


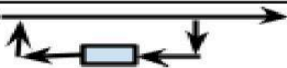

Conceptos y Paradigmas de Lenguajes de Programación 2023

Práctica Nro. 2

Sintaxis

Objetivo: conocer como se define léxicamente un lenguaje de programación y cuales son las herramientas necesarias para hacerlo

Ejercicio 1: Complete el siguiente cuadro:

Meta símbolos utilizados por		Símbolo utilizado en Diagramas sintacticos	Significado
BNF	EBNF		
palabra terminal	palabra terminal		Definición de un elemento terminal
		rectángulo 	Definición de un elemento no terminal
::=	::=	diagrama con rectángulos, óvalos y flechas	
	()	flecha que se divide en dos o más caminos	
< p > < p1 >			Repetición
			Repetición de 0 o más veces
	+		Repetición de 1 o más veces
	[]		

Nota: p y p1 son producciones simbólicas

Ejercicio 2: ¿Cuál es la importancia de la sintaxis para un lenguaje? ¿Cuáles son sus elementos?

Ejercicio 3: ¿Explique a qué se denomina regla lexicográfica y regla sintáctica?

Ejercicio 4: ¿En la definición de un lenguaje, a qué se llama palabra reservadas? ¿A qué son equivalentes en la definición de una gramática? De un ejemplo de palabra reservada en el lenguaje que más conoce. (Ada,C,Ruby,Python,...)

Conceptos y Paradigmas de Lenguajes de Programación 2023

Ejercicio 5: Dada la siguiente gramática escrita en BNF:

$G = (N, T, S, P)$

$N = \{ \langle \text{numero_entero} \rangle, \langle \text{digito} \rangle \}$

$T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$S = \langle \text{numero_entero} \rangle$

$P = \{$

$\langle \text{numero_entero} \rangle ::= \langle \text{digito} \rangle \langle \text{numero_entero} \rangle \mid \langle \text{numero_entero} \rangle \langle \text{digito} \rangle \mid \langle \text{digito} \rangle$

$\langle \text{digito} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\}$

a- Identifique las componentes de la misma

b- Indique porqué es ambigua y corríjala

Ejercicio 6: Defina en BNF (Gramática de contexto libre desarrollada por Backus- Naur) la gramática para la definición de una palabra cualquiera.

Ejercicio 7: Defina en EBNF la gramática para la definición de números reales. Inténtelo desarrollar para BNF y explique las diferencias con la utilización de la gramática EBNF.

Ejercicio 8: Utilizando la gramática que desarrolló en los puntos 6 y 7, escriba el árbol sintáctico de:

- a. Conceptos
- b. Programación
- c. 1255869
- d. 854,26
- e. Conceptos de lenguajes

Ejercicio 9: Defina utilizando diagramas sintácticos la gramática para la definición de un identificador de un lenguaje de programación. Tenga presente como regla que un identificador no puede comenzar con números.

Ejercicio 10:

a) Defina con EBNF la gramática para una expresión numérica, dónde intervienen variables y números. Considerar los operadores +, -, * y / sin orden de prioridad. No considerar el uso de paréntesis.

b) A la gramática definida en el ejercicio anterior agregarle prioridad de operadores.

c) Describa con sus palabras los pasos y decisiones que tomó para agregarle prioridad de operadores al ejercicio anterior.

Conceptos y Paradigmas de Lenguajes de Programación 2023

Ejercicio 11: La siguiente gramática intenta describir sintácticamente la sentencia for de ADA, indique cuál/cuáles son los errores justificando la respuesta.





```
N= {<sentencia_for>, <bloque>, <variable>, <letra>, <cadena>, <digito>, <otro>, <operacion>,
    <llamada_a_funcion>, <numero>, <sentencia> }
P= { <sentencia_for>::= for (i= IN 1..10) loop <bloque> end loop;
    <variable>::= <letra> | <cadena>
    <cadena>::= { ( <letra> | <digito> | <otro> ) }+
    <letra>::= ( a | .. | z | A | .. | Z )
    <digito>::= ( 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 )
    <bloque>::= <sentencia> | <sentencia> <bloque> | <bloque> <sentencia> ;
    <sentencia>::= <sentencia_asignacion> | <llamada_a_funcion> | <sentencia_if> |
    <sentencia_for> | <sentencia_while> | <sentencia_switch> }
```

Ejercicio 12: Realice en EBNF la gramática para la definición un tag div en html 5. (Puede ayudarse con el siguiente enlace (<https://developer.mozilla.org/es/docs/Web/HTML/Elemento/div>))

Ejercicio 13: Defina en EBNF una gramática para la construcción de números primos. ¿Qué debería agregar a la gramática para completar el ejercicio?

