Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

ANÁLISIS SEMÁNTICO

GRAMÁTICAS DE ATRIBUTOS Y TIPOS

Índice

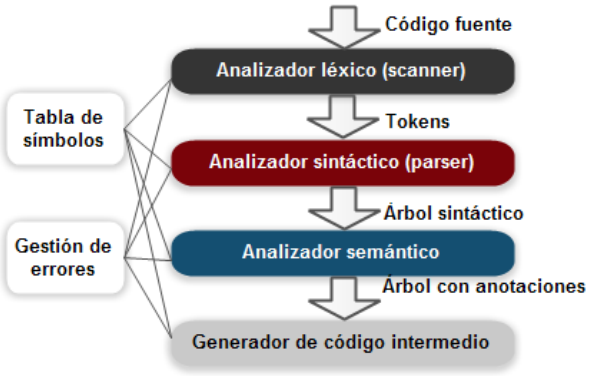
Presentación

El objetivo de este tema es comprender las tareas necesarias para realizar el análisis semántico.

Al finalizar este tema el estudiante será capaz de:

- Conocer el esquema conceptual para el análisis semántico.
- Conocer qué tipos de controles semánticos podemos llegar a realizar.
- Entender para qué se utilizan los atributos y las gramáticas de atributos.
- Aprender a utilizar las reglas semánticas.
- Entender las diferencias entre atributos sintetizados y atributos heredados.
- Comprender la necesidad de los grafos de dependencias.

Análisis semántico



Hemos aprendido a reconocer los **tokens** de un lenguaje y a partir de ellos verificar la estructura sintáctica de un programa construyendo un árbol sintáctico y ahora nos falta comprobar que estas sentencias sintácticamente correctas tienen validez semántica. Es decir, vamos a verificar que estas sentencias sintácticamente correctas tienen **coherencia en su contexto y por tanto tienen sentido.**

Hasta ahora hemos utilizado las **expresiones regulares** para el analizador léxico (gramáticas de tipo 3), y las **gramáticas independientes del contexto** (gramáticas de tipo 2) para el analizador sintáctico, pero con estas últimas no nos basta para hacer el análisis semántico. Es necesario definir unas gramáticas más ricas. Como son las **gramáticas de atributos.**

Una gramática de atributos, se basa en la identificación de los atributos que deben calcularse para realizar los controles semánticos. Estos controles semánticos se realizan utilizando **reglas semánticas** (también denominadas ecuaciones de atributos) basadas en los atributos definidos. Esto nos generará un **árbol con anotaciones,** es decir con valores en sus atributos y que será la entrada para el **generador de código intermedio.**

Para implementar esta gramática de atributos necesitamos apoyarnos en una estructura denominada **tabla de símbolos** y cuyo objetivo es asociar identificadores con sus propiedades o atributos.

Si cuando realizamos el proceso de validación semántica aparecen errores semánticos, es necesario poder comunicar al gestor de errores estos errores para actuar en consecuencia, dependiendo de la estrategia que se implemente.

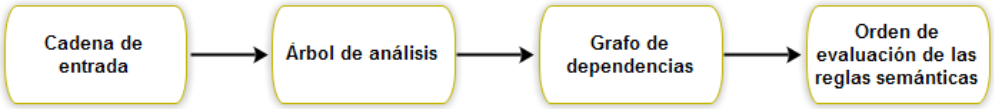
Atributos

Cada atributo es una variable que representa una propiedad de un símbolo (terminal o no terminal). Ejemplos de atributos que podemos necesitar son tipo (entero, real, etc.) o valor (25, 25,4, etc.).

Esquema conceptual

La forma de proceder conceptual es la siguiente (Aho at al, 1986):

1. Se analiza sintácticamente los componentes léxicos que constituyen la cadena de entrada.
2. Se construye el árbol de análisis sintáctico.
3. Se recorre el árbol para evaluar las reglas semánticas de cada uno de sus nodos.
4. La traducción de la cadena de componentes léxicos es el resultado obtenido al evaluar las reglas semánticas.



