

# Metaheurísticas

Unidad 3

Metaheurísticas basadas en Poblaciones

## Tema 3: Programación Genética

# Objetivos

- Conocer los principales conceptos de la programación genética
- Conocer los elementos más importantes de su diseño
- Capacidad para resolver un problema mediante una técnica basada en programación genética

# Bibliografía

[Koz02] J. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, 2002.

[Koz04] J. Koza, Genetic Programming II: Automatic Discovery of Reusable Programs. The MIT Press, 2004.

[Sim13] D. Simon, Evolutionary Optimization Algorithms, Wiley, 2013.

[Won00] M. L. Wong, K. S. Leung, Data mining using grammar based genetic programming and applications, Kluwer Academic Publishers, 2000.



# Motivación

- Los algoritmos genéticos, por regla general, incorporan en su estructura la solución asumida
- Normalmente, no se puede modificar esa estructura

La Programación Genética son un intento de generalizar a los Algoritmos Genéticos → Intentan **generalizar la mejor solución y también la estructura óptima**

# Índice

1. Introducción
2. Fundamentos
3. Otras consideraciones

# Introducción

principios básicos

1. Nos dan la flexibilidad de obtener soluciones a un amplio abanico de problemas
2. No restringen sus soluciones como otros enfoques evolutivos con respecto a sus estructuras
3. Emplean inducción en su evolución
4. Típica aplicación para regresión simbólica

# Introducción

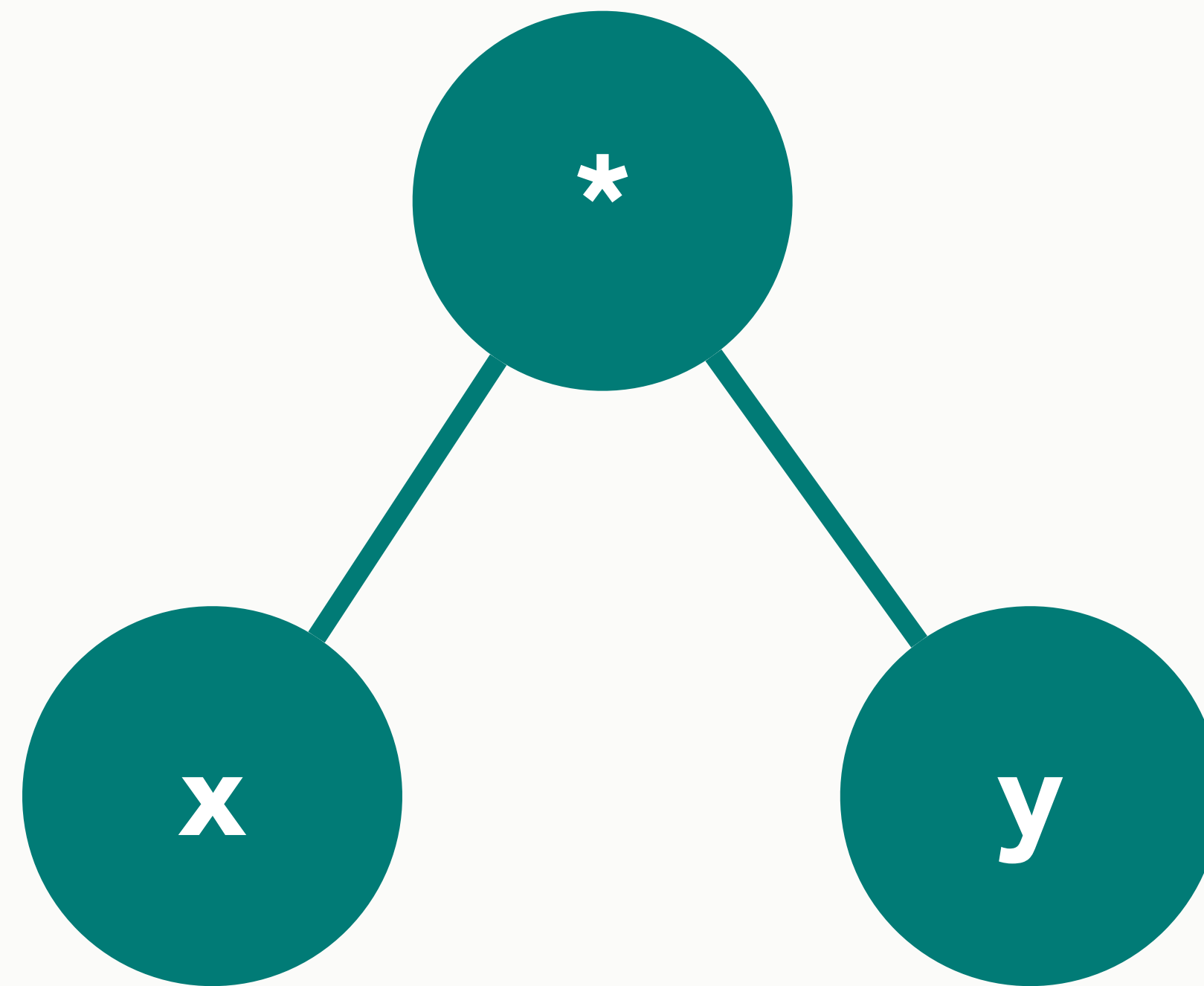
principios básicos

$$xy + |z|$$

# Introducción

principios básicos

$$xy + |z|$$

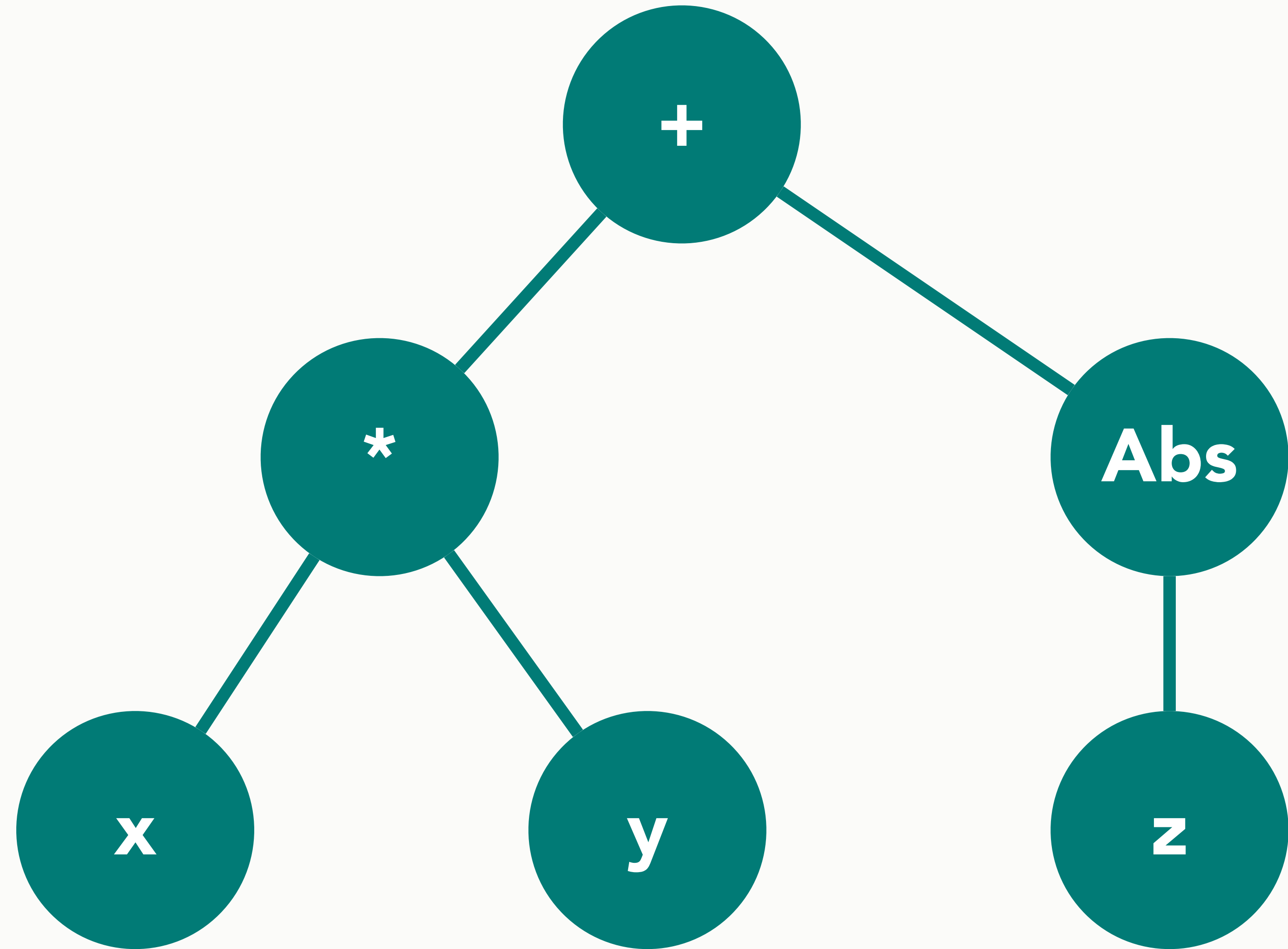




# Introducción

principios básicos

$$xy + |z|$$

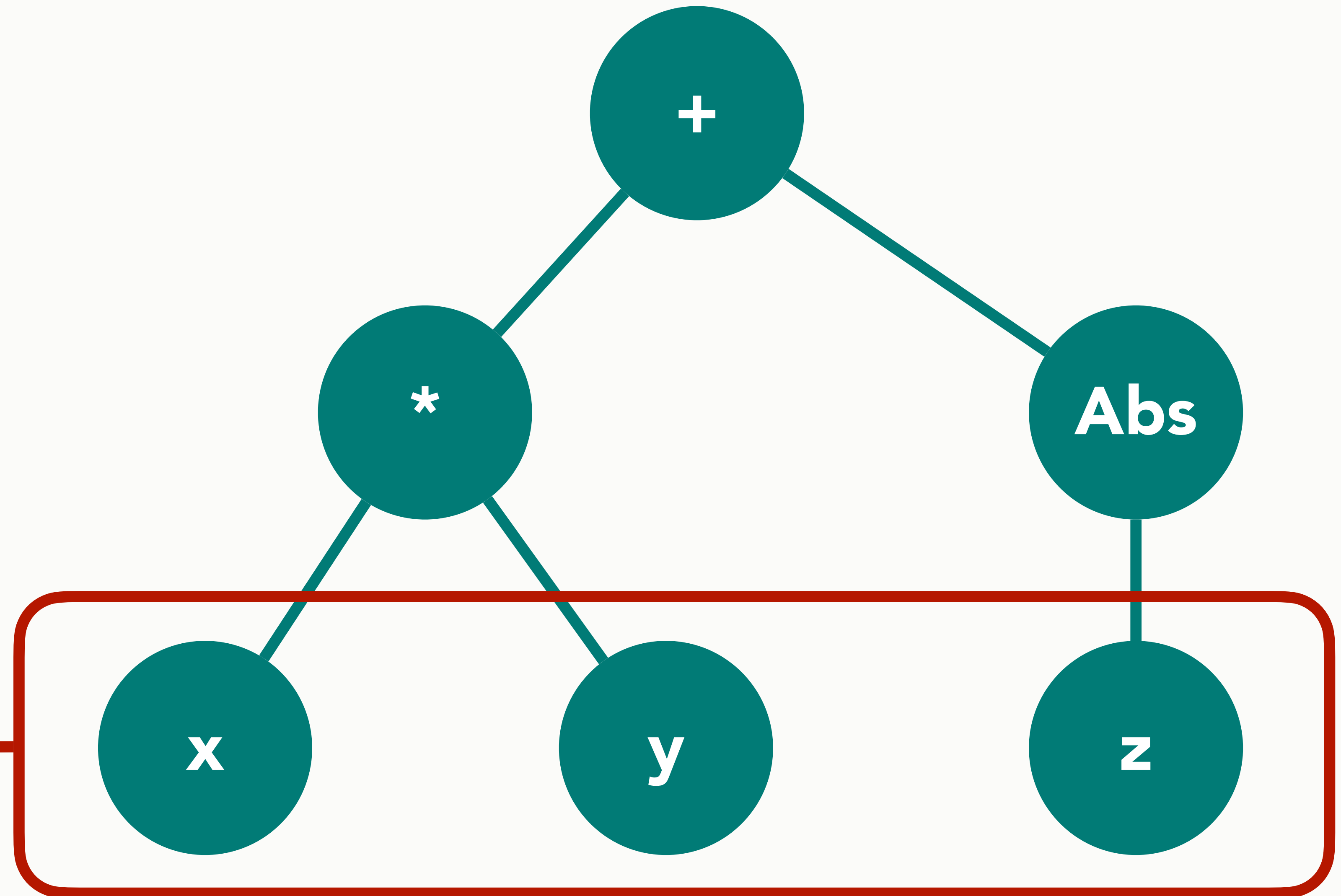


# Introducción

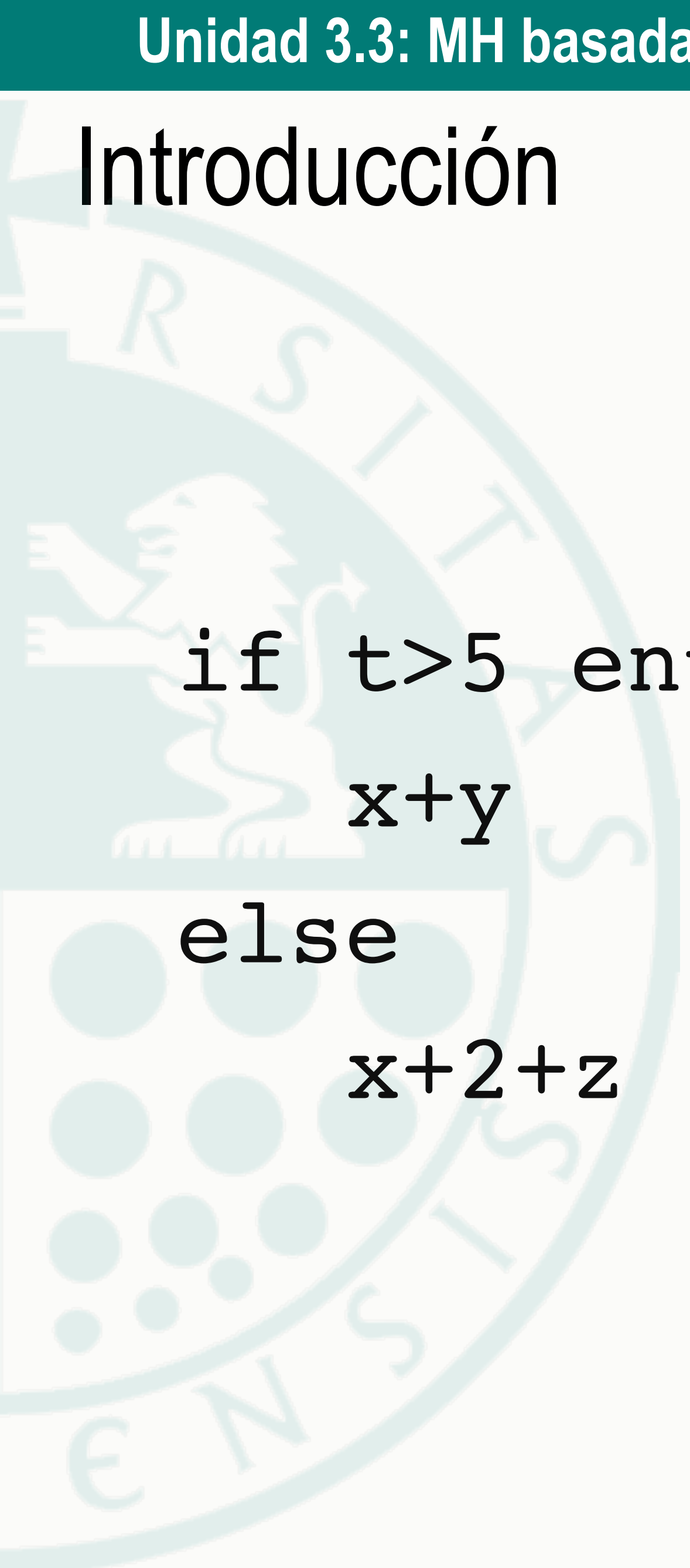
principios básicos

$xy + |z|$

**HOJAS**



# Introducción



```
if t>5 entonces  
    x+y  
else  
    x+2+z
```

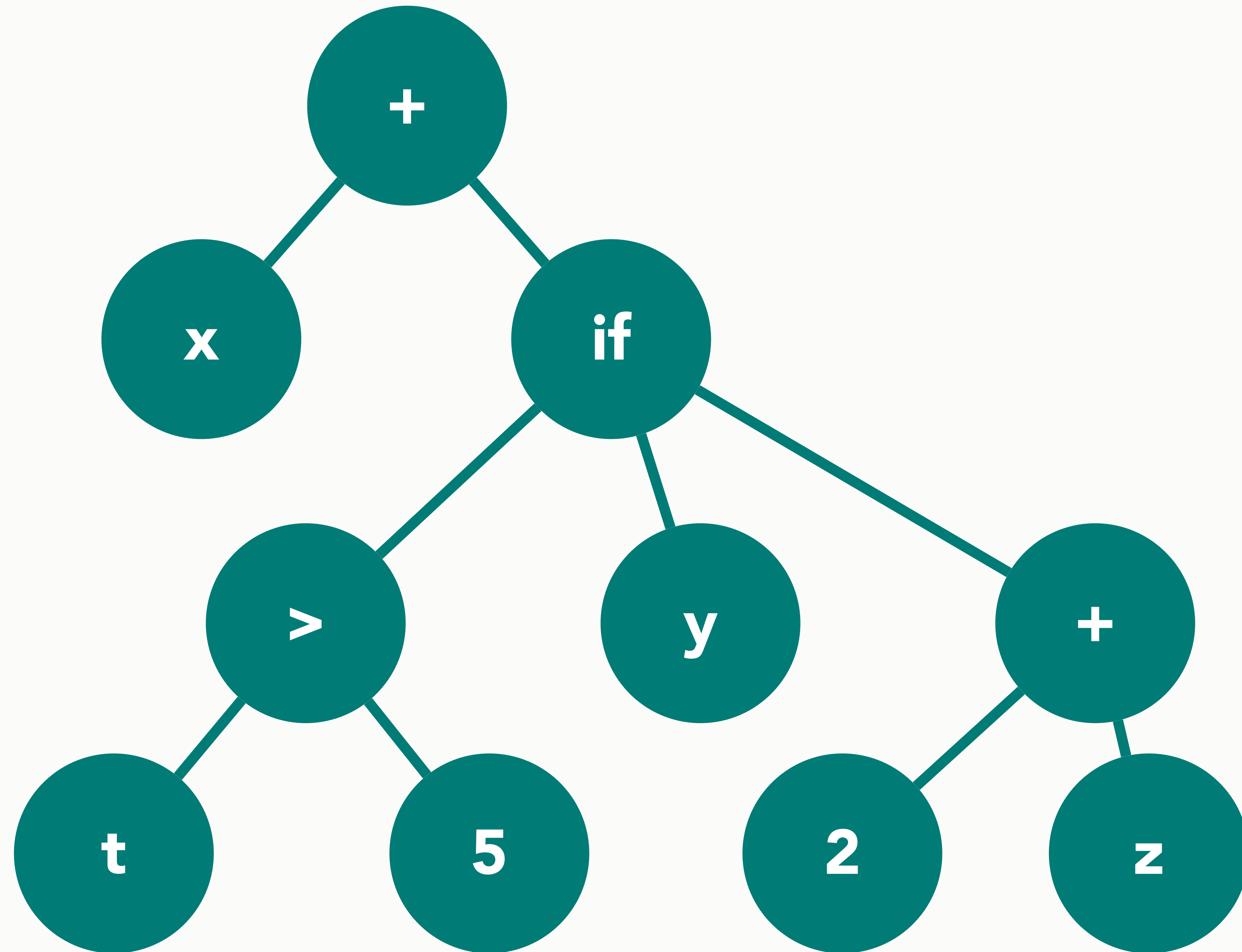
# Introducción

```
if t>5 entonces
```

```
  x+y
```

```
else
```

```
  x+2+z
```

