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 → 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 → Letra |
  Identificador Letra |
  Identificador GuionBajo Letra

  GuionBajo → _
  Letra → 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 en la parte derecha de la producción
- Se puede construir una gramática equivalente recursiva a derecha

Recursiva a derecha

- Identificador → Letra |
  Letra Resto
  Resto → Letra Resto |
  GuionBajo Letra Resto |
  ε
  GuionBajo → _
  Letra → 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	Expresión → Término
2	Expresión → Expresión + Término
3	Término → Factor
4	Término → Término * Factor
5	Factor → Número
6	Factor → (Expresión)
7	Número → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Ejemplo de derivación

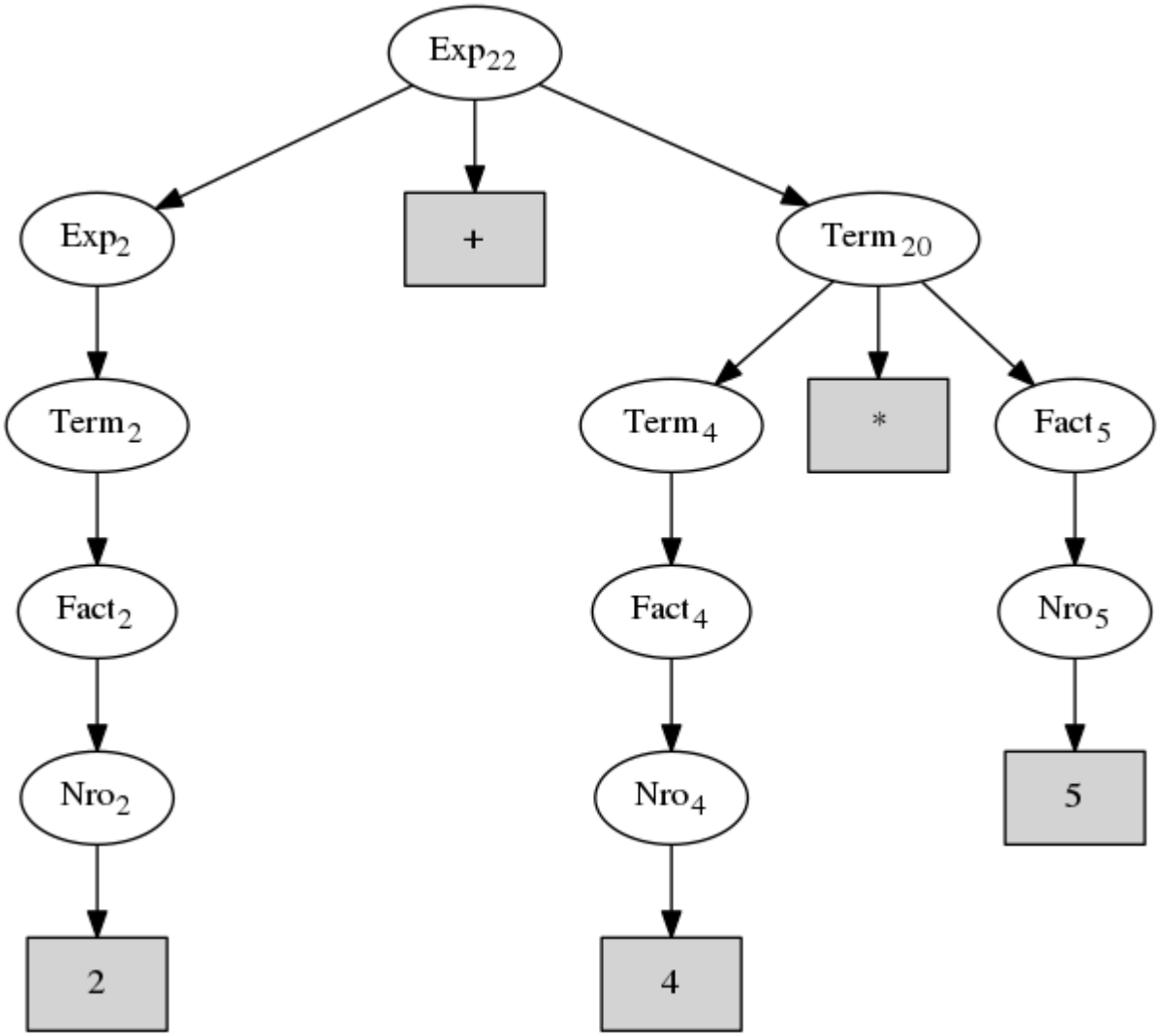
- Para la expresión 2 + 4 * 5

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

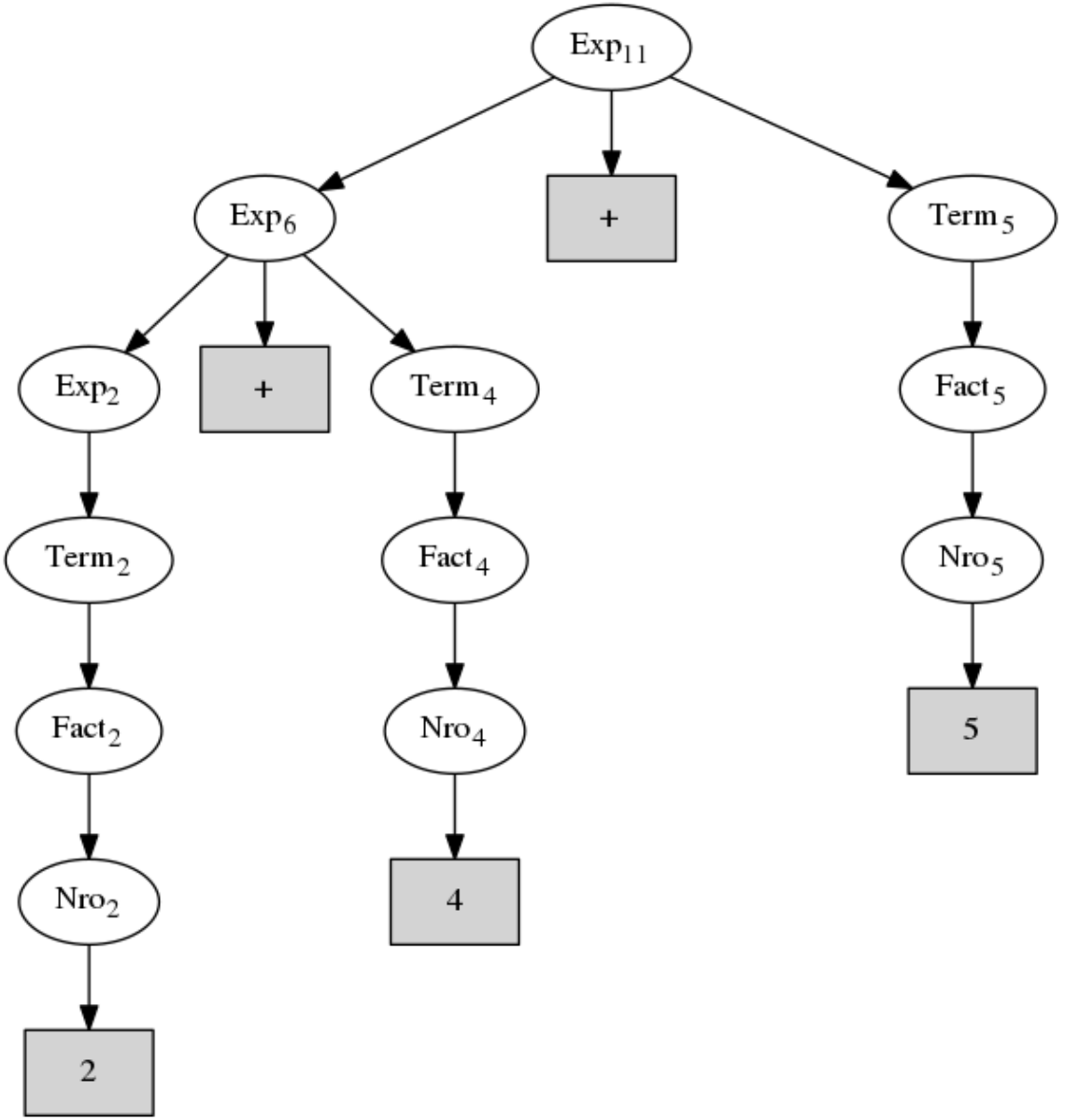
Reducción

Corresponde a Paso	Derivación a Reducir	Producción	Operación
11	2 + 4 * **5**	7 (N → 5)	
10	2 + 4 * **Número5**	5 (F → N)	
9	2 + **4** * Factor5	7 (N → 3)	
8	2 + **Número4** * Factor5	5 (F → N)	
7	2 + **Factor4** * Factor5	3 (T → F)	
6	2 + **Término4 * Factor5**	4 (T → T * F)	
5	**2** + Término20	7 (N → 2)	4 * 5 = 20
4	**Número2** + Término20	5 (F → N)	
3	**Factor2** + Término20	3 (T → F)	
2	**Término2** + Término20	1 (E → T)	
1	**Expresión2** + Término20	2 (E → E + T)	2 + 20 = 22
0	Expresión22	Axioma	

