



Universidad
de Huelva

Tema 5

Análisis semántico

- 5.1 Características del análisis semántico
- 5.2 Gramáticas atribuidas
- 5.3 Especificación de un traductor
- 5.4 Traductores descendentes
- 5.5 Traductores ascendentes
- 5.6 Estructuras básicas

5.1 Características del análisis semántico

5.2 Gramáticas atribuidas

5.3 Especificación de un traductor

5.4 Traductores descendentes

5.5 Traductores ascendentes

5.6 Estructuras básicas

Objetivos:

- Verificar la semántica de la cadena de entrada:
 - Verificar que las variables han sido declaradas previamente
 - Comprobación de tipos en las expresiones
 - Comprobación de los parámetros de una función
 - Elección de la función u operador en caso de sobrecarga o polimorfismo
- Emitir informes de errores semánticos
- Construir el Árbol de Sintaxis Abstracta

Principio de la traducción dirigida por la sintaxis:

- El significado (semántica) de una frase está directamente relacionado con su estructura sintáctica

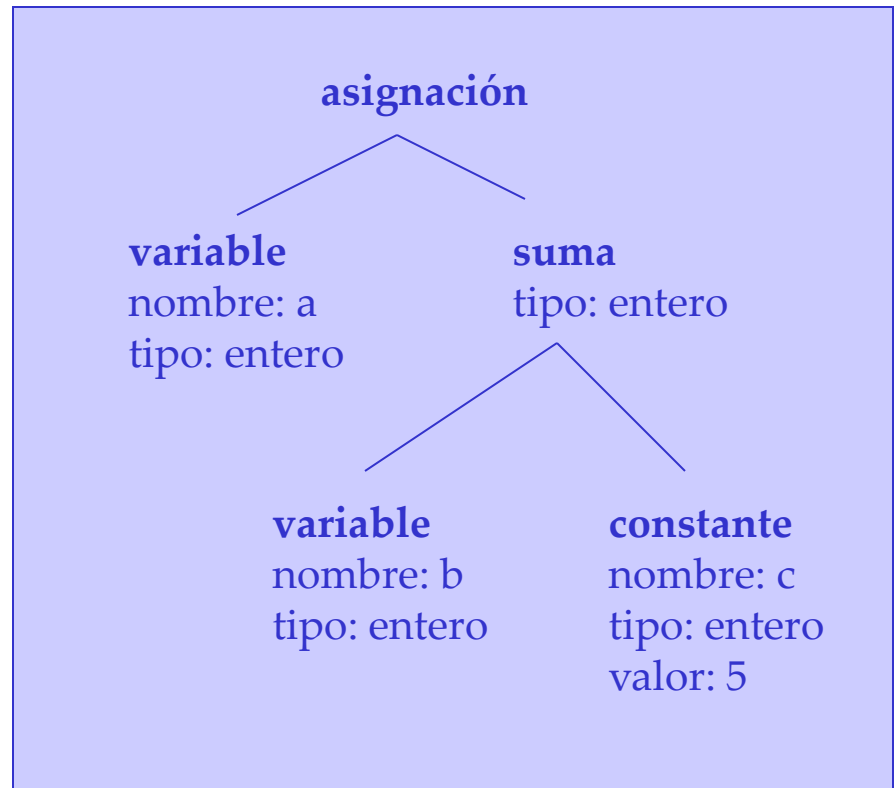
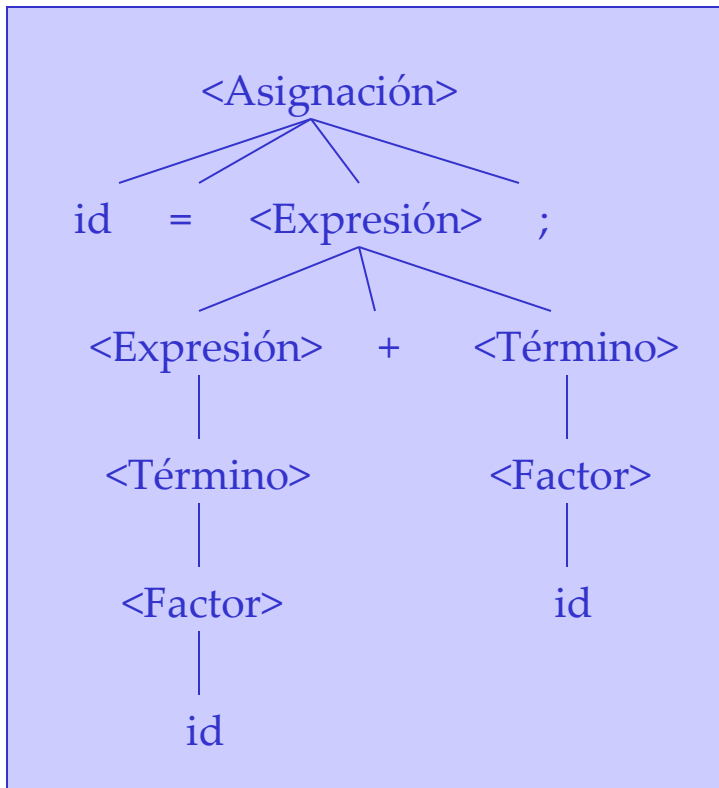
Especificación semántica

