Análisis Semántico

- Introducción
- Semántica dirigida por sintaxis
  - Gramáticas de Atributos
  - Ejemplos
- Evaluación de gramáticas
  - Grafo de Dependencias
  - Métodos de evaluación con análisis sintáctico
    - Gramáticas SA y LA
    - TDS y ETD
    - Evaluación Descendente con Analizador LL
    - Evaluación Ascendente con Analizador LR

Análisis Semántico

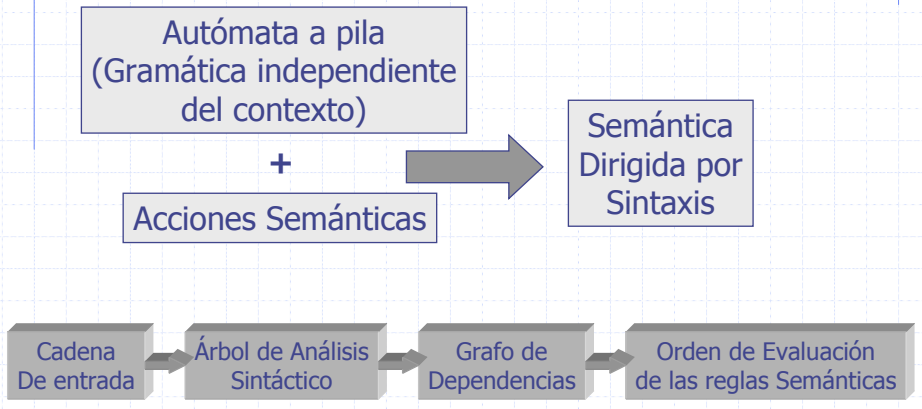
- Extensión del análisis sintáctico para la **comprensión** del programa
  - Comprobar que tiene sentido
  - Previo a su traducción
- Las gramáticas independientes del contexto (G2) no son suficientes para realizar el análisis semántico.
  - Comprobaciones de "larga distancia" en el árbol (contexto)
  - Es necesario definir un tipo de gramática más rica como las gramáticas de atributo: modelo de flujo de datos
- Esta fase modifica la tabla de símbolos y suele estar mezclada con la generación de código (traducción)
  - Destaca la verificación de tipos

Análisis Semántico

- Comprobaciones adicionales (estáticas)
  - Comprobación de tipos
    - La aplicación de los operadores y operandos deben ser compatibles
  - Comprobaciones de unicidad
    - Hay situaciones en los que un objeto solo puede definirse una vez exclusivamente. Las etiquetas de una sentencia case no deben repetirse, declaraciones de objetos,…
  - Comprobaciones relacionadas con nombres
    - El mismo nombre debe aparecer dos o más veces. Ej.: variables en funciones, …
  - Comprobaciones del flujo del control
    - Las proposiciones que hacen que se abandone el flujo del control de una construcción debe transferirse a otro punto. (*break, exit …*)

El Analizador Semántico

- Semántica dirigida por sintaxis



Semántica dirigida por sintaxis

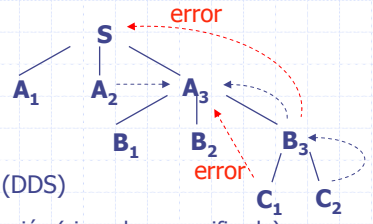
- Definición
  - Las gramáticas de atributo son gramáticas G2 a las que se añaden atributos y reglas de evaluación de atributos (reglas semánticas)
  - Cada atributo es una variable que representa una propiedad de un elemento del lenguaje
    - Habitualmente para cada símbolo X (terminal o no terminal)
    - Ej: X.Tipo, X.Valor, ...
    - Puede ser una cadena, número, tipo, posición de memoria, etc
- Reglas semánticas
  - Se asocian a las producciones sintácticas.
  - Ecuaciones de atributo (caso particular): sólo función de atributos de símbolos en la producción
  - Además existen condiciones semánticas que se ejecutan sobre estos atributos

Semántica dirigida por sintaxis

- Efectos de las acciones semánticas
  - Cálculo de valores de atributos
  - Guardar/Consultar información de la Tabla de Símbolos (variable global)
  - Generación de código
  - Notificación de mensajes de error
  - ...
- No es un proceso estándar como el análisis sintáctico
  - Varía de un lenguaje a otro
- Valores de atributos:
  - cada producción $A=\alpha$ se asocia con un conjunto de acciones semánticas representadas como una función:
    - $X.atr=f(Y_1.atr, ..., Y_n.atr)$
  - También se conocen estas acciones como "ecuaciones de atributos"

Gramáticas de Atributos

- Unión de gramática con atributos y reglas semánticas
  - Generar y transportar info: árboles "anotados" (o "adornados")
  - Las relaciones entre atributos de cada acción semántica solo entre símbolos de la producción



- Notaciones
  - Definición dirigida por la sintaxis (DDS)
    - Acciones asociadas a cada producción (sin orden especificado)
  - Esquema de Traducción (ETDS)
    - Acciones intercaladas en cada momento. Notación para implementar un traductor

