UNIVERSIDAD NACIONAL GUILLERMO BROWN

Conceptos y Paradigmas de Lenguajes de Programación

Clase 4

SINTAXIS y SEMÁNTICA

SEMÁNTICA

UNIVERSIDAD NACIONAL GUILLERMO BROWN

SEMÁNTICA

La ***semántica*** describe el significado de los símbolos, palabras y frases de un lenguaje ya sea lenguaje natural o lenguaje informático

- **Ejemplos:**
  - int vector [10];
  - if (a<b) max=a; else max=b;
- **Tipos de semántica**
  - Estática
  - Dinámica

Semántica estática

- No está relacionado con el significado del programa, está relacionado con las formas validas.
- Se las llama así porque el análisis para el chequeo puede hacerse en compilación.
- Para describir la sintaxis y la semántica estática formalmente sirven las denominadas gramáticas de atributos, inventadas por Knuth en 1968.
- Generalmente las gramáticas sensibles al contexto resuelven los aspectos de la semántica estática.

Semántica estática - Gramática de atributos

- A las construcciones del lenguaje se le asocia información a través de los llamados "**atributos**" asociados a los símbolos de la gramática correspondiente
- Los valores de los atributos se calculan mediante las llamadas "**ecuaciones o reglas semánticas**" asociadas a las producciones gramaticales.
- La evaluación de las reglas semánticas puede:
  - Generar Código.
  - Insertar información en la Tabla de Símbolos.
  - Realizar el Chequeo Semántico.
  - Dar mensajes de error, etc.

Semántica estática - Gramática de atributos

Los atributos están directamente relacionados con los símbolos gramaticales (terminales y no terminales)

La forma, general de expresar las gramáticas con atributos se escriben en forma tabular. Ej:

Regla gramatical	Reglas semánticas
Regla 1	Ecuaciones de atributo asociada
.	.
.	.
Regla n	Ecuaciones de atributo asociada

Semántica dinámica.

- Es la que describe el efecto de ejecutar las diferentes construcciones en el lenguaje de programación.
- Su efecto se describe durante la ejecución del programa.
- Los programas solo se pueden ejecutar si son correctos para la sintáxis y para la semántica estática.

¿Cómo se describe la semántica?

- No es fácil
- No existen herramientas estándar como en el caso de la sintáxis (diagramas sintácticos y BNF)
- Hay diferentes soluciones formales:
  - Semántica axiomática
  - Semántica denotacional

- Semántica operacional

Semántica axiomática

- Considera al programa como "**una máquina de estados**".
- La notación empleada es el "**cálculo de predicados**".
- Se desarrolló para probar la corrección de los programas.
- Los constructores de un lenguajes de programación es formalizan describiendo como su ejecución provoca un cambio de estado.

SEMÁNTICA

Semántica axiomática

- Un estado se describe con un predicado que describe los valores de las variables en ese estado
- Existe un **estado anterior** y un **estado posterior** a la ejecución del constructor.
- Cada sentencia se precede y se continúa con una expresión lógica que describe las restricciones relaciones entre los datos.
  - Precondición
  - Poscondición

Ejemplo: a/b

```
a | b
r   c
```

Semántica denotacional

- Se basa en la teoría de funciones recursivas
- Se diferencia de la axiomática por la forma que describe los estados, la axiomática lo describe a través de los predicados, la denotacional a través de funciones.
- Se define una correspondencia entre los constructores sintácticos y sus significados

Semántica Operacional

- El significado de un programa se describe mediante otro lenguaje de bajo nivel implementado sobre una máquina abstracta
- Los cambios que se producen en el estado de la máquina cuando se ejecuta una sentencia del lenguaje de programación definen su significado
- Es un método informal
- Es el más utilizado en los libros de texto
- PL/1 fue el primero que la utilizó

Semántica Operacional

Ejemplo:

Lenguajes	Máquina abstracta

for i := pri to ul do
begin
.....
end
	i := pri
	lazo if i > ul goto sal
	.....
	i := i +1
	goto lazo
	sal .....