- Se inserta en la especificación sintáctica
- Se amplían las gramáticas libres de contexto:
 - Se asocia información a los símbolos (atributos)
 - Se asocian acciones semánticas (código) a las producciones

Árbol de Sintaxis Abstracta

- Recoge toda la información sintáctica y semántica
- Elimina los símbolos que no aportan significado
- Los nodos reflejan los datos asociados a cada símbolo
- El objetivo de las acciones semánticas es construir este árbol

Ejemplo: a = b + c ;



5.1 Características del análisis semántico

5.2 Gramáticas atribuidas

5.3 Especificación de un traductor

5.4 Traductores descendentes

5.5 Traductores ascendentes

5.6 Estructuras básicas

Atributos:

- Elementos de los símbolos del lenguaje que almacenan información semántica:
 - Tipo de datos
 - Referencia a las variables
 - Etc.

Acciones semánticas

- Expresiones que permiten calcular el valor de los atributos

Tipos de atributos:

- Atributos sintetizados:
 - Aquellos cuyo valor se calcula a partir de la información de los nodos hijos.
 - Representan un flujo de información ascendente en el árbol.
 - Ej.: $\langle A \rangle \rightarrow \langle B \rangle \langle C \rangle \langle D \rangle \{ A.a = B.b + C.c + D.d \}$
- Atributos heredados:
 - Aquellos cuyo valor se calcula a partir de la información del nodo padre o de los hermanos.
 - Representan un flujo de información descendente.
 - Ej.: $\langle A \rangle \rightarrow \langle B \rangle \langle C \rangle \langle D \rangle \{ C.c = A.a + B.b \}$

- Los atributos sintetizados de un símbolo representan una información que se calcula después de reconocer sintácticamente ese símbolo.
- Los atributos heredados de un símbolo representan una información que es necesario conocer antes de reconocer sintácticamente el símbolo.
- Generalmente se considera que los atributos de los tokens (símbolos terminales) son siempre sintetizados. Estos atributos son los generados por el analizador léxico.

Ejemplo (declaración de variables en C):

Atributos	Estructura
D.s, L.s, Lp.s	class Lista{ int tipo; String nombre; Lista sig; }
T.s, L.h, Lp.h	int

Producción	Regla semántica asociada
$\langle D \rangle \rightarrow \langle T \rangle \langle L \rangle$ pyc	{ L.h = T.s; D.s = L.s }
$\langle T \rangle \rightarrow$ float	{ T.s = REAL }
$\langle T \rangle \rightarrow$ int	{ T.s = ENTERO }
$\langle L \rangle \rightarrow$ id $\langle Lp \rangle$	{ Lp.h = L.h; L.s.tipo = L.h; L.s.nombre = id.lexema; L.s.sig = Lp.s }
$\langle Lp \rangle \rightarrow$ coma id $\langle Lp_1 \rangle$	{ Lp ₁ .h = Lp.h; Lp.s.tipo = Lp.h; Lp.s.nombre = id.lexema; Lp.s.sig = Lp ₁ .s }
$\langle Lp \rangle \rightarrow \lambda$	{ Lp.s = null }