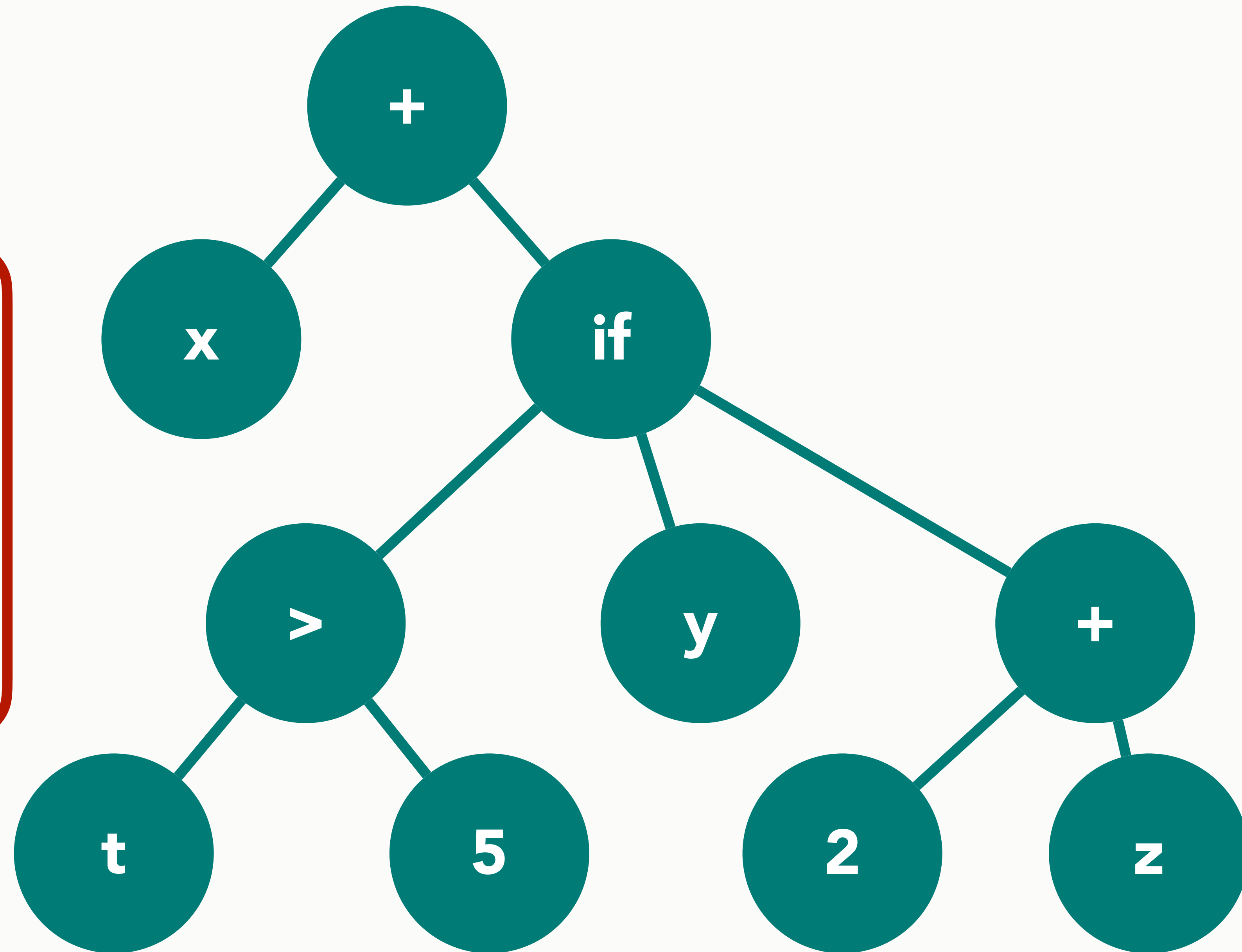




# Introducción

## FENOTIPO

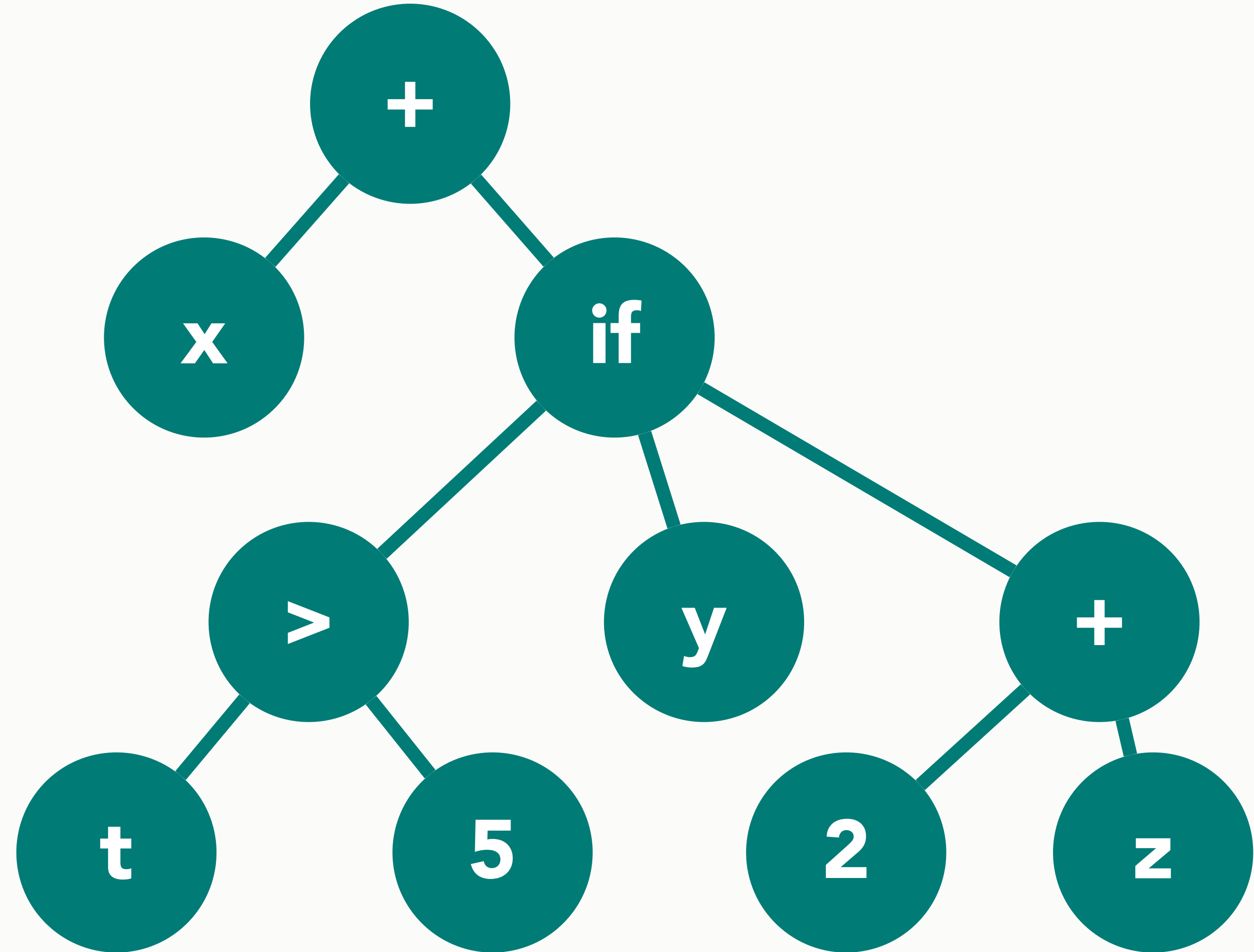
```
if t>5 entonces  
  x+y  
else  
  x+2+z
```



# Introducción

## FENOTIPO

```
if t>5 entonces  
  x+y  
else  
  x+2+z
```



GENOTIPO + x if > t 5 y + 2 z

# Introducción

pasos necesarios



# Introducción

pasos necesarios

Dependen  
del  
problema

Componentes  
del  
algoritmo

Diseñar una representación

Decidir la forma de iniciar la población

Diseñar una correspondencia entre genotipo y fenotipo

Diseñar de una forma de evaluar un individuo

Diseñar un operador de cruce

Diseñar un operador de mutación

Decidir la selección de los individuos

Decidir el reemplazamiento de los individuos

Decidir la condición de parada



# Fundamentos

representación

- La **función de adaptación** en la programación genética debe ser similar que en cualquier otro enfoque evolutivo
- Asocian a la representación *una gramática* que guíe el proceso de búsqueda
  - ★ La gramática es clave para obtener soluciones factibles
  - ★ De hecho también se denominan **grammar guided genetic programming**

# Fundamentos

representación mediante **gramática**

- Existen diversas formas de representar programas aunque la más utilizada son los árboles de expresiones
- Habitualmente se emplean lenguajes sencillos
- Basándose en una **gramática libre de contexto** para definir las sentencias válidas del lenguaje
- El cromosoma codifica la expresión del árbol y normalmente se recorren en **inorden**

# Fundamentos

representación mediante **gramática**

## NODOS TERMINALES

- Describen los símbolos que pueden aparecer en nodos terminales
- Compuesto normalmente por variables de entrada o constantes
- Muy importante el factor de balanceo del árbol

# Fundamentos

representación mediante **gramática**

## NODOS FUNCIÓN

- Describen los operadores y funciones que pueden aparecer en nodos no terminales, por ejemplo:
  - Operadores matemáticos estándar
  - Funciones específicas al problema
  - Condicionales y/o comparadores



# Fundamentos

representación mediante **gramática**

## NODOS FUNCIÓN

- Describen los operadores y funciones que pueden aparecer en nodos no terminales, por ejemplo:
  - Funciones lógicas
  - Funciones de asignación
  - Bucles
  - Subrutinas

# Fundamentos

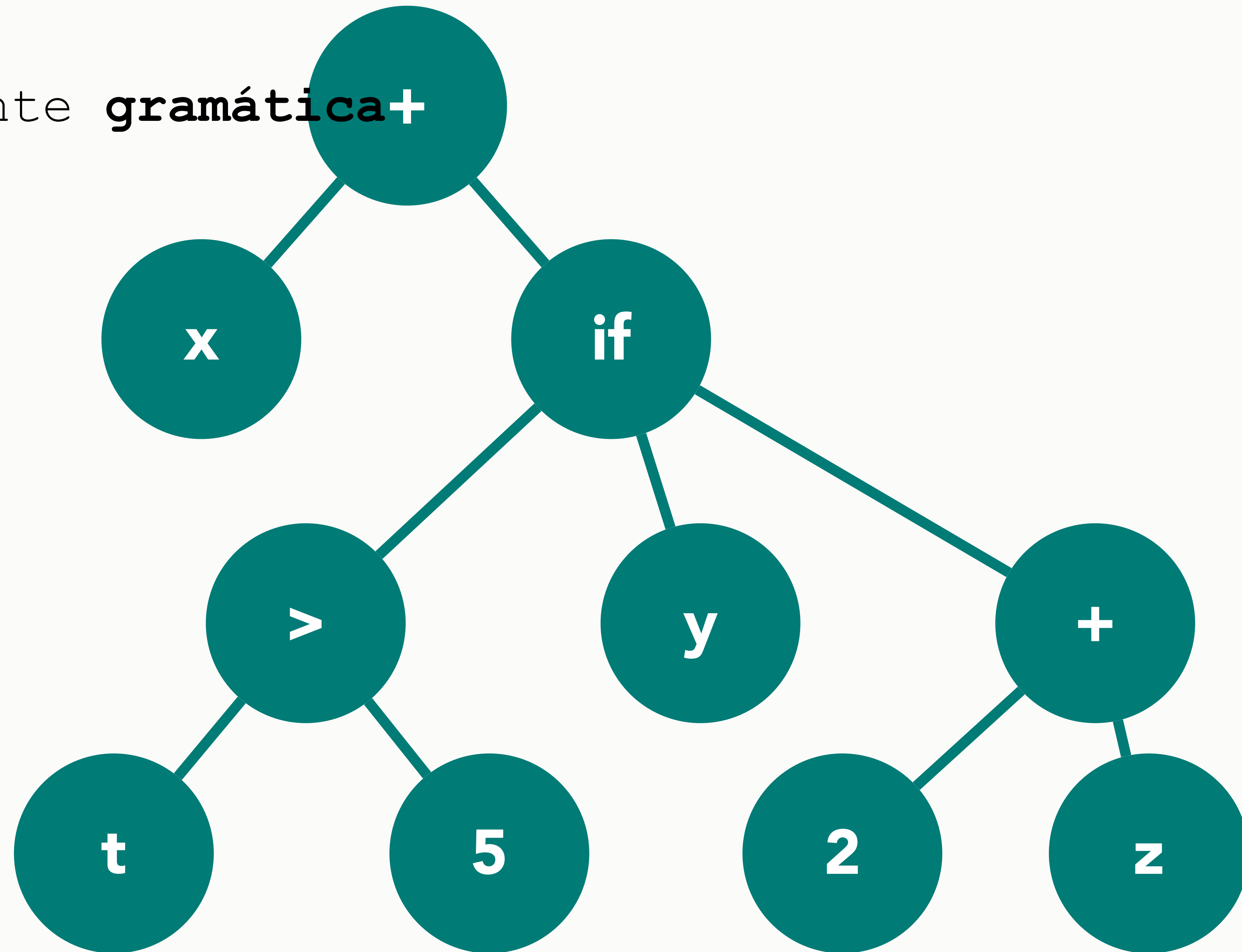
representación mediante **gramática+**

```
if t>5 entonces
```

```
  x+y
```

```
else
```

```
  x+2+z
```



# Fundamentos

representación mediante **gramática+**

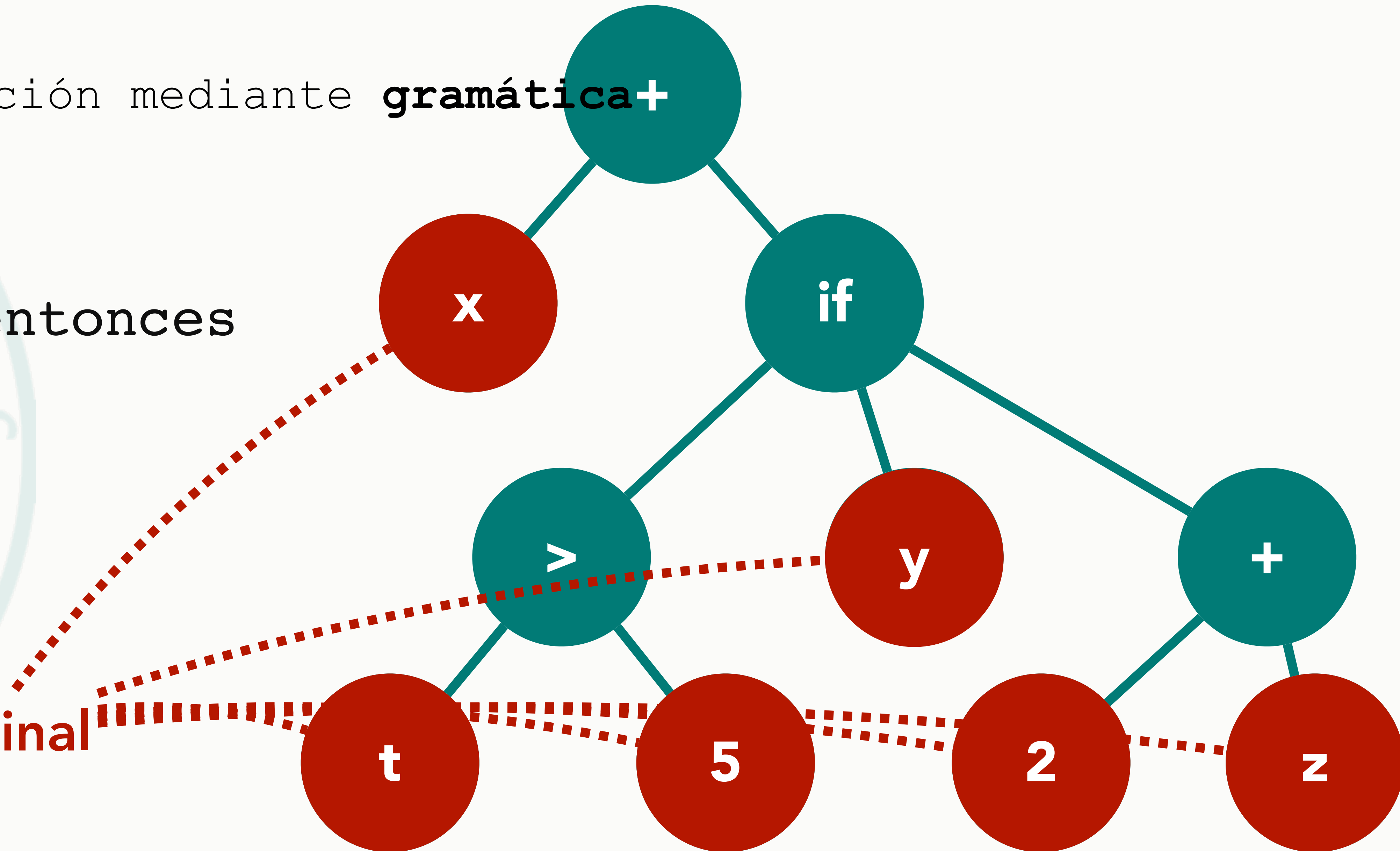
```
if t>5 entonces
```

```
  x+y
```

```
else
```

```
  x+2+z
```

