BACKUS-NAUR

Autor: Sara Valentina Restrepo Ramírez
Risaralda, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: sara.restrepo1Qutp.edu.co

***Resumen*— La notación de Backus-Naur, también conocida por sus denominaciones inglesas Backus-Naur form (BNF), Backus-Naur formalism o Backus normal form, es un metalenguaje usado para expresar gramáticas libres de contexto: es decir, una manera formal de describir lenguajes formales.**

Palabras clave- Backus, notación ,programación de alto nivel,

***Abstract*— The Backus-Naur notation, also known by its English denominations Backus-Naur form (BNF), Backus-Naur formalism or Backus normal form, is a metalanguage used to express context-free grammars: that is, a formal way of describing formal languages**

I. INTRODUCCIÓN

Backus-Naur Form: reglas generativas y descripción de un meta-lenguaje Backus-Naur es un metalenguaje de programación que se ha consti-tuido 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 par-tir de las teorías lingüísticas de Noam Chomsky. De acuerdo con lo que planteó el propio Chomsky cuando afirmó: «teaching linguistics to students of information theory at MIT, combined linguistics and mathematics, by taking what is essentially Thue's formalism as the basis for the description of the syntax of natural language. He also introduced a clear distinction between generative rules (those of context free grammars) and transformation rules» (1956), no directa-mente Backus, pero
<oración> ::= [<sí Naur tomó el BNF (por sus siglas en inglés) para el desarrollo de las gramáticas de los lenguajes de programa-ción, pero también para representar en un lenguaje formalizado par-tes de las estructuras gramáticas de la lengua natural. 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 con-texto (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 ejem-plos que explique en cuáles campos de investigación se implementa este metalenguaje. Palabras clave Metalenguaje Lenguaje base traductor (Assembler)La introducción puede contener:

I. CONTENIDO

Teoría simple (introducción)
Ejemplos
Biografías
impactos

Ejemplos

<vocal> ::= “a” | “e” | “i” | “o”
| “u” | “á” | “é” | “í” | “ó”
| “ú” | “ü”
<consonante> ::= “b” | “c” | “d”
| “f” | “g” | “h” | “j” | “k”
| “l” | “m” | “n” | “ñ” | “p”
| “q” | “r” | “s” | “t” | “v”
| “w” | “x” | “y” | “z”

Ejemplos «lingüísticos»

sujeto>] <predicado>
<sujeto> ::= <sintagma_nominal>
<predicado> ::=
(<verbo_copulativo> <atributo>)
|(<verbo_predicativo>
[<complemento_directo>]
[<complemento_indirecto>]
{<complemento_circunstancial>})

<letraEsp> ::=
<letraIng> | “Ñ” | “Á” | “É”
| “Í” | “Ó” | “Ú” | “Ü” | “ñ”
| “á” | “é” | “í” | “ó” | “ú”
| “ü”
<palabra> ::=

<letraEsp> {<letraEsp>}

<letraIng> ::= “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”

biografías

John Backus (Filadelfia, 3 de diciembre de 1924 - Oregón, 17 de marzo de 2007) fue un científico de la computación[1] estadounidense.
3 de diciembre de 1924 Ver y modificar los datos en
Filadelfia (Estados Unidos
17 de marzo de 2007 Ashland (Estados Unidos)
Nacionalidad Estadounidense
Educación
Educado en Universidad de Columbia
Universidad de Virginia
Información profesional
Ocupación Matemático e informático
Área
Ciencias de la computación
Empleador
IBM Ver y modificar los datos en
Obras notables
Speedcoding
Rama militar
Ejército de los Estados Unidos
Conflictos
Miembro de
Academia Nacional de Ciencias
Academia Estadounidense de las Artes y las Ciencias
Ganador del Premio Turing en 1977 por sus trabajos en sistemas de programación de alto nivel, en especial por su trabajo con FORTRAN.

Para evitar las dificultades de programación de las calculadoras de su época, en 1954 Backus se encargó de la dirección de un proyecto de investigación en IBM para el proyecto y realización de un lenguaje de programación más cercano a la notación matemática normal. De ese proyecto surgió el lenguaje FORTRAN, el primero de los lenguajes de programación de alto nivel que tuvo un gran impacto, incluso comercial, en la emergente comunidad informática.

Tras la realización de FORTRAN, Backus fue un miembro muy activo del comité internacional que se encargó del proyecto de lenguaje ALGOL. En ese contexto propuso una notación para la representación de las gramáticas usadas en la definición de un lenguaje de programación (las llamadas gramáticas libres de contexto). Tal notación se conoce como Notación de Backus-Naur (Backus-Naur Form o BNF) y une al nombre de Backus al de Peter Naur, un informático europeo del comité ALGOL que contribuyó a su definición.

En los años 1970, Backus se interesó sobre todo por la Programación funcional, y proyectó el lenguaje de programación FP, descrito en el texto que le sirvió para ganar el premio Turing, "Can Programming be Liberated from the Von Neumann Style?" Se trata de un lenguaje de uso fundamentalmente académico, que sin embargo animó un gran número de investigaciones. El proyecto FP, transformado en FL, se terminó cuando Backus se jubiló en IBM, en 1991.

John Backus falleció el sábado 17 de marzo de 2007, a la edad de 82 años en su casa en Ashland, Oregón por causas naturales, de acuerdo a la declaración de su familia.[2]

Peter Naur (Frederiksberg, 25 de octubre de 1928-3 de enero de 2016)1 fue un científico danés pionero en informática y ganador del Premio Turing en 2005.

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.2 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).3 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.

impactos

creación de la notación de Backus-Naur.

trabajos en sistemas de programación de alto nivel, en especial por sus trabajos con FORTRAN.

Conclusiones

John Backus fue un científico de la computación.
Backus y Naur ganaron el premio Turing en 1977 por sus trabajos en FORTRAN
El BNF se utiliza extensamente como notación para las gramáticas de los lenguajes de programación de los sistemas de comandos.

Referencias

http://webdiis.unizar.es/~latre/prog1f/2017-18/Problemas_1_Notacion_BNF.pdf

https://es.m.wikipedia.org/wiki/John_Backus