Aunque este es el esquema, lo lógico es ir construyendo el árbol y a la vez ir realizando los **controles semánticos.** De hecho, si el análisis semántico pudiera realizarse cuando está hecho todo el análisis sintáctico y por tanto se ha construido un árbol de sintaxis abstracta, entonces esta tarea de análisis semántico sería mucho más fácil, puesto que consistiría en la especificación del orden para el recorrido de dicho árbol, junto con los cálculos a realizar en cada nodo del recorrido. Esto implicaría dar varias pasadas al árbol de análisis (Louden, 2004).

Debemos ser conscientes de que el objetivo, por razones de eficiencia, es realizar todas las operaciones en una sola pasada (todas las fases, incluida la generación de código, suceden en una única lectura del código fuente). Para ello vamos a realizar las comprobaciones semánticas a la vez que se valida la estructura sintáctica, de tal forma que se generará un nodo si este es correcto tanto sintáctica como semánticamente. Es por ello que se añaden a las **reglas gramaticales** el cálculo necesario para establecer estos controles o comprobaciones semánticas.

El objetivo principal del analizador semántico de un procesador de lenguaje es asegurarse de que el programa analizado **satisfaga las reglas requeridas por la especificación del lenguaje,** para garantizar su correcta ejecución. El tipo y dimensión de análisis semántico requerido varía enormemente de un lenguaje a otro.

Lenguajes

En lenguajes interpretados como *Lisp* o *Smalltalk* casi no se lleva a cabo análisis semántico previo a su ejecución, mientras que en lenguajes como Ada, el analizador semántico deberá comprobar numerosas reglas que un programa fuente está obligado a satisfacer (Ortín Soler et al, 2004).

Tipos de controles semánticos

Para poder saber que atributos necesitamos, tenemos que identificar los tipos de controles semánticos a realizar. Estos pueden ser de dos tipos: estáticos y dinámicos.

Controles estáticos: los atributos pueden conocerse antes de la ejecución.

Comprobación de tipos	Los operadores y operandos deben ser compatibles, pudiendo llevar aparejados acciones adicionales para poder operar con ellos, como por ejemplo sumar un entero y un real.
Comprobaciones del flujo de control	Las proposiciones que hacen que se abandone el flujo de control de una estructura y este deba transferirse a otro punto. Ejemplo de esto son break y exit.
Comprobaciones de unicidad	Determinados objetos solo pueden definirse una vez. Un ejemplo son las etiquetas de una sentencia case y declaraciones de objetos.
Comprobaciones relacionadas con nombre	El mismo nombre debe aparecer dos veces, como por ejemplo en Ada, el nombre que aparece al inicio de un bloque, debe aparecer también al final.

Controles dinámicos: son las comprobaciones que se llevan a cabo en tiempo de ejecución.

Comprobaciones de asignación de variables	En un lenguaje como Lisp se hace de forma dinámica mientras que en Fortran77 se hace de forma estática.
Comprobaciones de rango	Por ejemplo los de array, donde los accesos que se hacen implican comprobar si se está dentro del rango del array o por ejemplo si en una operación de división el denominador vale 0.
Comprobaciones de referencias a punteros	El acceso a un puntero nulo, solo se conoce en tiempo de ejecución.

Como todas estas comprobaciones dependen de lenguaje y de la forma en la que se quieren realizar, no existen herramientas generales tipo flex o bison que permitan construir analizadores semánticos.

Estáticos y/o dinámicos

Dependiendo del lenguaje, unos controles pueden ser estáticos en un lenguaje y dinámicos en otros.

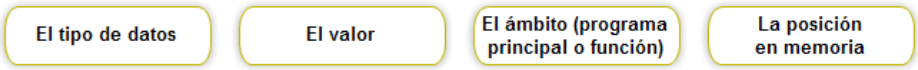
Controles estáticos

Son controles que se pueden realizar en tiempo de compilación.

