Sintaxis y Semántica del lenguaje.

Tema: Semántica de los lenguajes de programación

Procesamiento de lenguajes: compiladores e intérpretes

Semántica de un lenguaje

- Describir la semántica de un lenguaje es describir cuál es el efecto que tiene cada instrucción válida del mismo en ejecución.
- Cómo ejecuta el SO un programa? Qué efectos tiene sobre la memoria? Qué cambios se producen?
- ¿Cómo entiende cuál es el efecto de cada instrucción o su significado?
- La realidad es que solo se ejecutan órdenes en lenguaje de máquina a muy bajo nivel…….

Procesamiento de lenguajes

- Hay dos alternativas para procesar un lenguaje de alto nivel:

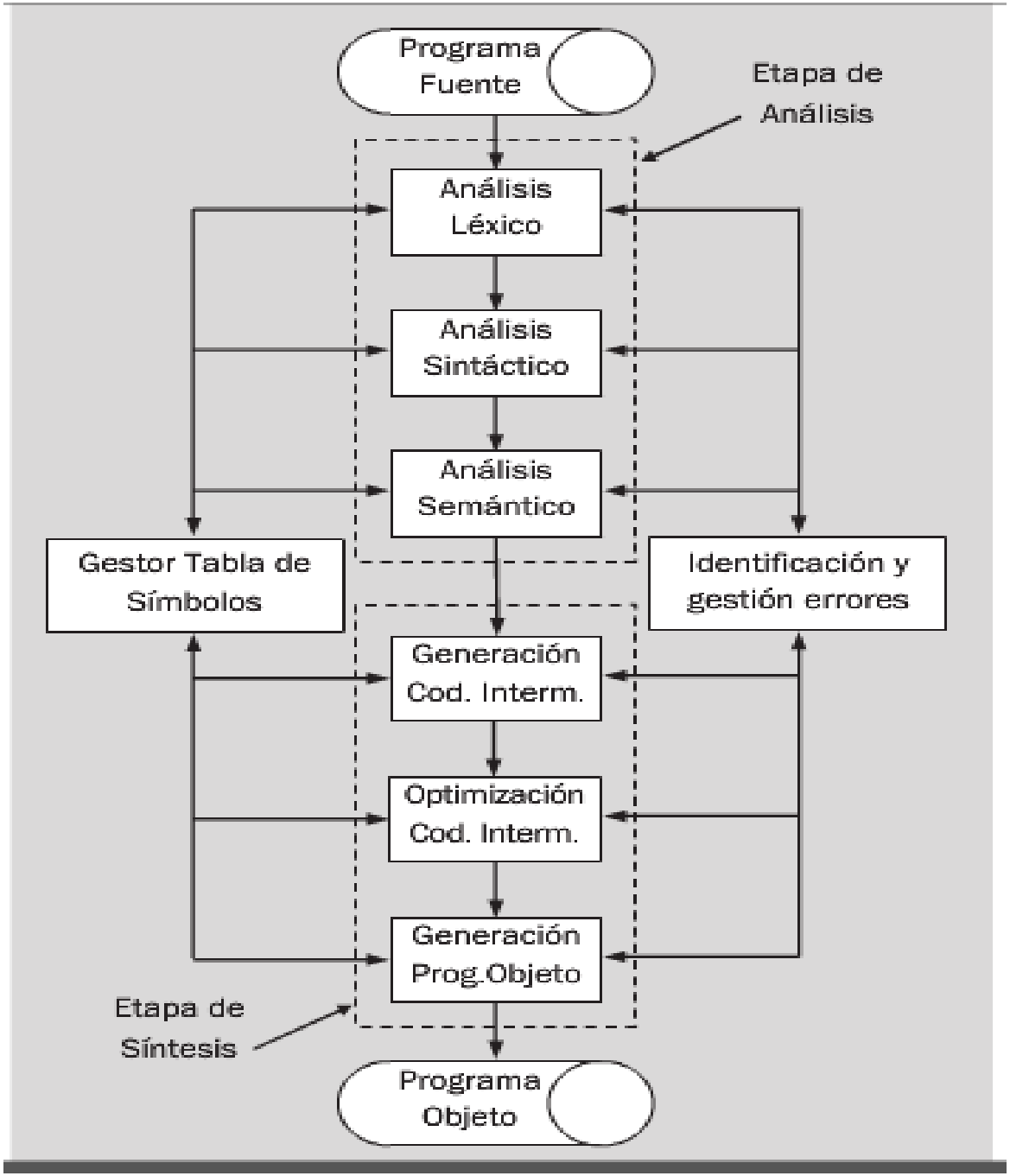
  - **Traducción o compilación:** es el proceso mediante el cual se traducen cada una de las instrucciones escritas en el lenguaje de alto nivel a lenguaje de máquina y se guarda el compilado.

