



Sintaxis y BNF

- Motivación
 - Lenguajes de programación usan LR para:
 - Identificadores
 - Constantes
 - Palabras reservadas
 - Lenguajes independientes del contexto para:
 - Expresiones
 - Sentencias
 - Estructuras de control
 - BNF (Backus Naur Form) se usa para describir un Lenguaje de programación en esos dos niveles
 - Léxico: Categorías léxicas o Tokens (LR)
 - Sintáctico: Categorías sintácticas (LIC)



Con notación similar a gramática

- Por ahora cambiaremos:
 - Solo escribimos las producciones
 - El axioma es el no terminal a izquierda de la primer producción
 - Los no terminales los representaremos con nombres significativos en lugar de con un único símbolo
- Los metasímbolos siguen siendo \rightarrow y **|**



Identificadores

- Suponga que los identificadores en un determinado lenguaje se forman con letras mayúsculas y guiones bajos. Los guiones bajos solo se permiten de a uno entre dos letras.

- Identificador \rightarrow Letra |
Identificador Letra |
Identificador GuionBajo Letra

GuionBajo \rightarrow _

Letra \rightarrow A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R
S|T|U|V|W|X|Y|Z



Recursivas a Izquierda

- Notar que en la gramática anterior el identificador se va formando de derecha a izquierda
- Se dice que la gramática es **recursiva a izquierda** porque el no terminal de la izquierda aparece a la izquierda de lo producido (la parte derecha)
- Se puede construir una gramática equivalente recursiva a derecha



Recursiva a derecha

- Identificador \rightarrow Letra |
Letra Resto

Resto \rightarrow Letra Resto |
GuionBajo Letra Resto |
 ε

GuionBajo \rightarrow _

Letra \rightarrow A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R
S|T|U|V|W|X|Y|Z



Expresiones y Sintaxis

- Una GIC es apropiada para balancear símbolos, por ejemplo, paréntesis
- Nos permite establecer **precedencia de operadores**, los más cercanos al axioma serán los de menor precedencia.
- Reducción es el proceso de evaluación y es inverso paso a paso al de la derivación
- Si intento “nivelar” precedencias puedo terminar con una gramática ambigua
- Nos permite establecer la **asociatividad de los operadores** (izquierda a derecha o al revés)



Gramática para expresiones

Regla Nro	Producción
1	$\text{Expresión} \rightarrow \text{Expresión} + \text{Término}$
2	$\text{Expresión} \rightarrow \text{Término}$
3	$\text{Término} \rightarrow \text{Término} * \text{Factor}$
4	$\text{Término} \rightarrow \text{Factor}$
5	$\text{Factor} \rightarrow \text{Número}$
6	$\text{Factor} \rightarrow (\text{Expresión})$
7	$\text{Número} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$



Ejemplo de derivación

- Para la expresión $2 + 4 * 5$

Paso	Regla	Derivación
0	Axioma	Expresión
1	1 ($E \rightarrow E + T$)	Expresión + Término
2	2 ($E \rightarrow T$)	Término + Término
3	4 ($T \rightarrow F$)	Factor + Término
4	5 ($F \rightarrow N$)	Número + Término
5	7 ($N \rightarrow 2$)	2 + Término
6	3 ($T \rightarrow T * F$)	2 + Término * Factor
7	4 ($T \rightarrow F$)	2 + Factor * Factor
8	5 ($F \rightarrow N$)	2 + Número * Factor
9	7 ($N \rightarrow 4$)	2 + 4 * Factor
10	5 ($F \rightarrow N$)	2 + 4 * Número
11	7 ($N \rightarrow 5$)	2 + 4 * 5