Ejemplo 1

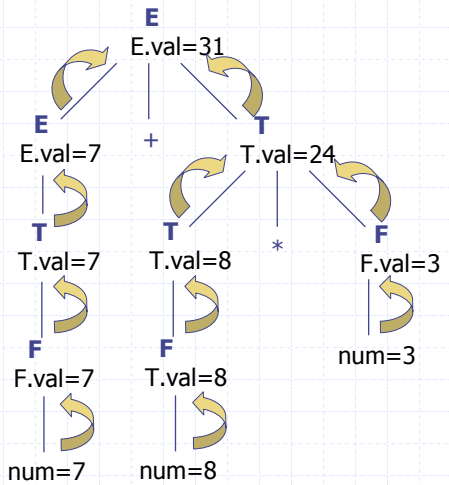
Evaluación de Expresiones numéricas

E::= E·+·T
E::= T
T::= T·*·F
T::= F
F::= num

1 atributo: val

Acc. sem.: op aritm.

Sentencia: 7+8*3



- Léxico: num + num * num
- Sintáctico: E+T→E+T*F
- Semántico: valor: 31

Ejemplo 1. DDS

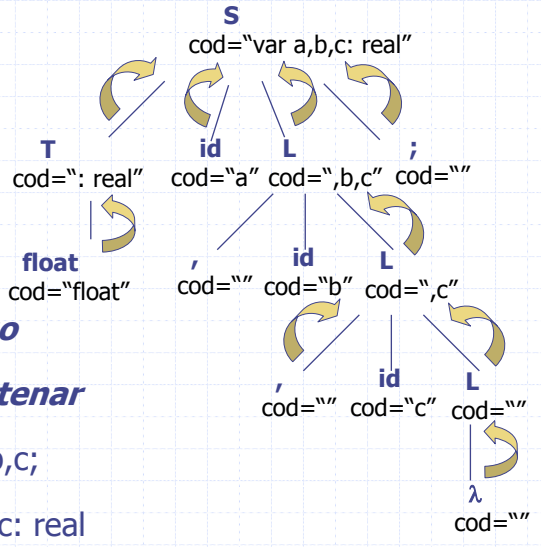
- Ejemplo (calculadora):

producción	Acciones semánticas
E::= E·+·T	E_0 .val =E_1 .val +T.val
E::= T	E.val =T.val
T::= T·*·F	T_0 .val =T_1 .val *F.val
T::= F	T.val =F.val
F::= num	F.val =num.val

Ejemplo 2

Traductor C->Pascal

S::=T id L ;
L::= , id L | λ
T::= float | int



1 atributo: código

Acc. sem.: concatenar

Sentencia: float a,b,c;

Resultado: var a,b,c: real

Ejemplo 2. DDS

- Ejemplo (traductor C->Pascal):

producción	Acciones semánticas
S::= T id L ;	S.cod =“var ” || id.cod || “:” || T.cod
L::= , id L	L_0.cod = “, “|| id.cod || L_1.cod
L::=λ	L.cod = “”
T::= float	T.cod = “real”
T::= int	F.cod = “int”

(|| representa concatenar)

Ejemplo 3

Valor numérico decimal/octal

E::=N base
N::= N D | D
base::=o | λ
D::= 0|1|…|9

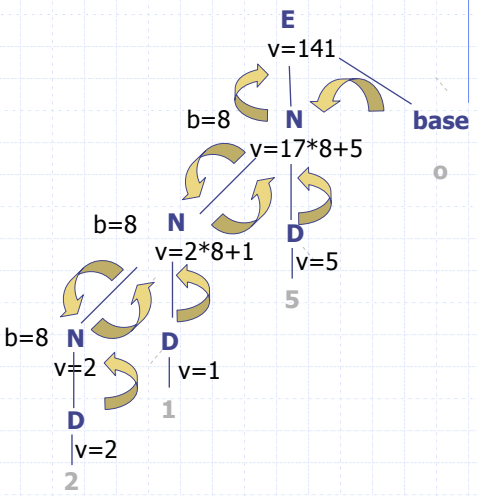
2 atributos:

base, valor

Acc. sem.: aritmética

Sentencia: 215o

Resultado: 141



Ejemplo 3. DDS

- Ejemplo (números enteros/octales):

producción	Acciones semánticas
E::= N base	E.base=base E.val=N.val
N::= N D	N_1.base=N_0.base N_0.val = N_1.val*N_1.base +D.val
N::= D	N.val = D.val
D::= "0"	D.val = 0
D::= "1"	D.val = 1
…	…
D::= "9"	D.val = 9