    Abarca varias etapas:

    - ANÁLISIS (front end)
      - análisis léxico,
      - sintáctico y
      - semántico de cada instrucción y
    - SÍNTESIS (back end)

    Cada lenguaje de programación requiere su propio compilador.

  - **Interpretación**: en este proceso se traduce línea a línea y se ejecuta directamente, sin guardar la traducción. Si la misma instrucción aparece varias veces seguida, se la decodifica cada vez. No genera programación objeto.



PROCESAMIENTO DE LENGUAJES

TRADUCCIÓN / COMPILACIÓN

ETAPA DE ANALISIS LEXICO

- En esta fase, la cadena de caracteres que constituye el programa fuente es leída carácter a carácter, para identificar y agrupar sus componentes léxicos. Se ignoran los comentarios.

- Estos componentes léxicos son los caracteres del alfabeto y secuencias de caracteres (cadenas) que tienen un significado colectivo (identificadores, variables, constantes, palabras claves del lenguaje, operadores aritméticos, etc.) que son denominados tokens.

- A medida que los componentes léxicos son identificados, se los clasifica y se los registra en la tabla de símbolos. La tabla de símbolos es una estructura que está destinada al almacenamiento y recuperación de todos los elementos que forman parte de un programa.

- En esta etapa para validar que los tokens estén bien conformados se aplican **autómatas finitos.**

TRADUCCIÓN / COMPILACIÓN

ETAPA DE ANALISIS LEXICO

Tabla de símbolos

Componentes léxicos o *tokens*	Lexemas	Descripción
Identificadores	ii, total0, valor3, ...	Primer símbolo alfabético seguido de otros símbolos o dígitos
Operadores relacionales	<, <=, ==, >, >=, !=	Símbolos menor, igual o mayor y sus combinaciones
Operadores aritméticos	+, -, *, /, %	Símbolos de operaciones aritméticas básicas
Operadores de manipulación de bits	&, |, ^, ~, <<, >>	Operadores AND, OR, XOR, NOT y de desplazamiento.
Instrucciones de estructuras de control	if, else, switch, while, for, do	Instrucciones selectivas y repetitivas
Valores numéricos	125, 3.1416, 6.12E02, 0x7F, 0157	Constantes numéricas enteras, reales, octales y hexadecimales

Inicialmente esta tabla es cargada por el analizador léxico.

Luego se completa en las restantes fases de análisis y será empleada intensivamente en la siguiente etapa de síntesis.

Traducción / compilación

ETAPA DE ANALISIS SINTACTICO

- Los componentes léxicos ya fueron identificados en la fase anterior, pero las combinaciones de estos componentes dan lugar a sentencias del programa fuente, por lo tanto ahora es necesario comprobar que dichas sentencias sean sintácticamente correctas.

- Es decir, se debe verificar que todas las sentencias puedan haber sido generadas por la gramática del lenguaje fuente. En caso contrario, el analizador debe informar sobre los errores sintácticos.

- Hace este análisis sintáctico **usando gramáticas y construye un árbol de derivación**. Con ello verifica que la secuencia de símbolos formen palabras válidas del lenguaje

TRADUCCIÓN / COMPILACIÓN

ETAPA DE ANALISIS SEMANTICO

- En el análisis semántico, se revisa al programa fuente, sintácticamente correcto, para reunir información sobre los tipos de las variables que serán utilizadas en la fase posterior de generación de código intermedio.

- Simultáneamente, se procuran identificar eventuales errores semánticos. Las comprobaciones en esta fase incluyen la detección y comunicación de numerosos errores que corresponden a:
  - i) comprobación de tipos,
  - ii) comprobación de flujos de control,
  - iii) comprobación de unicidad o coherencia en las denominaciones de identificadores,
  - iv) coherencia en los argumentos de subprogramas y
  - v) potenciales errores en tiempo de ejecución (variables no inicializadas, direccionamiento de arreglos fuera de límites, cocientes que pueden tomar valores nulos, etc.).