2+4*5 Visto como árbol

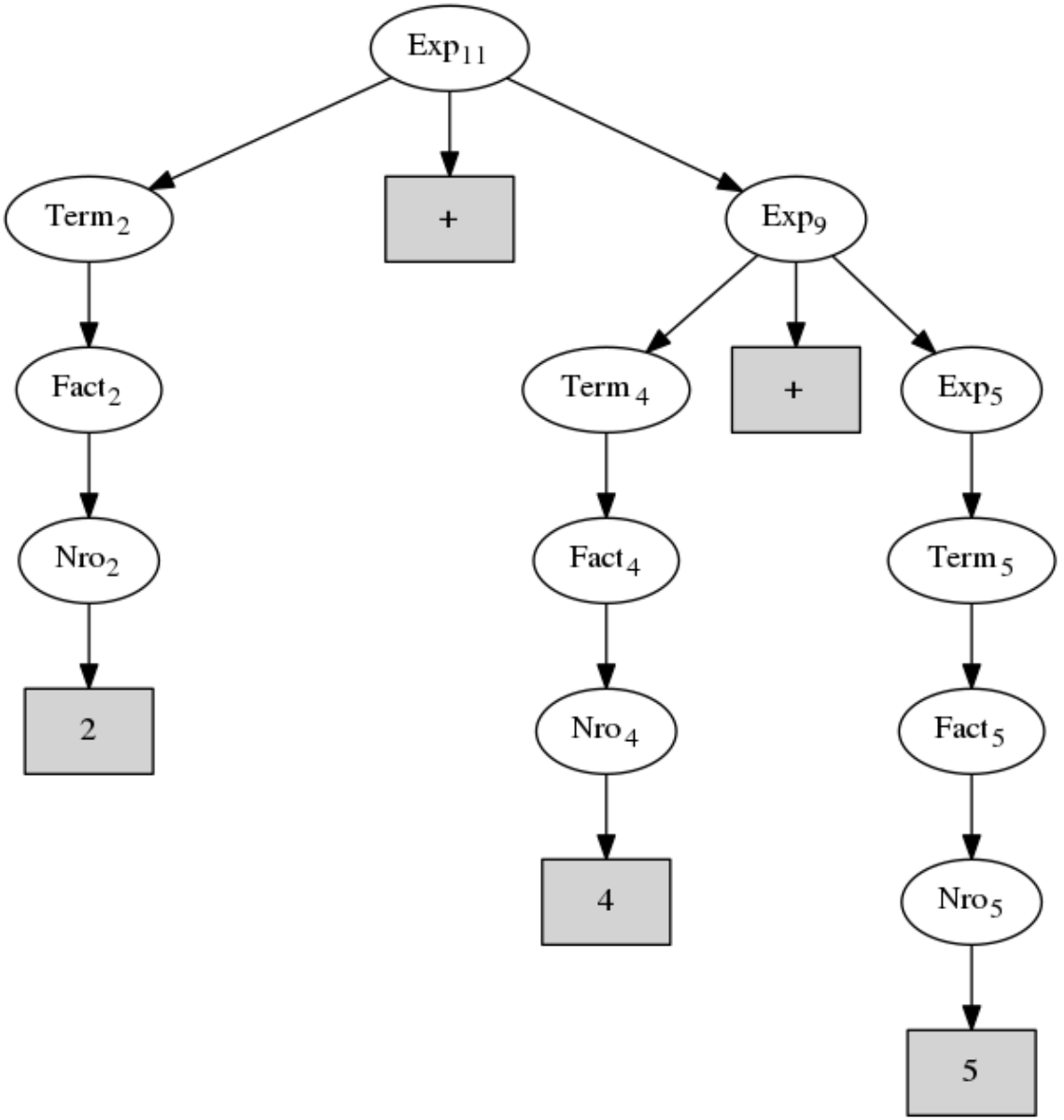


2+4+5 Asociación a Izquierda



Si asociara a Derecha

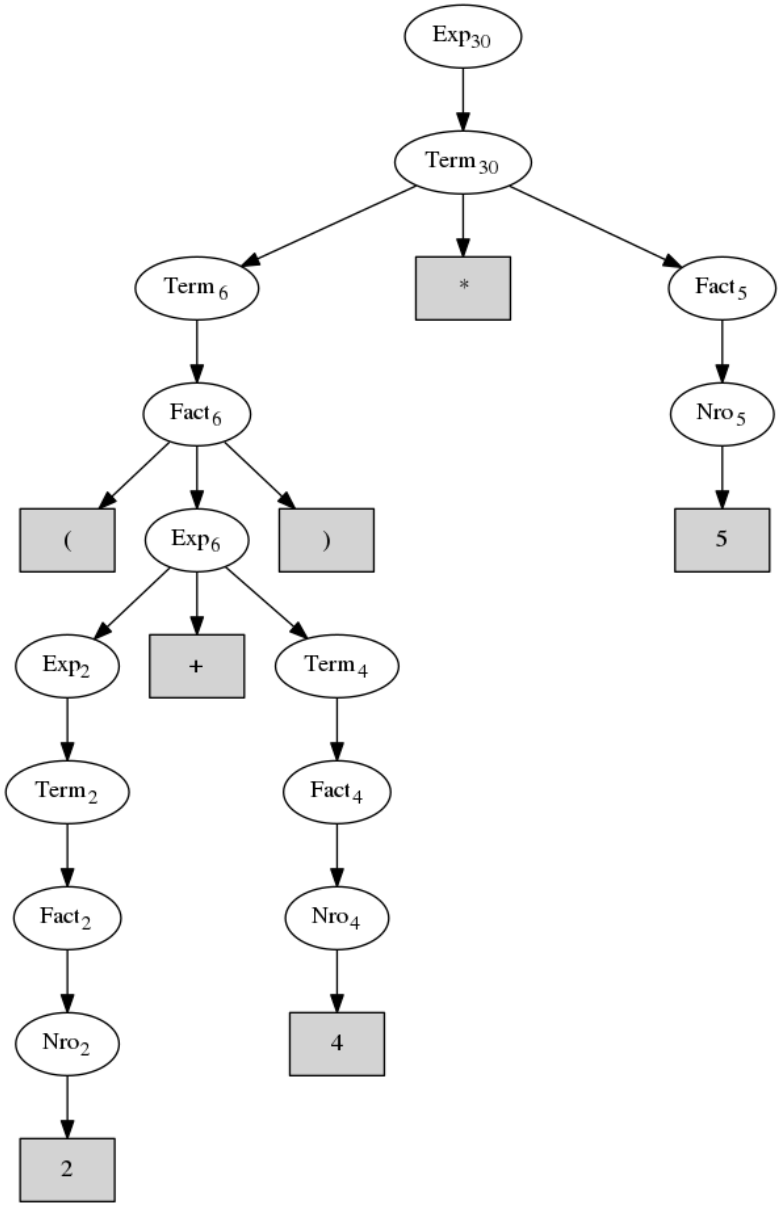
Exp → Term + Exp



Intento Forzar la suma primero

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

Árbol con suma primero



Gramática sin precedencia (ambigua)

Regla Nro	Regla
1	Exp → Exp + Exp
2	Exp → Exp * Exp
3	Exp → (Expr)
4	Expr → Nro
5	Nro → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