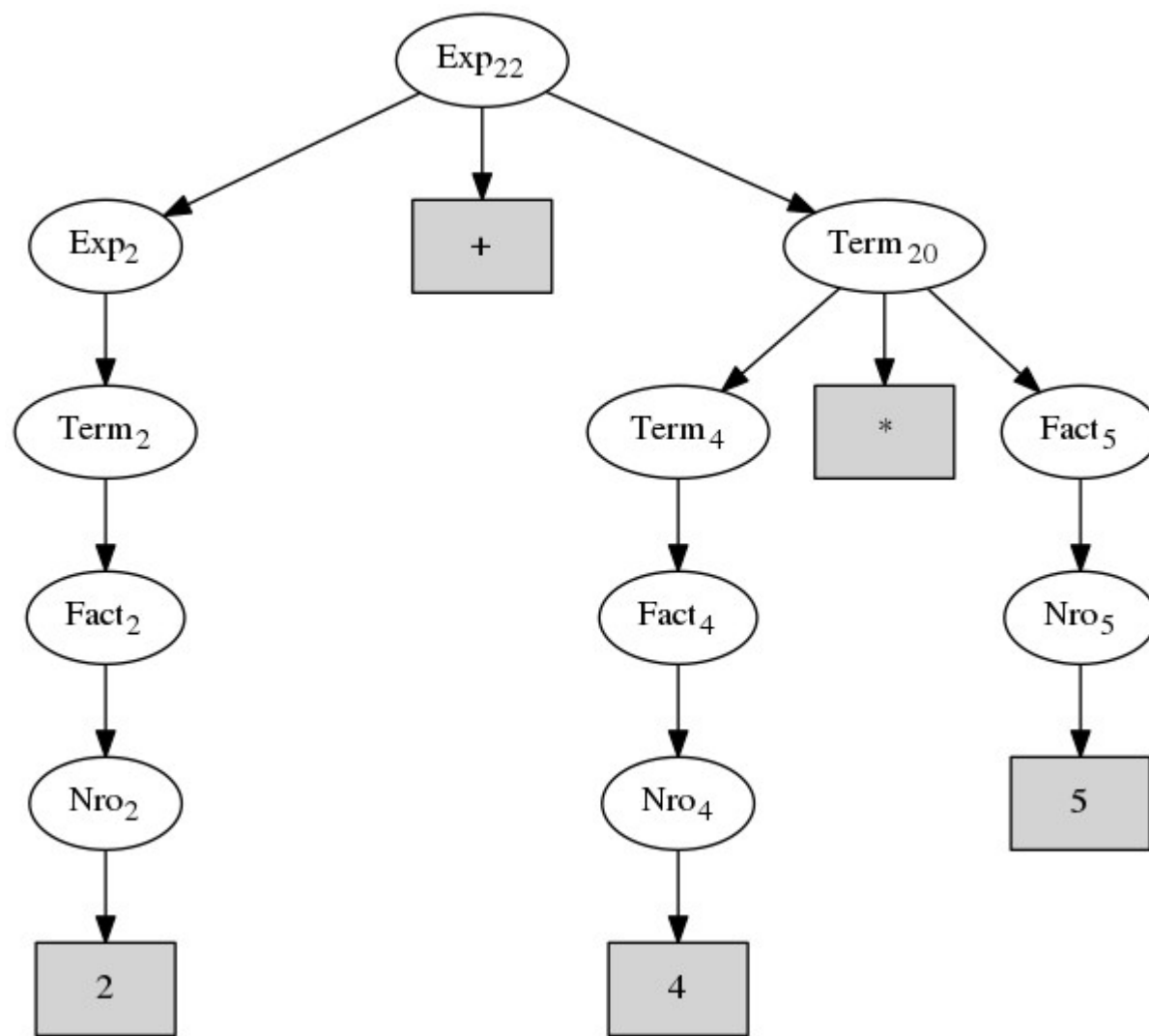


Reducción

Corresponde a Paso	Derivación a Reducir	Producción	Operación
11	$2 + 4 * 5$	7 ($N \rightarrow 5$)	
10	$2 + 4 * \text{Número}5$	5 ($F \rightarrow N$)	
9	$2 + 4 * \text{Factor}5$	7 ($N \rightarrow 4$)	
8	$2 + \text{Número}4 * \text{Factor}5$	5 ($F \rightarrow N$)	
7	$2 + \text{Factor}4 * \text{Factor}5$	4 ($T \rightarrow F$)	
6	$2 + \text{Término}4 * \text{Factor}5$	3 ($T \rightarrow T * F$)	
5	$2 + \text{Término}20$	7 ($N \rightarrow 2$)	$4 * 5 = 20$
4	$\text{Número}2 + \text{Término}20$	5 ($F \rightarrow N$)	
3	$\text{Factor}2 + \text{Término}20$	4 ($T \rightarrow F$)	
2	$\text{Término}2 + \text{Término}20$	2 ($E \rightarrow T$)	
1	$\text{Expresión}2 + \text{Término}20$	1 ($E \rightarrow E + T$)	$2 + 20 = 22$
0	Expresión22	Axioma	