**terminal**



# Fundamentos

representación mediante **gramática+**

```
if t>5 entonces
```

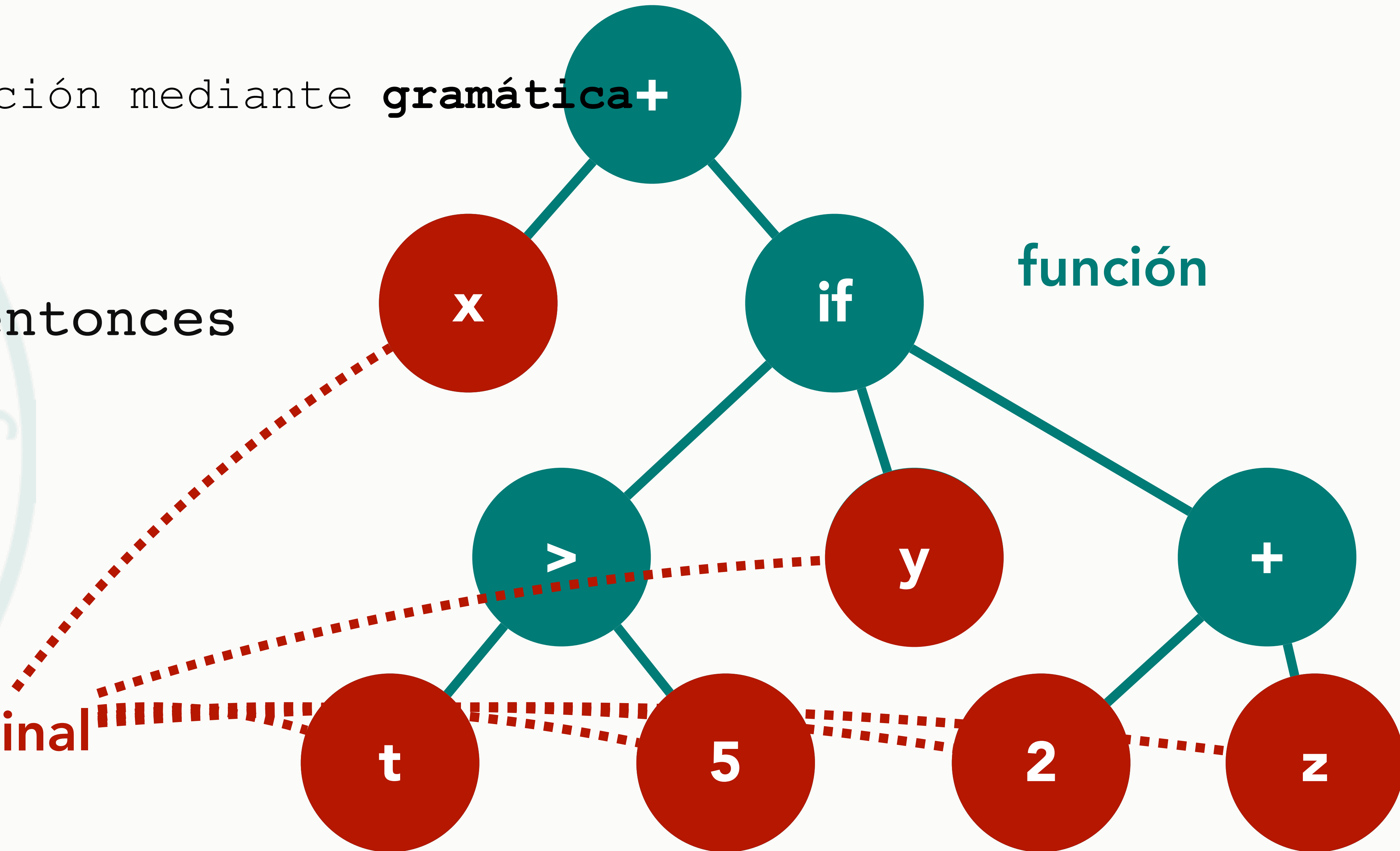
```
  x+y
```

```
else
```

```
  x+2+z
```

**terminal**

**función**





# Ejercicio

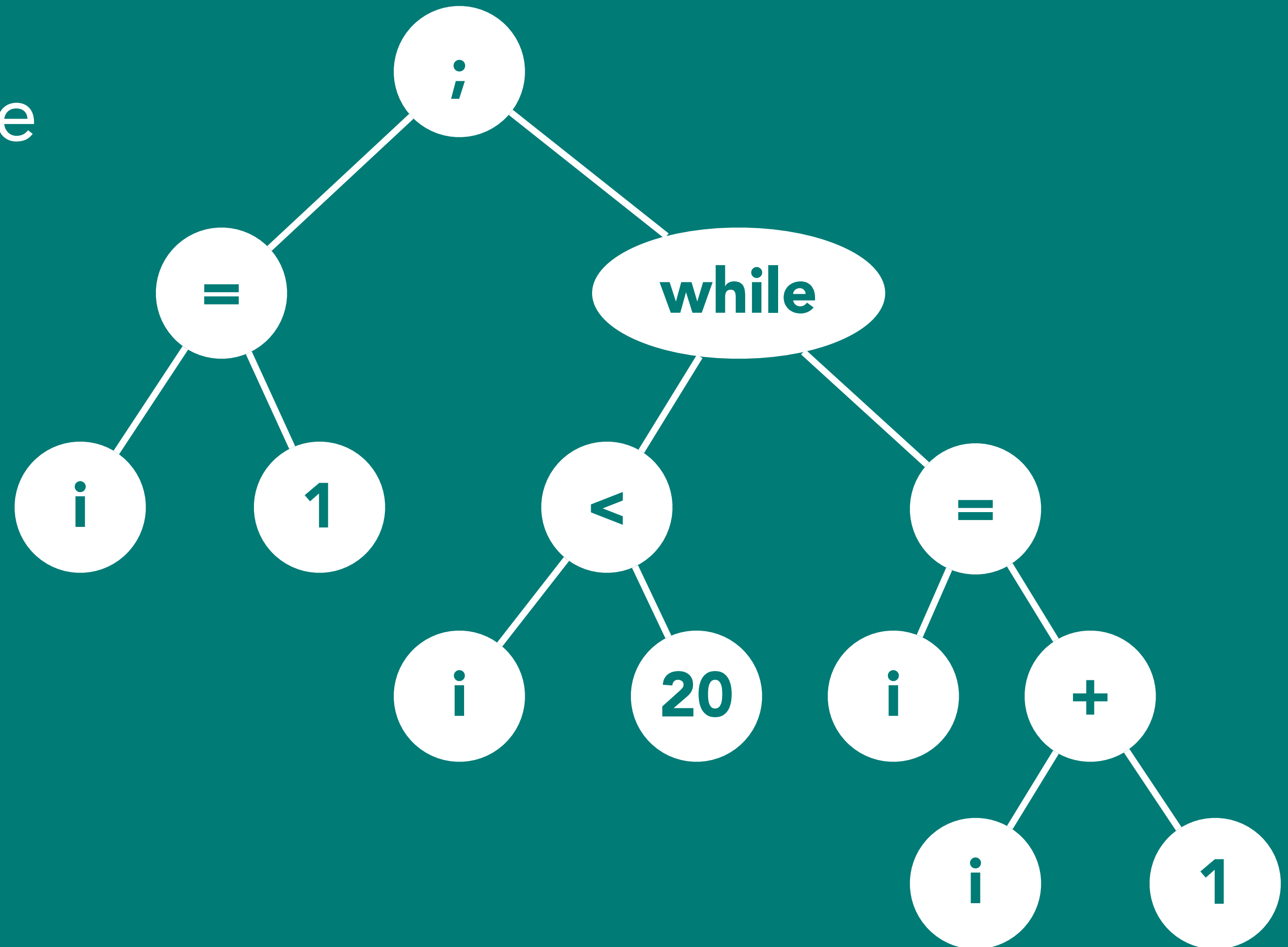
Representa el siguiente programa:

```
i=1;  
while(i<20){  
    i=i+1  
}
```

# Ejercicio

Representa el siguiente programa:

```
i=1;  
while(i<20){  
    i=i+1  
}
```



# Ejercicio

¿Y las siguientes expresiones?

$$2\pi i + \left( (x + 3) - \frac{y}{5 + 1} \right)$$

$$(x \wedge \text{true}) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$$

# Fundamentos

## inicialización

- La inicialización en este tipo de evolutivo se puede realizar de distintas formas



# Fundamentos

## inicialización

- La inicialización en este tipo de evolutivo se puede realizar de distintas formas

inicialización completa

# Fundamentos

## inicialización

- La inicialización en este tipo de evolutivo se puede realizar de distintas formas

inicialización completa

inicialización creciente

# Fundamentos

## inicialización

- La inicialización en este tipo de evolutivo se puede realizar de distintas formas

inicialización completa

inicialización creciente

inicialización combinada

# Fundamentos

inicialización **completa**

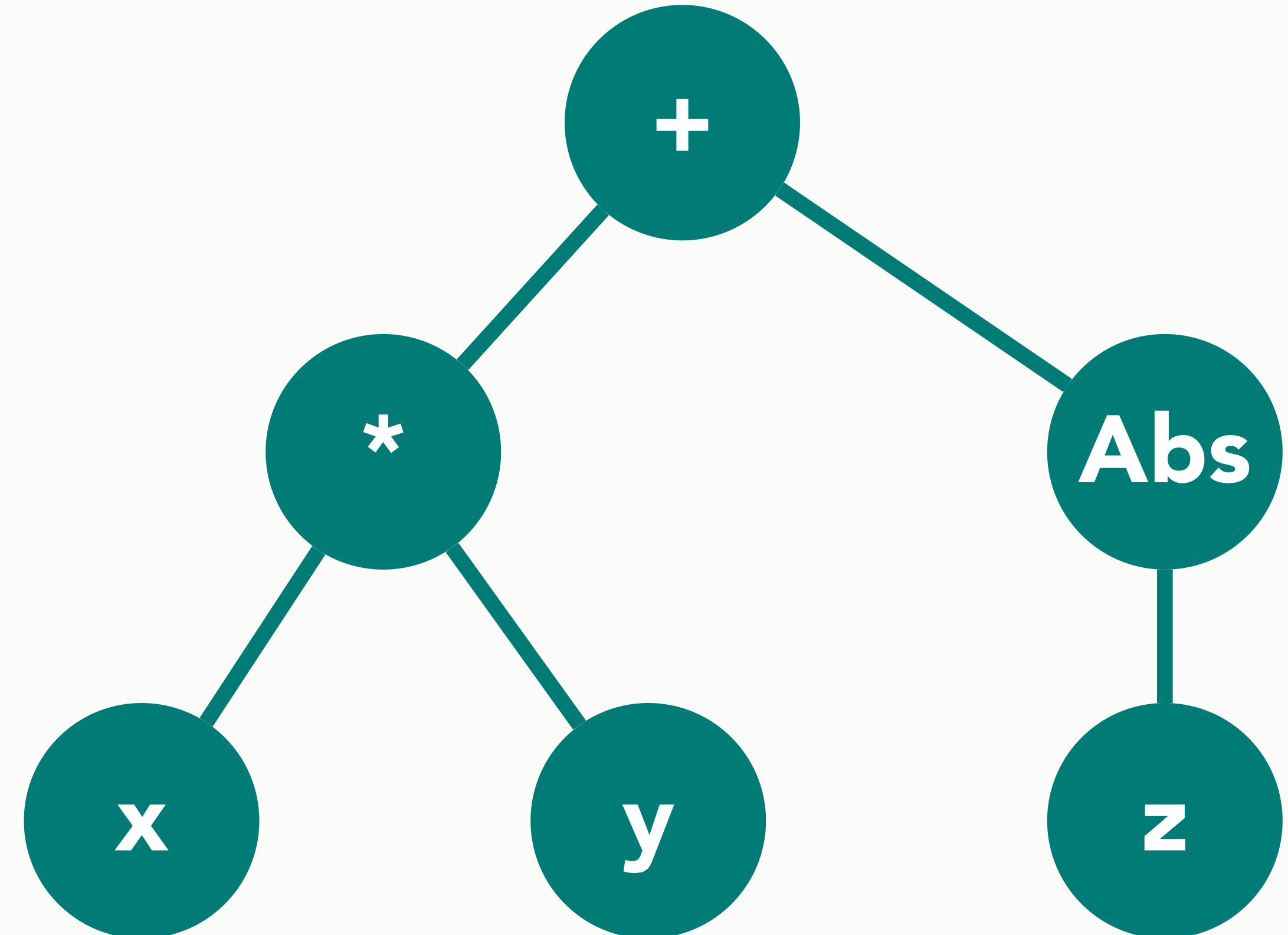
- Se basa en el concepto de profundidad del árbol
- Parámetro introducido por el usuario
- Todos los nodos terminales tienen la misma profundidad hasta la raíz



# Fundamentos

inicialización **completa**

- Se basa en el concepto de profundidad del árbol
- Parámetro introducido por el usuario
- Todos los nodos terminales tienen la misma profundidad hasta la raíz



# Fundamentos

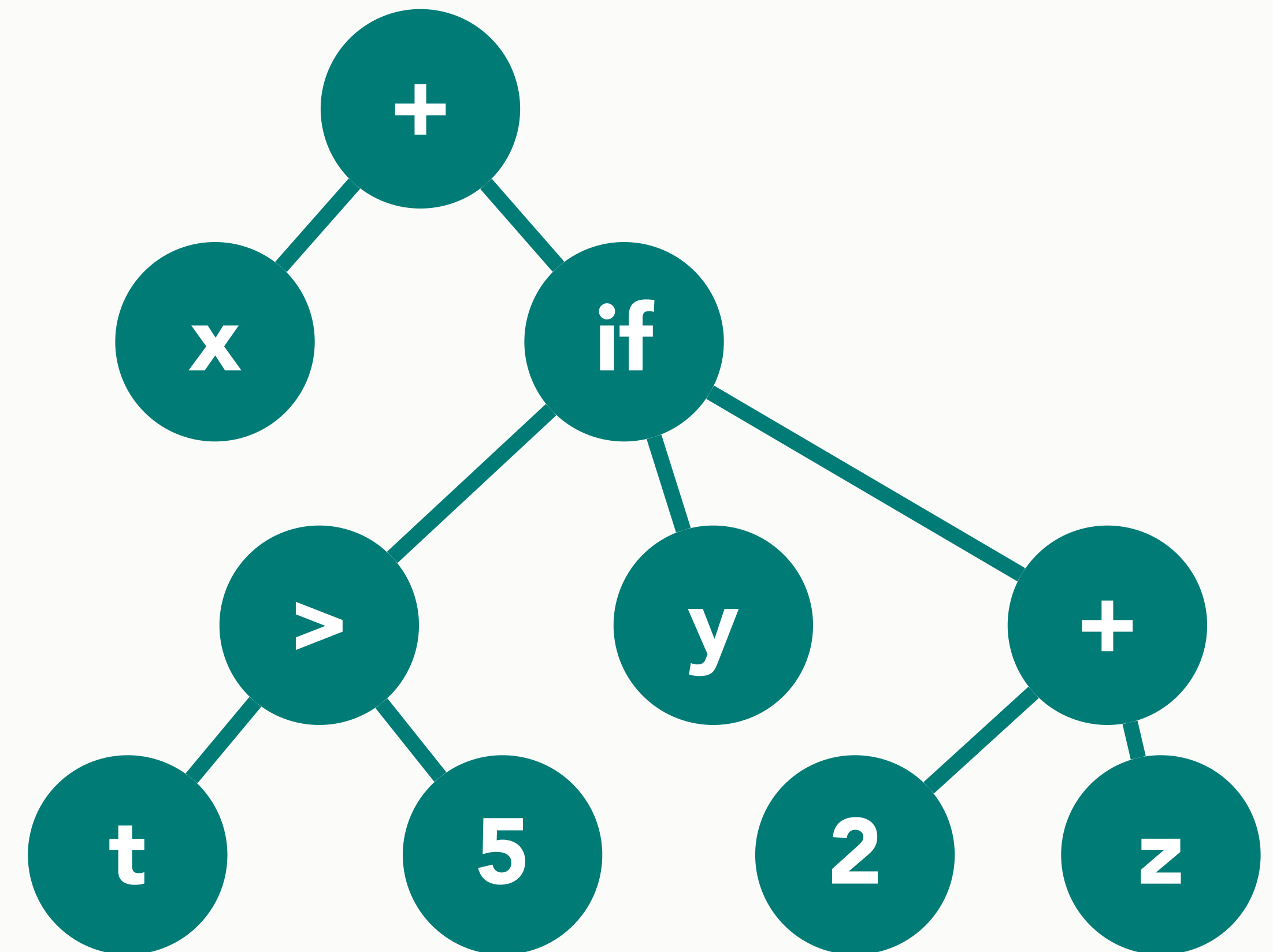
inicialización **creciente**

- Se basa en el concepto de profundidad del árbol
- Parámetro introducido por el usuario
- Todos los nodos terminales tienen menor y igual profundidad hasta la raíz

# Fundamentos

inicialización **creciente**

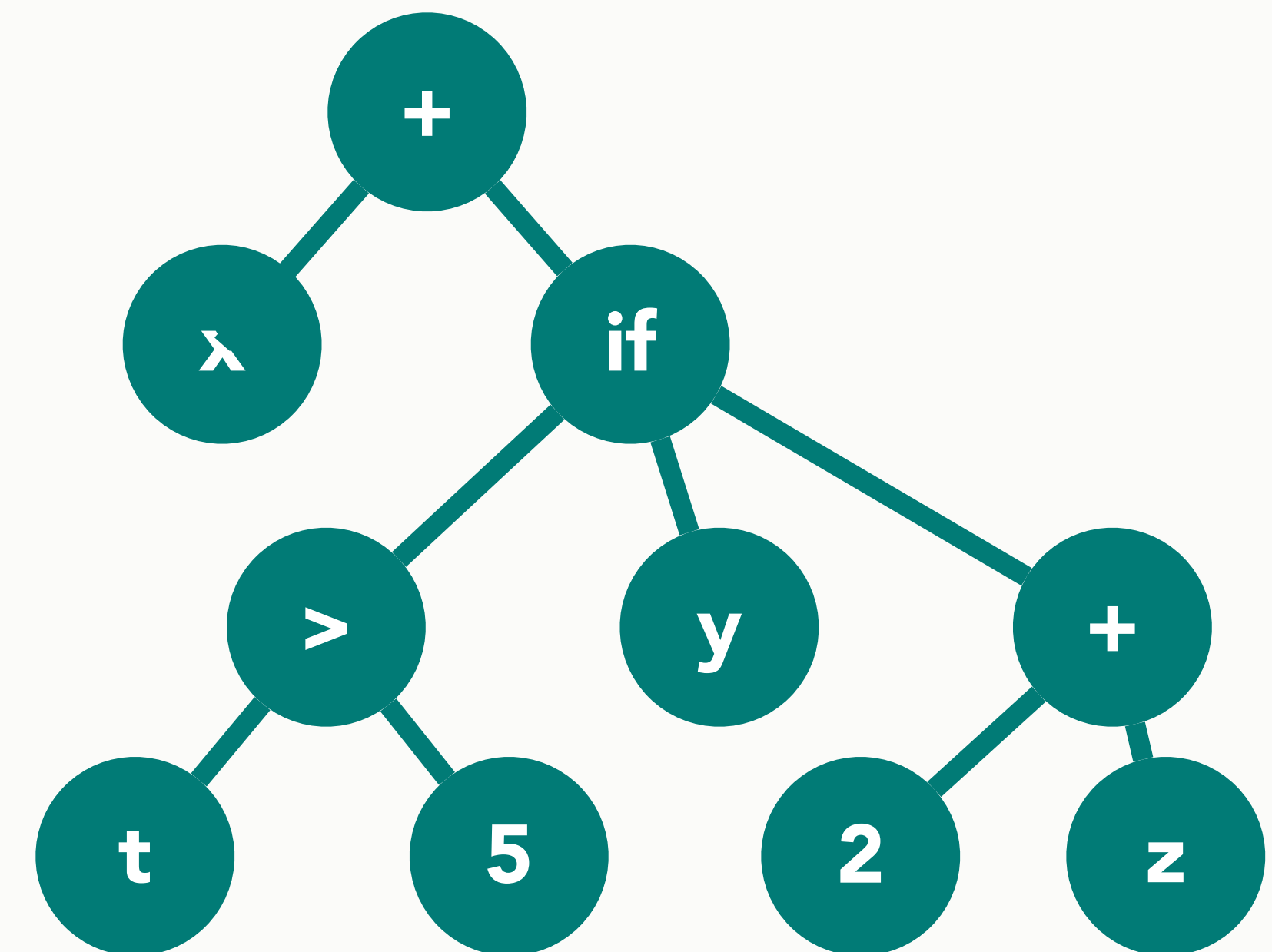
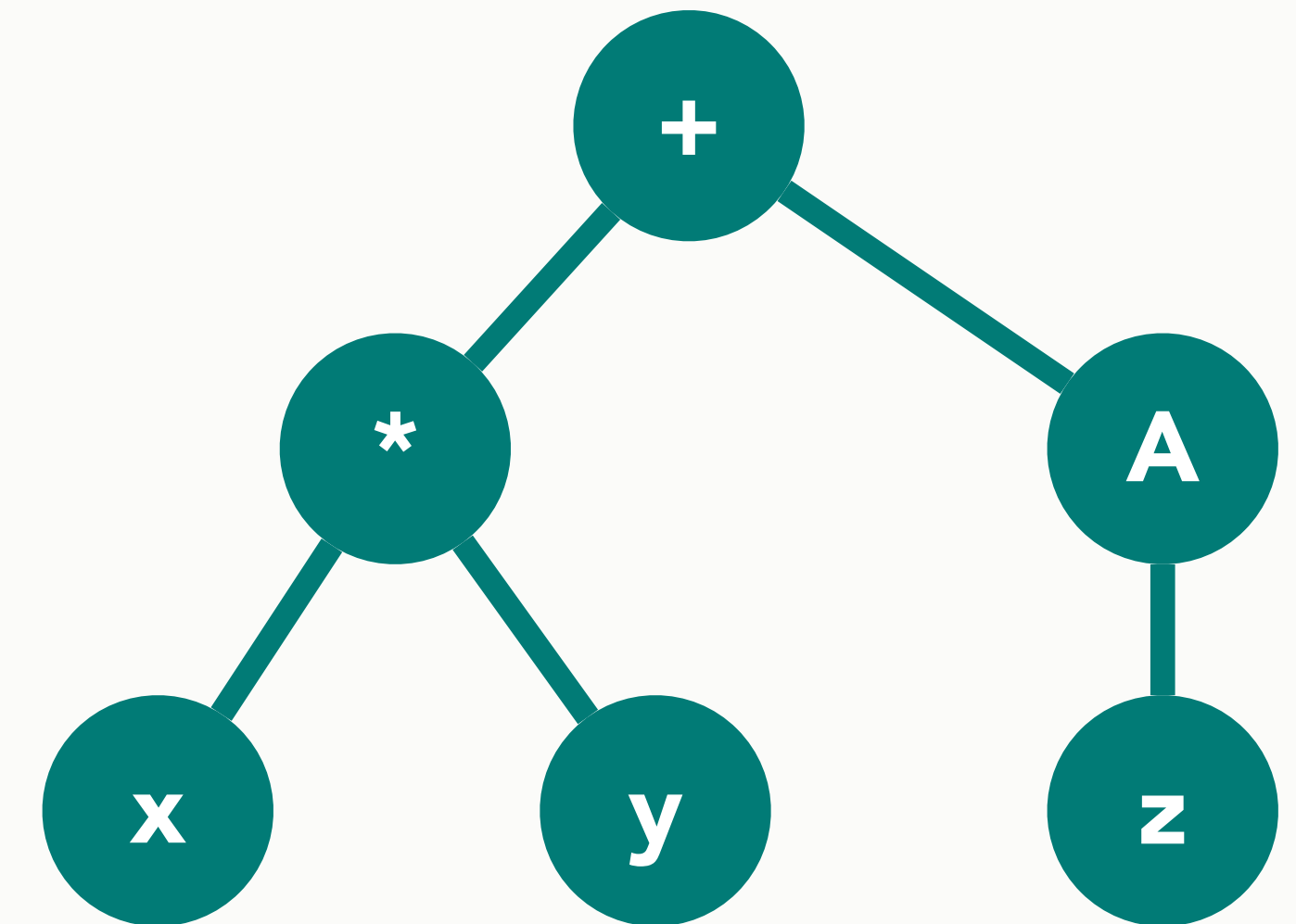
- Se basa en el concepto de profundidad del árbol
- Parámetro introducido por el usuario
- Todos los nodos terminales tienen menor y igual profundidad hasta la raíz



