Lenguajes de Programación

Unidad II

“Análisis y Descripción de Lenguajes de Programación”

Rogelio Ferreira Escutia

Introducción

- Durante los primeros días del diseño de lenguajes, cuando se estaban desarrollando los lenguajes FORTRAN, ALGOL, COBOL y LISP, alrededor de 1960, se pensaba que una sintaxis formal era todo lo que se requería para especificar programas.
- El concepto de una gramática libre de contexto o gramática de forma Backus-Naur (BNF) se desarrolló y utilizó con éxito para especificar la sintaxis de un lenguaje.
- En la actualidad, aún se utilizan como la técnica principal para describir los componentes de un programa.

Contenido

1) *Sintaxis*
2) *Etapas de Traducción*
3) *Análisis Léxico y Sintáctico*
4) *Análisis Semántico*
5) *Métodos Formales para la Especificación de Sintaxis*
    1) *Gramática Libre de Contexto*
    2) *BNF*
    3) *Arboles de Análisis Sintáctico*

1) Sintaxis

Sintaxis

- La Sintaxis es “La Disposición de palabras como elementos en una oración para mostrar su relación. Describe la serie de símbolos que constituyen programas válidos”.
- El enunciado “X = Y + Z” representa una serie válida de símbolos, en tanto que “XY +-” no representa una secuencia válida de símbolos para un programa en C.
- La sintaxis suministra información significativa que se necesita para entender un programa y proporciona información imprescindible para la traducción del programa fuente a un programa objeto

Sintaxis

- Como en la expresión ambigua en español “nada en el agua”, el solo desarrollo de una sintaxis de lenguaje es insuficiente para especificar sin ambigüedad la estructura de un enunciado.
- En un enunciado como X = 2 + 7.65 la sintaxis no nos puede decir si se asignó un valor a X, o si X se declaró como tipo real.

Criterios Generales de Sintaxis

- **Legibilidad: Un programa es legible si la estructura subyacente del algoritmo y los datos que el programa representa quedan de manifiesto al inspeccionar el texto del programa.**

- **Facilidad de Escritura: Las características sintácticas que hace que un programa sea fácil de escribir suelen hallarse en conflicto con las características que facilitan su lectura.**

- **Facilidad de Verificación: Entender cada enunciado de un programa es relativamente fácil, el proceso global de crear programas correctos es en extremo difícil, por lo que se necesitan técnicas que permitan probar que el programa es matemáticamente correcto.**

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

Criterios Generales de Sintaxis

- **Facilidad de Traducción: la traducción de los programas se dificulta conforme aumenta el número de construcciones sintácticas especiales. La clave para una traducción fácil es la regularidad de la estructura.**

- **Carencia de Ambigüedad: Una construcción ambigua permite dos o más interpretaciones distintas. El problema de ambigüedad surge por lo común no en la estructura de elementos individuales del programa, sino en la interacción entre diferentes estructuras**

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

Elementos Sintácticos de un Lenguaje

- **Conjunto de Caracteres: Existen varios conjuntos de caracteres de uso amplio, como el conjunto ASCII, cada uno con un conjunto diferentes de caracteres especiales, además de las letras y dígitos básicos.**

- **Identificadores: La sintaxis básica para identificadores, una cadena de letras y dígitos que comienzan con una letra, es la mas usada. Las variaciones entre lenguajes se dan principalmente en la inclusión opcional de caracteres especiales como punto (.) y guión (-)**

- **Símbolos de Operadores: Casi todos los lenguajes emplean los caracteres especiales (+) y (-) para representar las 2 operaciones aritméticas básicas. Casi todos los lenguajes adoptan alguna combinación y utilizan caracteres especiales para ciertos operadores.**

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

Elementos Sintácticos de un Lenguaje

- **Palabras clave y palabras reservadas: Una palabra clave es un identificador que se usa como una parte fija de la sintaxis de un enunciado, por ejemplo "IF". Una palabra clave es una palabra reservada y no se puede utilizar como un identificador. Casi todos los lenguajes emplean actualmente palabras reservadas con lo cual se mejora la capacidad de detección de errores de los traductores.**