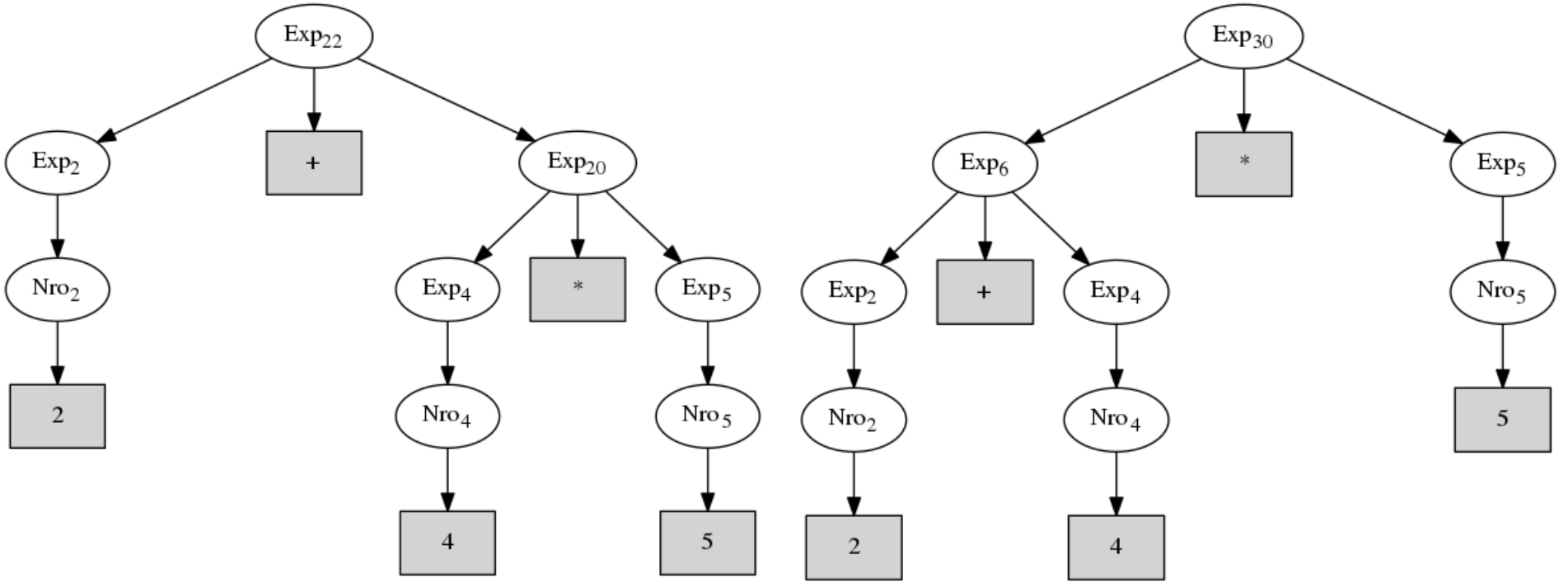
Derivación

Paso	Producción	Derivación Obtenida	Producción	Derivación Alternativa
0	Axioma	**Exp**	Axioma	**Exp**
1	1 (E → E + E)	**Exp** + Exp	2 (E → E * E)	Exp * **Exp**
2	4 (E → Nro)	**Nro** + Exp	4 (E → Nro)	Exp * **Nro**
3	5 (Nro → 2)	2 + **Exp**	5 (Nro → 5)	**Exp** * 5
4	2 (E → E * E)	2 + **Exp** * Exp	1 (E → E + E)	Exp + **Exp** * 5
5	4 (E → Nro)	2 + **Nro** * Exp	4 (E → Nro)	Exp + **Nro** * 5
6	5 (Nro → 3)	2 + 4 * **Exp**	5 (Nro → 3)	**Exp** + 4 * 5
7	4 (E → Nro)	2 + 4 * **Nro**	4 (E → Nro)	**Nro** + 4 * 5
8	5 (Nro → 5)	2 + 4 * 5	5 (Nro → 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 * **Nro5**	4		**Nro2** + 4 * 5	4	
6	2 + **4** * Exp5	5		**Exp2** + 4 * 5	5	
5	2 + **Nro4** * Exp5	4		Exp2 + **Nro4** * 5	4	
4	2 + **Exp4** * **Exp5**	2		**Exp2 + Exp4** * 5	1	
3	**2** + Exp20	5	4 * 5 = 20	Exp6 * **5**	5	2 + 4 = 6
2	**Nro2** + Exp20	4		Exp6 * **Nro5**	4	
1	**Exp2 + Exp20**	1	2 + 20 = 22	**Exp6** * **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**