5.1 Características del análisis semántico

5.2 Gramáticas atribuidas

5.3 Especificación de un traductor

5.4 Traductores descendentes

5.5 Traductores ascendentes

5.6 Estructuras básicas

Gramáticas con atributos por la izquierda

- Son un subconjunto de las gramáticas con atributos.
- Los atributos heredados se calculan exclusivamente a partir de los símbolos a la izquierda del atributo.
- Cualquier información heredada está disponible en el momento de ejecutar la reglas de producción correspondiente.
- Los flujos de información en el árbol de sintaxis abstracta se producen de izquierda a derecha, de arriba hacia abajo o de abajo hacia arriba. Nunca de derecha a izquierda.
- Permiten definir traductores de una sola pasada.

Esquemas de traducción dirigidos por la sintaxis (ETDS)

- Son un formalismo para construir traductores de una sola pasada
- Emplean gramáticas con atributos por la izquierda.
- Incluyen acciones semánticas encerradas entre llaves ({ ... }) en cualquier punto de las reglas de producción.
- Las acciones semánticas incluyen fragmentos de código en un lenguaje de programación.
- Las acciones semánticas pueden incluir instrucciones que no se refieran al cálculo de atributos.

5.1 Características del análisis semántico

5.2 Gramáticas atribuidas

5.3 Especificación de un traductor

5.4 Traductores descendentes

5.5 Traductores ascendentes

5.6 Estructuras básicas

Desarrollo de un ETDS basado en análisis descendente

- Los atributos sintetizados de cada símbolo S se almacenan en la estructura de datos (objeto) que devuelve la función asociada al símbolo S .
- Los atributos heredados de cada símbolo S se describen como parámetros de la función asociada al símbolo S .
- Las acciones semánticas incluidas en las reglas de producción de un símbolo S se añaden directamente al código de la función asociada al símbolo S .

Ejemplo de un ETDS basado en análisis descendente

- Reglas de producción
 - $\langle D \rangle \rightarrow \langle T \rangle \langle L \rangle$ **pyc**
 - $\langle T \rangle \rightarrow$ **float**
 - $\langle T \rangle \rightarrow$ **int**
 - $\langle L \rangle \rightarrow$ **id** $\langle L_p \rangle$
 - $\langle L_p \rangle \rightarrow$ **coma id** $\langle L_{p_1} \rangle$
 - $\langle L_p \rangle \rightarrow \lambda$
- Atributos
 - clase L { int tipo; String id; L siguiente; }
 - T.s (int) (sintetizado), L.h (int) (heredado), Lp.h (int heredado)
 - D.s (L) (sintetizado), L.s (L) (sintetizado), Lp.s (L) (sintetizado)

Ejemplo de un ETDS basado en análisis descendente

- Gramática atribuida

- $\langle D \rangle \rightarrow \langle T \rangle \{ L.h = T.s \} \langle L \rangle \text{ pyc } \{ D.s = L.s \}$
- $\langle T \rangle \rightarrow \text{float } \{ T.s = REAL \}$
- $\langle T \rangle \rightarrow \text{int } \{ T.s = ENTERO \}$
- $\langle L \rangle \rightarrow \{ L.s = \text{new } L(); \} \text{ id } \{ L..s.id = id; L.s.tipo = L.h; Lp.h = L.h \}$
 $\langle Lp \rangle \{ L..s.siguiete = Lp.s \}$
- $\langle Lp \rangle \rightarrow \{ Lp.s = \text{new } L(); \} \text{ coma id } \{ Lp.s.id = id; Lp.s.tipo = Lp.h; Lp_1.h = Lp.h \}$
 $\langle Lp_1 \rangle \{ Lp.s.siguiete = Lp_1.s \}$
- $\langle Lp \rangle \rightarrow \lambda \{ Lp.s = null \}$

Ejemplo de un ETDS basado en análisis descendente

- Función asociada a <D>

```
L parseD() {
    L D_s;
    switch(nextToken.getKind()) {
        case FLOAT:
        case INT:
            int T_s = parseT();    // las funciones devuelven objetos
            L L_s = parseL( T_s ); // los atributos heredados
            match(PYC);           // se pasan como argumentos
            D_s = L_s;
            break;
        default: error();
    }
    return D_s;
}
```