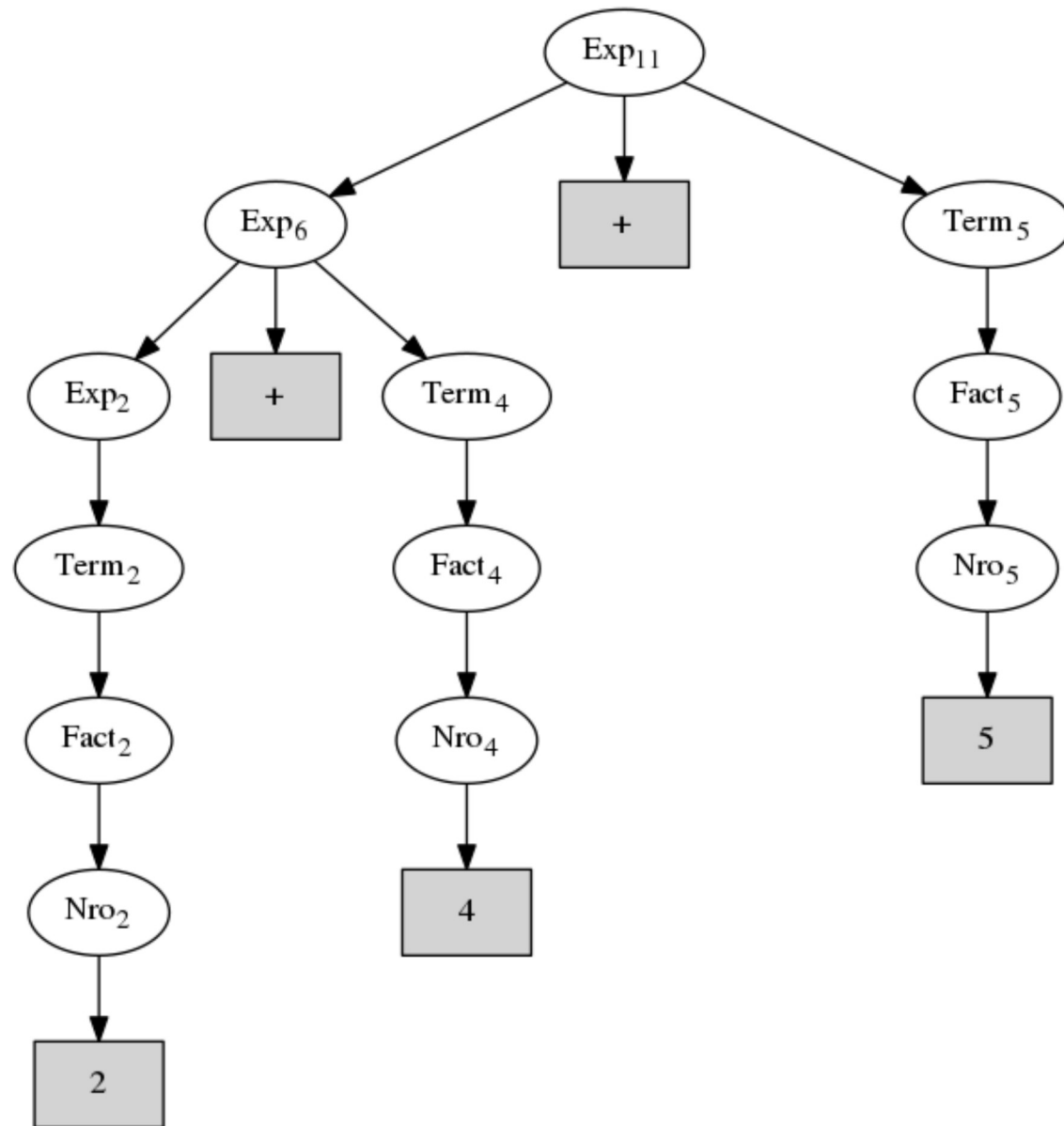


$2+4*5$ Visto como árbol





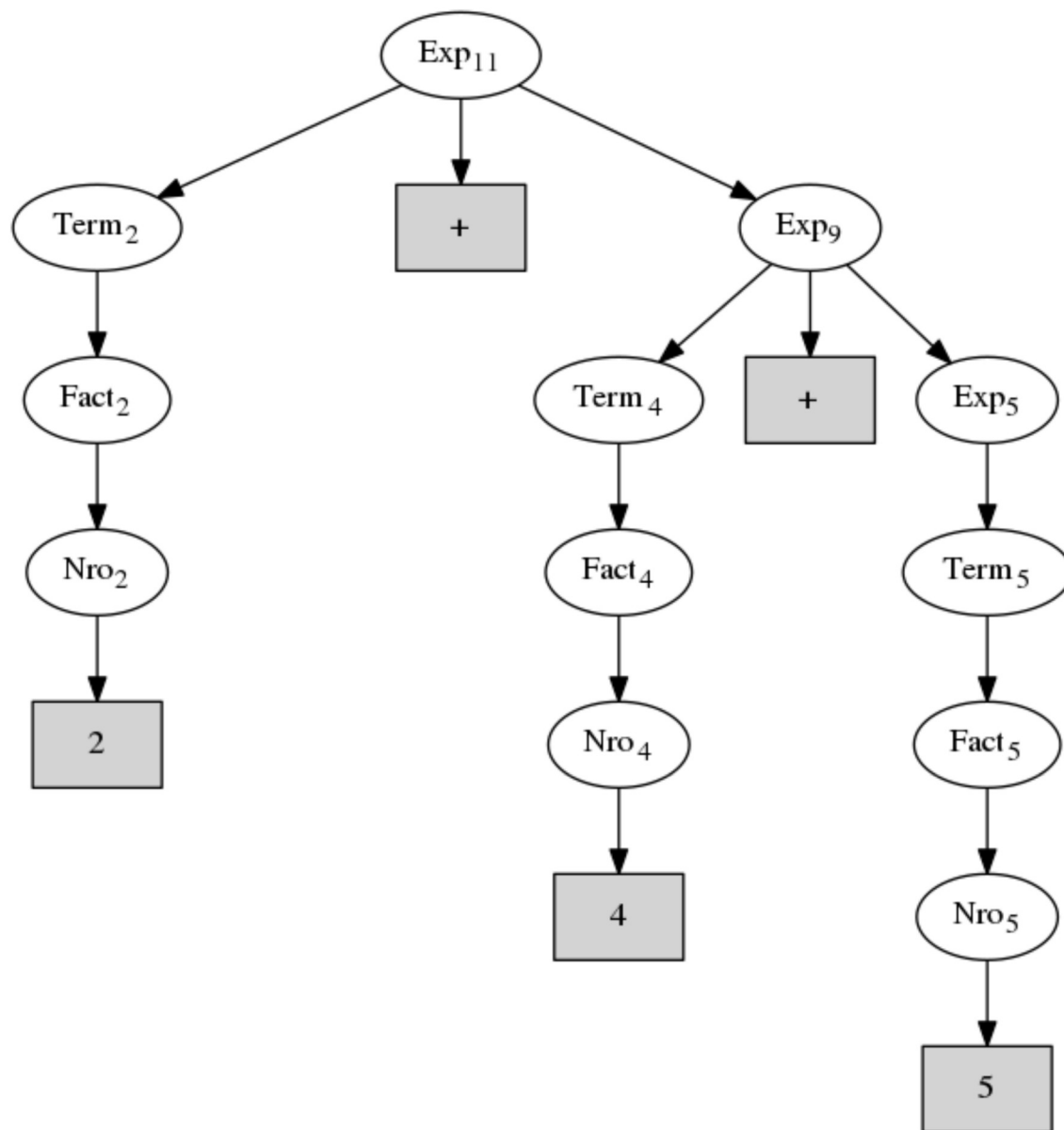
2+4+5 Asociación a Izquierda





Si asociara a Derecha

$\text{Exp} \rightarrow \text{Term} + \text{Exp}$



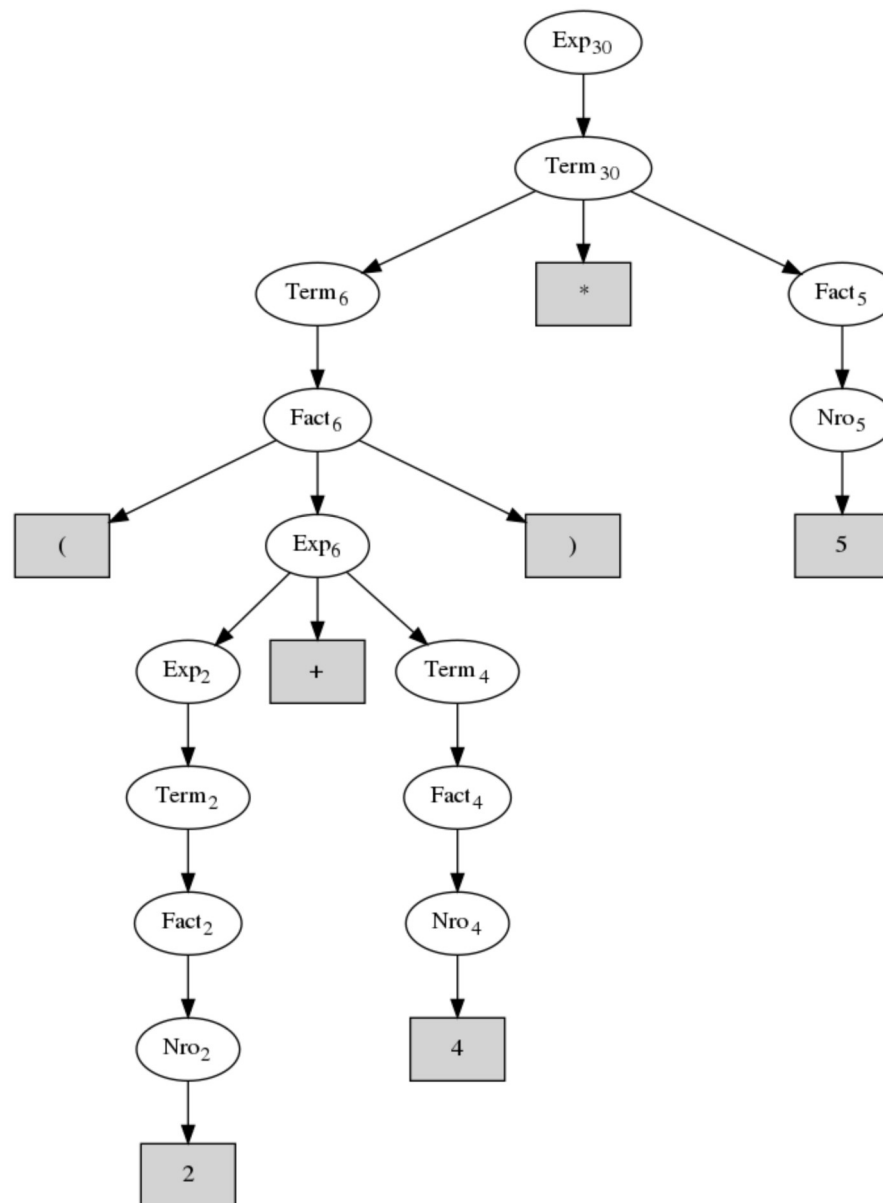


Intento Forzar la suma primero

P.	Regla	Derivación	Comentario
0	Axioma	Expresión	
1	2 : $E \rightarrow T$	Término	
2	3 : $T \rightarrow T * F$	Término * Factor	Introduzco * primero para que se ejecute último en la reducción
3	4 : $T \rightarrow F$	Factor * Factor	No puedo volver a Expresión
4	6: $F \rightarrow (E)$	(Expresión) * Factor	Salvo bajando hasta incorporar paréntesis