Ejercicio 14: Sobre un lenguaje de su preferencia escriba en EBNF la gramática para la definición de funciones o métodos o procedimientos (considere los parámetros en caso de ser necesario)

1. Cuadro

Meta símbolos utilizados por		Símbolo utilizado en Diagramas sintácticos	Significado
BNF	EBNF		
palabra terminal	palabra terminal		Definición de un elemento terminal
< >	< >	rectángulo 	Definición de un elemento no terminal
::=	::=	diagrama con rectángulos, óvalos y flechas	Meta-símbolo de definición que indica que el elemento a su izquierda se puede definir según el esquema de la derecha
	()	flecha que se divide en dos o más caminos	Selección de una alternativa
< p > < p1 >	{ }		Repetición
	*		Repetición de 0 o más veces
	+		Repetición de 1 o más veces
	[]		Opcional, está presente o no lo está

2. Sintaxis:

- La sintaxis es el conjunto de reglas que definen como componer letras, dígitos y otros caracteres para formar los programas, esta, establece reglas que definen cómo deben combinarse las componentes básicas, llamadas "word", para formar sentencias y programas. La sintaxis en un lenguaje de programación es fundamental ya que dicta las reglas y estructuras que deben seguirse para escribir instrucciones válidas en el código fuente, es decir, establece las reglas para que el programador se comunique con el procesador de forma correcta sintácticamente.

Elementos de la Sintaxis:

- Alfabeto o conjunto de caracteres.
- Identificadores.
- Operadores.
- Palabra clave y palabra reservada.
- Comentarios y uso de blancos.

3. Reglas de la Sintaxis:

- Reglas Léxicas/Lexicográficas: Conjunto de reglas para formar las "word", a partir de los caracteres del alfabeto.
- Reglas Sintácticas: Conjunto de reglas que definen cómo formar las "expresiones" y "sentencias".

4. Palabras Reservadas:

- En la definición de un lenguaje de programación, las "palabras reservadas" son aquellas que tienen un significado específico dentro del lenguaje y que no pueden ser utilizadas para otros propósitos, como nombres de variables, funciones o clases. Estas palabras están reservadas para ser utilizadas por el propio lenguaje y forman parte de su sintaxis. Estas, se corresponden con el conjunto de símbolos terminales de la gramática. Un ejemplo de palabra reservada en Python es "if". En Python, "if" se utiliza para definir una estructura condicional, y no puede ser utilizado como nombre de variable, función u otro identificador.

5. Dada la siguiente gramática escrita en BNF:

$G = (N, T, S, P)$

$N = \{ \langle \text{numero_entero} \rangle, \langle \text{digito} \rangle \}$

$T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$S = \langle \text{numero_entero} \rangle$

$P = \{$

$\langle \text{numero_entero} \rangle ::= \langle \text{digito} \rangle \langle \text{numero_entero} \rangle \mid \langle \text{numero_entero} \rangle \langle \text{digito} \rangle \mid \langle \text{digito} \rangle$

$\langle \text{digito} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

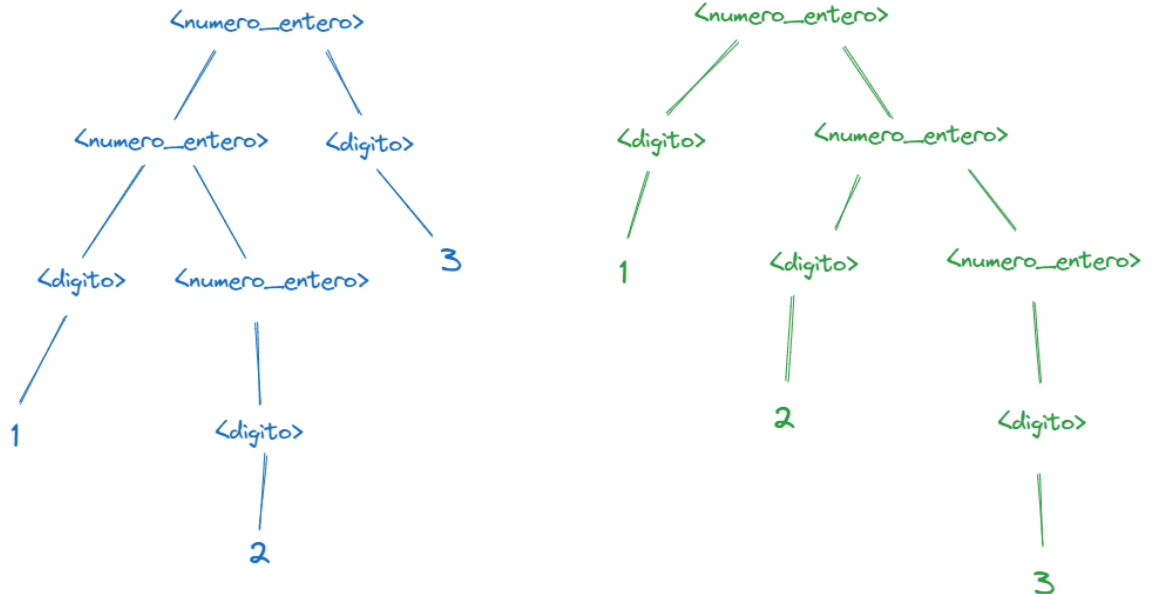
$\}$

a) Componentes:

- G: 4-tupla que define un conjunto de reglas finitas que define un conjunto infinito de posibles sentencias válidas en el lenguaje.
- N: Conjunto de símbolos no terminales.
- T: Conjunto de símbolos terminales.
- S: Símbolo distinguido de la gramática que pertenece a N.
- P: Conjunto de producciones.