Atributos y gramáticas de atributos

Como se ha indicado antes, un atributo es una variable que representa una propiedad de un símbolo (terminal o no terminal).

Ejemplos típicos de atributos son:



El proceso de calcular un atributo y asociar su valor calculado con la construcción del lenguaje en cuestión se define como fijación del atributo. El momento en el que se realiza este proceso de fijación, que puede ser compilación o ejecución se denomina **tiempo de fijación.**

Gramáticas de atributos

Las gramáticas independientes del contexto, las de tipo 2 de la jerarquía de Chomsky, son las que utilizamos para definir la sintaxis de un lenguaje de programación. De esta forma definimos la estructura que tendría cualquier programa escrito en ese lenguaje de programación elegido. Una vez hemos validado la estructura del programa, y para poder realizar determinados controles semánticos necesitamos utilizar información, representada por los atributos, que asociamos a los símbolos terminales y no terminales de la gramática que hemos establecido para ese lenguaje. Estas gramáticas se denominan gramáticas de atributos.

Cuando tenemos estas gramáticas de atributos decimos que somos capaces de hacer traducciones dirigidas por la sintaxis, puesto que el reconocimiento de la sintaxis de un programa (tanto si es descendente como ascendente el proceso de análisis) es la **guía** sobre la que los valores de los atributos van pasándose de un símbolo a otro de la gramática (tanto terminales como no terminales).

Para poder realizar este paso de valores de los atributos, para cada regla gramatical se establecen unas operaciones con estos atributos, y se denominan **reglas semánticas,** puesto que nos permiten realizar los controles semánticos que se han establecido para este lenguaje de programación.

Fijación

Los atributos que pueden fijarse antes de la ejecución, o lo que es lo mismo en tiempo de compilación, se denominan estáticos, mientras que los que pueden fijarse solo durante la ejecución se denominan dinámicos (Louden, 2004).

Reglas semánticas

La forma de denotar los atributos asociados a un símbolo es de la siguiente forma: si **X** es un símbolo gramatical y **a** es un atributo de este símbolo gramatical, escribimos **X.a** para indicar que el valor de **a** está asociado a este símbolo.

Atributo	Dominio (posibles valores)	Descripción
X.tipo	Entero, real, booleano, carácter, array, registro, puntero o función.	El tipo que tiene el símbolo no terminal X.
X. valor	Real	El valor que obtendrá el símbolo no terminal X.

En función de cómo se obtienen los atributos, estos pueden ser de dos tipos (posteriormente entraremos en más detalle):

1. **Sintetizados:** el valor que se asigna a un nodo depende del valor de los nodos hijos.
2. **Heredados:** el valor de un nodo depende del valor de los hermanos y del padre.

Las relaciones entre los valores que pueden tener los atributos de una gramática de atributos se especifican mediante reglas semánticas.

Como nos indica Louden (2004) las gramáticas de atributos se escriben en forma tabular, donde cada regla gramatical lleva asociada las reglas semánticas asociadas, y estas a su vez se especifican mediante las ecuaciones de atributo. La estructura es la siguiente:

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

Denotación de los atributos

Para cada atributo es necesario especificar el conjunto de valores que este puede tomar y a esto se le denomina **dominio del atributo,** así como una descripción del mismo (Ortín Soler et al, 2004).