# Fundamentos

inicialización **combinada**

- Se basa en el concepto de profundidad del árbol
- Parámetro introducido por el usuario
- La mitad de la población es completa y la otra mitad es creciente estricta





# Fundamentos

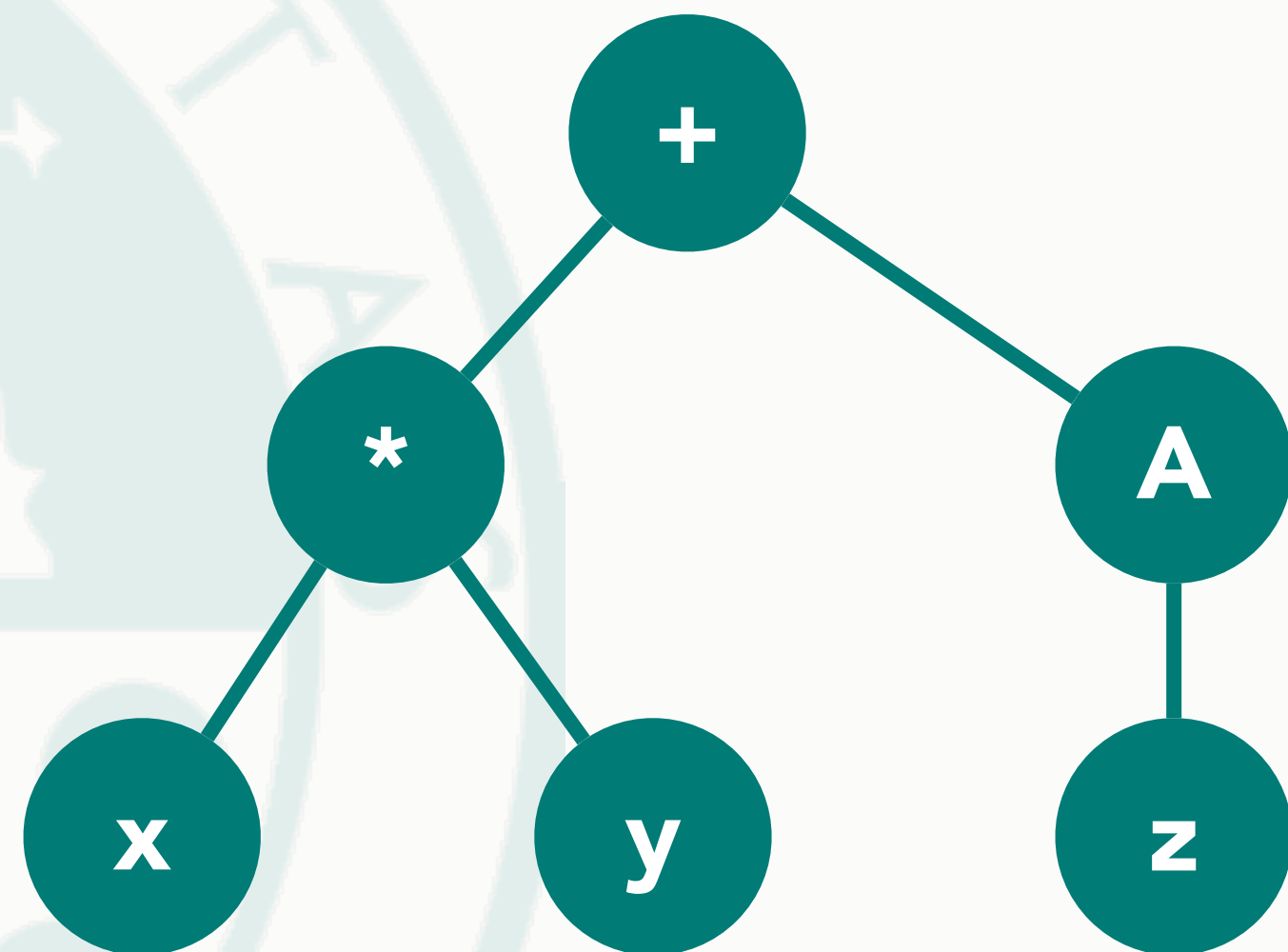
función de evaluación

- La **función de adaptación** en la programación genética debe ser similar que en cualquier otro enfoque evolutivo
- Se puede hacer uso de la recursividad para evaluar
- Asocian a la representación de la gramática de la representación
- Puede emplear enfoque mono o multiobjetivo
- **Dependen del problema**