- b) Una gramática es ambigua si una sentencia puede derivarse de más de una forma, teniendo en cuenta esto, hago el siguiente ejemplo con árboles de derivación:

EJEMPLO: 123 TOP-DOWN DE IZQUIERDA A DERECHA



Podemos ver que de dos formas distintas podemos conseguir la misma sentencia, haciendo que esta gramática sea ambigua. Lo que la hace ambigua es esta sentencia:

"`<numero_entero> ::= <digito><numero_entero> | <numero_entero><digito>`" ya que la asociatividad tiene que ser por la izquierda o por la derecha.

Corrección de la Gramática:

- $G = (N, T, S, P)$
 - $N = \{ \langle \text{numero_entero} \rangle, \langle \text{digito} \rangle \}$
 - $T = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
 - $S = \langle \text{numero_entero} \rangle$
 - $P = \{$
 - $\langle \text{numero_entero} \rangle ::= \langle \text{digito} \rangle \langle \text{numero_entero} \rangle \mid \langle \text{digito} \rangle$
 - $\langle \text{digito} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
 - $\}$

6. Definición en BNF

- $G = (N, T, S, P)$
 - $N = \{ \langle \text{palabra} \rangle, \langle \text{mayúscula} \rangle, \langle \text{minúscula} \rangle \}$
 - $T = \{ A, \dots, Z, a, \dots, z \}$
 - $S = \langle \text{palabra} \rangle$
 - $P = \{$
 - $\langle \text{palabra} \rangle ::= \langle \text{mayúscula} \rangle \langle \text{palabra} \rangle \mid \langle \text{minúscula} \rangle \langle \text{palabra} \rangle \mid$
 - $\langle \text{minúscula} \rangle \mid \langle \text{mayúscula} \rangle$

```

<mayúscula> ::= A | B | C | D | E | F | ... | X | Y | Z
<minúscula> ::= a | b | c | d | e | f | ... | x | y | z
}

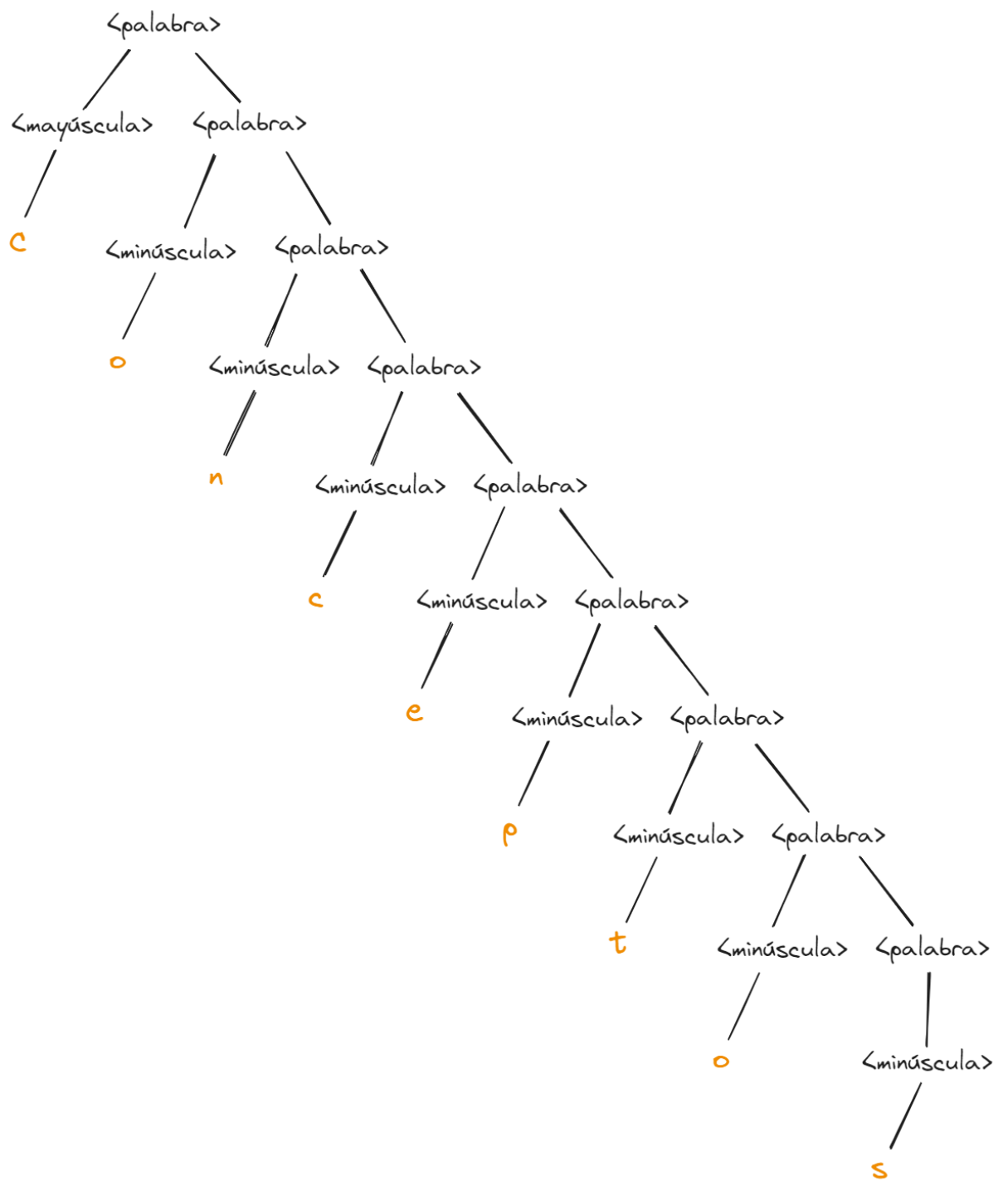
```