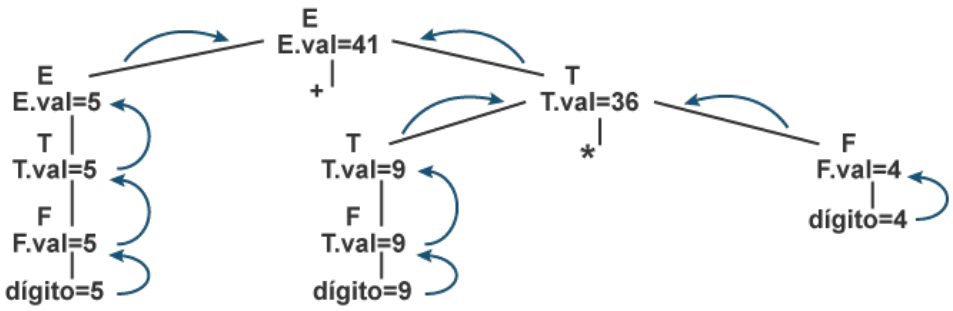
Reglas semánticas

También se denomina a las reglas semánticas ecuaciones de atributos, aunque una regla semántica puede estar constituida por una o varias ecuaciones de atributos.

Ejemplo de regla semántica

Considerando la gramática, la acción semántica que queremos realizar es obtener el valor de la siguiente operación: **5 + 9*4.** Por tanto el atributo que nos interesa es **valor.**

En primer lugar tenemos que especificar las reglas semánticas, donde hay que tener en cuenta que si un mismo símbolo aparece más de una vez en una regla gramatical, como por ejemplo E → E +T, donde vemos que E aparece más de una vez, debemos distinguirlo mediante subíndices, siendo por tanto E1 → E2 +T. Como se ha indicado el cálculo de este atributo se denomina traducción dirigida por la sintaxis, es por ello, que es necesario construir el árbol sintáctico para ver como van realizándose los cálculos con atributos, en este caso sintetizados puesto que se pasan los valores de nodos hijos a nodos padre:



El árbol de análisis sintáctico que muestra los valores de los atributos en cada nodo se denomina **árbol de análisis sintáctico con anotaciones** (otros autores lo denominan árbol sintáctico decorado).

En detalle Problemas de análisis semántico

Gramática

- E → E + T
- E → T
- T → T * F
- T → F
- F → dígito

Reglas semánticas

Regla gramatical	Reglas semánticas
$E_1 \rightarrow E2 + T$	E_1.valor = E_2.valor + T.valor
E → T	E.valor = T.valor
$T_1 \rightarrow T_2 * F$	T_1.valor = T_2.valor * F.valor
T → F	T.valor = F.valor
F → dígito	F.valor = dígito

Atributo = Valor

Atributo	Dominio (posibles valores)	Descripción
Valor	Entero	El valor que obtendrá los símbolos terminales y no terminales

En detalle

Problemas de análisis semántico

Con objeto de que le resulte más fácil al estudiante, en este tipo de problemas de análisis semánticos, es mejor construir primero el árbol y a partir de donde y como queramos que se hagan los cálculos, se deciden las reglas semánticas asociadas a cada regla gramatical.

Atributos sintetizados vs heredados

La **evaluación de los atributos basados en reglas** depende de la forma de recorrer el árbol de análisis. Dependiendo del tipo de atributo que escojamos **(sintetizados o heredados)** este recorrido del árbol será más eficiente, condicionando incluso la necesidad de realizar más de una pasada.

Atributos sintetizados: su valor se calcula únicamente a partir de los valores de los atributos (sintetizados/heredados) pertenecientes a sus hijos en el árbol de análisis. En el caso del ejemplo anterior, vemos como en un recorrido ascendente del árbol y por tanto de las reglas de la gramática que lo componen se van realizando los cálculos que se requieren.

Atributos heredados: su valor se calcula a partir de los valores de los atributos (sintetizados/heredados) pertenecientes al padre o a los hermanos de ese nodo en el árbol de análisis. A diferencia de los atributos sintetizados, el **orden** en el que se calculan los atributos heredados de los hijos es importante, debido a que pueden existir dependencias entre los atributos de los hijos. También se denomina a las gramáticas que utilizan **atributos heredados gramáticas de atributos por la izquierda o L-gramáticas,** en donde se tienen que dar las siguientes condiciones:

1ª condición	En una gramática de atributos por la izquierda, todo atributo heredado de cualquier producción depende solo de atributos heredados de símbolos por la izquierda.
2ª condición	Puede depender de atributos heredados del lado izquierdo de la producción. Es decir, si tenemos una producción A → αβ, los atributos heredados de αβ solo dependería de atributos heredados de A.
3ª condición	La información nunca va de derecha a izquierda.

