

# Backus Naur Form

Autor: Juan David Arce Martinez

*Facultad de ingenierías, Universidad Tecnológica de Pereira, Pereira, Colombia*

Correo-e: juandavid.arce@utp.edu.co

**Resumen**— Backus Naur Form es un metalenguaje que permite expresar las construcciones admitidas por un lenguaje formal, es decir, permite especificar la sintaxis de un lenguaje formal. Al ser un estándar de facto, no existe una versión normalizada. Por esa razón, en la documentación de distintos lenguajes pueden encontrarse distintas adaptaciones de BNF, aunque suelen ser bastante similares. Generalmente, aparece en la documentación de cualquier lenguaje formal (shell script, Java, C#, SQL, etc.), para especificar su sintaxis.

**Palabras clave**—Backus, Naur, Metalenguaje, Programa.

**Abstract**— Backus Naur Form is a metalanguage that allows to obtain the constructions admitted by a formal language, that is, it allows to specify the syntax of a formal language. Being a de facto standard, there is no standardized version.

For this reason, in the documentation of the different languages, adaptations of BNF can be differentiated, although they are often quite similar. Generally, it appears in the documentation of any formal language (shell script, Java, C #, SQL, etc.), to specify its syntax.

**Key Word** — Backus, Naur, Metalanguage, Program

## I. INTRODUCCIÓN

Backus-Naur Form: reglas generativas y descripción de un meta-lenguaje Backus-Naur es un metalenguaje de programación que se ha constituido a lo largo del tiempo en uno de los sistemas de notación técnica más frecuentemente usados en computación. Tal como se asegura en una de sus páginas web, la fuente y creación del mismo, surgió a partir de las teorías lingüísticas de Noam Chomsky. De acuerdo con lo que planteó el propio Chomsky cuando afirmó: «Enseñando lingüística a estudiantes de teoría de la información en el MIT, lingüística combinada y matemáticas, tomando lo que es esencialmente el formalismo de Thue como base para la descripción de la sintaxis del lenguaje natural. También introdujo una clara distinción entre reglas generativas (las de gramáticas libres de contexto) y reglas de transformación.» (1956), no directamente Backus, pero sí Naur tomó el BNF (por sus siglas en inglés) para el desarrollo de las gramáticas de los lenguajes de programación, pero también para representar en un lenguaje formalizado partes de las estructuras gramáticas de la lengua natural

## II. CONTENIDO

[1]. John Backus nació en Filadelfia el 3 de diciembre de 1924, y creció cerca de allí en Wilmington, Backus asistió a la prestigiosa escuela Hill Pottstown en Pensilvania. No era un

buen estudiante, y sus años en la escuela se caracterizaron por una serie de fracasos. No obstante a las bajas calificaciones y la poca asistencia, Backus se graduó de la Hill School en 1942 y entra en la Universidad de Virginia.

Su padre, quería que siguiera sus pasos, estudiando Química, Backus comenzó el estudio de la Química pero por un corto tiempo, en este tiempo disfrutó junto a su padre los aspectos teóricos de la ciencia, pero no le gustaba el trabajo de laboratorio. Al final de su segundo semestre, la asistencia a clases se redujo a una vez por semana, esta actitud condujo a las autoridades escolares a no permitirle a Backus continuar con los estudios. Ingresó en el Ejército en 1942. Backus fue destacado en Fort Stewart, Georgia, pero su rendimiento en una prueba de aptitud cambió el curso de su carrera militar. El Ejército decidió inscribirlo en un programa de preingeniería de la Universidad de Pittsburgh. Otra prueba de aptitud, esta vez sobre las capacidades médicas, lo llevó a Haverford College, donde fue a estudiar Medicina. Backus trabajó en un hospital de la ciudad atlántica, como parte del programa de estudios premédicos. Durante ese tiempo, se le diagnosticó un tumor cerebral y se le colocó una placa de platino en su cabeza. En marzo de 1945, entró en la Escuela de Medicina en Nueva York, pero se dio cuenta de que la Medicina no era para él y sólo duró nueve meses.