5	1: $E \rightarrow E + T$	(Expresión + Término) * Factor	
6	2 : $E \rightarrow T$	(Término + Término) * Factor	
... Trabajando los pasos que faltan ...			
13	5: $F \rightarrow N$	(2 + 4) * Número	
14	7: $N \rightarrow 5$	(2 + 4) * 5	



Árbol con suma primero





Gramática sin precedencia (ambigua)

Regla Nro	Regla
1	$\text{Exp} \rightarrow \text{Exp} + \text{Exp}$
2	$\text{Exp} \rightarrow \text{Exp} * \text{Exp}$
3	$\text{Exp} \rightarrow (\text{Exp})$
4	$\text{Exp} \rightarrow \text{Nro}$
5	$\text{Nro} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$



Derivación

Paso	Producción	Derivación Obtenida	Producción	Derivación Alternativa
0	Axioma	Exp	Axioma	Exp
1	1 ($E \rightarrow E + E$)	Exp + Exp	2 ($E \rightarrow E * E$)	Exp * Exp
2	4 ($E \rightarrow Nro$)	Nro + Exp	4 ($E \rightarrow Nro$)	Exp * Nro
3	5 ($Nro \rightarrow 2$)	2 + Exp	5 ($Nro \rightarrow 5$)	Exp * 5
4	2 ($E \rightarrow E * E$)	2 + Exp * Exp	1 ($E \rightarrow E + E$)	Exp + Exp * 5
5	4 ($E \rightarrow Nro$)	2 + Nro * Exp	4 ($E \rightarrow Nro$)	Exp + Nro * 5
6	5 ($Nro \rightarrow 3$)	2 + 4 * Exp	5 ($Nro \rightarrow 3$)	Exp + 4 * 5
7	4 ($E \rightarrow Nro$)	2 + 4 * Nro	4 ($E \rightarrow Nro$)	Nro + 4 * 5
8	5 ($Nro \rightarrow 5$)	2 + 4 * 5	5 ($Nro \rightarrow 2$)	2 + 4 * 5