- Función asociada a <T>

```
int parseT() {  
    int T_s;  
    switch(nextToken.getKind()) {  
        case FLOAT:  
            match(FLOAT);  
            T_s = REAL;  
            break;  
        case INT:  
            match(INT);  
            T_s = ENTERO;  
            break;  
        default: error();  
    }  
    return T_s;  
}
```

- Función asociada a <L>

```
L parseL(int L_h) {
    L L_s;
    switch(nextToken.getKind()) {
        case ID:
            L_s = new L();
            Token tk = match(ID); // El analizador léxico devuelve objetos
            L_s.tipo = L_h; L_s.id = tk.lexema;
            L Lp_s = parseLp(L_h);
            L_s.siguiete = Lp_s;
            break;
        default: error();
    }
    return L_s;
}
```

- Función asociada a <Lp>

```
L parseLp(int Lp_h) {  
    L Lp_s;  
    switch(nextToken.getKind()) {  
        case COMA: Lp_s = new L();  
                    match(COMA);  
                    Token tk = match(ID);  
                    Lp_s.tipo = Lp_h; Lp_s.id = tk.lexema;  
                    L Lp1_s = parseLp(Lp_h);  
                    Lp_s.siguiete = Lp1_s;  
                    break;  
        case PYC: Lp_s = null; break;  
        default: error();  
    }  
    return Lp_s;  
}
```

5.1 Características del análisis semántico

5.2 Gramáticas atribuidas

5.3 Especificación de un traductor

5.4 Traductores descendentes

5.5 Traductores ascendentes

5.6 Estructuras básicas

Desarrollo de un ETDS basado en análisis ascendente

- A cada estado de la tabla se le puede asociar un símbolo (terminal o no terminal) que corresponde al símbolo que provoca la transición a ese estado.
- Los atributos sintetizados de cada símbolo S se almacenan en una estructura de datos (objeto).
- Se utiliza una pila de atributos, paralela a la pila de estados, para almacenar estas estructuras.
- Las acciones semánticas situadas al final de una regla se ejecutan al reducir la regla.

Desarrollo de un ETDS basado en análisis ascendente

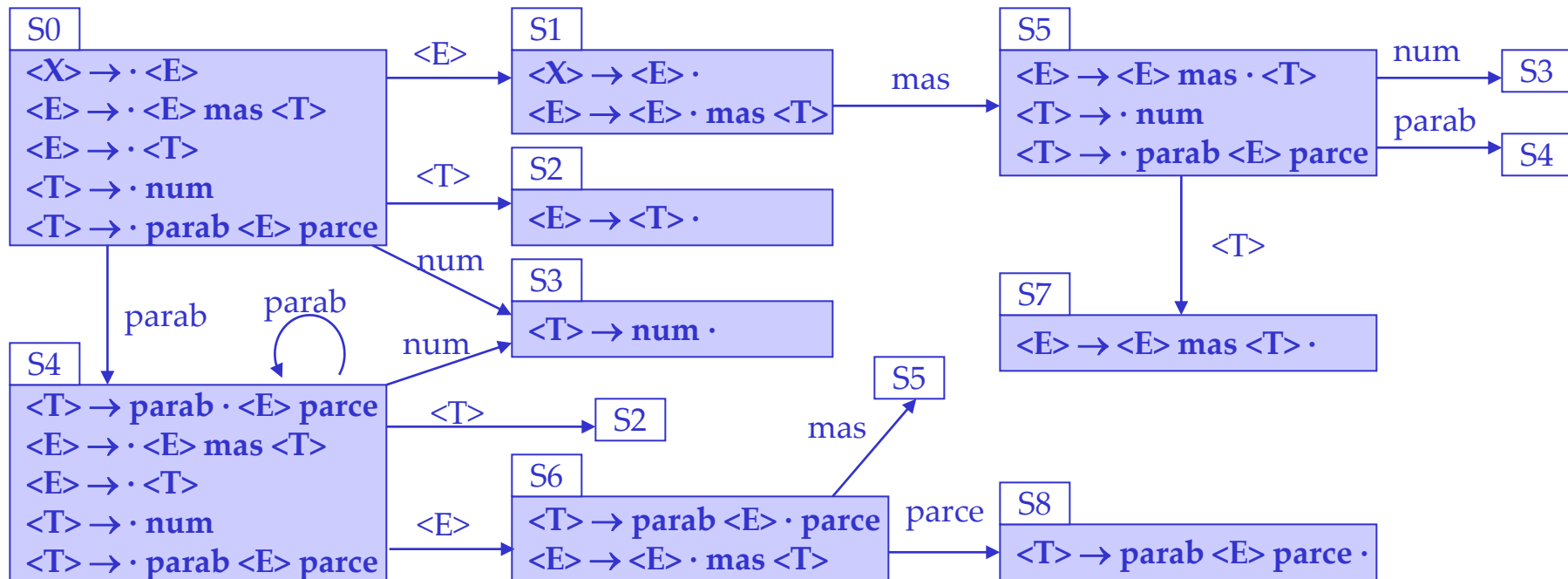
- Las acciones semánticas situadas en puntos intermedios se sustituyen por símbolos ficticios (marcadores) a los que se asocia una regla de producción λ seguida de la acción semántica.
- Conocida la longitud de una regla, se puede saber la posición de cada estructura de datos tomando como base la cima de la pila.
- Los atributos heredados de un símbolo no se pueden tratar directamente, sino que se introducen como atributos sintetizados del símbolo anterior.

