, \dots, x_n, t + 1) = g(h(x_1, \dots, x_n, t), x_1, \dots, x_n, t)$$

*Nota:* En este contexto vamos a considerar que una función 0-aria es una constante.

## Funciones p.r.

Una función es **primitiva recursiva (p.r.)** si se puede obtener a partir de las funciones iniciales por un número finito de aplicaciones de composición y recursión primitiva.

## Ejercicio 2

¿Cómo se comportan las siguientes funciones definidas por recursión primitiva a partir de  $f$  y  $g$ ?

a.

- $f = n = 0$
- $g = u_2^2$
- $h : \mathbb{N} \rightarrow \mathbb{N}$  queda dada por:

$$\begin{aligned} h(0) &= f = 0 \\ h(t+1) &= g(h(t), t) = u_2^2(h(t), t) = t \end{aligned}$$

## Ejercicio 2

¿Cómo se comportan las siguientes funciones definidas por recursión primitiva a partir de f y g?

a.

- $f = n = 0$
- $g = u_2^2$
- $h : \mathbb{N} \rightarrow \mathbb{N}$  queda dada por:

$$\begin{aligned} h(0) &= f = 0 \\ h(t+1) &= g(h(t), t) = u_2^2(h(t), t) = t \end{aligned}$$

Como el predecesor de naturales:

$$h(x) = x - 1$$

## Ejercicio 2

¿Cómo se comportan las siguientes funciones definidas por recursión primitiva a partir de  $f$  y  $g$ ?

b.

- $n = 1$
- $f = s$
- $g(x,y,z) = x + z$
- $h : \mathbb{N}^2 \rightarrow \mathbb{N}$  queda dada por:

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

## Ejercicio 2

¿Cómo se comportan las siguientes funciones definidas por recursión primitiva a partir de  $f$  y  $g$ ?

b.

- $n = 1$
- $f = s$
- $g(x,y,z) = x + z$
- $h : \mathbb{N}^2 \rightarrow \mathbb{N}$  queda dada por:

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

Notar que para todo  $x, y \in \mathbb{N}$

$$h(x, y) = (x + 1) + \sum_{i=1}^y (i - 1)$$

# Predicados primitivos recursivos

Los **predicados p.r.** son simplemente funciones que toman valores entre 0 y 1.

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

# Predicados primitivos recursivos

Los **predicados p.r.** son simplemente funciones que toman valores entre 0 y 1.

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

Por ejemplo, el predicado  $x \leq y$  es p.r. porque se puede definir como:

$$\alpha(x - y)$$

Con  $\alpha$ :

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

# Predicados primitivos recursivos

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

## Demostración



TAREA PARA LA CASA

## Definición por casos(2)

Sean  $h, g : \mathbb{N}^n \rightarrow \mathbb{N}$  funciones p.r. y sea  $p : \mathbb{N}^n \rightarrow \{0, 1\}$  un predicado p.r. La función:

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

es p.r.

### Demostración

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

¿Nos tomamos un break?



# Codificación de pares

Definimos la función primitiva recursiva

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

## 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  $\langle x, y \rangle = z$  son:

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

## Proposición

Los observadores de pares son primitivas recursivas.

Ejemplo:

- $\langle 2, 5 \rangle = 2^2(2.5 + 1) - 1 = 43$
- $l(43) = 2$
- $r(43) = 5$

## Ejercicio 3

Mostrar que la siguiente función es primitiva recursiva:

$$\begin{aligned}fib(0) &= 0 \\fib(1) &= 1 \\fib(n + 2) &= fib(n + 1) + fib(n)\end{aligned}$$

## Ejercicio 3

Mostrar que la siguiente función es primitiva recursiva:

$$\begin{aligned}fib(0) &= 0 \\fib(1) &= 1 \\fib(n+2) &= fib(n+1) + fib(n)\end{aligned}$$

## Ejercicio 3

Pistas:

- Si sólo podemos acceder al "valor anterior", necesitaríamos guardar en el "valor anterior", toda la información necesaria para generar el valor actual.