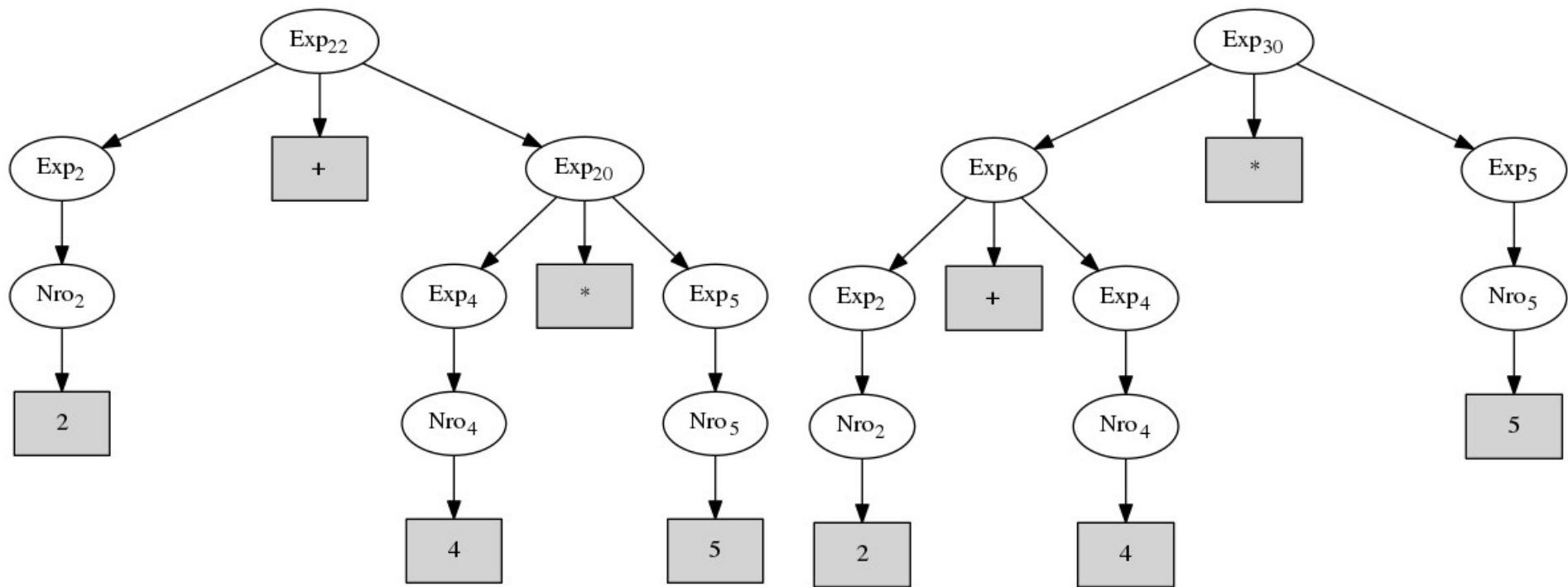


Reducción

Paso	Derivación a Reducir	Prod	Operación	Derivación a Reducir	Prod	Operación
8	$2 + 4 * 5$	5		$2 + 4 * 5$	5	
7	$2 + 4 * \mathbf{Nro5}$	4		$\mathbf{Nro2} + 4 * 5$	4	
6	$2 + 4 * \text{Exp5}$	5		$\mathbf{Exp2} + 4 * 5$	5	
5	$2 + \mathbf{Nro4} * \text{Exp5}$	4		$\text{Exp2} + \mathbf{Nro4} * 5$	4	
4	$2 + \mathbf{Exp4} * \mathbf{Exp5}$	2		$\mathbf{Exp2} + \mathbf{Exp4} * 5$	1	
3	$2 + \text{Exp20}$	5	$4 * 5 = 20$	$\text{Exp6} * 5$	5	$2 + 4 = 6$
2	$\mathbf{Nro2} + \text{Exp20}$	4		$\text{Exp6} * \mathbf{Nro5}$	4	
1	$\mathbf{Exp2} + \mathbf{Exp20}$	1	$2 + 20 = 22$	$\mathbf{Exp6} * \mathbf{Exp5}$	2	$6 * 5 = 30$
0	Exp22	Axioma		Exp30	Axioma	