7. Definición en EBNF y BNF

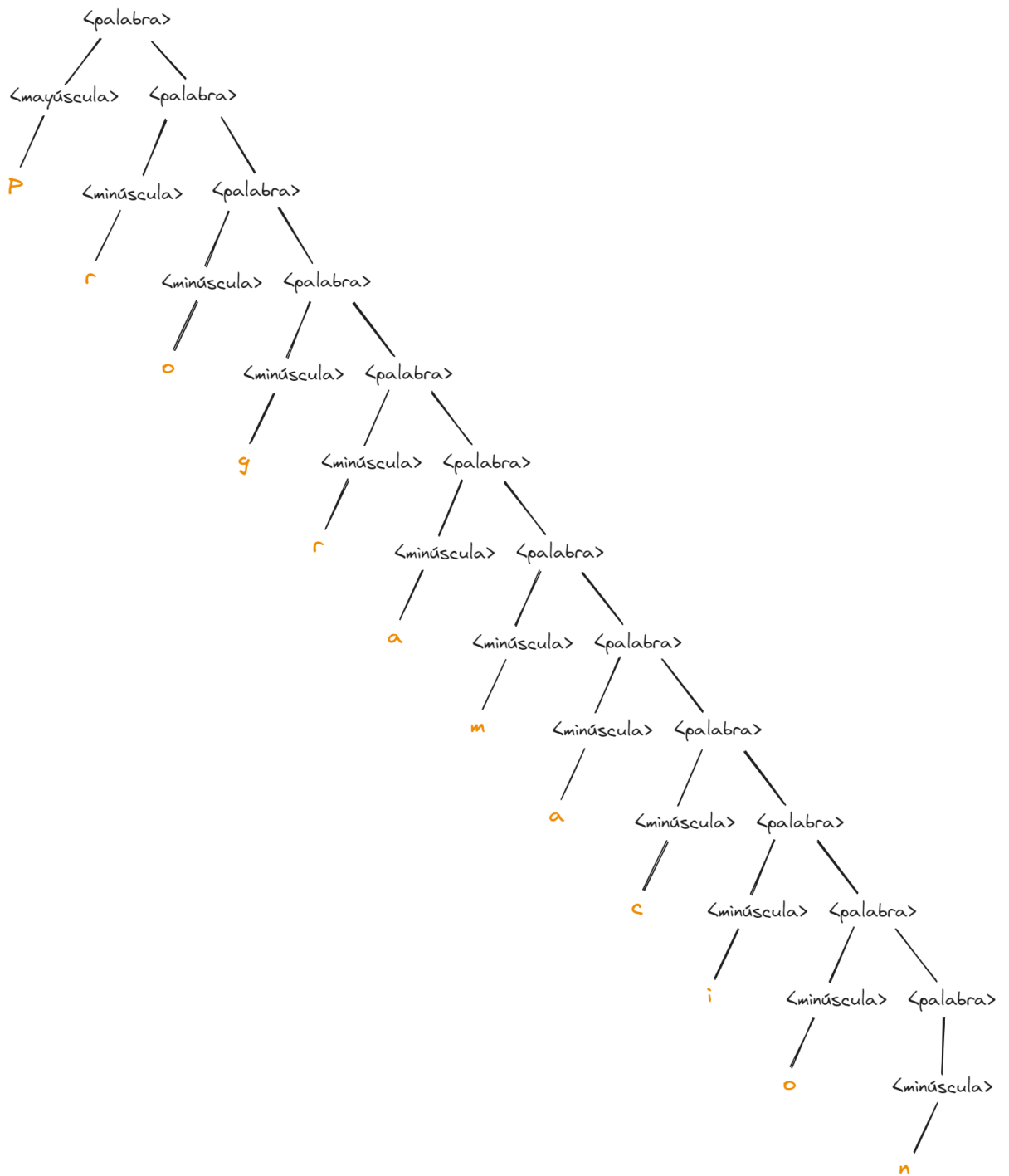
- EBNF:
 - $G = \{N, T, S, P\}$
 - $N = \{\langle \text{numero_real} \rangle, \langle \text{dígito} \rangle\}$
 - $T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, "+", "-", ","\}$
 - $S = \langle \text{numero_real} \rangle$
 - $P = \{$
 - $\langle \text{numero_real} \rangle ::= [(+|-)] \{\langle \text{dígito} \rangle\}^+ [., \{\langle \text{dígito} \rangle\}^+]$
 - $\langle \text{dígito} \rangle ::= (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)$
 - $\}$
- BNF:
 - $G = \{N, T, S, P\}$
 - $N = \{\langle \text{numero_real} \rangle, \langle \text{entero_sig} \rangle, \langle \text{entero} \rangle, \langle \text{decimal} \rangle, \langle \text{dígito} \rangle\}$
 - $T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, "+", "-", ","\}$
 - $S = \langle \text{numero_real} \rangle$
 - $P = \{$
 - $\langle \text{numero_real} \rangle ::= \langle \text{entero_sig} \rangle | \langle \text{entero_sig} \rangle \langle \text{decimal} \rangle$
 - $\langle \text{decimal} \rangle ::= , \langle \text{entero} \rangle$
 - $\langle \text{entero_sig} \rangle ::= + \langle \text{entero} \rangle | - \langle \text{entero} \rangle | \langle \text{entero} \rangle$
 - $\langle \text{entero} \rangle ::= \langle \text{dígito} \rangle | \langle \text{dígito} \rangle \langle \text{entero} \rangle$
 - $\langle \text{dígito} \rangle ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$
 - $\}$
- La diferencia más clara entre las dos formas es la cantidad de símbolos no terminales que necesita la BNF para poder definir lo mismo que definimos en la EBNF con menos símbolos, además del hecho que la BNF en mi opinión se lee peor que la EBNF.