Una de las formas más simples de implementar analizadores semánticos las brindan las gramáticas con atributos.

Traducción / compilación

ETAPA DE SINTESIS

Una vez finalizada la etapa de análisis se pasa a la etapa final:

- Traduce cada instrucción a código assembler (código intermedio)
- Traduce a código de máquina

El resultado final es el programa objeto compilado.

Dentro de los lenguajes de programación que son compilados tenemos la familia **C** que incluye a **C++, Objective C, C#** y también otros como **Fortran, Pascal, Haskell y Visual Basic**.

INTERPRETACIÓN

- El intérprete realiza la traducción ***línea a línea*** a medida que las va procesando y no guarda su traducción:

  - Toma una instrucción
  - La traduce
  - La ejecuta

  Si la misma instrucción aparece varias veces, al no haberla guardado, debe volver a traducirla.

  Ruby, Python, PHP (se interpreta del lado del servidor), **JavaScript** y otros como **Perl, Smalltalk, Lisp, Prolog** son todos interpretados.

Traducción vs interpretación

- Traducción → ocupa más memoria pero es un proceso más rápido: cada instrucción ya traducida se almacena y si vuelve a aparecer se usa su traducción, no se la decodifica de nuevo.

- La interpretación → demanda menos espacio de memoria, es más flexible a la hora de corregir pero es más lento el proceso al traducir mientras se ejecuta. Además debe traducir cada instrucción repetida cada vez que aparece.

Lenguajes con procesamiento híbrido o mixto

- En base a los beneficios de cada forma de procesamiento de los lenguajes, han aparecido lenguajes que combinan ambas estrategias: aplican sobre el código fuente un proceso de compilación hasta obtener código intermedio y luego interpretan línea a línea dicho código.

Java es un caso particular ya que hace uso de una máquina virtual que se encarga de la traducción del código fuente por lo que a veces es denominado compilado e interpretado. El JDK (Java Development Kit) es el compilador de Java que lo traduce a bytecode y el JRE (Java Runtime Environment) es quien interpreta y ejecuta ese código intermedio.

Gramáticas atribuidas para análisis semántico

ALGO MAS SOBRE EL ANALISIS SEMANTICO

- Las gramáticas con atributos (GA) o gramáticas atribuidas deben su nombre a que se apoyan en la **asignación de atributos a las distintas construcciones sintácticas** de un lenguaje.

- **Son gramáticas independientes de contexto**, a las que se les incorporan:
  - atributos o propiedades a sus símbolos terminales y no terminales,
  - reglas para su evaluación y
  - condiciones que éstas deben cumplir.

- La muy estrecha relación con la gramática hace que el análisis semántico que se desprende de este enfoque sea denominado dirigido por la sintaxis

Gramáticas atribuidas para análisis semántico

- Los **atributos** son variables que representan determinadas propiedades de los símbolos terminales y no terminales.

- A cada regla de producción RP (regla sintáctica BNF) se le asocia un número finito de **reglas semánticas RS** que especifican la forma en que se modifican los atributos con la aplicación de la regla sintáctica.

- Las **condiciones C** están asociadas a cada una de las reglas de producción RP. Estas condiciones deben ser cumplidas por los valores de los atributos.
  - *Una sentencia sintácticamente correcta también lo será semánticamente, si y solo si, todos los atributos satisfacen las condiciones C del contexto*

Gramáticas atribuidas para análisis semántico

- Algunos de los ejemplos típicos de atributos en una gramática son:
  - el tipo de una variable,
  - su valor,
  - la dirección asignada de memoria,
  - el número de argumentos de una función,
  - los tipos de estos argumentos,
  - .....