Backus dejó el Ejército en 1946 tras una operación adicional para reemplazar la placa en la cabeza, que nunca había ajustado correctamente. Sin saber qué hacer en su vida, alquila un pequeño apartamento en Nueva York. Le gustaba la música, y quería comprar un buen equipo de audio de alta fidelidad. El equipo que buscaba no existía por aquel entonces, por lo que matrícula en la escuela de técnicos de radio, para aprender a construir uno. Mientras estudiaba, Backus ayuda a un instructor a realizar cálculos matemáticos en estudios de ajustes de curvas para los amplificadores de audio. El trabajo era tedioso, pero descubre aptitud e interés en las matemáticas, El descubrimiento de la vocación hacia las Matemáticas, hace que Backus matricule en la Universidad de Columbia para estudiar con seriedad esta ciencia en el año 1946. En la primavera de 1949, Backus visitó el Centro de Computación de IBM en Madison Avenue, donde se encontraba la Selective Sequence Electronic Calculator, (SSEC), una de las primeras computadoras IBM. Backus le menciona a la guía que estaba buscando un trabajo. Ella le animó a hablar con el director del proyecto, y fue contratado para trabajar en la SSEC. Por otro lado La N de la notación BNF, usada en la descripción de la sintaxis de la mayoría de los lenguajes de programación, se usa en alusión a su apellido. Naur contribuyó en la creación del lenguaje de programación Algol 60 en donde se introdujo por primera vez la noción de recursión.[2] Empezó su carrera como un astrónomo, pero su encuentro con las computadoras lo hizo cambiar de carrera. A

Naur no le agradaba el término anglosajón Ciencias de la computación ("Computer Science" en inglés) y sugiere llamarlo Datalogía ("Datalogy" en inglés). Este término fue adoptado en Dinamarca y Suecia.

Trabajó en el Regnecentralen (empresa de computación danesa), en el Instituto Niels Bohr y en la Universidad Técnica de Dinamarca. De 1968 a 1998 trabajó como profesor en la Universidad de Copenhague. Es conocido por su crítica al uso de los métodos formales en programación. Así mismo, basado en su inclinación desde el empirismo, critica el uso que le dan los filósofos a la lógica para describir la ciencia. Critica igualmente a psicólogos que todavía se basan en teorías del conductismo y el constructivismo.

Se retiró en 1999 a la edad de 70 años. En los últimos años estuvo desarrollando una teoría del pensamiento humano que denominó Teoría Sinapsis-Estado de Vida Mental ("A Synapse-State Theory of Mental Life" en inglés). En su discurso de aceptación del Premio Turing, Peter Naur concluye así:

La informática nos presenta una forma de descripción muy útil para describir una gran variedad de fenómenos de este mundo, pero el pensamiento humano no es uno de ellos, siendo la razón que el pensamiento humano es básicamente una cuestión de la plasticidad de los elementos del sistema nervioso, mientras que las computadoras - máquinas de Turing - no tienen elementos plásticos. Para describir el pensamiento humano se necesita una forma muy diferente, no digital, como lo demuestra la Teoría Sinapsis-Estado.

[3]Primeramente, John Backus y posteriormente Peter Naur establecieron una sintaxis que (a nuestro juicio) responde a la lógica matemática (básicamente como todo lenguaje) y a una estructura algorítmica. El objetivo fue que esa gramática les permitiera crear una base de instrucción para describir y modelar distintos tipos de textos, a saber: documentos que estén bajo formato, conjunto de instrucciones de programación, protocolos de comunicación, lenguajes de programación, notación para las gramáticas y sintaxis de los lenguajes de programación de la computadora, etc. Por supuesto como todo metalenguaje, el Backus-Naur es artificial pero se estructura a semejanza de la lengua natural con el objetivo central de que sea implementado en lo que en la lingüística computacional se denomina como gramáticas de libre contexto (o por sus palabras en inglés context-free grammars). El objetivo que se plantea (durante el desarrollo de la primera etapa de nuestra investigación) es estudiar de qué manera se estructura y funciona el Backus-Naur. Por otro lado tratar de establecer un cuerpo de ejemplos que explique en cuáles campos de investigación se implementa este metalenguaje. Al describir idiomas, la forma Backus-Naur (BNF) es una notación formal para codificar gramáticas destinadas al consumo humano.

Muchos lenguajes de programación, protocolos o formatos tienen una descripción BNF en su especificación.

Cada regla en la forma de Backus-Naur tiene la siguiente estructura:

`n a m e ::= e x p a n s i o n`

El símbolo `::=` significa "se puede expandir en" y "se puede reemplazar con".

En algunos textos, un nombre también se denomina símbolo no terminal.