Ejemplo de un ETDS basado en análisis ascendente

- Reglas de producción
 - $\langle E \rangle \rightarrow \langle E \rangle \text{ mas } \langle T \rangle$
 - $\langle E \rangle \rightarrow \langle T \rangle$
 - $\langle T \rangle \rightarrow \text{num}$
 - $\langle T \rangle \rightarrow \text{parab } \langle E \rangle \text{ parce}$
- Ejemplo
 - $7 + (8 + 2 + (3 + 1))$

Ejemplo de un ETDS basado en análisis ascendente

- Autómata Reconocedor de Prefijos Viables



Ejemplo de un ETDS basado en análisis ascendente

- Tabla de análisis SLR

Símbolo	Estado	mas	num	parab	parce	\$	<E>	<T>
-	0		d3	d4			1	2
<E>	1	d5				aceptar		
<T>	2	r2			r2	r2		
num	3	r3			r3	r3		
parab	4		d3	d4			6	2
mas	5		d3	d4				7
<E>	6	d5			d8			
<T>	7	r1			r1	r1		
parce	8	r4			r4	r4		

Ejemplo de un ETDS basado en análisis ascendente

- Traza de la cadena 7+5

Pila de atributos	Pila de estados	Entrada	Acción
-	0	num mas num \$	d3
- num	0 3	mas num \$	r3
- <T>	0 2	mas num \$	r2
- <E>	0 1	mas num \$	d5
- <E> mas	0 1 5	num \$	d3
- <E> mas num	0 1 5 3	\$	r3
- <E> mas <T>	0 1 5 7	\$	r1
- <E>	0 1	\$	aceptar

Ejemplo de un ETDS basado en análisis ascendente

- Gramática atribuida
 - $\langle E \rangle \rightarrow \langle E \rangle \text{ mas } \langle T \rangle \{ \$.op = SUMA; \$.izquierda = \$1; \$.derecha = \$3; \}$
 - $\langle E \rangle \rightarrow \langle T \rangle \{ \$.op = TERMINO; \$.izquierda = \$1; \$.derecha = NULL; \}$
- Notación (utilizada en YACC)
 - $\$$ es la estructura de datos de la parte izquierda de la regla.
 - $\$1 .. \N es la estructura de datos del símbolo 1 .. N de la parte derecha.
 - Para una regla de N símbolos, la estructura $\$M$ se encuentra en la posición N-M desde la cima de la pila.
 - No admite atributos heredados.