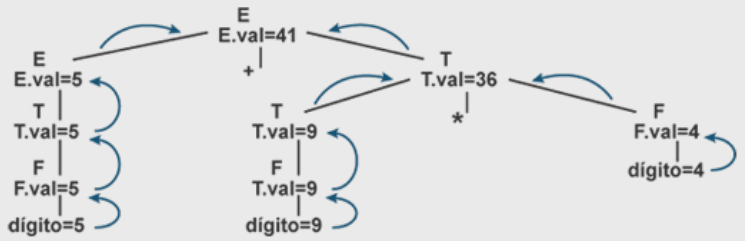
Aunque siempre es posible reescribir una gramática de atributos para que utilice solo atributos sintetizados, a veces es más natural utilizar atributos heredados.

Es importante destacar, que una vez se decide el tipo de atributo este se mantiene a lo largo de toda la gramática. Es decir, un atributo es sintetizado o heredado todo el tiempo y en todas las reglas semánticas en las que se implemente.

Atributos

La elección del tipo de atributo es importante desde el punto de vista de diseño del compilador.

Árbol de análisis



Una gramática en la que todos los atributos son sintetizados se conoce como gramática con atributos S (S-gramáticas).
En este caso el atributo valor va "subiendo" en el árbol, suponiendo un recorrido ascendente del mismo (análisis sintáctico ascendente), y se puede observar como en el nodo T àT * F, se obtiene el valor, en función de los nodos hijo.

Ejemplo con atributos heredados

Basándonos en la definición de atributos heredados, veremos un ejemplo en el que se distribuye la información sobre los tipos a los distintos identificadores de una declaración (Aho et al, 1986).

Supongamos el caso siguiente, en el que una declaración de identificadores incluye el tipo y la lista de identificadores, y donde el tipo puede ser entero o real y la lista de identificadores puede ser un identificador o varios unidos por comas, de tal forma que el tipo afecta a toda la lista. Es decir, queremos que el tipo que se defina lo herede toda la lista. La gramática que sintetiza este caso es la siguiente, y la cadena de entrada es **int a, b, c.**

Vamos a utilizar un procedimiento, denominado **insertatipo** para insertar el tipo de cada identificador en la tabla de símbolos para el atributo **nombre** de cada identificador.

La forma en la que se va crear el árbol de análisis es con un analizador sintáctico descendente, viendo por tanto en el árbol de análisis mediante las flechas discontinuas como se va produciendo la herencia del tipo para cada uno de los identificadores.

Podemos ver en al árbol como el atributo tipo es sintetizado y se pasa a L.her como heredado para que sea insertado en la tabla de símbolos.

NOTA: antes de insertar un atributo en la tabla de símbolos se debe validar si este existe previamente, en el ejemplo no se ha incorporado puesto que el objetivo era entender el funcionamiento de los atributos heredados

En detalle Funcionamiento de los atributos heredados

Identificadores

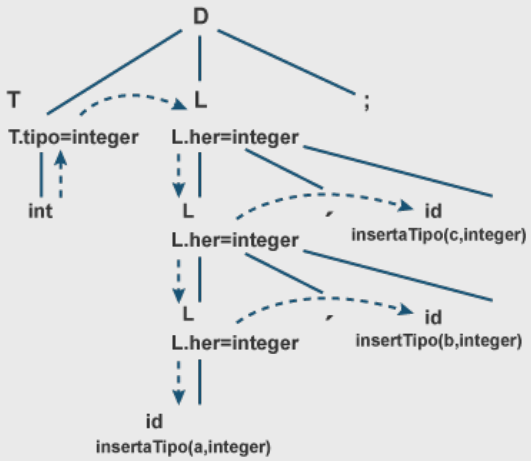
La gramática que sintetiza este caso es la siguiente y la cadena de entrada es **int a, b, c;**