Cada nombre en la forma de Backus-Naur está rodeado por corchetes angulares, `< >`ya sea que aparezca en el lado izquierdo o derecho de la regla.

`Una` es una expresión que contiene símbolos terminales y símbolos no terminales, unidas por secuencia y elección.

`e x p a n s i o n`

Un símbolo de terminal es un literal como ( "+"o "function") o una clase de literales (como integer).

Simplemente yuxtaponer expresiones indica secuenciación.

Una barra vertical | indica elección.

Por ejemplo, en BNF, la gramática de expresión clásica es:

```
<expr> ::= <term> "+" <expr>
          | <term>
```

```
<term> ::= <factor> "*" <term>
          | <factor>
```

```
<factor> ::= "(" <expr> ")"
           | <const>
```

```
<const> ::= entero
```

Naturalmente, podemos definir una gramática para las reglas en BNF:

```
r u l e → n a m e ::= e x p a n s i o n
n a m e → < i d e n t i f i E r >
e x p a n s i o n → e x p a n s i o n e x p a n s i o n
e x p a n s i o n → e x p a n s i o n | e x p a n s i o n
e x p a n s i o n → n a m e
e x p a n s i o n → t e r m i n a l
```

Podríamos definir identificadores utilizando la expresión regular [-A-Za-z\_0-9]+.

Un terminal podría ser un literal entre comillas (como "+", "switch"o "<<=") o el nombre de una clase de literales (como integer).

El nombre de una clase de literales generalmente se define por otros medios, como una expresión regular o incluso una prosa.

La forma extendida de Backus-Naur (EBNF) es una colección de extensiones de la forma de Backus-Naur.

No todos ellos son estrictamente un superconjunto, ya que algunos cambian la relación de regla de definición ::= a =, mientras que otros eliminan los paréntesis angulares de los no terminales.

Más importante que las pequeñas diferencias sintácticas entre las formas de EBNF son las operaciones adicionales que permite en las expansiones.

#### Opción

En EBNF, los corchetes alrededor de una expansión , indica que esta expansión es opcional. [ expansion ]

Por ejemplo, la regla:

<term> ::= [ "-" ] <factor>

Permite negar factores.

#### Repetición

En EBNF, las llaves indican que la expresión se puede repetir cero o más veces.

Por ejemplo, la regla:

<args> ::= <arg> { "," <arg> }

define una lista de argumentos separados por comas convencionales.

#### Agrupamiento

Para indicar la precedencia, las gramáticas EBNF pueden usar paréntesis (), para definir explícitamente el orden de expansión.

Por ejemplo, la regla:

<expr> ::= <term> ("+" | "-") <expr>

Define una forma de expresión que permite tanto la suma como la resta.

#### Concatenación

En algunas formas de EBNF, el ,operador denota explícitamente la concatenación, en lugar de confiar en la yuxtaposición.

### III. CONCLUSIONES

El formato Backus-Naur (BNF) es un sistema notacional para especificar tipos de datos o categorías sintácticas, también especifica la sintaxis de los lenguajes de programacion mediante reglas de producción o de re-escritura. Este formato es muy importante para la tecnología y específicamente para

el campo de la programación, a pesar de que tenga mucho tiempo desde su creacion este sigue siendo importante y lo seguirá siendo por mucho tiempo, es de gran importancia que los individuos involucrados en el campo de la tecnología y específicamente el de la informática se vinculen y entiendan este formato ya que es un pilar importante.

### REFERENCIAS

#### Referencias de publicaciones periódicas:

- [1] ", Jhon Backus" Lecture for the ACM Turing award session 25, mar,2012 Available:  
[https://www.ecured.cu/John\\_Backus](https://www.ecured.cu/John_Backus)
- [2] Sue Gee (4 de enero de 2016). Peter «Naur Dies Aged 87».
- [3] Carlos Dimeo."Backus-Naur form". University of Bielsko-Biala.. [Online]. mayo 2015 Available:  
[https://www.researchgate.net/publication/325881080\\_Bac\\_kus\\_Naur\\_Form\\_reglas\\_generativas\\_y\\_descripcion\\_de\\_un\\_m\\_eta-lenguaje](https://www.researchgate.net/publication/325881080_Bac_kus_Naur_Form_reglas_generativas_y_descripcion_de_un_m_eta-lenguaje)