- **Comentarios: son mensajes que nos permiten mejorar la legibilidad de los programas sin interferir con su funcionamiento. Los comentarios pueden ser codificados de varias maneras dependiendo del lenguaje, como renglones separados en Fortran, "/*" y "*/" en C, al final de un renglón como en Ada, en un renglón (al principio o al final) con "//" como en C++, o "!" en Fortran 90.**

Elementos Sintácticos de un Lenguaje

- **Espacios en Blanco: varían ampliamente de lenguaje a lenguaje. En Fortran no son significativos, excepto en cadena de caracteres. En Snobol 4 la concatenación es por medio de un espacio en blanco y también se utiliza como separador entre elementos de un enunciado. En C se pasan por alto.**

- **Delimitadores: es un elemento sintáctico que se usa simplemente para señalar el principio o final de alguna unidad sintáctica, como un enunciado o expresión.**

- **Formatos de campos libres y fijos: una sintaxis es de campo libre si los enunciados de un programa se pueden escribir en cualquier parte de un renglón o las interrupciones entre renglones. Una sintaxis de campo fijo, utiliza la posición sobre un renglón de entrada para transmitir información.**

Elementos Sintácticos de un Lenguaje

- **Expresiones: son funciones que acceden a objetos de datos en un programa y devuelven algún valor. Las expresiones son los bloques sintácticos básicos de construcción a partir de los cuales se construyen enunciados.**

- **Enunciados: constituyen el componente sintáctico más destacado en los lenguajes imperativos. Su sintaxis tiene un efecto decisivo sobre la regularidad, legibilidad y facilidad de escritura generales del lenguaje.**

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

2) Etapas de Traducción

Traducción

- El proceso de traducción de un programa, de su sintaxis original a una forma ejecutable, es medular en toda implementación de lenguajes de programación.

- Casi todos los lenguajes se podrían implementar con solo una traducción trivial, si uno estuviera dispuesto a escribir un intérprete de software y a aceptar velocidades lentas de ejecución. En la mayoría de los casos, sin embargo, la ejecución eficiente es un objetivo tan deseable que se hacen esfuerzos importantes para traducir los programas a estructuras ejecutables con eficiencia, en especial código de máquina interpretable por hardware.