Árboles de las derivaciones





BNF

- Debida a
 - John Backus que adopta las técnicas generativas de Chomsky en su informe sobre el lenguaje ALGOL en 1959
 - Peter Naur que en su informe sobre ALGOL 60 simplifica la notación original de Backus
- Originalmente BNF significaba **Backus Normal Form** pero a instancias de Donald Knuth se cambio a **Backus Naur Form**
- Reporte de Algol 60:
<https://www.masswerk.at/algol60/report.htm#1>



BNF

- Descripción
 - Usar nombres representativos (Naur)
 - Los no terminales se escriben entre $<$ y $>$ (Backus)
 - El metasímbolo de producción es $::=$ (Naur)
 - Se suele leer $::=$ como “es un/a”
 - Se separan opciones (Backus) con el metasímbolo $|$ (Naur)

- Ejemplo (tomado de Algol)

$\langle \text{identificador} \rangle ::= \langle \text{letra} \rangle |$

$\langle \text{identificador} \rangle$

$\langle \text{letra} \rangle |$

$\langle \text{identificador} \rangle \langle \text{dígito} \rangle$

$\langle \text{letra} \rangle ::= A | B | C | \dots | Z | a | b | c | \dots | z$

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



EBNF

- Muchas variantes
- En 1974 Niklaus Wirth junto a Kathleen Jensen escriben “Pascal User Manual Report” y hacen las siguientes extensiones
 - Las palabras reservadas del lenguaje las distingue subrayándolas
 - Usa { } para indicar la repetición de cero o más veces de lo esté entre las llaves (clausura de Kleene)
- Además ofrece una notación alternativa, el diagrama de sintaxis



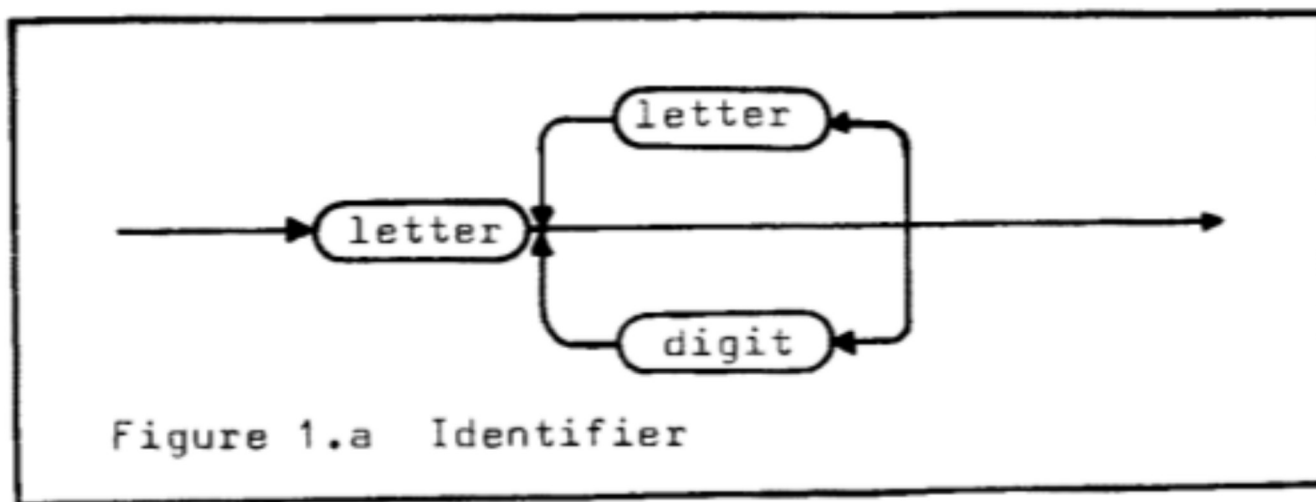
Identificador

- Vimos en BNF

$\langle \text{identificador} \rangle ::= \langle \text{letra} \rangle \mid$
 $\langle \text{identificador} \rangle \langle \text{letra} \rangle \mid$
 $\langle \text{identificador} \rangle \langle \text{dígito} \rangle$

- Ahora podemos escribir lo mismo como

$\langle \text{identificador} \rangle ::= \langle \text{letra} \rangle \{ \langle \text{letra o dígito} \rangle \}$
 $\langle \text{letra o dígito} \rangle ::= \langle \text{letra} \rangle \mid \langle \text{dígito} \rangle$