Ejemplo de un ETDS basado en análisis ascendente

- Notación
 - Las acciones semánticas situadas al final de una regla se ejecutan al reducir esa regla
 - Las acciones semánticas intermedias se sustituyen por nuevas reglas (marcadores). Por ejemplo:
 - $\langle E \rangle \rightarrow \langle E \rangle \{ \$$.izq = \$1; \} \text{ mas } \{ \$$.op = SUMA; \} \langle T \rangle \{ \$$.dcha = \$3; \}$
 - se transforma en:
 - $\langle E \rangle \rightarrow \langle E \rangle \langle M1 \rangle \text{ mas } \langle M2 \rangle \langle T \rangle \{ \$$.dcha = \$3; \}$
 - $\langle M1 \rangle \rightarrow \lambda \{ \$$.izq = \$1; \}$
 - $\langle M2 \rangle \rightarrow \lambda \{ \$$.op = SUMA; \}$

5.1 Características del análisis semántico

5.2 Gramáticas atribuidas

5.3 Especificación de un traductor

5.4 Traductores descendentes

5.5 Traductores ascendentes

5.6 Estructuras básicas

Generación de una estructura de lista



- Gramática LL(1) BNF
 - $\langle \text{Lista} \rangle \rightarrow \mathbf{id} \langle \text{SigueLista} \rangle$
 - $\langle \text{SigueLista} \rangle \rightarrow \mathbf{coma id} \langle \text{SigueLista} \rangle$
 - $\langle \text{SigueLista} \rangle \rightarrow \lambda$
- Gramática LL(1) EBNF
 - $\langle \text{Lista} \rangle \rightarrow \mathbf{id (coma id)^*}$
- Gramática LR(1)
 - $\langle \text{Lista} \rangle \rightarrow \mathbf{id}$
 - $\langle \text{Lista} \rangle \rightarrow \langle \text{Lista} \rangle \mathbf{coma id}$

Generación de una estructura de lista

- Gramática LL(1) BNF
 - $\langle \text{Lista} \rangle \rightarrow \mathbf{id} \langle \text{SigueLista} \rangle$
 $\{ e = \text{new Element}(), e.\text{id} = \text{id}, e.\text{next} = \text{SigueLista}.s, \text{Lista}.s = e \}$
 - $\langle \text{SigueLista} \rangle \rightarrow \mathbf{coma id} \langle \text{SigueLista}_1 \rangle$
 $\{ e = \text{new Element}(), e.\text{id} = \text{id}, e.\text{next} = \text{SigueLista}_1.s, \text{SigueLista}.s = e \}$
 - $\langle \text{SigueLista} \rangle \rightarrow \lambda \{ \text{SigueLista}.s = \text{null} \}$
- Gramática LL(1) EBNF
 - $\langle \text{Lista} \rangle \rightarrow \mathbf{id} \{ e = \text{new Element}(), e.\text{id} = \text{id}, e.\text{next} = \text{null}, \text{Lista}.s = e, \text{last} = e \}$
 $(\mathbf{coma id} \{ e = \text{new Element}(), e.\text{id} = \text{id}, e.\text{next} = \text{null} ,$
 $\text{last.next} = e, \text{last} = \text{last.next} \})^*$

Generación de una estructura de lista

• Gramática LR(1)

– $\langle \text{Lista} \rangle \rightarrow \mathbf{id}$

$\{ e = \text{new Element}(), e.\text{id} = \text{id}, e.\text{next} = \text{null}, \text{Lista}.s = e, \text{Lista}.\text{last} = e \}$

– $\langle \text{Lista} \rangle \rightarrow \langle \text{Lista}_1 \rangle \mathbf{coma id}$

$\{ e = \text{new Element}(), e.\text{id} = \text{id}, e.\text{next} = \text{null}, \text{Lista}.s = \text{Lista}_1.s,$
 $\text{Lista}_1.\text{last}.\text{next} = e, \text{Lista}.\text{last} = e \}$

• Gramática LR(1)

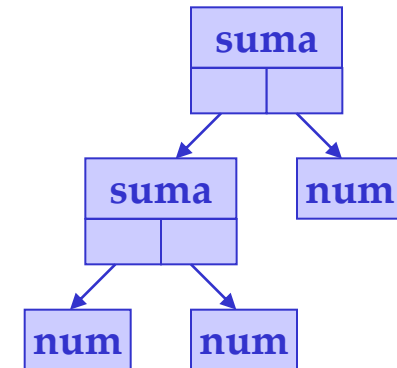
– $\langle \text{Lista} \rangle \rightarrow \mathbf{id} \{ e = \text{new Element}(), e.\text{id} = \text{id}, e.\text{next} = \text{null}, \text{Lista}.s = e \}$

– $\langle \text{Lista} \rangle \rightarrow \mathbf{id coma} \langle \text{Lista}_1 \rangle$

