UNIVERSIDAD NACIONAL GUILLERMO BROWN

Conceptos y Paradigmas de Lenguajes de Programación

Clase 3 - 2025

Cómo definir la sintaxis

Recordemos conceptos

- Sintaxis

Conjunto de reglas formales que especifican la forma correcta de escribir sus sentencias. Se expresan a través de reglas gramaticales y/o diagramas sintácticos.

- Semántica

Especifica el significado de las sentencias de un programa sintácticamente válido escrito en dicho lenguaje

¿Cómo describir la sintaxis de un LP?

- Todo lenguaje L está formado por un conjunto de sentencias.

- Las sentencias están construidas con caracteres que pertenecena un alfabeto ∑ dado.

- Por lo tanto, la sintaxis de un lenguaje se define por dos conjuntos de reglas:

- Reglas léxicas
- Reglas sintácticas

Cómo definir la sintaxis

Se necesita una descripción finita para definir un conjunto infinito (conjunto de todos los programas bien escritos)

Formas para definir la sintaxis:

- Lenguaje natural. Ej.: Fortran

- Utilizando la gramática libre de contexto, definida por Backus y Naun: BNF. Ej: Algol
BNF: es un metalenguaje y utiliza **gramáticas** para establecer las reglas de escritura de un lenguaje Dado.

- **Diagramas sintácticos**, que son equivalentes a BNF pero mucho mas intuitivos

BNF (Backus Naun Form)

Es un metalenguaje usado para expresar gramáticas libres de contexto: es decir, una manera formal de describir lenguajes formales.

El BNF se utiliza extensamente como notación para las gramáticas de los lenguajes de programación, de los sistemas de comando y de los protocolos de comunicación

- Es una notación formal para describir la sintaxis
- Es un metalenguaje
- Utiliza metasímbolos ■ < > ::= |
- Define las reglas por medio de “producciones”

Ejemplo:

< digito > : := 0|1|2|3|4|5|6|7|8|9

Metalenguaje

- No terminal Se define como Terminales

BNF (Backus Naun Form) - Ejemplo

BNF para una dirección postal de los EE. UU.

```
<dirección postal> ::= <nombre> <dirección> <apartado postal>

<nombre> ::= <personal> <apellido> [<trato>] <EOL> | <personal> <nombre>

<personal> ::= <primer nombre> | <inicial> "."

<direccion> ::= [<dpto>] <numero de la casa> <nombre de la calle> <EOL>

<apartado postal> ::= <ciudad> "," <código estado> <código postal> <EOL>
```

- Una dirección postal consiste en un nombre, seguido por una dirección, seguida por un apartado postal.
- Una parte «personal» consiste en un nombre o una inicial seguido(a) por un punto.

Gramática

Qué es:

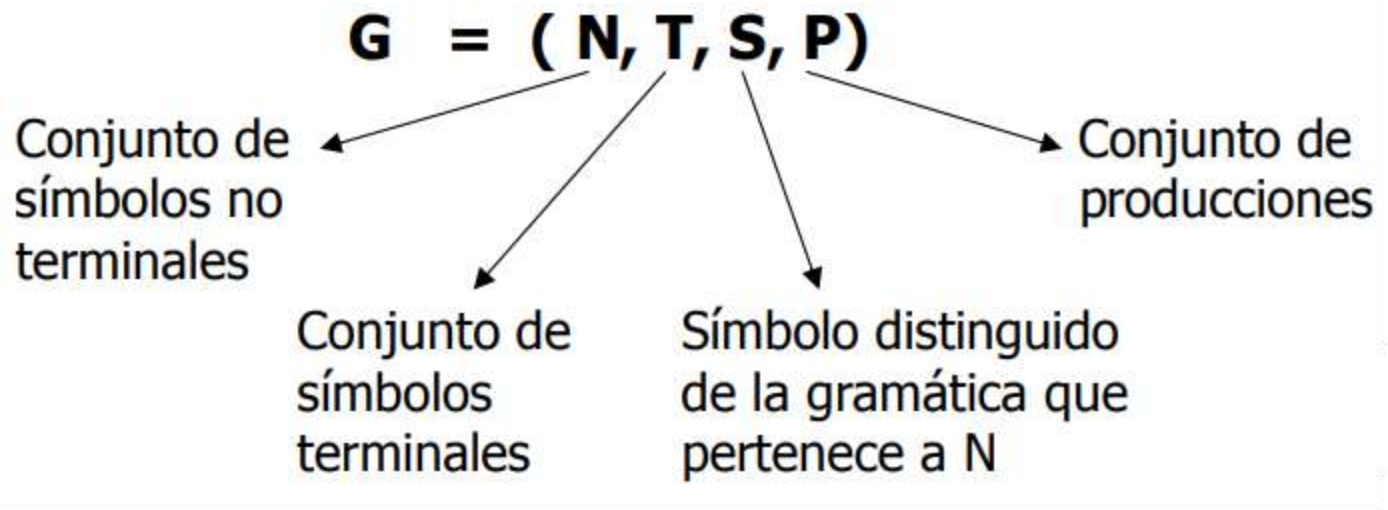
- Es la descripción formalizada de las frases de un lenguaje y está basada en reglas gramaticales. (Chomsky).