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

Traducción

- La traducción se divide en 2 partes principales

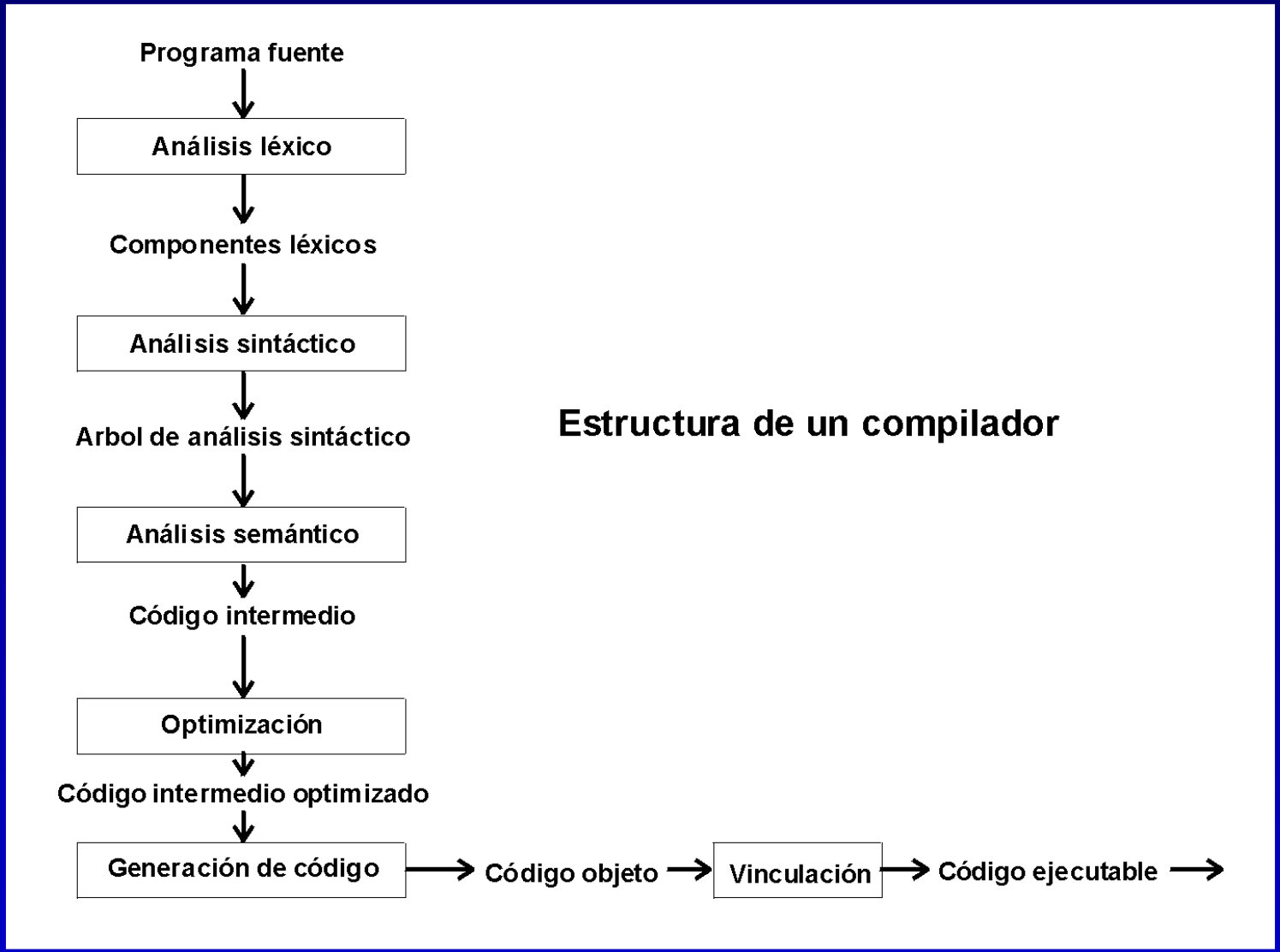
  1) Análisis del programa fuente de entrada
  2) Síntesis del programa objeto ejecutable

- Un compilador básico realiza lo anterior de la manera siguiente:

  1) Descomponer el programa de entrada en los componentes que lo constituye y obtener información, como el uso de nombres de variable.
  2) Genera el programa objeto a partir de la información recolectada anteriormente.

- Si la velocidad de compilación es importante (como en un compilador educativo) se emplea una estrategia de un paso, en donde, mientras se analiza, se convierte de inmediato a código objeto, en caso contrario se pueden implementar varias pasos.

Traducción


Programa fuente
Análisis léxico
Componentes léxicos
Análisis sintáctico
Arbol de análisis sintáctico
Análisis semántico
Código intermedio
Optimización
Código intermedio optimizado
Generación de código
Código objeto
Vinculación
Código ejecutable
Estructura de un compilador

3) Análisis Léxico y Sintáctico

Análisis Léxico

- **El analizador léxico es la fase fundamental de cualquier traductor, ya que se encarga de agrupar una serie de caracteres en sus componentes fundamentales, como son: identificadores, delimitadores, símbolos de operadores, números, palabras clave, espacios en blanco, comentarios, etc.**

- **Lo que realiza básicamente es, leer renglones sucesivos del programa de entrada, los descompone en elementos léxicos individuales y alimenta estos elementos léxicos a las etapas superiores del traductor para su uso en los niveles superiores de análisis**

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

Análisis Sintáctico

- **La segunda etapa de la traducción es el análisis sintáctico (conocido como parsing).**
- **En esta etapa se identifican las estructuras de programa mas grandes (enunciados, declaraciones, expresiones, etc.) usando los elementos léxicos producidos por el analizador léxico.**
- **Por lo general, el análisis sintáctico se alterna ordinariamente con el análisis semántico. Primero el analizador sintáctico identifica una serie de elementos léxicos que forman una unidad sintáctica como una expresión, posteriormente se llama al analizador semántico para que procese esa unidad**