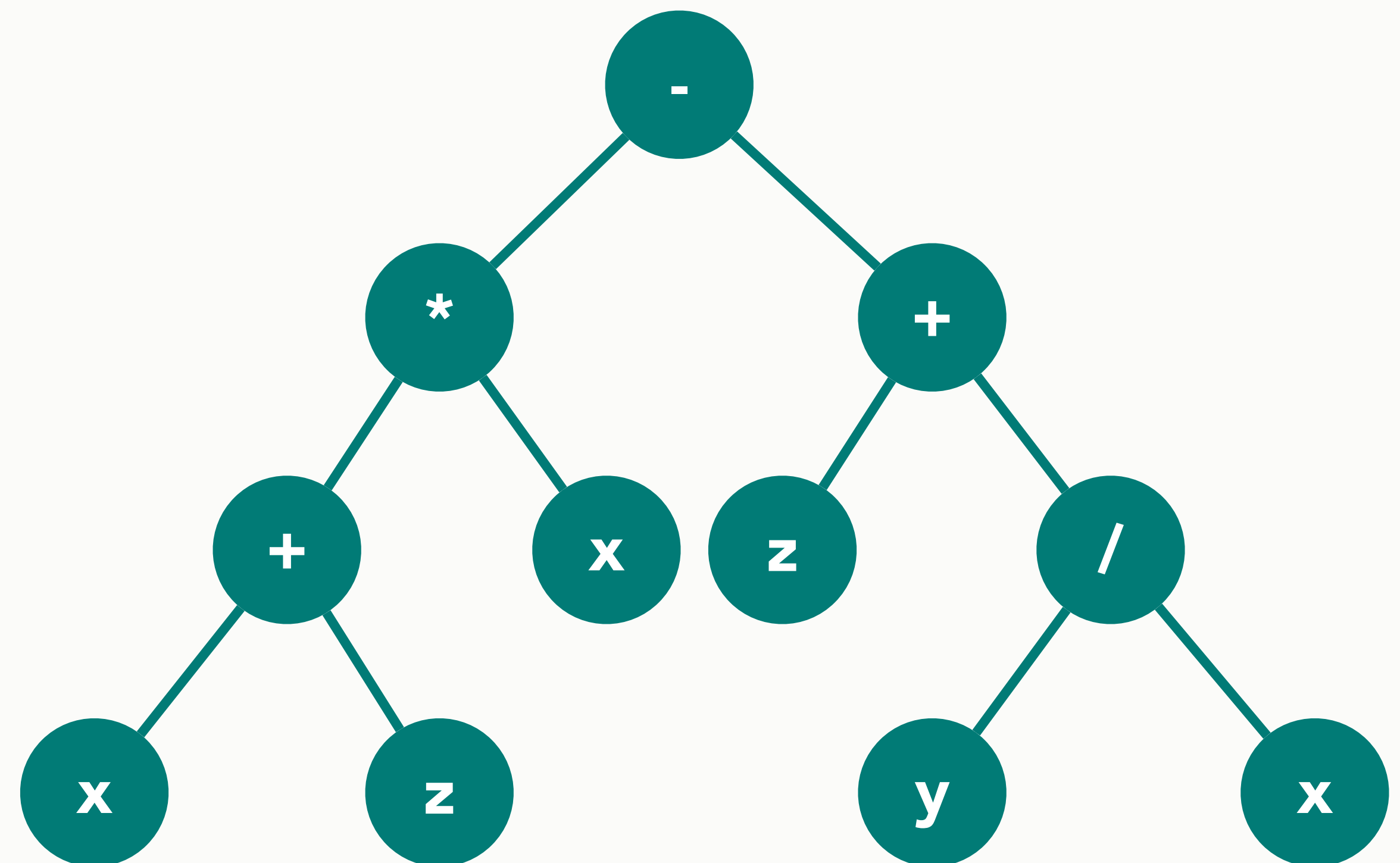
# Fundamentos

operador de cruce

**Padre 1**



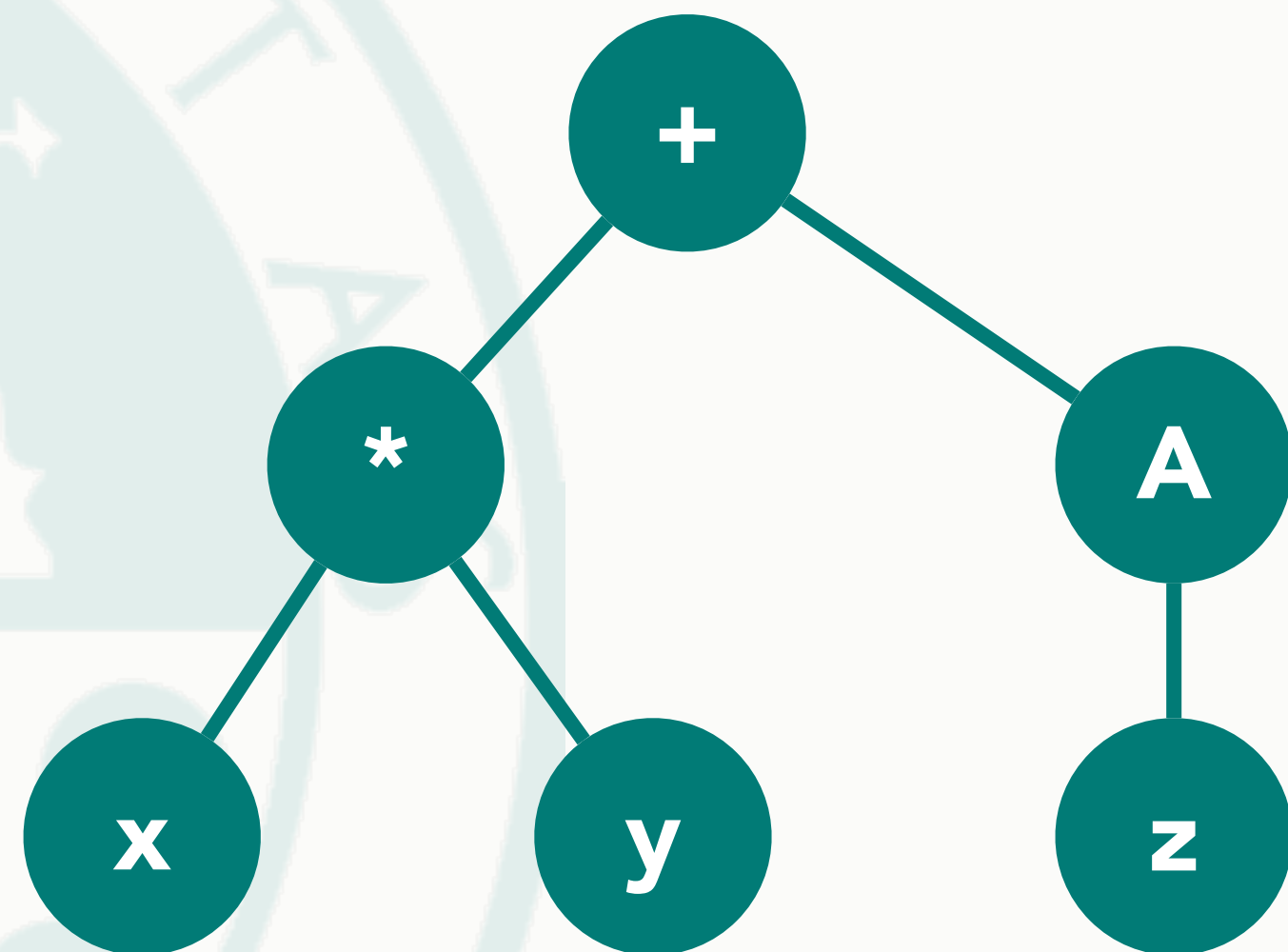
**Padre 2**



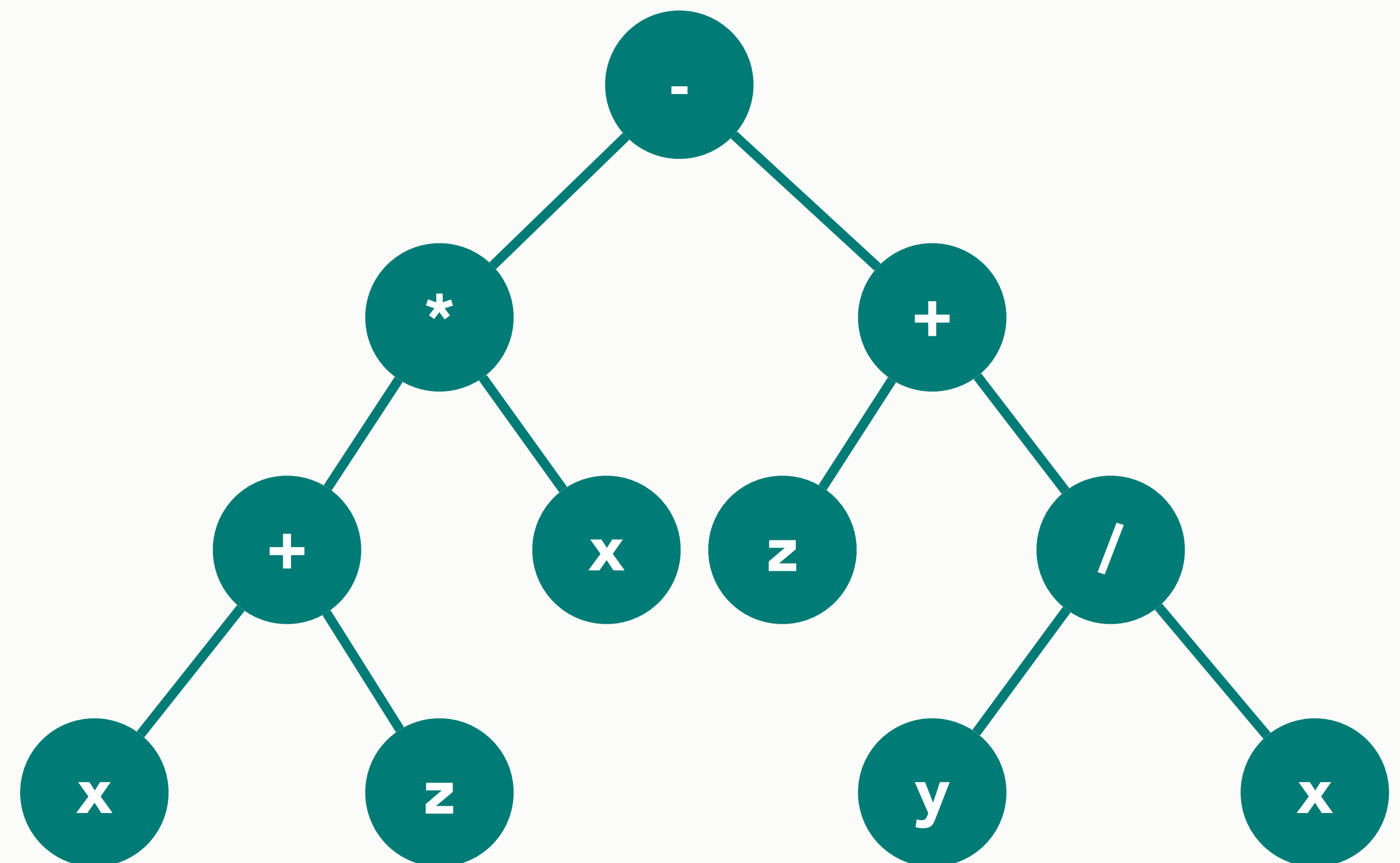
# Fundamentos

operador de cruce

**Padre 1**



**Padre 2**

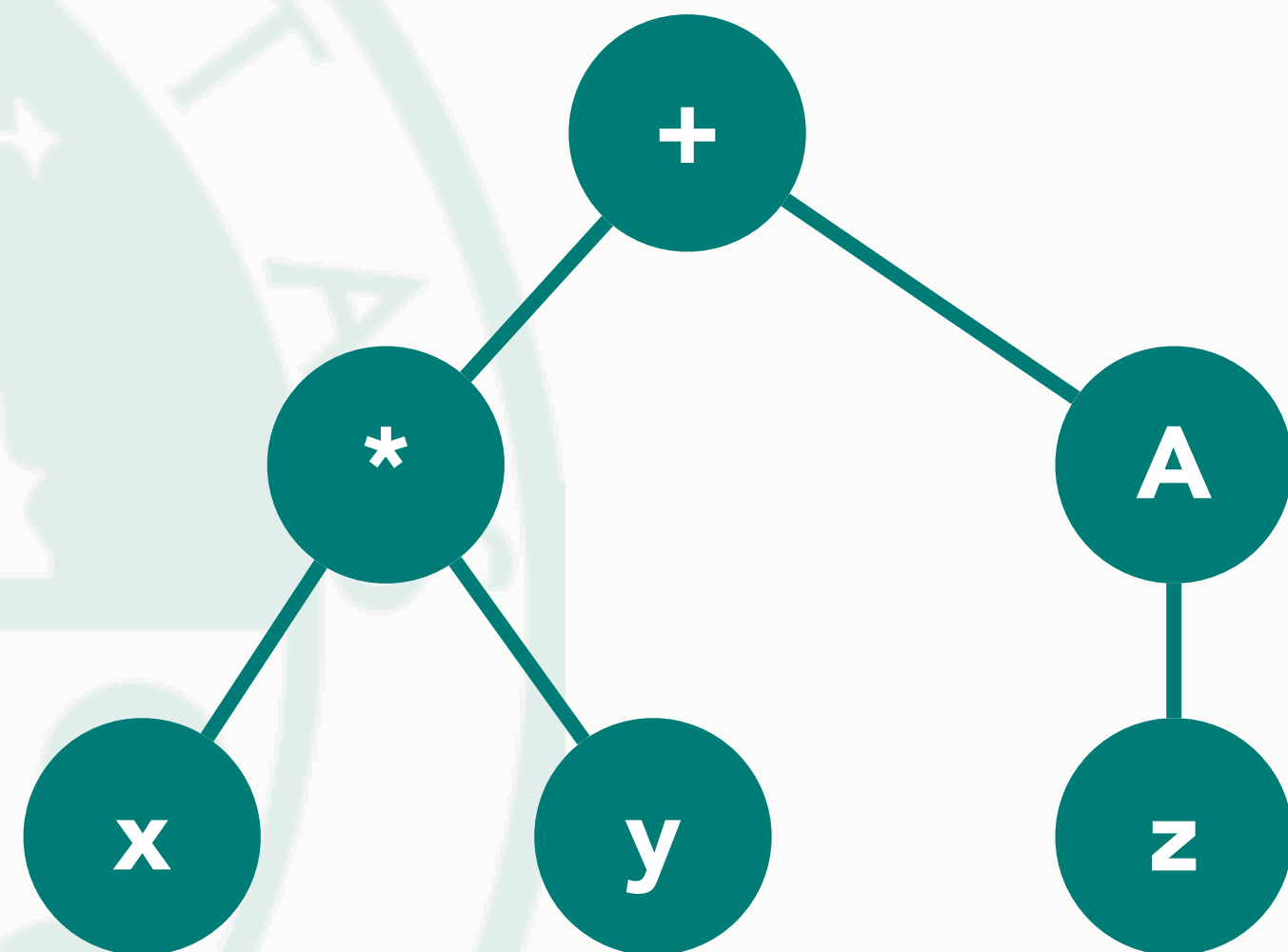


**¿FENOTIPO?**

# Fundamentos

operador de cruce

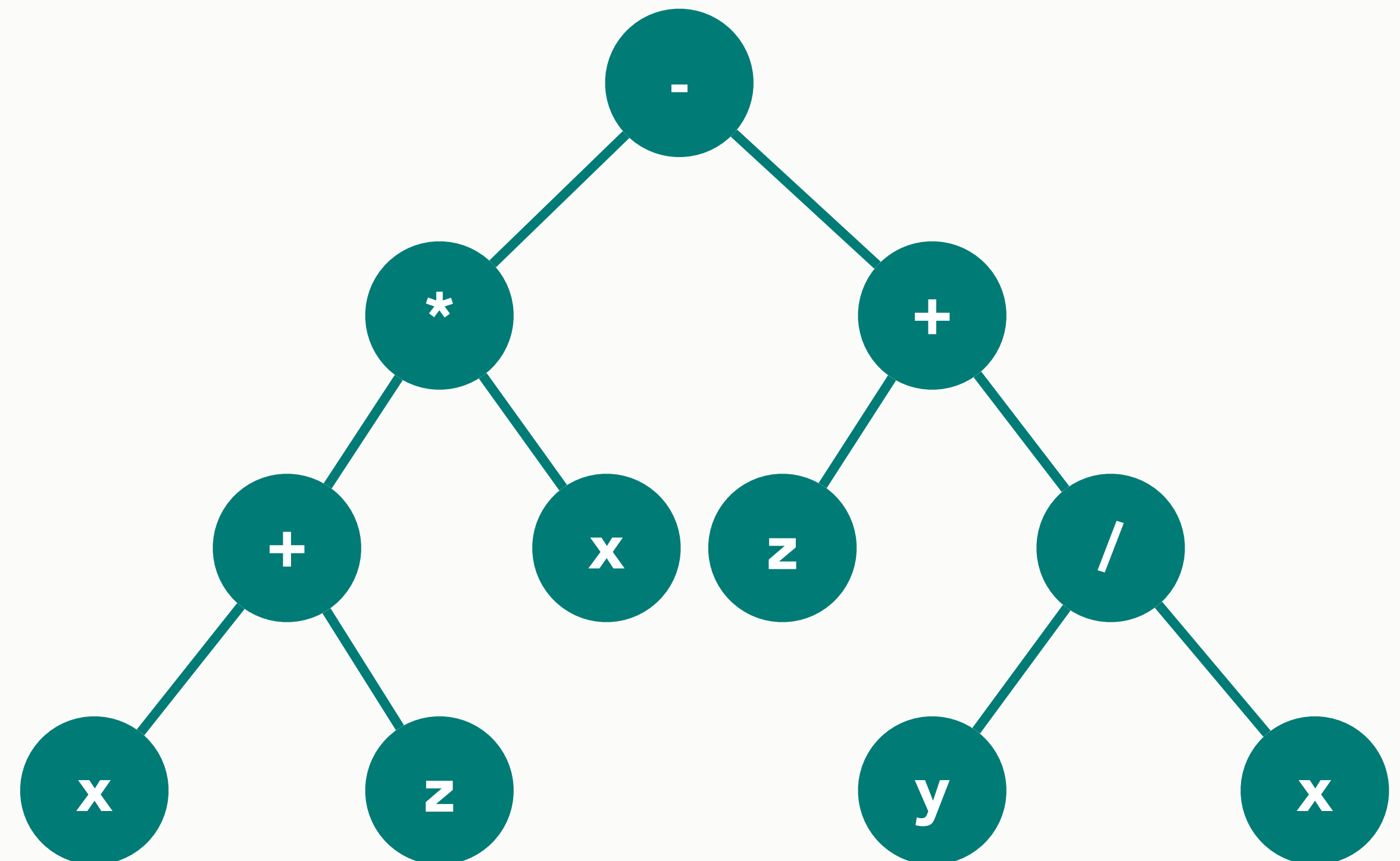
**Padre 1**



$xy + |z|$

**¿FENOTIPO?**

**Padre 2**

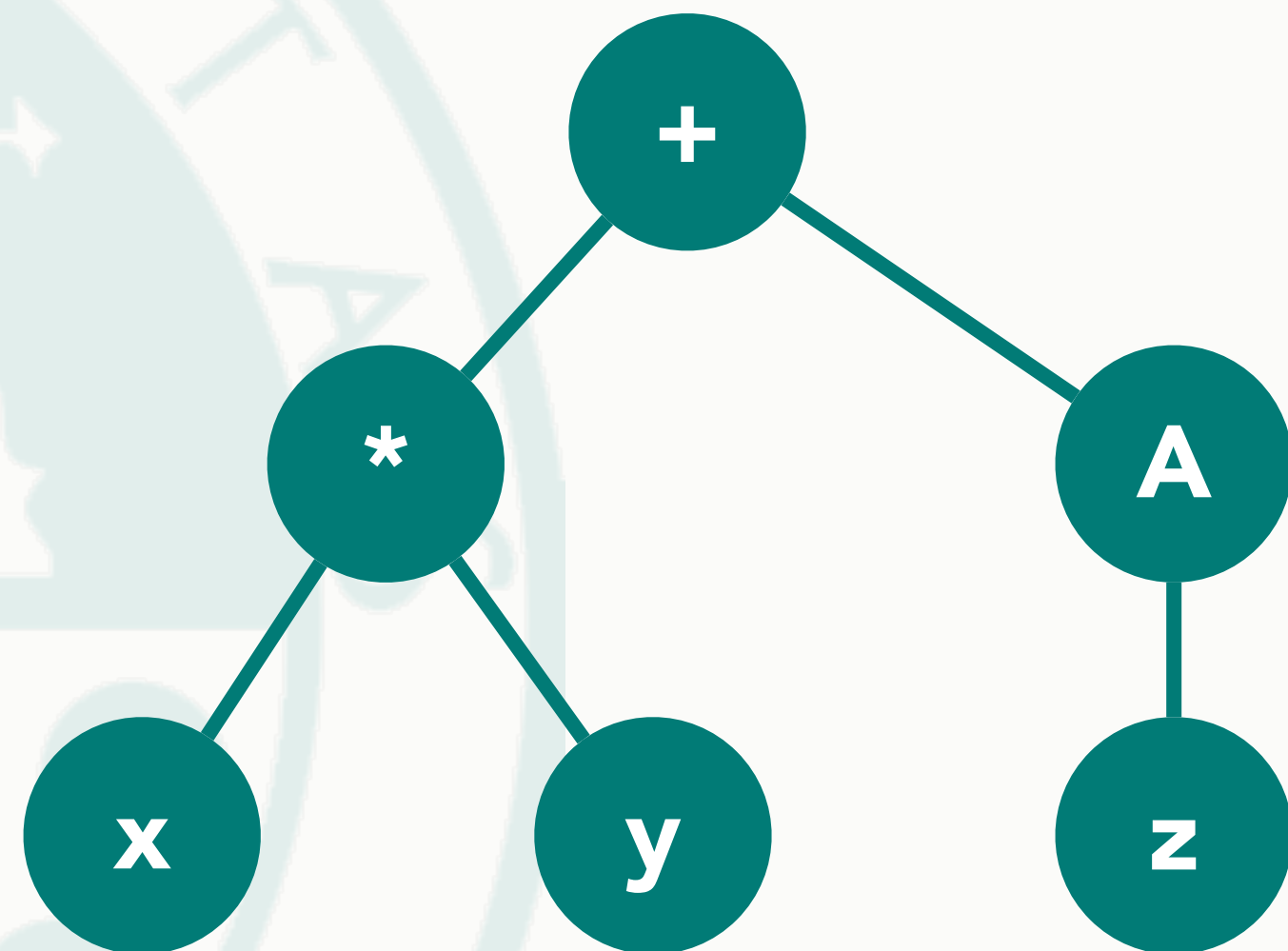




# Fundamentos

operador de cruce

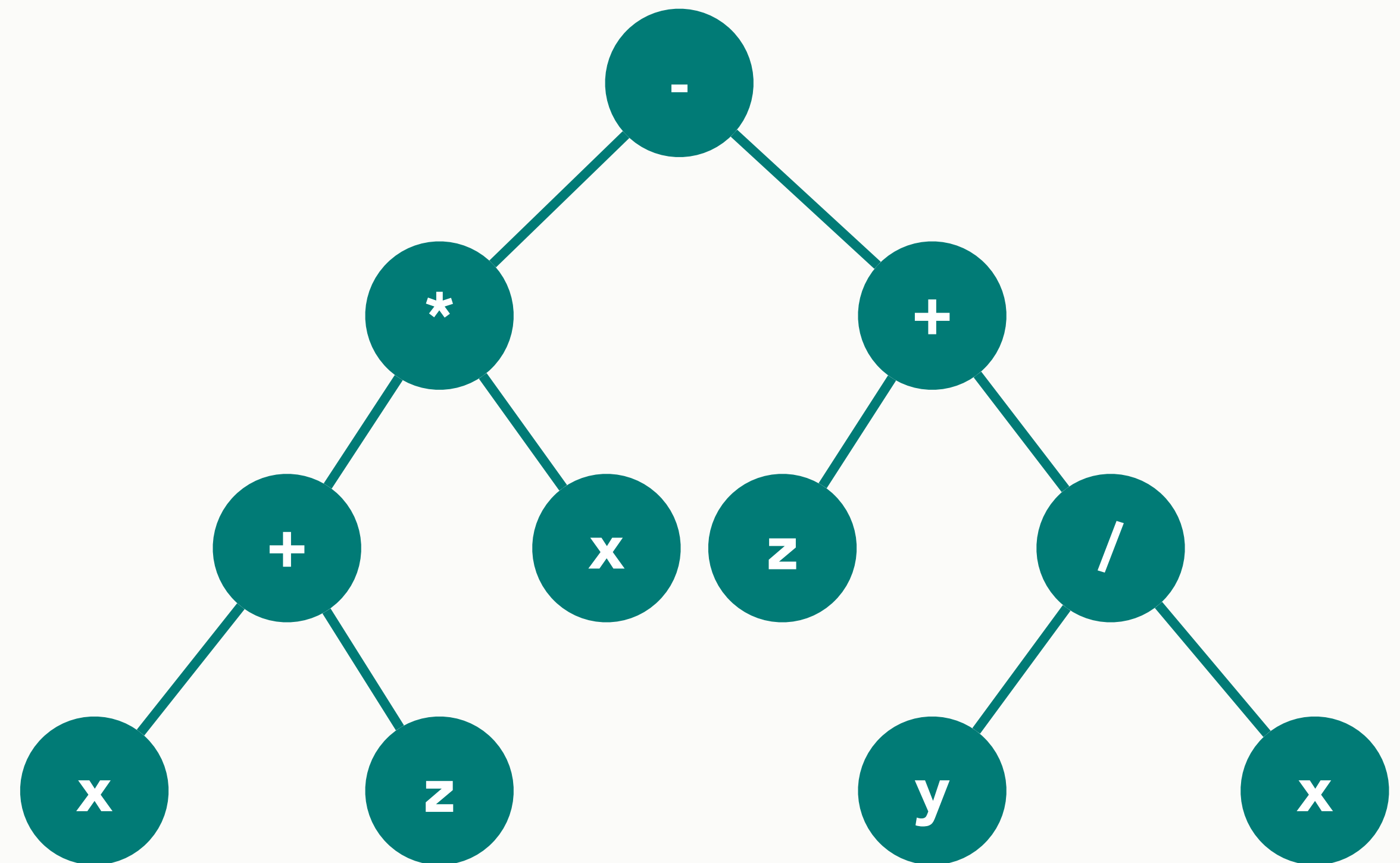
**Padre 1**



$xy + |z|$

**¿FENOTIPO?**

**Padre 2**

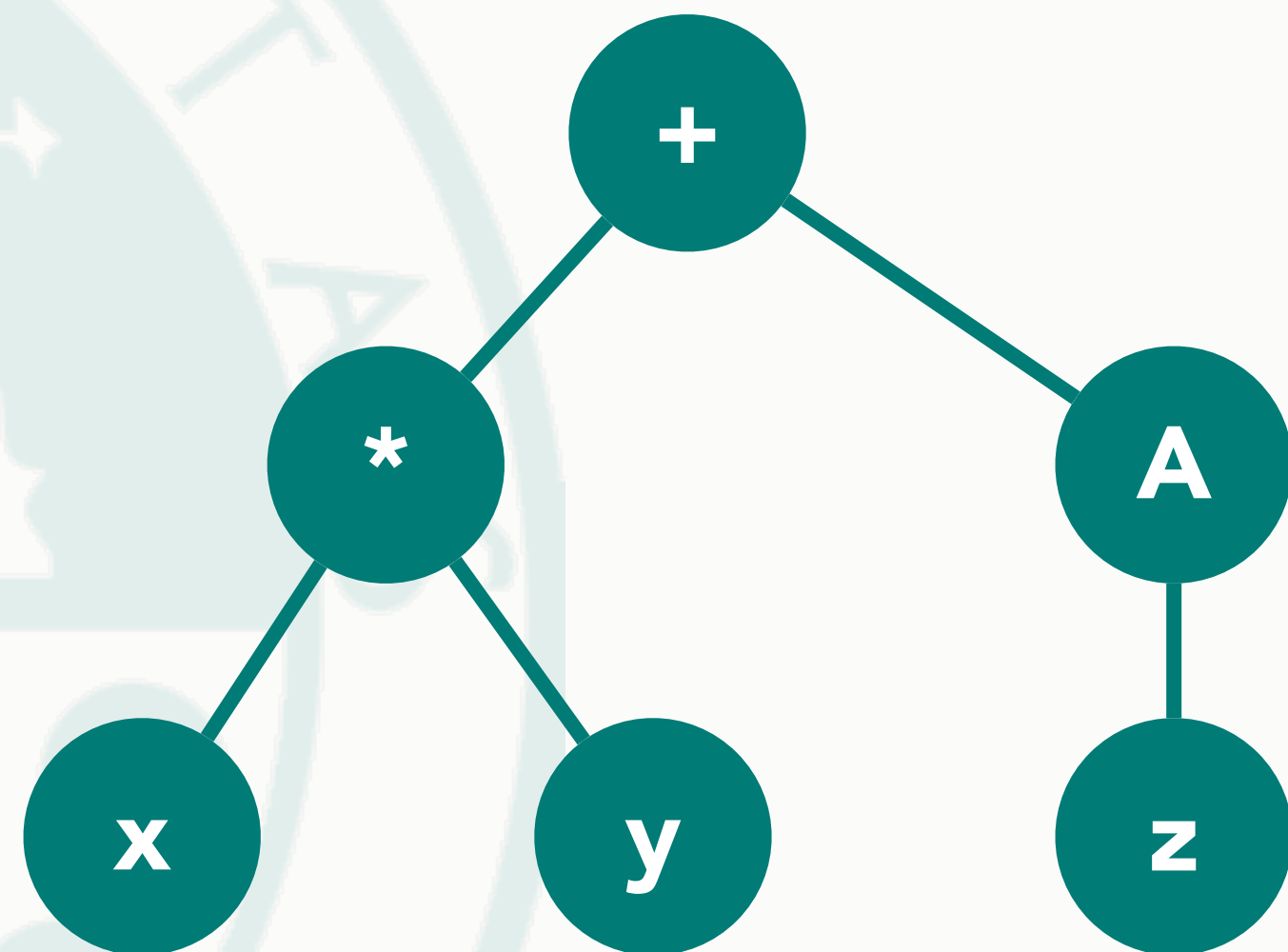


$(x + z) * x - (z + y/x)$

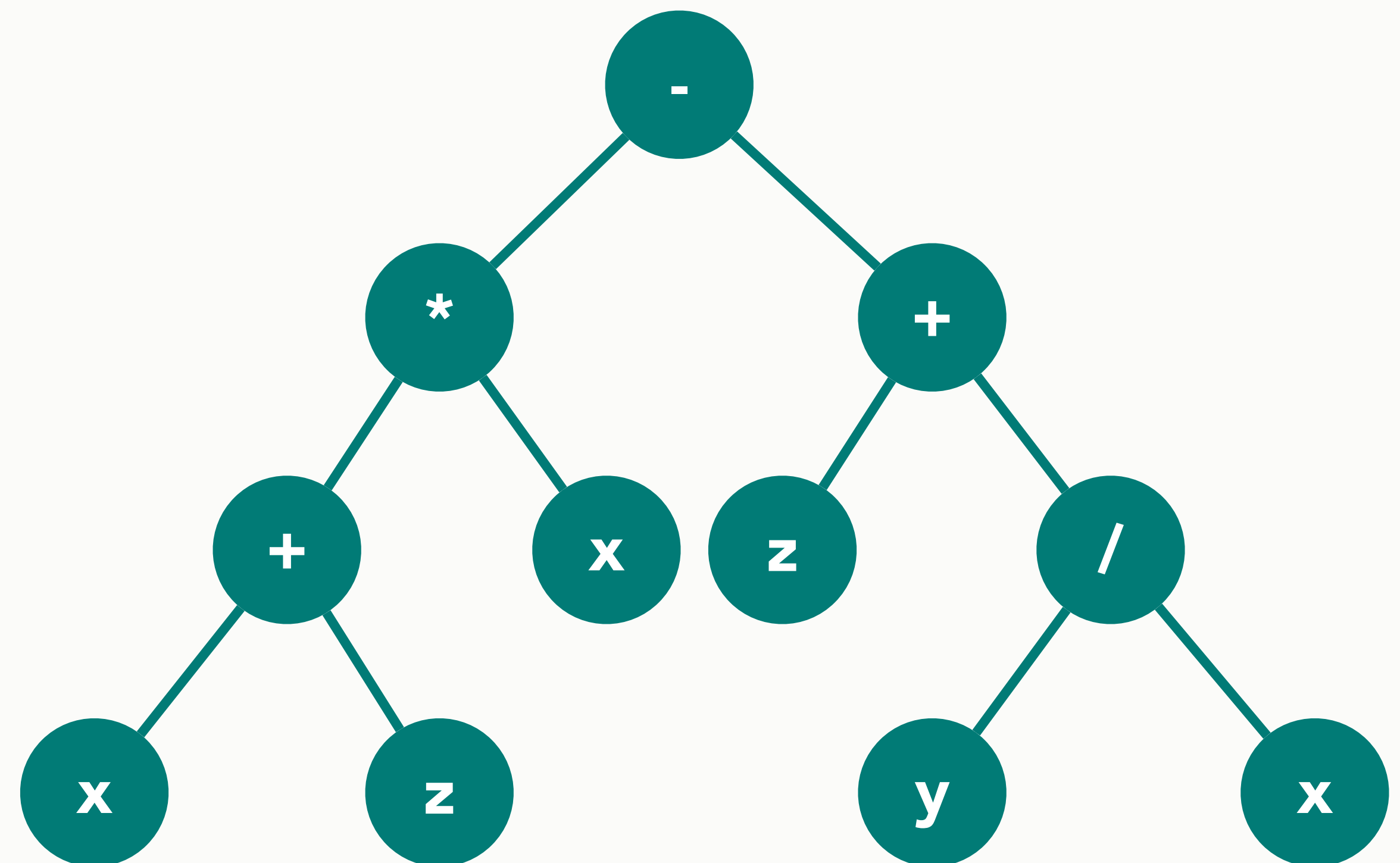
# Fundamentos

operador de cruce

**Padre 1**



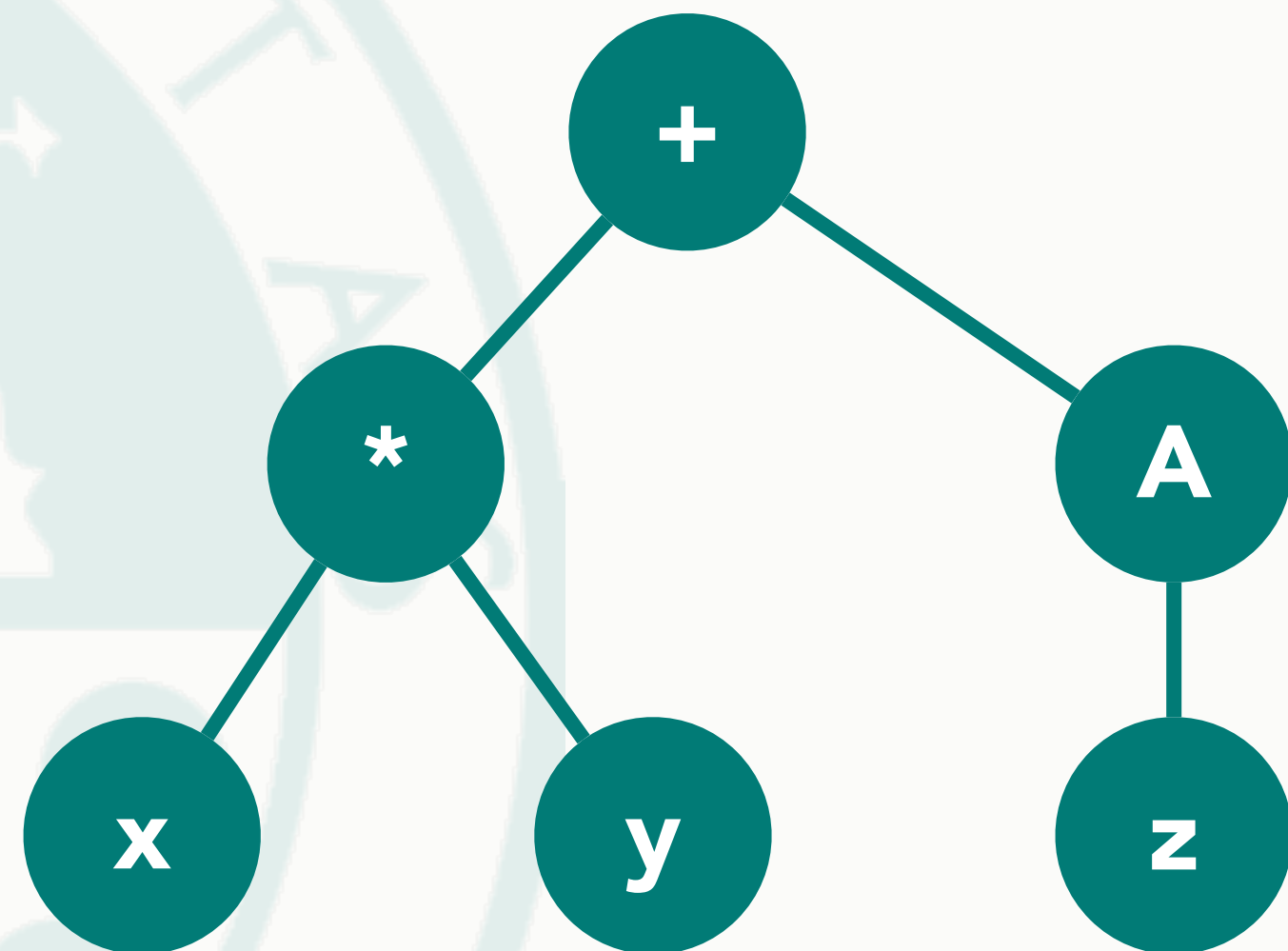
**Padre 2**



# Fundamentos

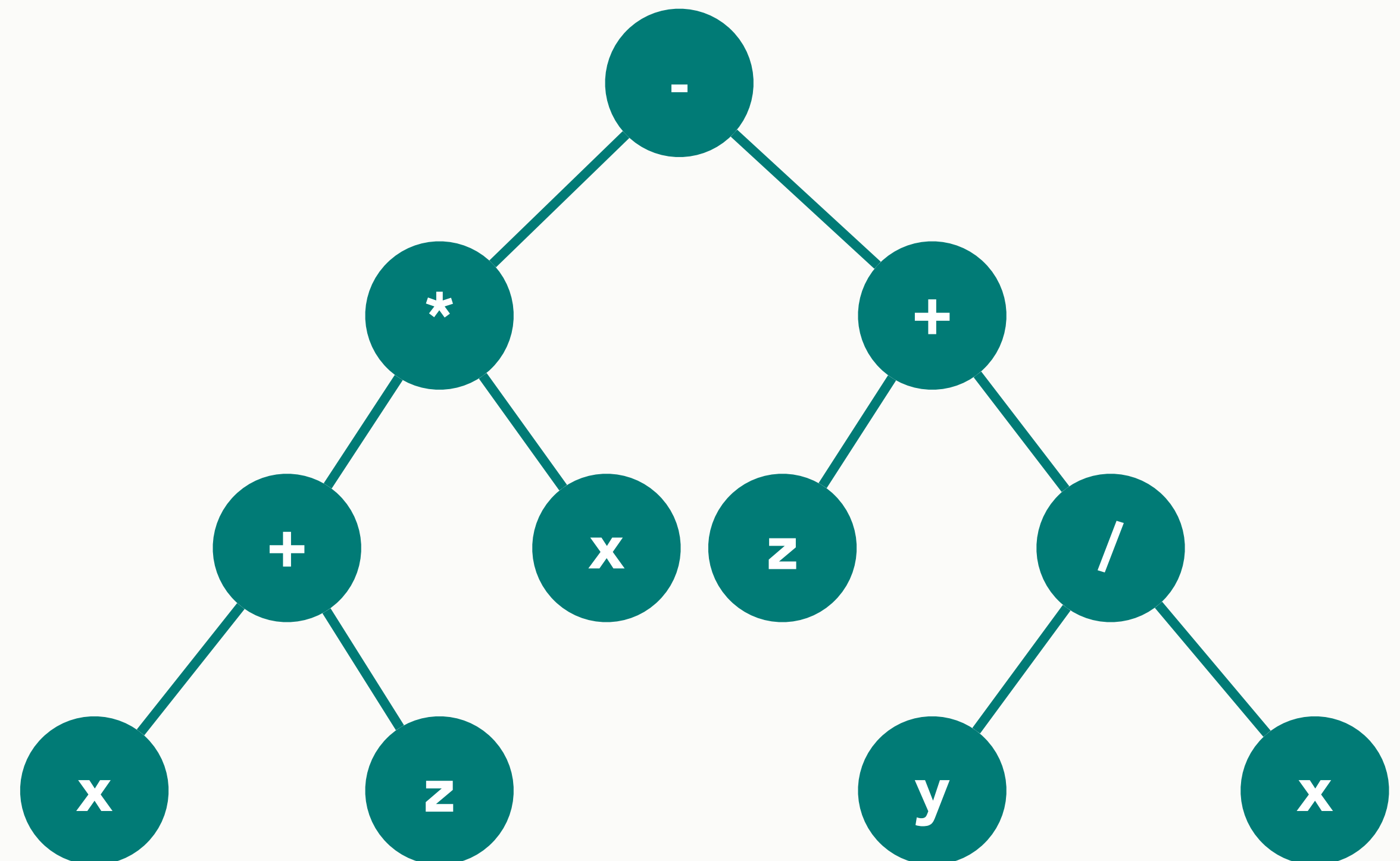
operador de cruce