PROCESAMIENTO DE UN LENGUAJE TRADUCCIÓN

- Las computadoras ejecutan lenguajes de bajo nivel llamado "lenguaje de máquina".
- Un poco de historia…
  - Programar en código de máquina
- Uso de código mnemotécnico (abreviatura con el propósito de la instrucción). "Lenguaje Ensamblador" y "Programa Ensamblador"
- Aparición de los "Lenguajes de alto nivel"

UNIVERSIDAD NACIONAL GUILLERMO BROWN

PROCESAMIENTO DE UN LENGUAJE INTERPRETACIÓN Y COMPILACIÓN

- ¿Cómo los programas escritos en lenguajes de alto nivel pueden ser ejecutados sobre una computadora cuyo lenguaje es muy diferente y de muy bajo nivel?.
- Alternativas de traducción:
  - **Interpretación**
  - **Compilación**

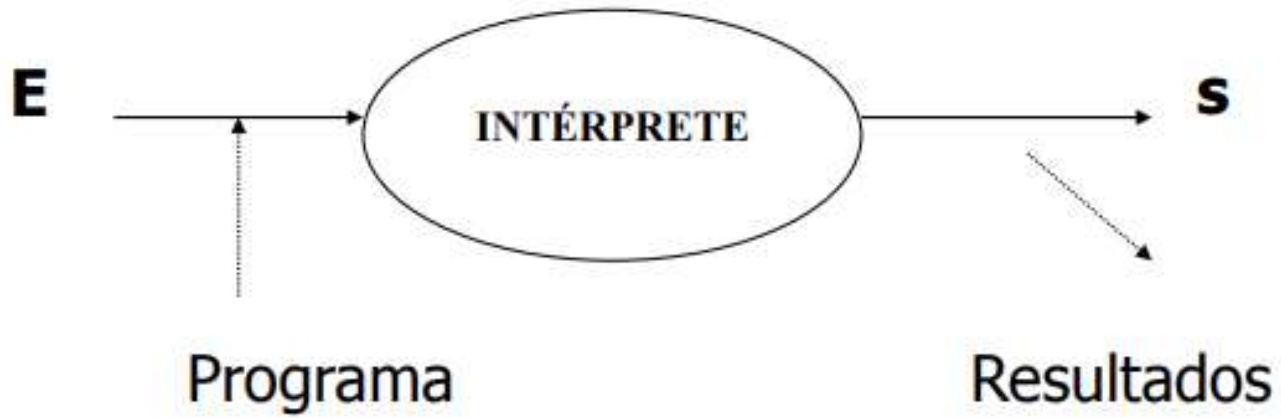
INTERPRETACIÓN

Intérprete:

- Lee,
- Analiza
- Decodifica y
- Ejecuta una a una las sentencias de un programa escrito en un lenguaje de programación.

- Ej: Lisp, Smalltalk, Basic, Python, etc.)
- Por cada posible acción hay un subprograma que ejecuta esa acción.
- La interpretación se realiza llamando a estos subprogramas en la secuencia adecuada.