## Ejercicio 3

Pistas:

- Si sólo podemos acceder al "valor anterior", necesitaríamos guardar en el "valor anterior", toda la información necesaria para generar el valor actual.
- ¿Cómo podemos "codificar" varios valores en uno sólo?

## Ejercicio 3

Pistas:

- Si sólo podemos acceder al "valor anterior", necesitaríamos guardar en el "valor anterior", toda la información necesaria para generar el valor actual.
- ¿Cómo podemos "codificar" varios valores en uno sólo?
- Quizás con una función auxiliar que guarde la información necesaria

## Ejercicio 3

Solución:

- Armar una auxiliar  $fib'$  que ”codifique” la información necesaria (los 2 pasos anteriores)
- Probar que  $fib'$  es p.r.
- Reescribir  $fib$  como composición de funciones pr (usando  $fib'$ )
- Probar que  $fib$  es p.r.

## Ejercicio 3

Sea  $fib'$ :

$$fib'(n) = \langle fib(n), fib(n + 1) \rangle$$

Osea nos guardamos toda la información que necesitamos.

## Ejercicio 3

Sea  $fib'$ :

$$fib'(n) = \langle fib(n), fib(n + 1) \rangle$$

Osea nos guardamos toda la información que necesitamos.

Y ahora veamos si  $fib'$  es primitiva recursiva.

Analicemos primero el valor de la función en 0.

$$fib'(0) = \langle fib(0), fib(1) \rangle = \langle 0, 1 \rangle \text{ (una constante, es p.r.)}$$

## Ejercicio 3

Veamos ahora cómo luce  $fib'$  para el caso  $t + 1$ .

## Ejercicio 3

Veamos ahora cómo luce  $\text{fib}'$  para el caso  $t + 1$ .

$$\text{fib}'(t + 1) = \langle \text{fib}(t + 1), \text{fib}(t + 2) \rangle \text{ (por definición de } \text{fib}')$$

## Ejercicio 3

Veamos ahora cómo luce  $\text{fib}'$  para el caso  $t + 1$ .

$$\begin{aligned}\text{fib}'(t + 1) &= \langle \text{fib}(t + 1), \text{fib}(t + 2) \rangle \text{ (por definición de } \text{fib}') \\ &= \langle \text{fib}(t + 1), \text{fib}(t + 1) + \text{fib}(t) \rangle \text{ (por definición de } \text{fib})\end{aligned}$$

## Ejercicio 3

Veamos ahora cómo luce  $fib'$  para el caso  $t + 1$ .

$$fib'(t + 1) = \langle fib(t + 1), fib(t + 2) \rangle \text{ (por definición de } fib')$$

$$= \langle fib(t + 1), fib(t + 1) + fib(t) \rangle \text{ (por definición de } fib)$$

$$= \langle r(fib'(t)), r(fib'(t)) + l(fib'(t)) \rangle \text{ (por definición de } fib')$$

(composición, sumas, crear y observar tuplas y todo sobre  $fib'(t)$ ).

## Ejercicio 3

Sabemos que  $fib'$  es p.r.

## Ejercicio 3

Sabemos que  $\text{fib}'$  es p.r.

- Reescribimos  $\text{fib} : \text{fib}(n) = l(\text{fib}'(n))$

## Ejercicio 3

Sabemos que  $fib'$  es p.r.

- Reescribimos  $fib : fib(n) = l(fib'(n))$
- $fib$  es composición de funciones p.r.

## Ejercicio 3

Sabemos que  $\text{fib}'$  es p.r.