**Padre 1**



**Seleccionamos el  
punto de corte**

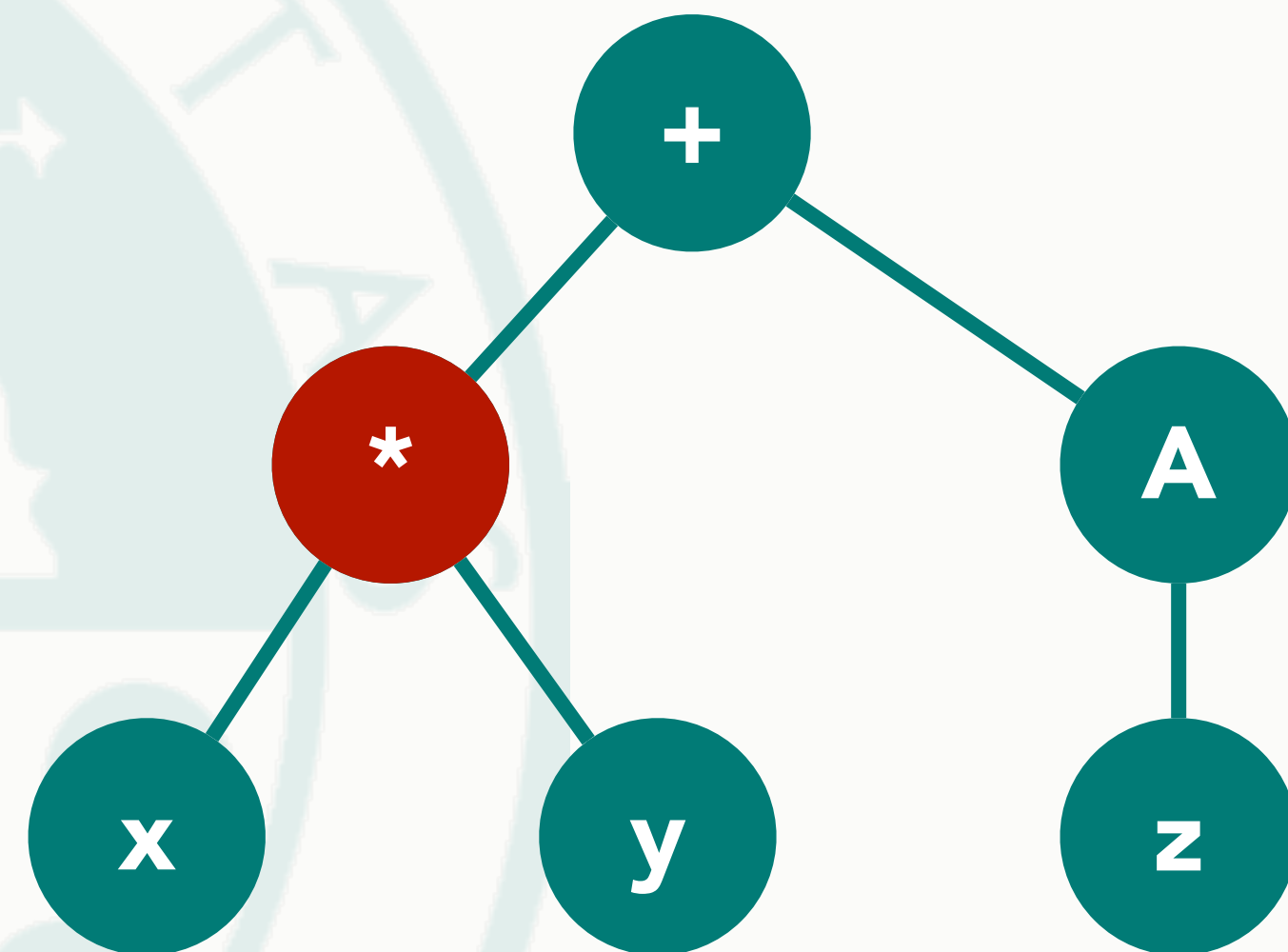
**Padre 2**



# Fundamentos

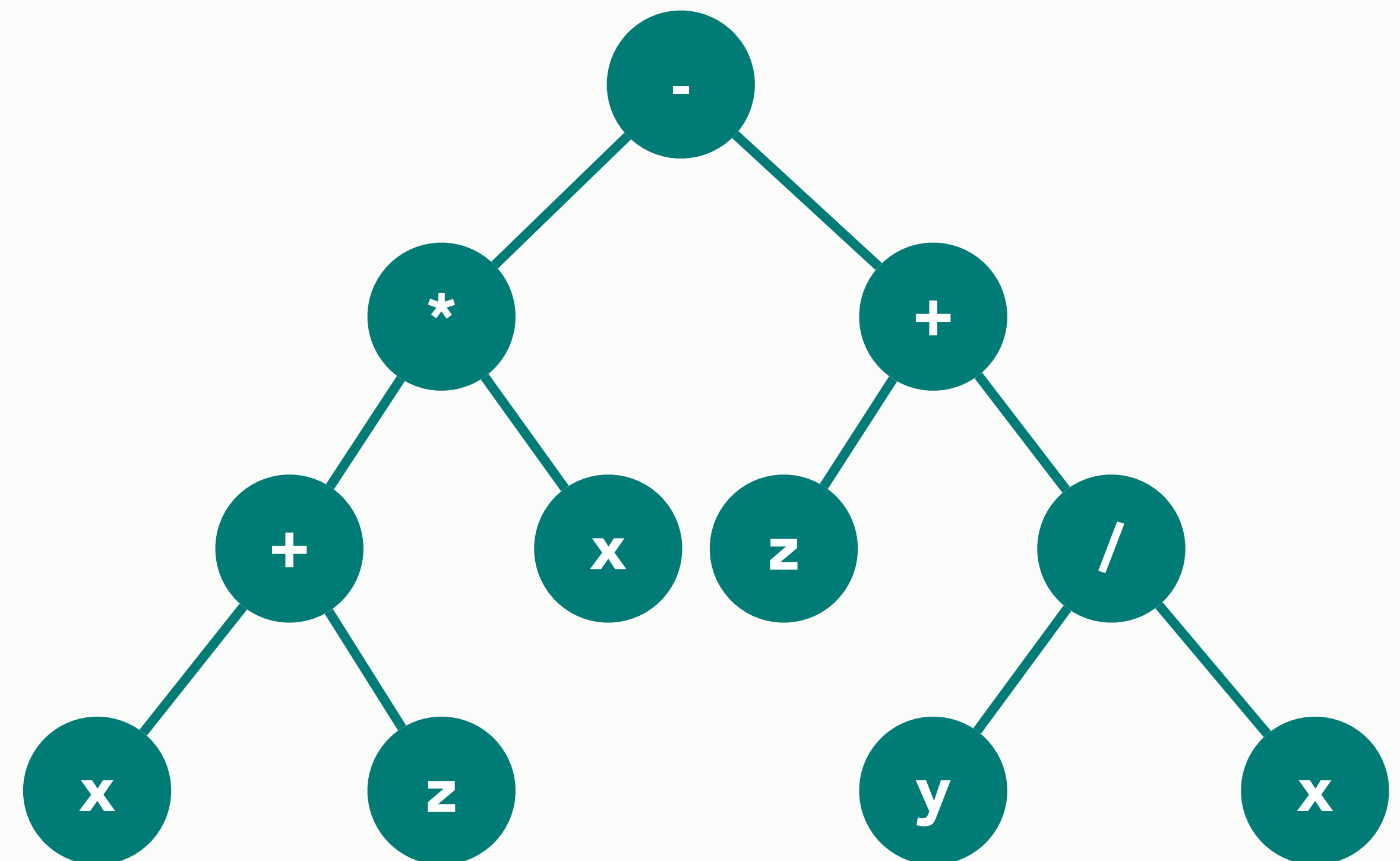
operador de cruce

**Padre 1**



**Seleccionamos el  
punto de corte**

**Padre 2**

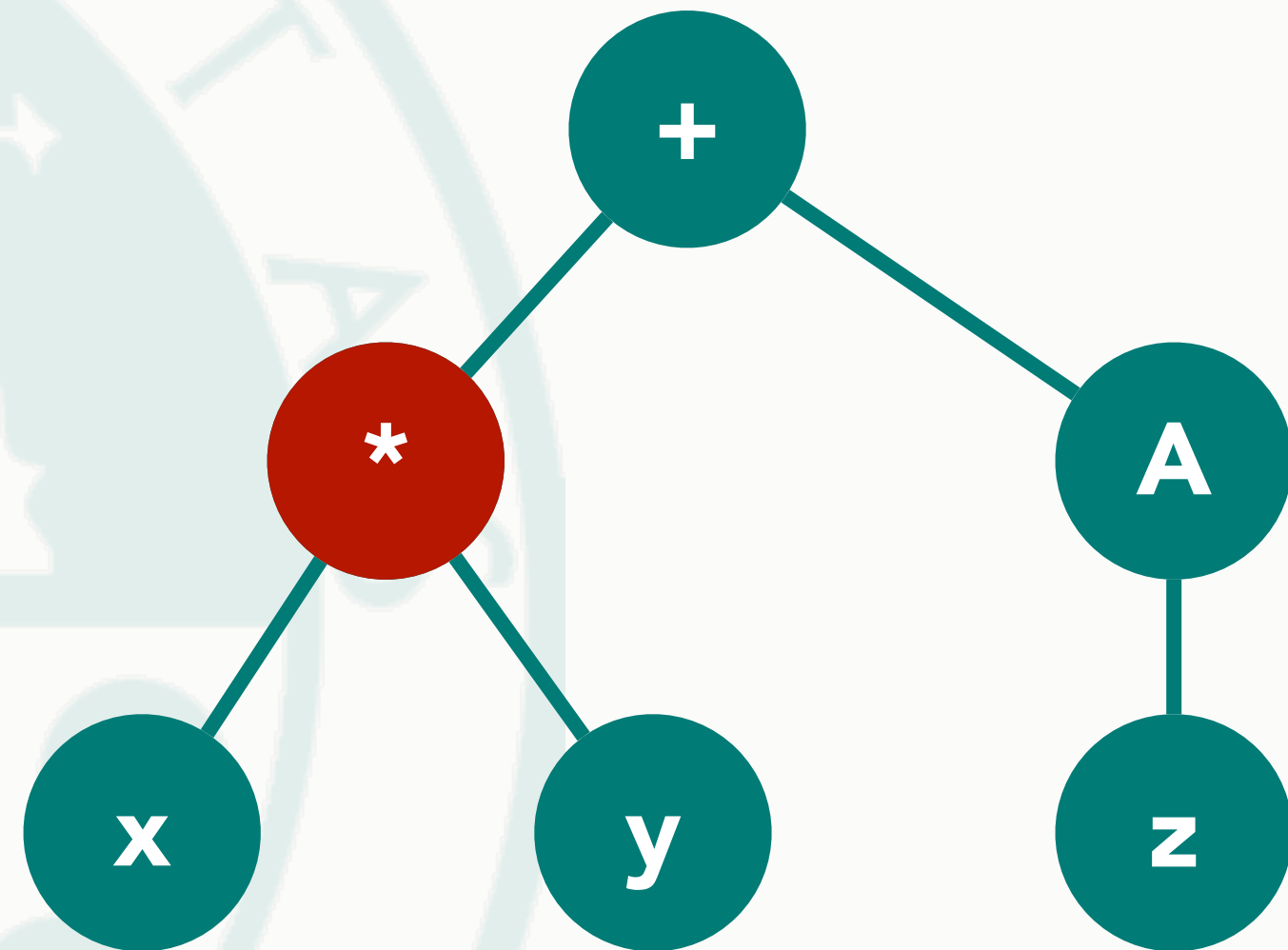




# Fundamentos

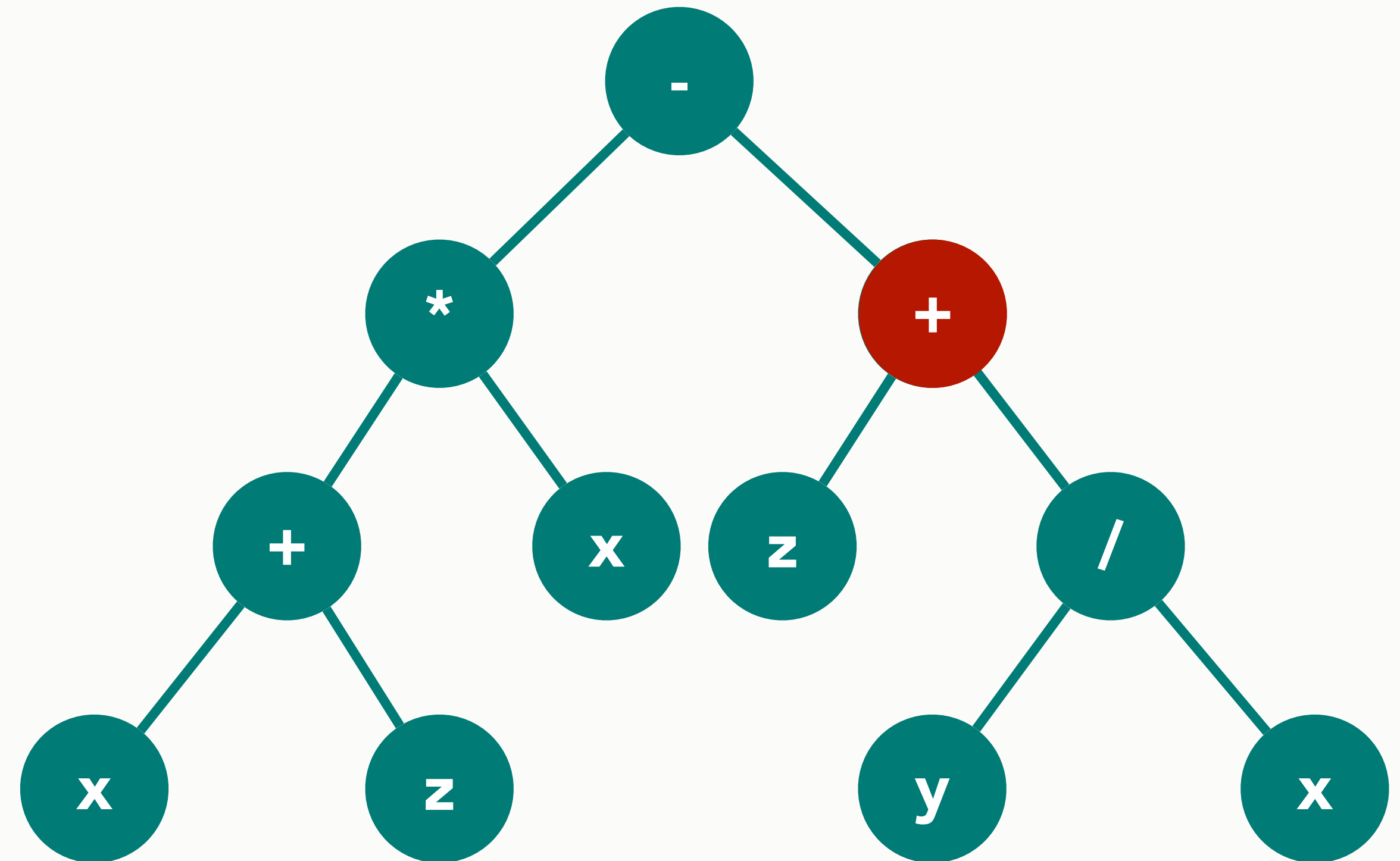
operador de cruce

**Padre 1**



**Seleccionamos el  
punto de corte**

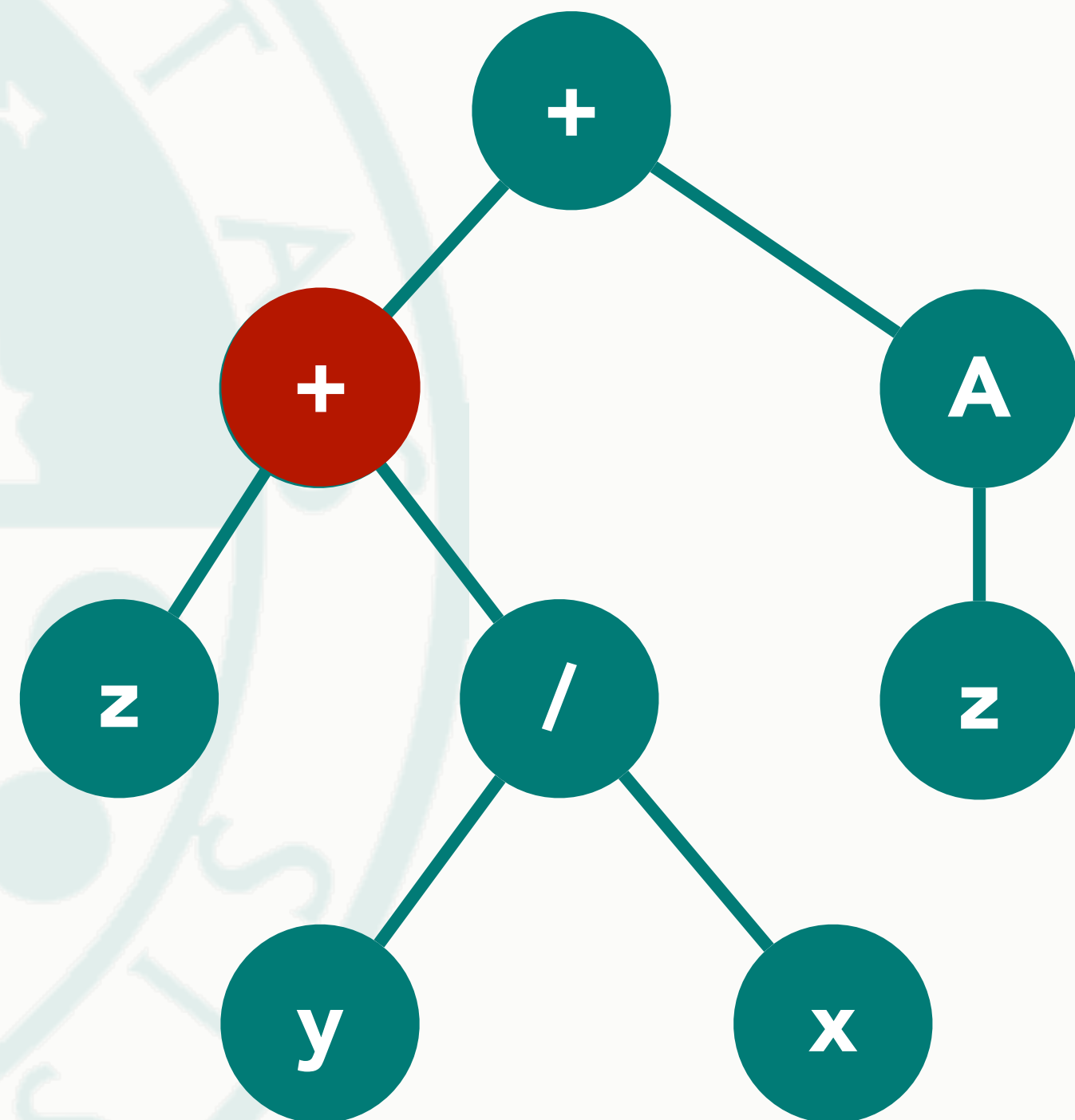
**Padre 2**



# Fundamentos

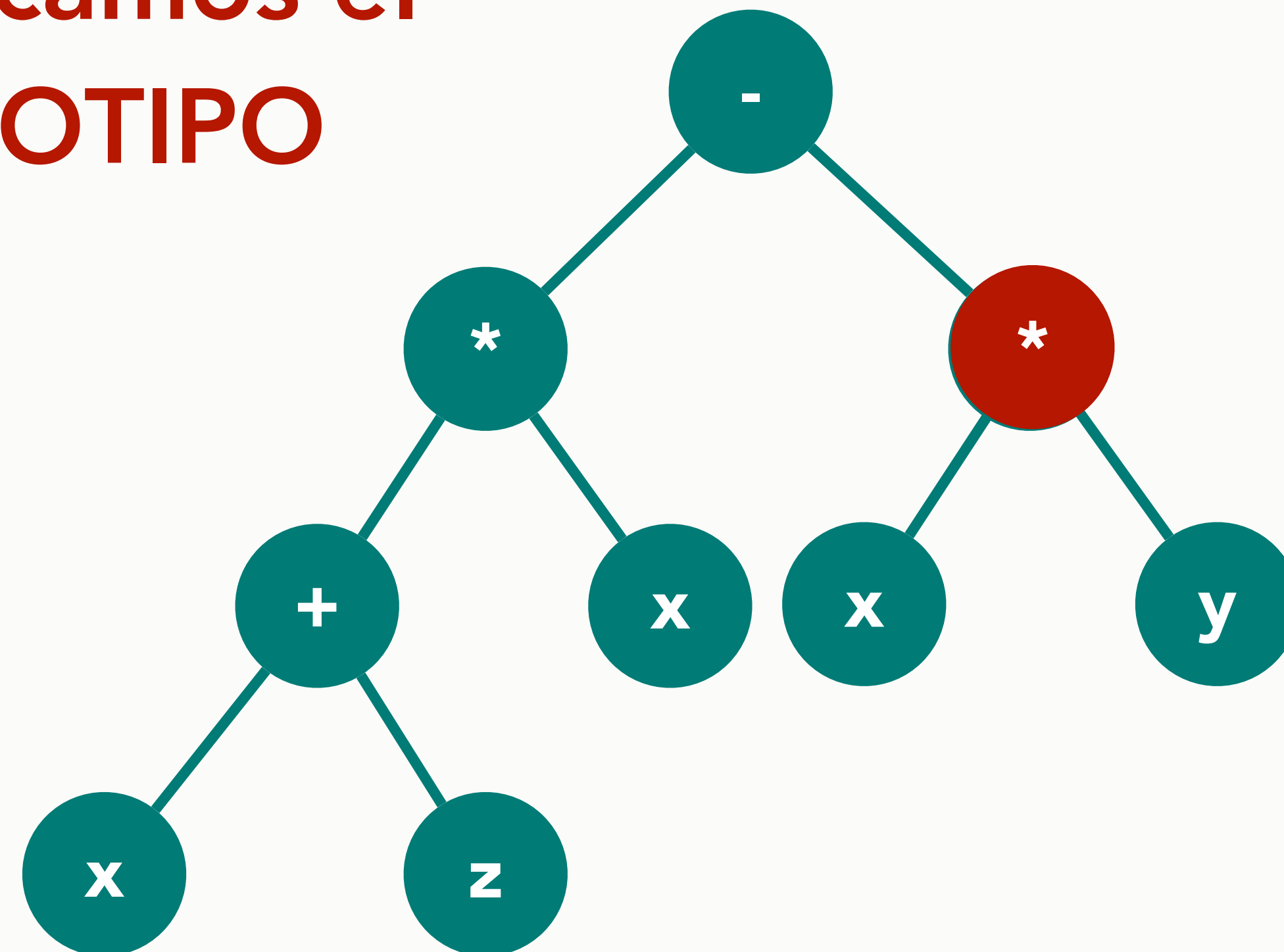
operador de cruce

Hijo 1



Modificamos el  
GENOTIPO

Hijo 2



# Fundamentos

consideraciones de la selección de los puntos de corte

- **Mayor eficiencia** si se sesga la **probabilidad** de selección de los arcos conectados a **nodos terminales** para hacerla **menor** que la de los conectados a **nodos internos**
- Valor adecuado **probabilidad selección de arcos en nodos terminales** menor que  $1/\text{número de arcos del árbol}$