- Es el conjunto de reglas finita que define un conjunto infinito de posibles sentencias válidas en el lenguaje.

Gramática

Como está formada

- Una gramática esta formada por una 4-tupla **G = (N, T, S, P)**



G= { N, T, S, P}

- Un conjunto N de símbolos **NO** terminales, que deben ser definidos a través de una regla gramatical
- Un conjunto T de símbolos terminales, caracteres del Alfabeto

- Un símbolo inicial **S**, que pertenece a N

- Un conjunto de reglas o plantillas de reconocimiento y/o generación de las sentencias de un lenguaje, denominadas **reglas de producción (P) A veces También denominada RP.**

SINTAXIS

Árboles Sintácticos -> busquemos en IA

un **árbol de sintaxis abstracta** (**AST**), o simplemente un **árbol de sintaxis**, es una representación de árbol de la estructura sintáctica simplificada del código fuente escrito en cierto lenguaje de programación.

Supongamos la siguiente oración:

“Juan un canta Zamba”

- Es una oración sintácticamente incorrecta
- No todas las oraciones que se pueden armar con los terminales son válidas
- Se necesita de un Método de análisis (reconocimiento) que permita determinar si un string dado es válido o no en el lenguaje: **Parsing**.
- El **parse**, para cada sentencia construye un “árbol sintáctico o árbol de derivación”

Árboles Sintácticos

Arbol de derivación Un árbol de derivación permite mostrar gráficamente cómo se puede derivar cualquier cadena de un lenguaje a partir del símbolo distinguido de una gramática que genera ese lenguaje. Un árbol es un conjunto de puntos, llamados nodos, unidos por líneas, llamadas arcos. Un arco conecta dos nodos distintos.

Para ser un árbol un conjunto de nodos y arcos debe satisfacer ciertas propiedades:

- hay un único nodo distinguido, llamado raíz (se dibuja en la parte superior) que no tiene arcos incidentes.
- todo nodo c excepto el nodo raíz está conectado con un arco a otro nodo k, llamado el padre de c (c es el hijo de k). El padre de un nodo, se dibuja por encima del nodo.
- todos los nodos están conectados al nodo raíz mediante un único camino.
- los nodos que no tienen hijos se denominan hojas, el resto de los nodos se denominan nodos interiores.

El árbol de derivación tiene las siguientes propiedades:

- el nodo raíz está rotulado con el símbolo distinguido de la gramática;
- cada hoja corresponde a un símbolo terminal o un símbolo no terminal;
- cada nodo interior corresponde a un símbolo no terminal. nodo raíz nodos interiores hojas Para cada cadena del lenguaje generado por una gramática es posible construir (al menos) un árbol de derivación, en el cual cada hoja tiene como rótulo uno de los símbolos de la cadena.