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

4) Análisis Semántico

Análisis Semántico

- **Aquí se procesan las estructuras sintácticas reconocidas por el analizador sintáctico y la estructura del código objeto ejecutable comienza a tomar forma.**
- **En esta etapa ya puede salir el programa ejecutable final, aunque todavía puede ser manipulada por la etapa de optimización antes de generar la versión final del ejecutable.**
- **El analizador semántico se divide ordinariamente en un conjunto de analizadores semánticos mas pequeños, cada uno de los cuales maneja un tipo particular de construcciones de programa.**
- **Los analizadores semánticos interactúan entre ellos mismos a través de información que se guarda en diversas estructuras de datos, en particular en la tabla central de símbolos.**

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

Funciones del Analizador Semántico

- **Mantenimiento de tablas de símbolos: contiene una entrada por cada identificador diferente encontrado en el programa fuente. La tabla contiene además del identificador, datos adiciones respecto a los atributos de ese identificador, como puede ser tipo de variable, nombre del arreglo, nombre del subprograma, etc.**

- **Inserción de información implícita: existe cierta información en los programas fuente que esta implícita y debe de hacerse explícita en los programas objeto de nivel mas bajo, esto puede ser, como cuando se declara una variable de un cierto tipo.**

- **Detección de errores: el analizador semántico debe ser capaz de reconocer los errores cuando se presenten y generar mensajes de error apropiados**

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

Funciones del Analizador Semántico

- **Procesamiento de Macros: una macro es un trozo de texto de programa que se ha definido por separado y que se va a insertar en el programa durante la traducción, siempre que se encuentre una llamada de macro en el programa fuente.**

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

5) Métodos Formales para la Especificación de Sintaxis

Gramática

- La definición formal de la sintaxis de un lenguaje de programación se conoce ordinariamente como una gramática, en analogía con la terminología común para los lenguajes naturales.
- Una gramática se compone de un conjunto de reglas (llamadas producciones) que especifican las series de caracteres (o elementos léxicos que forman programas permitidos.
- Una gramática formal es simplemente una gramática que se especifica usando una notación definida de manera estricta.
- Algunas de las gramáticas utilizadas en compiladores son las "BNF" y los "Arboles de Análisis Sintáctico".

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

1) Gramática Libre de Contexto

Gramática Libre de Contexto

- **Una gramática libre de contexto consiste en una serie de reglas gramaticales.**
- **Las reglas están formadas del lado izquierdo de un solo nombre de estructura seguida del símbolo "→", y del lado derecho hay una secuencia de elementos que pueden ser símbolos u otros nombres de estructura.**
- **Los nombres de las estructuras se conocen como no terminales, ya que se subdividen en estructuras adicionales.**
- **A las palabras o símbolos de tokens también se les conoce como terminales, dado que jamás se subdividen.**
- **Las reglas gramaticales también se llaman producciones puesto que producen las cadenas del lenguaje utilizando derivaciones.**

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

Gramática Libre de Contexto

- **Una gramática libre de contexto también tiene un no terminal distinguido, conocido como símbolo inicial. Este no terminal representa toda la estructura que se está definiendo (como una oración o un programa) y es el símbolo con el que se inician todas las derivaciones**

- **Se denominan "Libres de Contexto" debido a que los no terminales aparecen de manera individual del lado izquierdo de las producciones, esto significa que cada no terminal puede ser reemplazado por cualquier opción del lado derecho independientemente de dónde pudiera aparecer el no terminal.**

- **Dicho de otra manera, no existe "contexto" bajo el cual sólo pudieran ocurrir ciertos reemplazos.**

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

Gramática Libre de Contexto

- **Ejemplos:**

- **Exprese la descripción de expresiones aritméticas enteras simples con adición y multiplicación:**

$$\begin{aligned}
&Expresión \rightarrow Expresión + Expresión \mid Expresión * Expresión \mid (Expresión) \mid número \\
&número = número\ \ dígito \mid dígito \\
&dígito = 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}$$

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

2) Gramática BNF

Gramática BNF

- En los años 50´s, el lingüista Noam Chomsky desarrolló una forma gramatical que denominó "gramática libre del contexto" para la definición de la sintaxis de lenguajes naturales.

- La gramática BNF (Backus Naur Form) debe su nombre a que fue desarrollada por John Backus en 1960 para describir la sintaxis del lenguaje Algol60, mientras que Peter Naur era el presidente del comité que desarrolló Algol60.