Gramáticas atribuidas para análisis semántico

- Por ejemplo, se citan las siguientes reglas semánticas que corresponden a sentencias ejecutables:

REGLA SINTÁCTICA BNF	ACCIÓN SEMÁNTICA
A := B + C	A.valor = B.valor + C.valor
	A.tipo = mayor_tipo(B.tipo, C.tipo)
A := B * C	A.valor = B.valor * C.valor
	A.tipo = mayor_tipo(B.tipo, C.tipo)

Gramáticas atribuidas para análisis semántico

- Ejemplo de condiciones
- Las condiciones que corresponden a los operadores de división para el caso del lenguaje Pascal, que como se sabe son diferentes según se trate de una división de enteros o de números reales:

REGLA SINTÁCTICA BNF	CONDICIÓN SEMÁNTICA
A := B / C	C.valor > 0
	A.tipo, B.tipo = real, C.tipo = real o entero
A := B div C	C.valor > 0
	A.tipo, B.tipo, C.tipo = entero

Gramáticas atribuidas para análisis semántico

- Ejemplo:
- Si la GLC tiene las siguientes reglas de producción
  - <ASIGNACION> := <iden> = <EXPRESION>
  - <EXPRESION>:= la vista en teoría

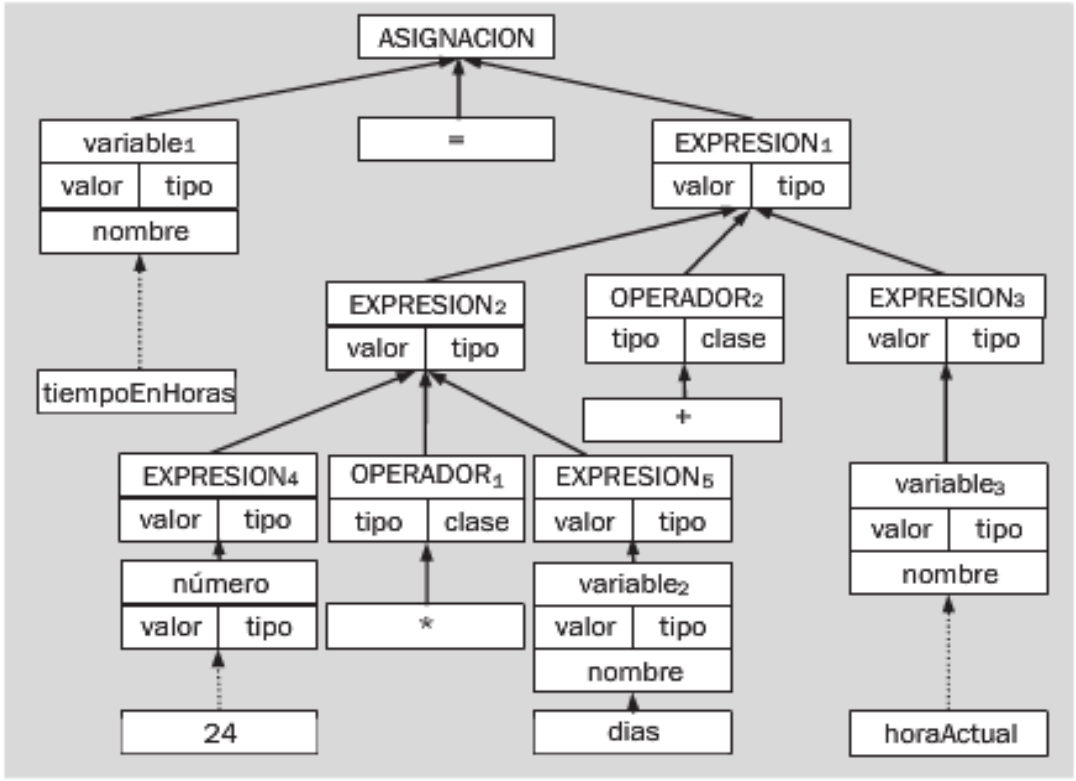
- Con la RS agregada quedaría:
  - <ASIGNACION> := <iden> = <EXPRESION>
  - { iden.valor = <EXPRESION>.valor
  - iden.tipo = <EXPRESION>.tipo }
  - ……..

- El analizador sintáctico verifica que la frase en la que intervienen esos tokens recibidos, esté correctamente escrita de acuerdo a la gramática, obteniendo un árbol derivación sintáctico.
- Este árbol es tomado por el analizador semántico que determinará, para cada símbolo, el valor de sus atributos. A partir de esta información, se aplican las reglas semánticas definidas con anterioridad.

Gramáticas atribuidas para análisis semántico

Ejemplo:

```
tiempoEnHoras = 24 * dias + horaActual,
```



: Árbol de derivación sintáctica con atributos.

A este árbol con atributos se le aplican las RS y las condiciones y se evalúa semánticamente.