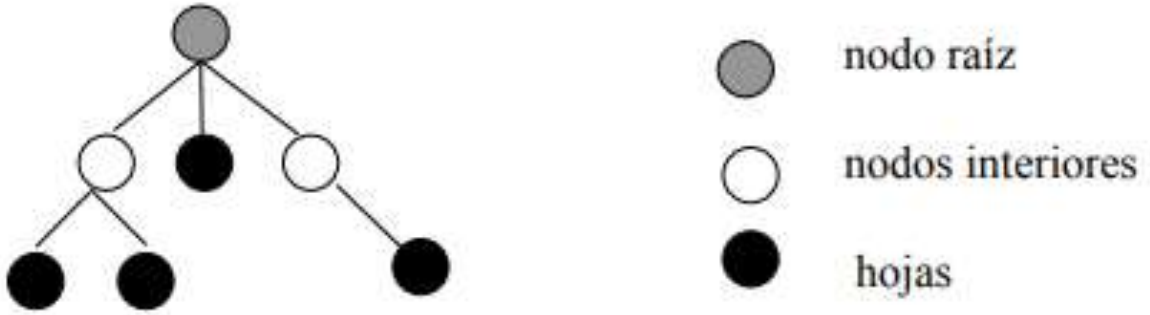
Arboles Sintácticos (AST)

- Siendo el producto en la fase análisis sintáctico de un compilador, el AST tiene varias propiedades que son inestimables a los siguientes pasos del proceso de compilación.
- Comparado al código fuente, un AST **no** incluye ciertos elementos, como puntuación no esencial y delimitadores (corchetes, punto y coma, paréntesis, entre otros.).
- Una diferencia importante es que el AST puede ser editado y mejorado con propiedades y anotaciones para cada elemento que contiene. Esta edición y anotación es imposible con el código fuente de un programa, ya que esto implicaría cambiarlo.
- Al mismo tiempo, un AST usualmente contiene información extra sobre el programa, debido a las etapas consecutivas de análisis del compilador. Un ejemplo simple de la información adicional presente en un AST es la posición de un elemento en el codigó fuente. Esta información es usada en caso de un error en el codigó, para notificar al usuario la locación de un error.