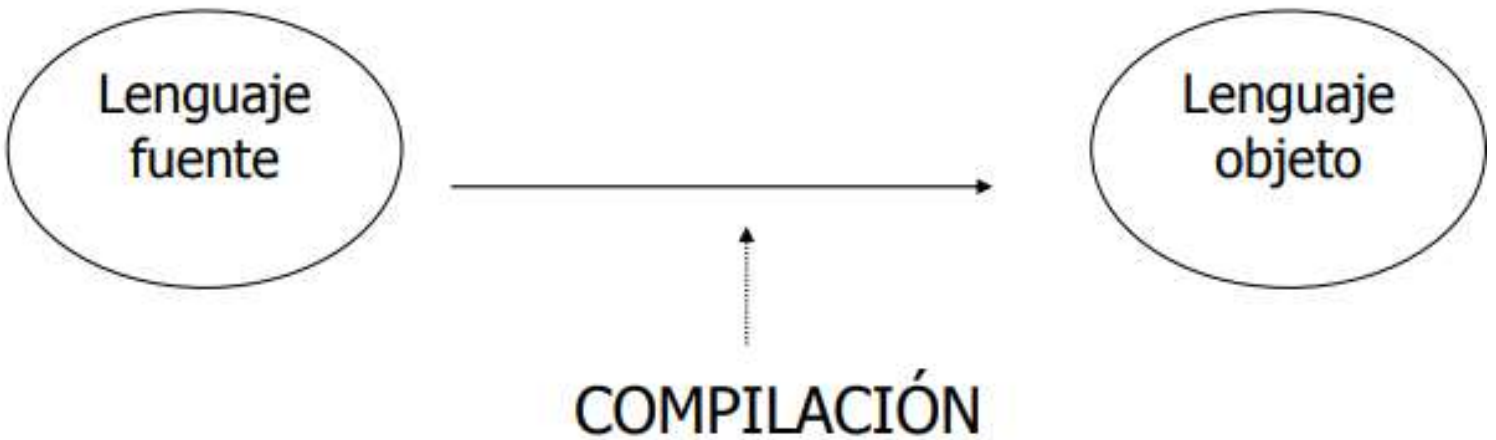
INTERPRETACIÓN



- Un intérprete ejecuta repetidamente la siguiente secuencia de acciones:
  - Obtiene la próxima sentencia
  - Determina la acción a ejecutar
  - Ejecuta la acción

COMPILACIÓN

Los programas escritos en un lenguaje de alto nivel se traducen a una versión en lenguaje de máquina antes de ser ejecutados.



TRADUCCIÓN

La compilación lleva varios pasos.

Ej: Pasos que prodría realizarse en una traducción:

- Compilado a assembler ← Compilador
- Ensamblado a código reubicable ← Assembler
- Linkeditado ← Link-editor
- Cargado en la memoria ← Loader

TRADUCCIÓN

Tipos de traductores:

- **Compilador**
  - *Lenguaje fuente*: Lenguaje de alto nivel
  - *Lenguaje objeto*: Cualquier lenguaje de máquina de una máquina real, o lenguaje assembler, o algún lenguaje cercano a ellos
- **Assembler**
  - *Lenguaje fuente*: Lenguaje assembler
  - *Lenguaje objeto:* Alguna variedad de lenguaje de máquina

TRADUCCIÓN

- **Link-editor**
  - *Lenguaje fuente*: Módulos en lenguaje de máquina en forma reubicable
  - *Lenguaje objeto*: Una simple unidad en forma reubicable con todos los módulos linkeditados juntos.
- **Loader**
  - *Lenguaje fuente*: Programa en forma reubicable con la tabla de datos
  - Lenguaje objeto: Lenguaje de máquina (código ejecutable)

TRADUCCIÓN

En ciertos lenguajes como C, se ejecuta antes del compilador otro traductor llamada **"Macro-Procesador o Pre Procesador"**

- **Macro:** fragmento de texto fuente que lleva un nombre
  - En el programa se utiliza el nombre de la macro
  - El nombre de la macro será reemplazada por su código cuando se procesen las macros

- Comparación entre > Traductor e Intérprete

  - **Forma en cómo ejecuta:**
    - ***Intérprete:***
      - Ejecuta el programa de entrada directamente
    - ***Compilador:***
      - Produce un programa equivalente en lenguaje objeto

  - **Forma en qué orden ejecuta:**
    - ***Intérprete:***
      - Sigue el orden lógico de ejecución
    - ***Compilador:***
      - Sigue el orden físico de las sentencias

UNIVERSIDAD NACIONAL GUILLERMO BROWN

TRADUCCIÓN

- **Tiempo de ejecución:**
  - ***Intérprete:***
    - Por cada sentencia se realiza el proceso de decodificación para determinar las operaciones a ejecutar y sus operandos.
    - Si la sentencia está en un proceso iterativo, se realizará la tarea tantas veces como sea requerido
    - La velocidad de proceso se puede ver afectada
  - ***Compilador:***
    - No repetir lazos, se decodifica una sola vez

- **Eficiencia:**
  - ***Intérprete:***
    - Más lento en ejecución
  - ***Compilador:***
    - Más rápido desde el punto de vista del hard

TRADUCCIÓN

- **Espacio ocupado:**
  - ***Intérprete:***
    - Ocupa menos espacio, cada sentencia se deja en la forma original
  - ***Compilador:***
    - Una sentencia puede ocupar cientos de sentencias de máquina

- **Detección de errores:**
  - ***Intérprete:***
    - Las sentencias del código fuente pueden ser relacionadas directamente con la que se esta ejecutando.
  - ***Compilador:***
    - Cualquier referencia al código fuente se pierde en el código objeto

TRADUCCIÓN

Combinación de ambas técnicas:

- **Los compiladores y los interpretes se diferencian en la forma que ellos reportan los errores de ejecución.**
- Algunos ambientes de programación contienen las dos versiones **interpretación y compilación**.
  - Utilizan el ***intérprete*** en la etapa de desarrollo, facilitando el diagnóstico de errores.
  - Luego que el programa ha sido validado se ***compila*** para generar código mas eficiente.