BNF lenguaje C

- Cambia la notación, según el estándar
 - Los no terminales van en cursiva (italic)
 - El metasímbolo de producción es :
 - Para indicar opciones pone una por renglón (en lugar de usar |)
 - Puede poner varias opciones en un renglón si se precede del metasímbolo *one of*, usado generalmente para conjuntos de caracteres
 - Los terminales van en negrita (en K&R van en tipografía monoespaciada)
 - Si un término es opcional se lo indica con el subíndice *opt*
 - No hay equivalente al cero o más repeticiones de la BNF usada en Pascal



Ejemplo

identifier :

identifier-nondigit

identifier identifier-nondigit

identifier digit

identifier-nondigit: one of

_	a	b	c	d	e	f	g	h	i	j	k	l	m
	n	o	p	q	r	s	t	u	v	w	x	y	z
	A	B	C	D	E	F	G	H	I	J	K	L	M
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

digit: one of

0	1	2	3	4	5	6	7	8	9
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

Según el estándar las implementaciones deben distinguir los identificadores al menos hasta el carácter 31 (external)



Otras BNF

- Hay muchas variantes, algunas de ellas son
 - = como metasímbolo de producción
 - [] para indicar cero o una vez (opción)
 - Indicar los strings terminales con comillas, simples o dobles
 - (* *) como comentarios
 - Distintas modificaciones caligráficas para terminales y/o no terminales (negrita, cursiva, etc)



BNF otros usos

- Por ejemplo para describir el formato de datos
 - Caso concreto: JSON (JavaScript Object Notation): <https://www.json.org/json-en.html>

object

{ }

{ members }

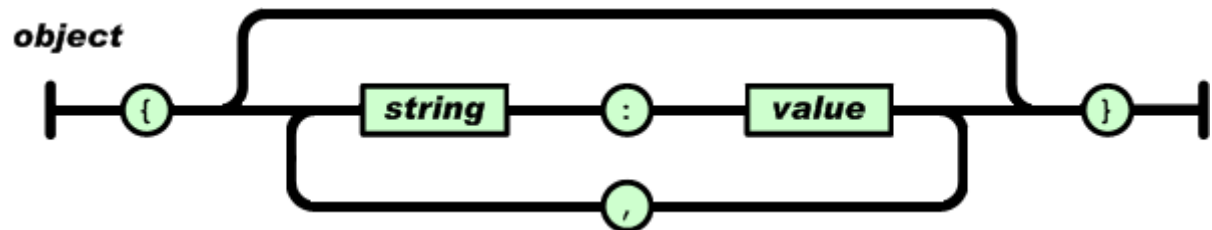
members

pair

pair , members

pair

string : value





BNF otros usos

- Otros que usan Alguna variante de EBNF
 - W3C (World Wide Web Consortium): En la definición de XML usa una EBNF que describe en el mismo documento
<https://www.w3.org/TR/xml11/#sec-notation>
 - En 1996 se publicó la ISO/IEC 14977 que define una versión actualizada de EBNF (Disponible en el dropbox, carpeta documentación): Utiliza las llaves { } como la EBNF que vimos y [] para elementos opcionales, y otros varios agregados.
[http://standards.iso.org/ittf/PubliclyAvailableStandards/s026153_ISO_IEC_14977_1996\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/s026153_ISO_IEC_14977_1996(E).zip)



BNF otros usos, opiniones

- Otros que usan Alguna variante de EBNF
 - IETF (Internet Engineering Task Force) define la “*Augmented BNF for Syntax Specifications: ABNF*” en su RFC5234: <http://tools.ietf.org/html/rfc5234>
 - La usa en algunas de sus RFC, por ejemplo la RFC5511 “*Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications*”: <http://tools.ietf.org/html/rfc5511>
 - Críticas al estándar ISO/IEC: <https://dwheeler.com/essays/dont-use-iso-14977-ebnf.html>



Licencia

*Esta obra, © de Eduardo Zúñiga, está protegida legalmente bajo una licencia Creative Commons, **Atribución-CompartirDerivadasIgual 4.0 Internacional**.
<http://creativecommons.org/licenses/by-sa/4.0/>*

***Se permite: copiar, distribuir y comunicar públicamente la obra; hacer obras derivadas y hacer un uso comercial de la misma.
Siempre que se cite al autor y se herede la licencia.***