BNF

- Descripción
  - Los no terminales se escriben entre < y >
  - El metasímbolo de producción es ::=
    - Se suele leer ::= como “es un/a”
  - Se mantiene el metasímbolo |
- Ejemplo (tomado de Algol)

<identificador> ::= <letra> |
<identificador> <letra> |
<identificador> <dígito>

<letra> ::= A | B | C | … | Z | a | b | c | …… | z

<dígito> ::= 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

<identificador> ::= <letra> |
<identificador> <letra> |
<identificador> <dígito>

- Ahora podemos escribir lo mismo como

<identificador> ::= <letra> {<letra o dígito>}
<letra o dígito> ::= <letra> | <dígito>

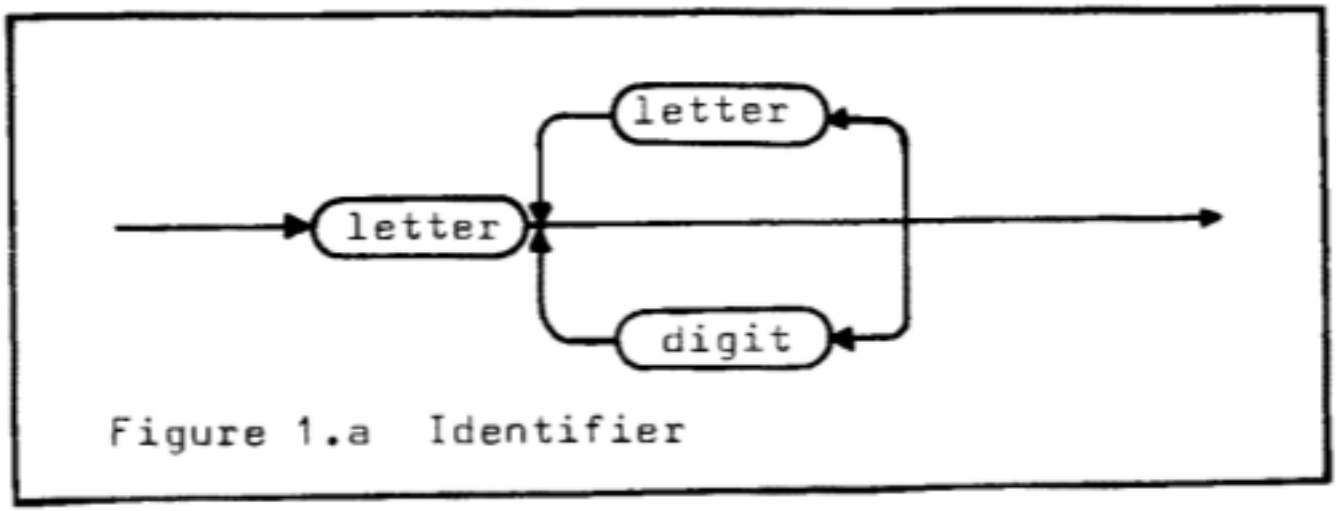


Figure 1.a Identifier

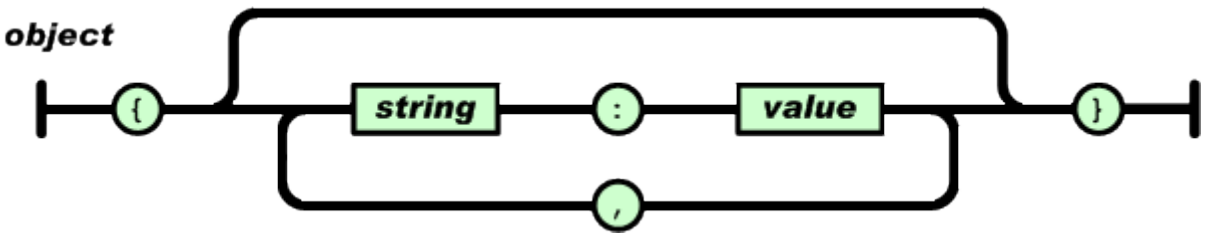
Otras BNF

- 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): http://json.org/

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 http://www.w3.org/TR/2006/REC-xml11-20060816/#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

BNF otros usos

- 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

Licencia