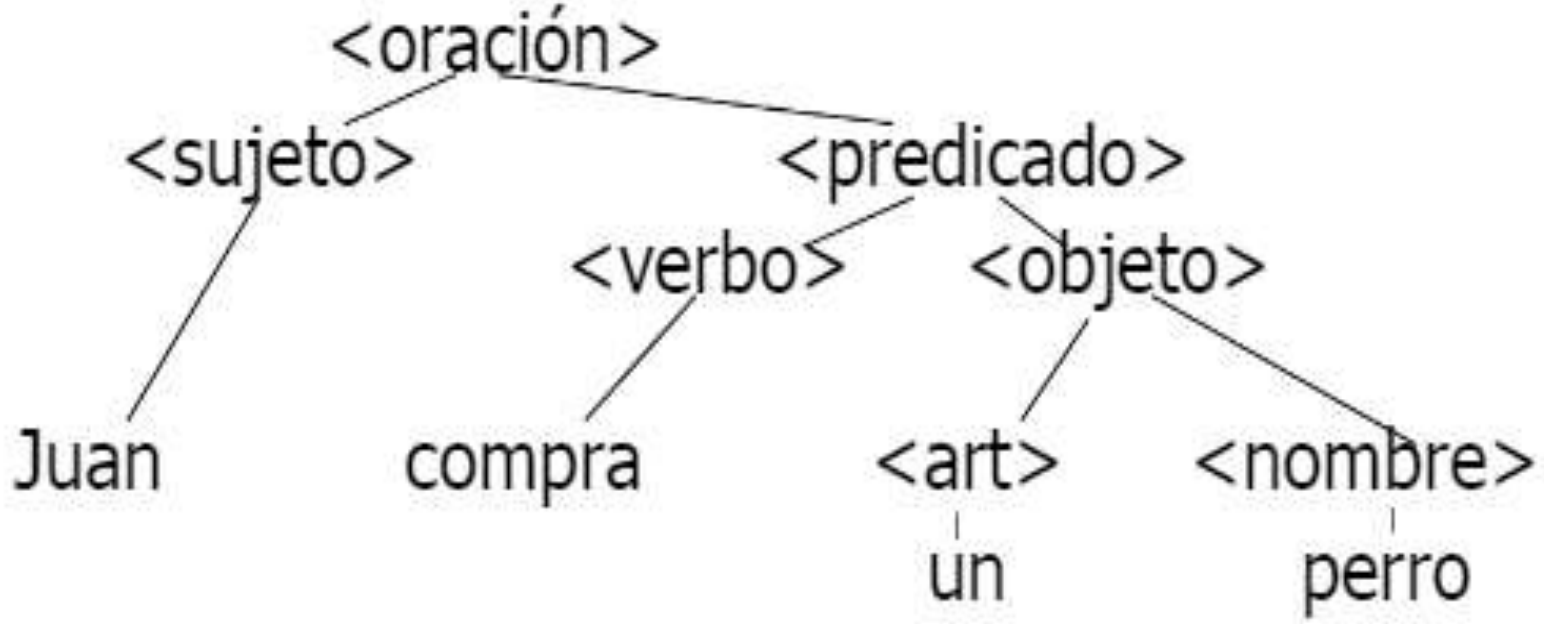
Árboles sintácticos

Existen Dos maneras de construirlo:

- Método botton-up
  - De izquierda a derecha
  - De derecha a izquierda
- Método top-dow
  - De izquierda a derecha
  - De derecha a izquierda



Ejemplo: árbol sintáctico de “oración”. Top-down de izquierda a derecha



Sintaxis - Árbol de derivación:

Un árbol de derivación es una forma de visualizar las relaciones que tienen las cabezas de reglas junto con los cuerpos de estas. En un árbol de derivación, los elementos del cuerpo de una regla se convierte en nodos hijos de la cabeza de la regla

Ejemplo - top-down de izquierda a derecha:

<oración> => <sujeto><predicado>
=> Juan <predicado>
=> Juan <verbo><objeto>
=> Juan compra <objeto>
=> Juan compra art><sustan>
=> Juan compra un <sustan>
=> Juan compra un perro

Sintaxis - Árbol de derivación:

En lenguajes formales y lingüística computacional, un árbol de sintaxis abstracta (AST), o simplemente un árbol de sintaxis, es una representación de árbol de la estructura sintáctica simplificada del código fuente escrito en cierto lenguaje de programación

Sintaxis - Gramáticas libres de contexto y sensibles al contexto :

int e; **a := b + c;**

■ Según nuestra gramática son sentencias sintácticamente válidas, aunque puede suceder que a veces no lo sea semánticamente.

■ Una gramática libre de contexto es aquella en la que no realiza un análisis del contexto.

■ Una gramática sensible al contexto analiza este tipo de cosas. (Algol 68).

Sintaxis - Otras formas de describir la sintaxis libres de contexto:

EBNF (Esta gramática es la BNF extendida)

Los metasímbolos que incorporados son:

[ ] elemento optativo puede o no estar

(|) selección de una alternativa

{} repetición

* 0 o mas veces + una o mas veces

Ejemplo con EBNF: Definición números enteros en BNF y en EBNF

BNF

<enterosig> ::= + <entero> | - <entero> | <entero>

<entero> ::= <digito>| <entero><digito>

Recursión

EBNF

<enterosig>::= [(+|-)] <digito>{<digito>}*

Eliminó la recursión y es mas fácil de entender

https://www.youtube.com/watch?v=feNOEcRiHOs

Pero todo esto, Cómo se aplica a un programa?

S: <programa>::=begin <listasent> end.

<listasent>::= <sent> | <listasent>; <sent>
<sent>::= <if> | <while>| .....

A partir de la descripción BFN el compilador valida a cada una de las sentencias de L, realizando el análisis léxico y sintáctico correspondiente.

Bibliografía

https://www.youtube.com/watch?v=7sy9sTKv2p8&t=2519s

Programming Language Concepts. Carlo Ghezzi, Mehdi Jazayeri. Ed. John Wiley & Sons, 3rd edition, 1998 (versiones disponibles en inglés y castellano)

Concepts of Programming Languages. Robert W. Sebesta. Publisher: Addison-Wesley, 2007.

Autómatas y lenguajes. Un enfoque de diseño. Ramon Brena Pinnero. Teconológico de Monterrey, 2003.

Introducción a la Teoría de Autómatas, Lenguajes y Computación. Hopcroft, Motwani y Ullman. Ed. Pearson, 2002.

Lenguajes, Gramáticas y Autómatas. Un Enfoque Práctico. P. Isasi, P. Martínez y D. Borrajo. Ed. Addison-Wesley 2001