- Reescribimos  $\text{fib} : \text{fib}(n) = l(\text{fib}'(n))$
- $\text{fib}$  es composición de funciones p.r.
- Por lo tanto,  $\text{fib}$  es p.r



**VAMO A BAJAR UN  
CAMBIO**

ONO



**TRANQUILOS**



**YA FALTA POCO**

## Sumatorias y productorias

Si  $f : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$  son funciones p.r., entonces también lo son las siguientes funciones:

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

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

# Sumatorias y productorias

Si  $f : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$  son funciones p.r., entonces también lo son las siguientes funciones:

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

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

## Demostración

-Cuando esté en casa seré productivo, haré todas mis tareas.  
-Yo apenas llego la a casa.



# Codificación de secuencias

El **número de Gödel** de la secuencia

$$a_1, \dots, a_n$$

# Codificación de secuencias

El **número de Gödel** de la secuencia

$$a_1, \dots, a_n$$

es el número:

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

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

# Codificación de secuencias

El **número de Gödel** de la secuencia

$$a_1, \dots, a_n$$

es el número:

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

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

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

# Propiedades de la codificación de secuencias

## **Teorema**

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

# Propiedades de la codificación de secuencias

## **Teorema**

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

## **Demostración**

Por la factorización única en primos.

# Propiedades de la codificación de secuencias

## Teorema

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

## Demostración

Por la factorización única en primos.

## Observadores

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

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

Ejemplos:

- $[1, 3, 3, 2, 2][2] = 3 = 40020750[2]$
- $[1, 3, 3, 2, 2][6] = 0 = 40020750[6]$
- $\|[1, 3, 3, 2, 2]\| = 5 = |40020750|$

## Ejercicio 488

Sea  $f : \mathbb{N} \rightarrow \mathbb{N}$  una función p.r. y  $P : \mathbb{N} \rightarrow \{0, 1\}$  un predicado p.r.  
Probar que  $map_f : \mathbb{N} \rightarrow \mathbb{N}$  que interpreta la entrada como una lista  
(sin ceros al final) y devuelve la lista con el resultado de aplicar  $f$  a  
cada uno de sus elementos.

Ejemplo, si  $doble(x) = 2x$

$$map_{doble}([1, 2, 3, 5]) = [2, 4, 6, 10]$$

## Ejercicio 488

### Idea

Desarmamos la lista, aplicamos  $f$  y construimos la lista de vuelta.

## Ejercicio 488

### Idea

Desarmamos la lista, aplicamos  $f$  y construimos la lista de vuelta.  
Abusando de la notación,

$$map_f(l) = [f(l[1]), f(l[2]), \dots, f(l[|l|])]$$

## Ejercicio 488

Para formalizar  $\text{map}_f(l) = [f(l[1]), f(l[2]), \dots, f(l[|l|])]$ . Basta escribirlo como:

$$\text{map}_f(l) = \prod_{i=1}^{|l|} p_i^{f(l[i])}$$

# Lenguajes primitivos recursivos

Con lo visto hoy podemos determinar también si un lenguaje es primitivo recursivo. Y diremos que es primitivo recursivo cuando la función característica del lenguaje es primitiva recursiva.

# Lenguajes primitivos recursivos

Con lo visto hoy podemos determinar también si un lenguaje es primitivo recursivo. Y diremos que es primitivo recursivo cuando la función característica del lenguaje es primitiva recursiva.

Ejemplos de lenguajes primitivos recursivos:

- Lenguajes regulares
- Lenguajes libres de contexto

# Lenguajes primitivos recursivos

Con lo visto hoy podemos determinar también si un lenguaje es primitivo recursivo. Y diremos que es primitivo recursivo cuando la función característica del lenguaje es primitiva recursiva.

Ejemplos de lenguajes primitivos recursivos:

- Lenguajes regulares
- Lenguajes libres de contexto

Ejemplos de lenguajes que no son primitivos recursivos:

- Gramáticas sin restricciones
- Otras, pues no todas las funciones computables son primitivas recursivas

# Bibliografía

- Davis, Martin D. & Weyuker, Elaine J. (1983). Computability, Complexity, and Languages. Fundamentals of Theoretical Computer Science. Cap 3.

**CUANDO POR FIN TERMINAS**