TRADUCCIÓN

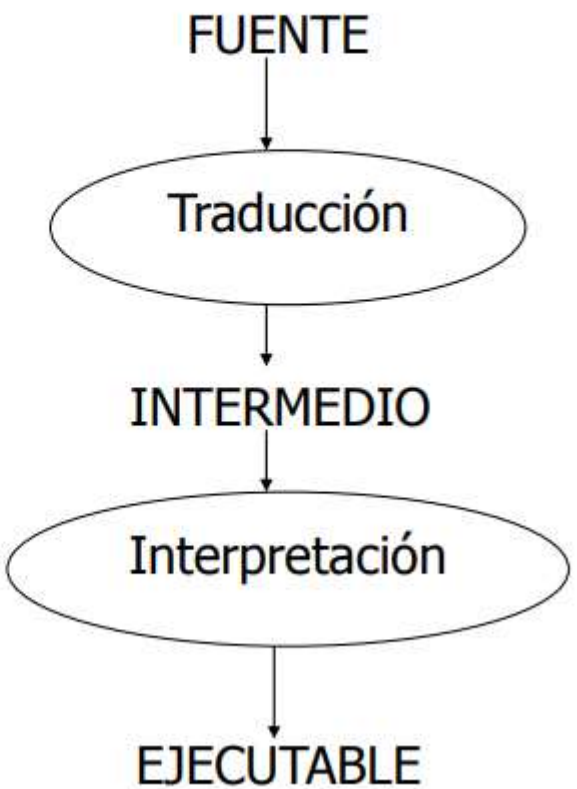
Combinación de ambas técnicas

Otro forma de combinarlos:

- **Traducción a un código intermedio que luego se interpretará.**
  - Sirve para generar **código portable**, es decir, código fácil de transferir a diferentes máquinas.
  - **Ejemplos:** Java, genera un código intermedio llamado "bytecodes", que luego es interpretado por la máquina cliente.

TRADUCCIÓN

- **Combinación de ambas técnicas:**



COMPILADORES

- Al compilar los programas la ejecución de los mismos es más rápida. Ej. de programas que se compilan: C Ada, Pascal, etc.
- Los compiladores pueden ejecutare en un solo paso o en dos pasos.
- En ambos casos cumplen con varias etapas, las principales son
  - **Análisis**
    - **Análisis léxico (Scanner)**
    - **Análisis sintáctico (Parser)**
    - **Análisis semántico (Semántica estática)**
  - **Síntesis**
    - **Optimización del código**
    - **Generación del código**

Generación de código intermedio

COMPILADORES

- **Análisis del programa fuente**
  - **Análisis léxico (Scanner):**
    - Es el que lleva mas tiempo
    - Hace el análisis a nivel de palabra
  - **Análisis sintáctico (Parser):**
    - El análisis se realiza a nivel de sentencia.
    - Se identifican las estructuras; sentencias, declaraciones, expresiones, etc. ayudándose con los tokens.
    - El analizador sintáctico se alterna con el análisis semántico. Usualmente se utilizan técnicas basadas en gramáticas formales.
    - Aplica una gramática para construir el árbol sintáctico del programa.

Análisis semántica (semántica estática):

- Es la fase medular
- Es la mas importante
- Las estructuras sintácticas reconocidas por el analizador sintáctico son procesadas y la estructura del código ejecutable toma forma.
- Se realiza la comprobación de tipos
- Se agrega la información implícita (variables no declaradas)
- Se agrega a la tabla de símbolos los descriptores de tipos, etc. a la vez que se hacen consultas para realizar comprobaciones.
- Se hacen las comprobaciones de nombres. Ej: toda variable debe estar declarada.
- Es el nexo entre el análisis y la síntesis