$\{ e = \text{new Element}(), e.\text{id} = \text{id}, e.\text{next} = \text{Lista}_1.s, \text{Lista}.s = e \}$

Generación de una estructura de árbol binario

- Gramática LL(1) BNF
 - $\langle \text{Árbol} \rangle \rightarrow \text{num } \langle \text{SigueÁrbol} \rangle$
 - $\langle \text{SigueÁrbol} \rangle \rightarrow \text{más num } \langle \text{SigueÁrbol} \rangle$
 - $\langle \text{SigueÁrbol} \rangle \rightarrow \lambda$
- Gramática LL(1) EBNF
 - $\langle \text{Árbol} \rangle \rightarrow \text{num } (\text{más num })^*$
- Gramática LR(1)
 - $\langle \text{Árbol} \rangle \rightarrow \text{num}$
 - $\langle \text{Árbol} \rangle \rightarrow \langle \text{Árbol}_1 \rangle \text{ más num}$



Generación de una estructura de árbol binario

- Gramática LL(1) BNF
 - $\langle \text{Árbol} \rangle \rightarrow \text{num} \{ \text{SigueÁrbol.h} = \text{num} \}$
 $\langle \text{SigueÁrbol} \rangle \{ \text{Árbol.s} = \text{SigueÁrbol.s} \}$
 - $\langle \text{SigueÁrbol} \rangle \rightarrow \text{más num} \{ \text{SigueÁrbol}_1.h = \text{new Suma}(\text{SigueÁrbol.h}, \text{num}) \}$
 $\langle \text{SigueÁrbol}_1 \rangle \{ \text{SigueÁrbol.s} = \text{SigueÁrbol}_1.s \}$
 - $\langle \text{SigueÁrbol} \rangle \rightarrow \lambda \{ \text{SigueÁrbol.s} = \text{SigueÁrbol.h} \}$
- Gramática LL(1) EBNF
 - $\langle \text{Árbol} \rangle \rightarrow \text{num} \{ \text{Árbol.s} = \text{num} \}$
 $(\text{más num} \{ \text{Árbol.s} = \text{new Suma}(\text{Árbol.s}, \text{num}) \})^*$

Generación de una estructura de árbol binario

- Gramática LR(1)
 - $\langle \text{Árbol} \rangle \rightarrow \text{num} \{ \text{Árbol.s} = \text{num} \}$
 - $\langle \text{Árbol} \rangle \rightarrow \langle \text{Árbol}_1 \rangle \text{ más num} \{ \text{Árbol.s} = \text{new Suma}(\text{Árbol}_1.\text{s}, \text{num}) \}$

Generación de una estructura de vector

```
class Vector { public Vector(); public add(Element e); }
```

- Gramática LL(1) BNF
 - $\langle \text{Lista} \rangle \rightarrow \mathbf{num} \langle \text{SigueLista} \rangle$
 $\{ v = \text{new Vector}(), v.\text{add}(\text{num}), \text{SigueLista}.h = v, \text{Lista}.s = v \}$
 - $\langle \text{SigueLista} \rangle \rightarrow \mathbf{más num}$
 $\{ \text{SigueLista}.h.\text{add}(\text{num}), \text{SigueLista}_1.h = \text{SigueLista}.h \}$
 $\langle \text{SigueLista}_1 \rangle$
 - $\langle \text{SigueLista} \rangle \rightarrow \lambda \{ \}$
- Gramática LL(1) EBNF
 - $\langle \text{Lista} \rangle \rightarrow \{ v = \text{new Vector}() \} \mathbf{num} \{ v.\text{add}(\text{num}) \}$
 $(\mathbf{más num} \{ v.\text{add}(\text{num}) \})^* \{ \text{Lista}.s = v \}$

Generación de una estructura de vector

```
class Vector { public Vector(); public add(Element e); }
```

- Gramática LR(1)
 - $\langle \text{Lista} \rangle \rightarrow \text{num} \{ v = \text{new Vector}(), v.\text{add}(\text{num}), \text{Lista.s} = v \}$
 - $\langle \text{Lista} \rangle \rightarrow \langle \text{Lista}_1 \rangle \text{ más num} \{ \text{Lista.s} = \text{Lista}_1.\text{s}, \text{Lista.s.add}(\text{num}) \}$