- D → T L;
- T → int
- T → float
- L → L, id
- L → id

En detalle

Entender el funcionamiento de los atributos heredados

Regla gramatical	Regla semántica
D à T L	L.her=T.tipo
T à int	T.tipo=integer
T à float	T.tipor=real
L_1 à L_2, id	L_2.her=L_1.her insertatipo (id.nombre, L_2.her)
L à id	insertatipo (id.nombre, L.her)



Grafo de dependencias

En las gramáticas de atributos, estos no pueden evaluarse en cualquier orden, sobre todo cuando se mezclan atributos heredados y sintetizados. Con el objeto de que se pueda evaluar, de una forma visual, esta dependencia entre atributos de distinto tipo, es por lo que se utilizan los **grafos de dependencias.**

¿Cómo se construye el grafo de dependencias?

Cada producción, de la forma $A \rightarrow X_1 .. X_n$ define una parte del grafo que se construye a partir de una cadena de entrada concreta.

El proceso (para cada producción) es el siguiente:

1. Se crea un nodo por cada atributo (a) de cada símbolo de la producción (X), representándose los atributos heredados a la izquierda y los sintetizados a la derecha.
2. Para cada regla semántica se hacen arcos desde cada nodo hijo al nodo padre.

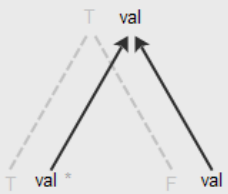
Ejemplo Grafo de dependencias

Documentos Cadena int a, b, c

Ejemplo

Grafo de dependencias

Regla gramatical	Reglas semánticas
T → T * F	T.val = T.val * F.val



Las líneas de puntos representan el árbol de análisis sintáctico y no son parte del grafo de dependencias, por eso aparecen difuminadas, aunque es necesario apoyarse en el árbol de análisis para entender el flujo de los atributos. En este caso en en el que solo hay un atributo y además es sintetizado, puesto que aparece a la derecha del símbolo gramatical, no sería necesario utilizar un grafo de dependencias. Es necesario cuando se mezclan los tipos de atributos, con el objeto de captar errores en el proceso de utilización de los atributos

Resumen

En este tema hemos comprendido las tareas necesarias para realizar un análisis semántico, entendiendo los pasos necesarios citados en el esquema conceptual.

Hemos aprendido a distinguir entre controles estáticos (tiempo de compilación) y controles dinámicos (tiempo de ejecución). Sabiendo en que consisten, de entre los controles estáticos podemos realizar comprobaciones: de tipos, del flujo de control, de unicidad o relacionados con el nombre. En cuanto a los controles dinámicos, las comprobaciones a realizar son: de asignación de variables, de rango o de referencias a punteros.

Por otro lado hemos visto varios tipos de atributos a tener en cuenta como son el tipo de datos, el valor, el ámbito o la posición en memoria, que junto con las gramáticas de atributos nos permite incorporar reglas semánticas basadas en estos atributos que se calculan según se va produciendo el proceso de análisis, permitiendo por tanto realizar en una sola pasada todas las fases del compilador.

También hemos aprendido a utilizar atributos sintetizados (paso de atributos de nodos hijo a nodos padre) y heredados (el valor de un nodo depende del valor de los hermanos y del padre) para realizar los controles semánticos y aplicarlos a las reglas semánticas.

Por último hemos aprendido a utilizar los grafos de dependencias cuando en una misma gramática tenemos atributos sintetizados y heredados, con el objeto de validar que el flujo de atributos es el correcto.