- La BNF y la gramática libre de contexto son equivalentes en cuanto a poder, las diferencias corresponden sólo a la notación, es por eso que los términos BNF y gramática libre del contexto son ordinariamente intercambiables en el estudio de la sintaxis. En la actualidad se utiliza la BNF original, la BNF extendida (EBNF) y los diagramas sintácticos.

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

BNF

- Una gramática BNF se compone de un conjunto finito de reglas de gramática las cuales definen un lenguaje, que específicamente en computación, sería un lenguaje de programación.
- Un lenguaje, desde el punto de vista sintáctico, se compone de un conjunto de programas sintácticamente correctos, cada uno de los cuales es simplemente una serie de caracteres.
- Un programa sintácticamente correcto no necesita necesariamente tener sentido.

• 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

Lenguaje

- **Un lenguaje es cualquier conjunto de cadenas de caracteres (de longitud finita) con caracteres elegidos de algún alfabeto finito fijo de símbolos.**

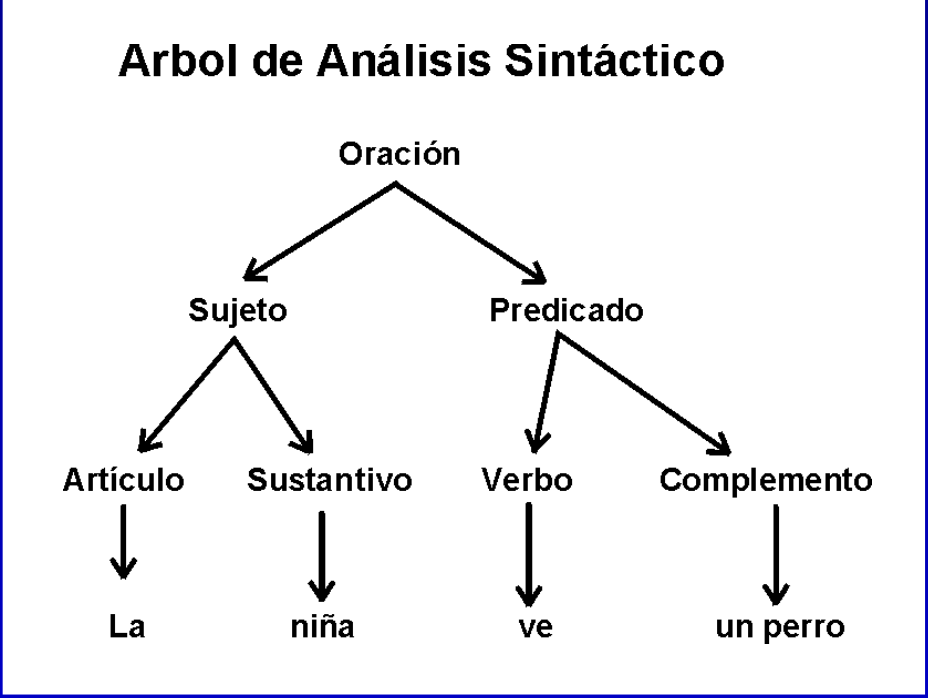
- **Bajo la definición anterior, todos los siguientes son lenguajes:**

  El conjunto de todos los enunciados de asignación en C.
  El conjunto de todos los programas en C.
  El conjunto compuesto de secuencias de letras a y b.

3) Arboles de Análisis Sintáctico

Arboles de Análisis Sintáctico

- Este método describe de manera gráfica el proceso de reemplazo dentro de una derivación.
- Ejemplo: Escribir el árbol de análisis sintáctico para la oración “La niña ve un perro”:



- 44) Lenguajes de Programación: Diseño e Implementación, Terrence W. Pratt, Prentice Hall

Unidad II – Análisis y Descripción de Lenguajes de Programación

Rogelio Ferreira Escutia

Instituto Tecnológico de Morelia
Departamento de Sistemas y Computación

Correo: ***rogelio@itmorelia.edu.mx***
rogeplus@gmail.com

Página Web: ***http://sagitario.itmorelia.edu.mx/~rogelio/***
http://www.xumarhu.net/

Twitter: ***http://twitter.com/rogeplus***
Facebook: ***http://www.facebook.com/groups/xumarhu.net/***