- Descendientes **más diferentes** de los padres
- Segundo árbol debe verificar las **restricciones sintácticas** como tipo y tamaño

# Fundamentos

consideraciones sobre el operador de cruce

- **Operador muy disruptivo**, es decir, los árboles tienden a crecer indefinidamente si no se limita el tamaño
- No hacer el cruce si **el subárbol** escogido en el segundo árbol hace que el descendiente **supere el tamaño máximo**
- Incorporar **criterios para penalizar una excesiva complejidad** de los programas a la función de adaptación



# Fundamentos

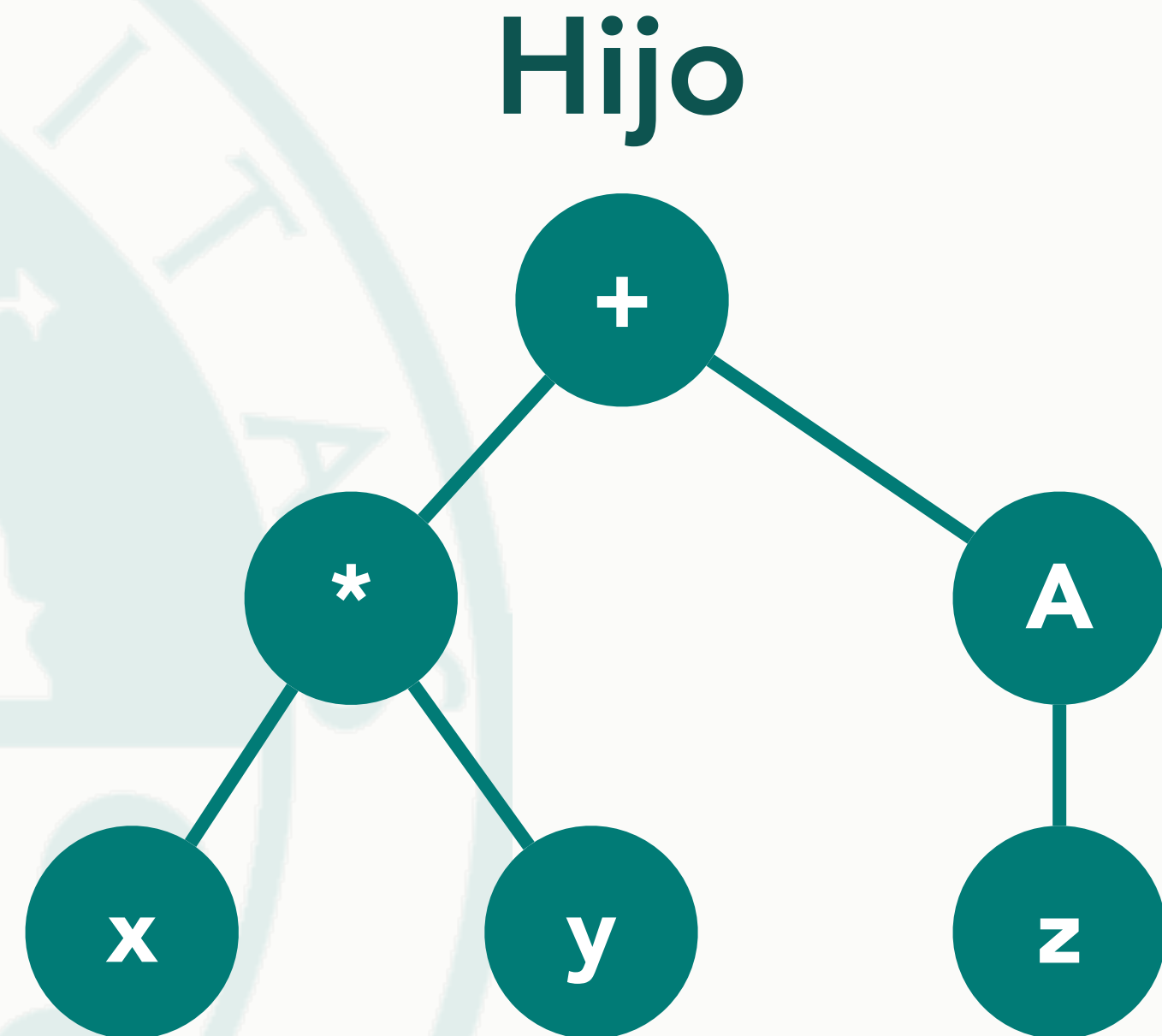
consideraciones sobre el operador de cruce

- El cruce no genera dos individuos iguales como en los algoritmos genéticos
- Puede dar descendientes inadecuados aunque los padres fuesen muy buenos
- Exploración muy buena pero no explota adecuadamente
- Necesitamos poblaciones mucho mayores