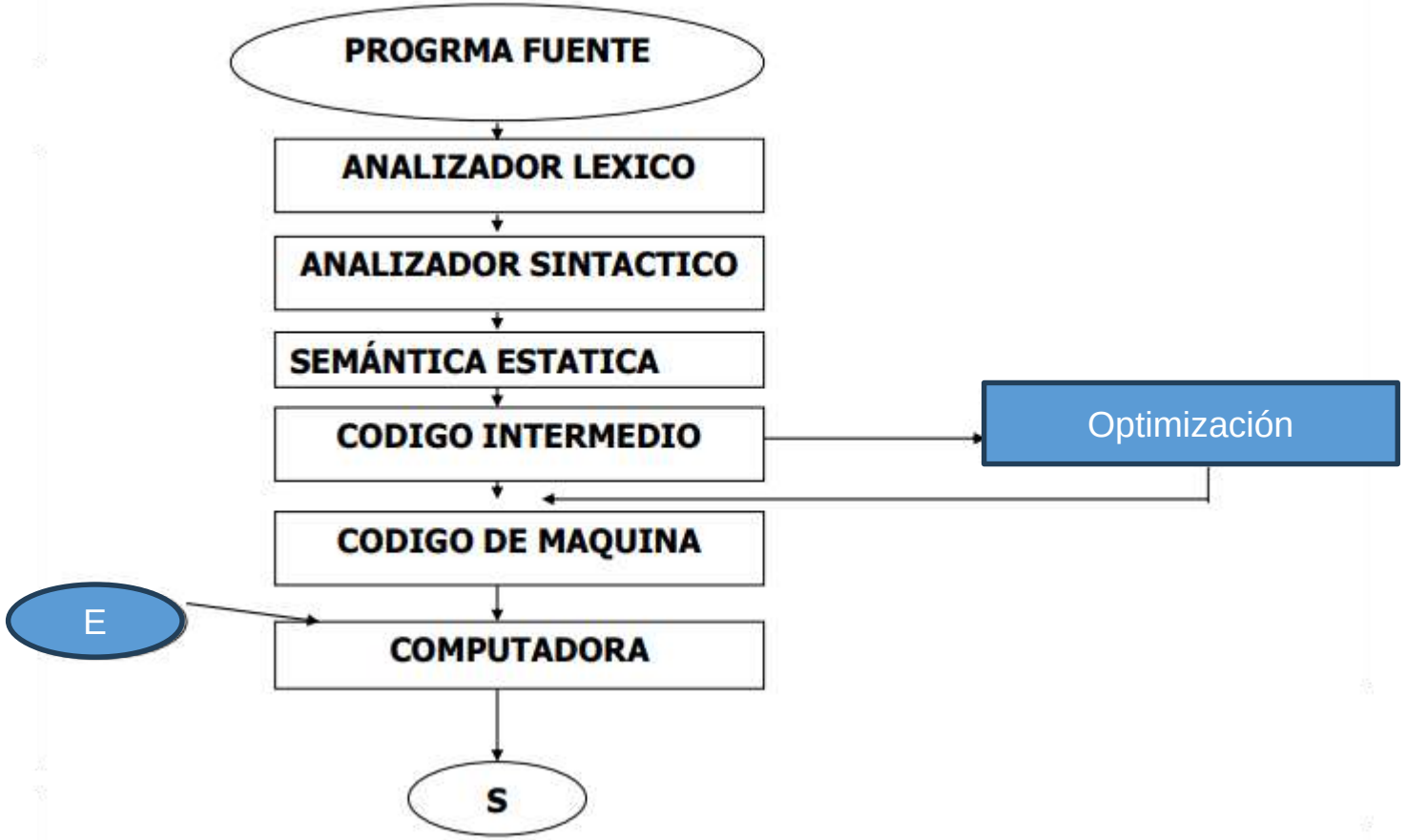
COMPILADORES

Generación de código intermedio:

- Características de esta representación
  - Debe ser fácil de producir
  - Debe ser fácil de traducir al programa objeto

- **Síntesis:**
  - En esta etapa se construye el programa ejecutable.
  - Se genera el código necesario y se optimiza el programa generado.
  - Si hay traducción separada de módulos, es en esta etapa cuando se linkedita.
  - Se realiza el proceso de optimización. Optativo

COMPILADORES



- Dudas - consultas??