8. Árboles sintácticos (Top-Down de izquierda a derecha):

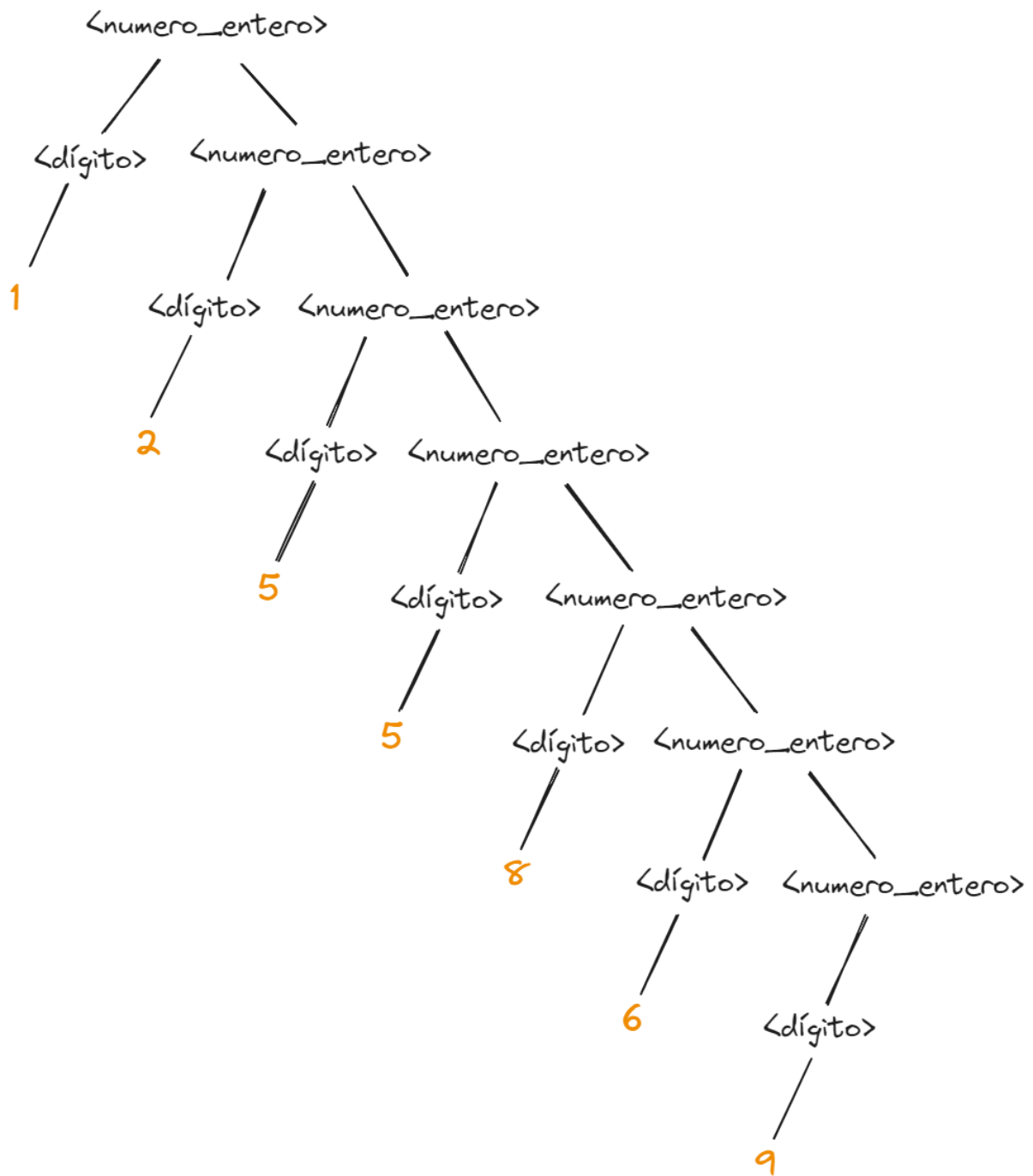
- Conceptos:



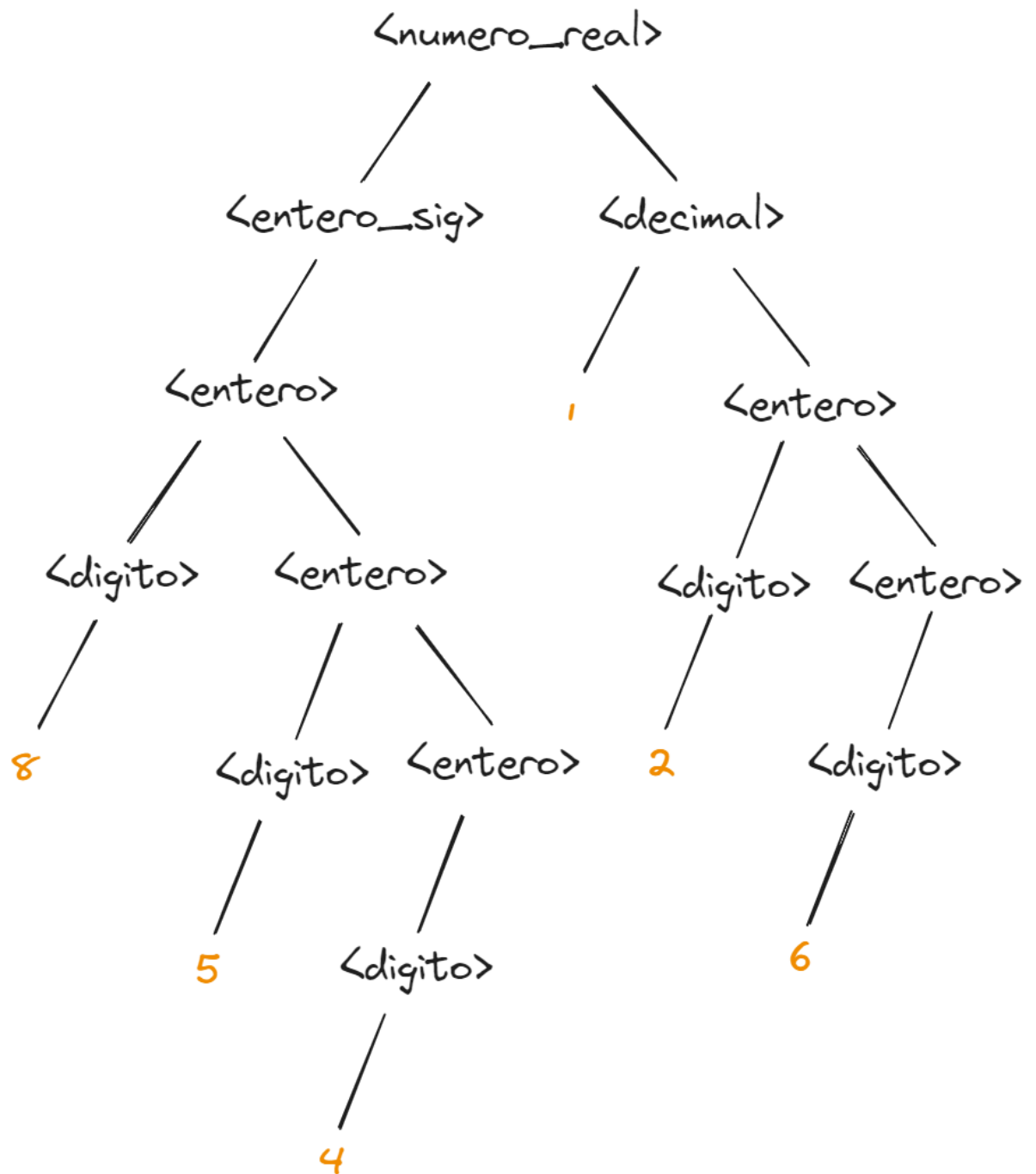
- Programación:



- 1255869:

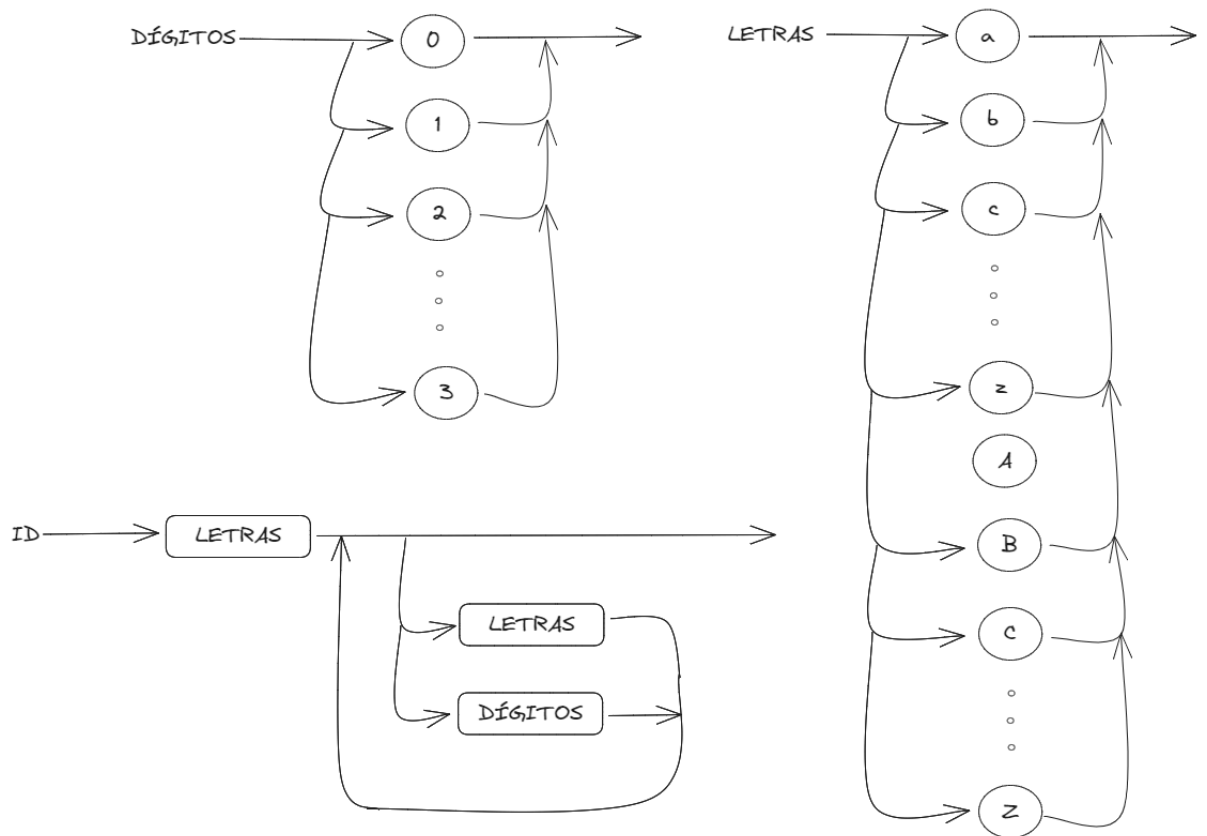


- 854,26:



- Conceptos de lenguajes:
Con las gramáticas definidas no se puede representar una cadena de texto, sino palabras individuales.

9. Diagrama sintáctico para un ID:



10. Expresiones numéricas:

a) Definición en EBNF sin orden de prioridad:

- $G = \{N, T, S, P\}$
 $N = \{ \langle \text{expr} \rangle, \langle \text{variable} \rangle, \langle \text{operador} \rangle, \langle \text{número} \rangle, \langle \text{dígito} \rangle, \langle \text{letra} \rangle \}$
 $T = \{ a, \dots, z, A, \dots, Z, 0, \dots, 9, "+", "-", "*", "/" \}$
 $S = \langle \text{expr} \rangle$
 $P = \{$
 $\langle \text{expr} \rangle ::= (\langle \text{variable} \rangle \mid \langle \text{número} \rangle) \{ \langle \text{operador} \rangle (\langle \text{id} \rangle \mid \langle \text{número} \rangle)$
 $\}^+$
 $\langle \text{número} \rangle ::= [(+ \mid -)] \{ \langle \text{dígito} \rangle \}^+$
 $\langle \text{variable} \rangle ::= \langle \text{letra} \rangle \{ (\langle \text{letra} \rangle \mid \langle \text{dígito} \rangle) \}^*$
 $\langle \text{dígito} \rangle ::= (0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9)$
 $\langle \text{operador} \rangle ::= (+ \mid - \mid * \mid /)$
 $\langle \text{letra} \rangle ::= (a \mid \dots \mid z \mid A \mid \dots \mid Z)$
 $\}$

b) Definición en EBNF con orden de prioridad:

- $G = \{N, T, S, P\}$

$N = \{ \langle \text{expr_con_p} \rangle, \langle \text{expr_sin_p} \rangle, \langle \text{expr_general} \rangle, \langle \text{variable} \rangle, \langle \text{operador_con_p} \rangle, \langle \text{número} \rangle, \langle \text{dígito} \rangle, \langle \text{letra} \rangle \}$
 $T = \{ a, \dots, z, A, \dots, Z, 0, \dots, 9, "+", "-", "*", "/" \}$
 $S = \langle \text{expr} \rangle$
 $P = \{$
 $\langle \text{expr_general} \rangle ::= \langle \text{expr_con_p} \rangle \{ \langle \text{operador_sin_p} \rangle \langle \text{expr_con_p} \rangle \}^+$
 $\langle \text{expr_con_p} \rangle ::= (\langle \text{variable} \rangle \mid \langle \text{número} \rangle) \{ \langle \text{operador_con_p} \rangle (\langle \text{id} \rangle \mid \langle \text{número} \rangle) \}^+$
 $\langle \text{número} \rangle ::= [(+|-)] \{ \langle \text{dígito} \rangle \}^+$
 $\langle \text{variable} \rangle ::= \langle \text{letra} \rangle \{ (\langle \text{letra} \rangle \mid \langle \text{dígito} \rangle) \}^*$
 $\langle \text{dígito} \rangle ::= (0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9)$
 $\langle \text{operador_con_p} \rangle ::= (* \mid /)$
 $\langle \text{operador_sin_p} \rangle ::= (+ \mid -)$
 $\langle \text{letra} \rangle ::= (a \mid \dots \mid z \mid A \mid \dots \mid Z)$
 $\}$

- c) Pensé que una expresión se podía dividir en expresiones con prioridad que usen operadores con prioridad, y expresiones sin prioridad que usen operadores sin prioridad, a partir de esa idea llegué a la conclusión de que debería tener una expresión general donde se puedan realizar 3 tipos de acciones:
- Trabajar únicamente con expresiones sin prioridad, las veces que se quiera.
 - Trabajar únicamente con expresiones con prioridad, las veces que se quiera.
 - Trabajar con expresiones con prioridad y expresiones sin prioridad las veces que se quiera, pero haciendo primero las que todas las que tengan prioridad.