# Fundamentos

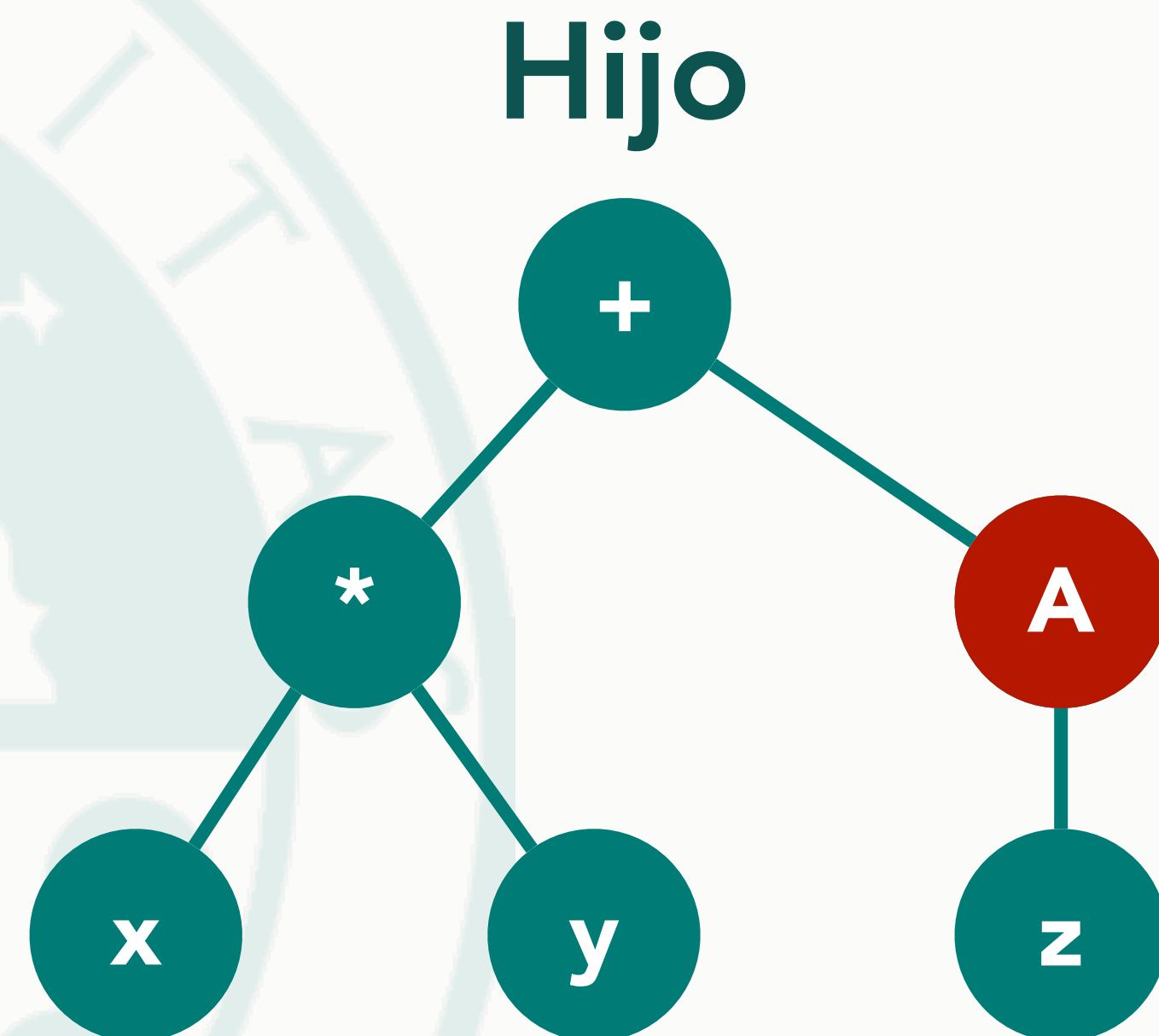
operador de mutación



$xy + |z|$

# Fundamentos

operador de mutación

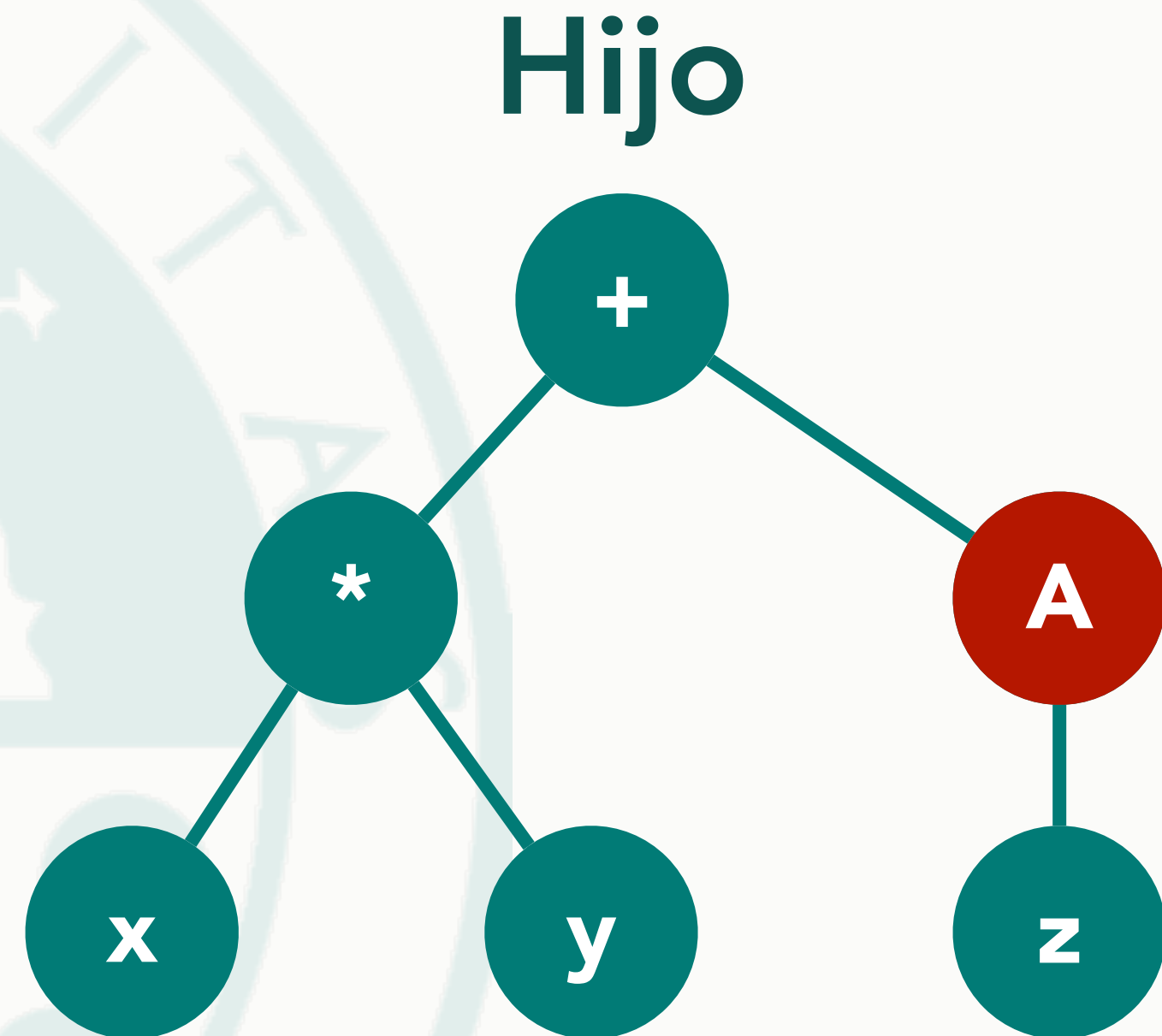


Seleccionamos el  
nodo a mutar

$$xy + |z|$$

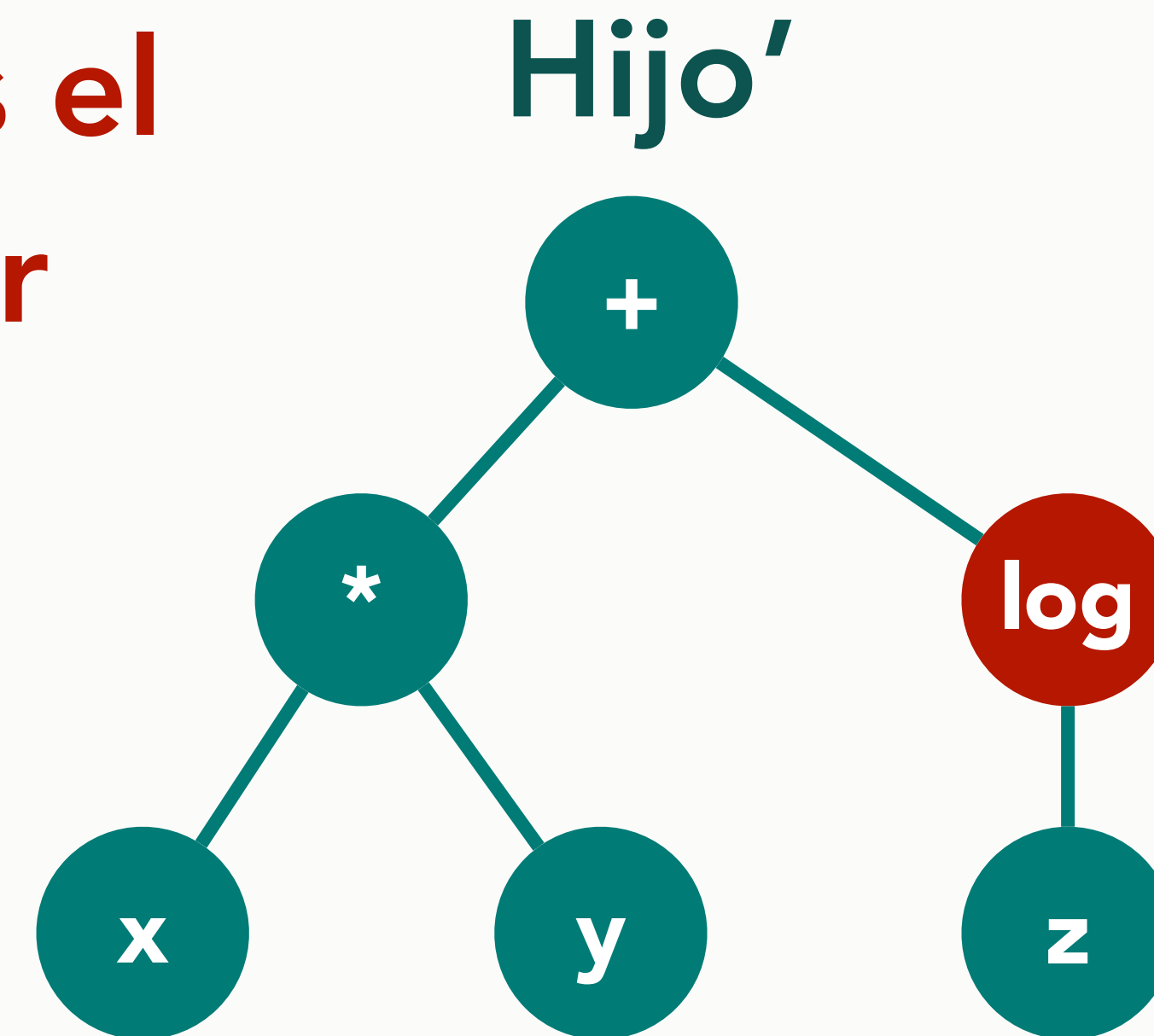
# Fundamentos

operador de mutación



$$xy + |z|$$

**Seleccionamos el  
nodo a mutar**



$$xy + \log_{10}(z)$$

# Fundamentos

operador de mutación

Otros métodos utilizados son:

- Reemplazamiento completo del subárbol a un nodo
- Expansión de un nodo terminal
- Permutación de nodos
- Mutación de constantes

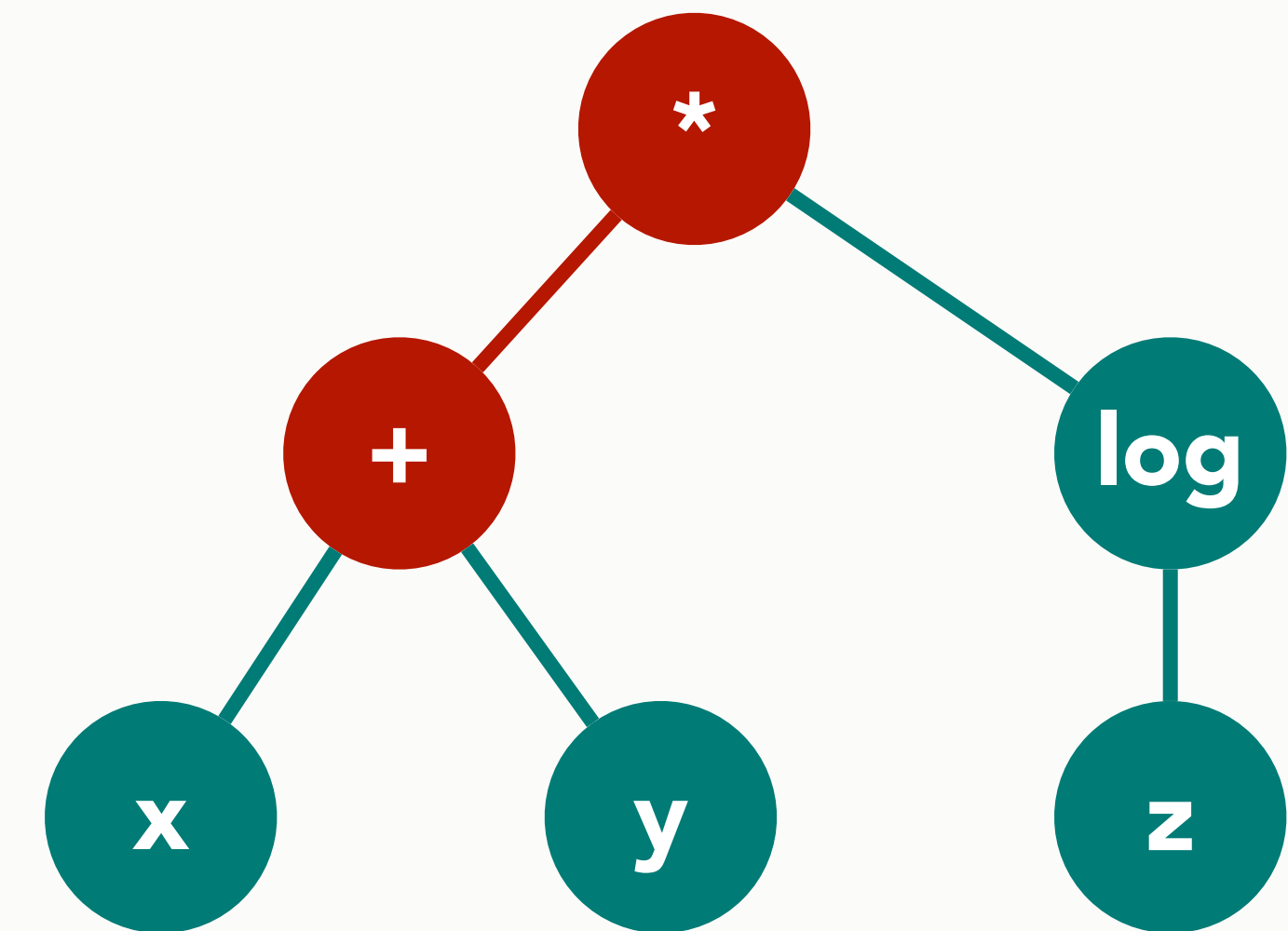
# Fundamentos

operador de mutación

Otros métodos utilizados son:

- Reemplazamiento completo del subárbol a un nodo
- Expansión de un nodo terminal
- Permutación de nodos
- Mutación de constantes

## Permutación



$$(x + y) * \log_{10}(z)$$



# Fundamentos

criterio de parada

- Concepto a resolver similar al resto de enfoques evolutivos
- Puede darse condicionado a diferentes factores
  - Número de evaluaciones
  - Número de generaciones
  - Tiempo de cómputo
  - Cambios en el fitness

## Otras consideraciones

- La selección del punto de corte para el cruce se puede utilizar con mecanismos que permitan mejorar la ganancia de los padres
- La selección para el reemplazamiento es similar al del resto de algoritmos evolutivos
- La recolección de repetidos es importante en este paradigma por la cantidad de recursos consumidos

# Otras consideraciones

parámetros

- El tamaño de la población en la programación genética es mayor que en los algoritmos genéticos, por ejemplo, podríamos de hablar de 500 individuos frente a los 50/100 de los genéticos
- Probabilidades de cruce y mutación varían según las necesidades de resolución del problema, en algunos casos son adaptativos
- Se podría utilizar élite en este paradigma

# Otras consideraciones

parámetros

- Un concepto fundamental es la profundidad de los árboles generados, así como el de los árboles descendientes
- En general se permite la elección del nodo terminal para poder realizar combinaciones



# Otras consideraciones

- Buenos resultados en problemas con representaciones matemáticas
  - ★ Descubrir relaciones trigonométricas
  - ★ Detectar/Contrastar teoremas científicos
  - ★ Inducir forma simbólica de secuencias
  - ★ Controlar el mínimo tiempo en un sistema de aceleración



# Ejercicio

Define la gramática y el genotipo de la expresión cuadrática siguiente

$$\frac{(\sqrt{b^2 - 4ac} - b)}{2a}$$

# Ejercicio: Regresión simbólica

Evolucionar una función que aproxime  
10 puntos de una parábola mediante  
programación genética

$$y = ?$$

Gramática

$S \rightarrow \text{número}$

$S \rightarrow x$

$S \rightarrow + S S$

$S \rightarrow * S S$

Terminales  $\in [-1, 1]$

Parámetros	
Tamaño	100
Cruce y mutación	95% y 5%
Selección	Elitista. Torneo
Generaciones	100
Altura máxima	5
Inicialización	Aleatoria

# Ejercicio: Regresión simbólica

Empleamos el error cuadrático medio como función de adaptación o fitness con respecto a la regresión simbólica en 10 puntos de la función

$$ECM = \frac{1}{n} \sum_{i=1}^n (Y_i - \widehat{Y}_i)^2$$

VALOR PREDICHO  
VALOR REAL

$y = ?$

Puntos			
x	y	x	y
-1	1	0	1
-0.8	0.84	0.2	1.24
-0.6	0.76	0.4	1.56
-0.4	0.76	0.6	1.96
-0.2	0.84	0.8	2.44
		1	3

# Ejercicio: Regresión simbólica

$$ECM = \frac{1}{n} \sum_{i=1}^n (Y_i - \widehat{Y}_i)^2$$

VALOR  
REAL

VALOR  
PREDICHO

$$y' = -0.5x$$

Puntos						
x	y	y'		x	y	y'
-1	1	0.5	(1-0.5)^2	0	1	0
-0.8	0.84	0.4	(0.84-0.4)^2	0.2	1.24	-0.1
-0.6	0.76	0.3		0.4	1.56	-0.2
-0.4	0.76	0.2		0.6	1.96	-0.3
-0.2	0.84	0.1		0.8	2.44	-0.4
				1	3	-0.5

## Ejercicio: Regresión simbólica

Empleamos el error cuadrático medio como función de adaptación o fitness con respecto a la regresión simbólica en 10 puntos de la función

$$y = x^2 + x + 1$$

$$ECM = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

**VALOR REAL**      **VALOR PREDICHO**



## Ejercicio de regresión simbólica

- Inicializa cuatro individuos, represéntalos y calcula su fitness

- $x+1$

- $x^2+1$

- $2$

- $x$

- El mejor individuo en la generación 100 es este, repite el proceso anterior

- $(-0.733)*(-0.733)*(-0.262)*(-0.262)*x^2 + (-0.733)*(-0.733)*x^2$

# Metaheurísticas

## Grado en Ingeniería Informática

### Universidad de Jaén

### Cristóbal J. Carmona

### Curso 2023/2024

Esta obra está protegida con licencia  
Creative Commons Atribución-NoComercial 4.0 Internacional