11. Análisis de la siguiente Gramática:

$N = \{ \langle \text{sentencia_for} \rangle, \langle \text{bloque} \rangle, \langle \text{variable} \rangle, \langle \text{letra} \rangle, \langle \text{cadena} \rangle, \langle \text{digito} \rangle, \langle \text{otro} \rangle, \langle \text{operacion} \rangle, \langle \text{llamada_a_funcion} \rangle, \langle \text{numero} \rangle, \langle \text{sentencia} \rangle \}$
 $P = \{ \langle \text{sentencia_for} \rangle ::= \text{for (i= IN 1..10) loop } \langle \text{bloque} \rangle \text{ end loop};$
 $\langle \text{variable} \rangle ::= \langle \text{letra} \rangle \mid \langle \text{cadena} \rangle$
 $\langle \text{cadena} \rangle ::= \{ (\langle \text{letra} \rangle \mid \langle \text{digito} \rangle \mid \langle \text{otro} \rangle) \}^+$
 $\langle \text{letra} \rangle ::= (a \mid \dots \mid z \mid A \mid \dots \mid Z)$
 $\langle \text{digito} \rangle ::= (1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid 0)$
 $\langle \text{bloque} \rangle ::= \langle \text{sentencia} \rangle \mid \langle \text{sentencia} \rangle \langle \text{bloque} \rangle \mid \langle \text{bloque} \rangle \langle \text{sentencia} \rangle ;$
 $\langle \text{sentencia} \rangle ::= \langle \text{sentencia_asignacion} \rangle \mid \langle \text{llamada_a_funcion} \rangle \mid \langle \text{sentencia_if} \rangle \mid \langle \text{sentencia_for} \rangle \mid \langle \text{sentencia_while} \rangle \mid \langle \text{sentencia_switch} \rangle \}$

- Errores:
 - La gramática debe de ser una 4-tupla, faltan poner en este caso los símbolos terminales y el símbolo distinguido.

- En N, faltan agregar: <sentencia_asignacion>, <sentencia_while>, <sentencia_switch> y <sentencia_if>.
- En P, faltan definir: <sentencia_for>, <otro>, <operacion>, <llamada_a_funcion> y <numero>.
- La sentencia "<bloque> ::= <sentencia> | <sentencia><bloque> | <bloque><sentencia>" es ambigua por la asociatividad.

12. EBNF Tag DIV:

- $G = \{N, T, S, P\}$
 $N = \{<div>, <style>, <class>, <cadena>, <carácter>\}$
 $T = \{ "a", "...", "z", "A", "...", "Z", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "+", "-", "/", "*", "<", ">", ",", ":", "{", "}", "[", "]", "_", " ", "<div>", "</div>", "style=", "class=" \}$
 $S = <div>$
 $P = \{$
 $<div> ::= "<div>" \{ (<style> | <class>) \}^* \{ (<cadena> | <div> | <tag>) \}^*$
 $"</div>"$
 $<style> ::= "style=" [<cadena>]$
 $<class> ::= "class=" [<cadena>]$
 $<cadena> ::= \{ <carácter> \}^+$
 $<carácter> ::= ("a" | "..." | "z" | "A" | "..." | "Z" | "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | " " | "<" | ">" | "," | ":" | "{" | "}" | "[" | "]" | "-" | "_" | "+" | "*" | "/")$
 $\}$

Nota: Se asume que las demás <tag> de HTML ya existían.

13. EBNF de números primos

- La gramática a usar para este ejercicio sería la de los números enteros, lo que habría que agregar sería la semántica que controle cuándo el número es primo y cuándo no.

14. EBNF definición de funciones en Python

- $G = \{N, T, S, P\}$
 $N = \{<función>, <parámetros>, <nombre>, <dígito>, <letra>\}$
 $T = \{ "a", "...", "z", "A", "...", "Z", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "(", ")", ",", ":", "def" \}$
 $S = <función>$
 $P = \{$
 $<función> ::= "def" <nombre> <parámetros> ":" \{ <sentencia> \}^* [$
 $<return>]$
 $<parámetros> ::= "(" \{ <nombre> \{ "<nombre>" \}^* \}^* ")"$

```
"<nombre> ::= <letra>{ ( <dígito> | <letra> ) }*  
<dígito> ::= ( "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" )  
<letra> ::= ( "a" | "..." | "z" | "A" | "..." | "Z" )  
}
```

Nota: Asumo que <sentencia> y <return> ya fueron definidos previamente