Ejemplo 4

Traductor infija->postfija

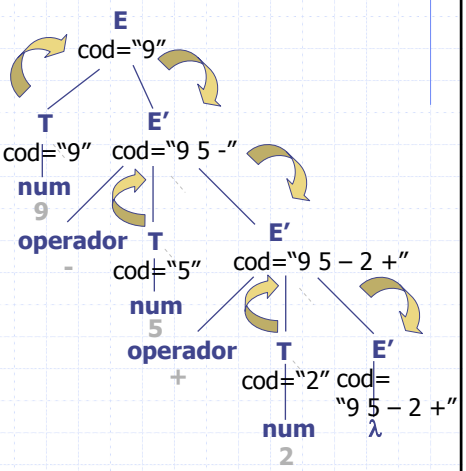
E::=T E'
E'::= op T E'
E'::=λ
T::= num

1 atributo: código

Acc. sem.: concatenar

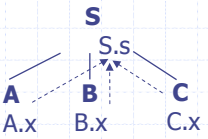
Sentencia: 9-5+2

Resultado: 9 5 – 2 +



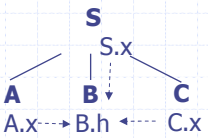
Gramáticas de Atributos. DDS

- Dos tipos de atributos
  - Sintetizados
    - El valor a asignar a un nodo depende del valor de los nodos hijos



S.s=f(A.x, B.x, C.x)

  - Heredados
    - Se pasan a niveles inferiores del árbol. Su valor depende del valor de los hermanos y del padre



B.h=g(A.x, B.x, C.x)

- El atributo mantiene el carácter en toda la gramática
- Los tokens sólo tienen atributos sintetizados

Evaluación de la gramática

- Evaluación de los atributos (anotación)
  - Métodos Dinámicos
    - Análisis con Grafo de Dependencias
    - El orden se obtiene de un ordenamiento topológico de los nodos, en función de las dependencias en el grafo
    - Se realiza en el momento de compilación: depende del programa
    - Si hay ciclos no funciona (análisis gramáticas circulares)
  - Métodos Estáticos
    - Orden de evaluación se decide en tiempo de construcción del compilador
    - Métodos basados en reglas, las acciones semánticas asociadas con las producciones se analizan a mano
    - Métodos de "una pasada": simultáneamente con el análisis sintáctico: No necesitan el árbol sintáctico

Grafos de Dependencias

- Los atributos no pueden evaluarse en cualquier orden
  - Si un atributo *b* depende de un atributo *c*, entonces se debe evaluar la regla semántica para *b* después de la regla semántica que define a *c*
- Las interdependencias entre atributos heredados y sintetizados de un árbol de análisis sintáctico se pueden representar mediante un grafo dirigido llamado **Grafo de Dependencias**

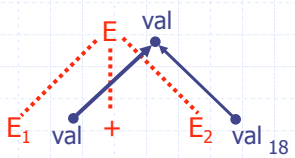
Grafos de Dependencias

- **Creación**: cada producción $A->X_1...X_n$ define una parte del grafo, se construye a partir de la cadena concreta

1. Se crea un nodo por cada atributo $X_i.a_j$ de cada símbolo de la producción
   - Los atributos heredados van a la izquierda y los sintetizados a la derecha
2. Para cada regla semántica $X_i.a_j=f(..., X_k.a_l,...)$ se hacen arcos desde cada nodo $X_k.a_l$ hacia el nodo $X_i.a_j$
   - Esto se repite para cada k y l afectados por la regla

- Ejemplo:

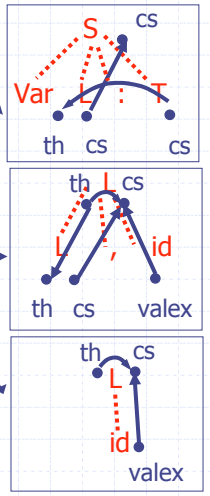
Producción	Regla Semántica
$E \rightarrow E+E$	$E_0.val:=E_1.val+E_2.val$



Grafos de Dependencias

- traductor decl Pascal->C:

producción	Acciones semánticas
S::= var L : T;	L.th=T.ts S.cs=L.cs
L::= L, id	L_0.cs = L_1.cs ||L_0.th||id.valex||";" L_1.th= L_0.th
L::=id	L.cs =L.th||id.valex||";"
T::= real	T.ts = "float"
T::= integer	T.ts = "int"

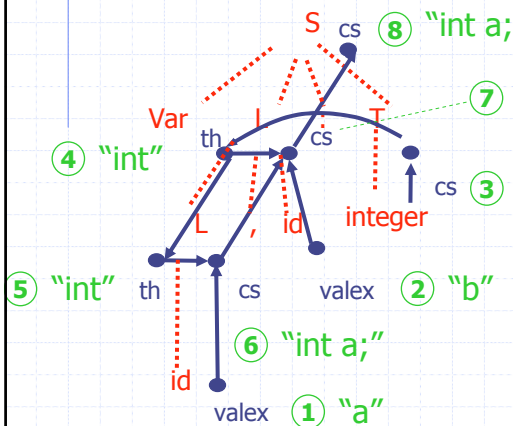


- Atributos:
  - S: S.cs (código sintetizado)
  - L: L.th (tipo heredado), L.cs (código sintetizado)
  - T: T.ts (tipo sintetizado)

Ejemplo Grafo de Dependencias

- traducción: var a,b: integer; ⇒ int a; int b;

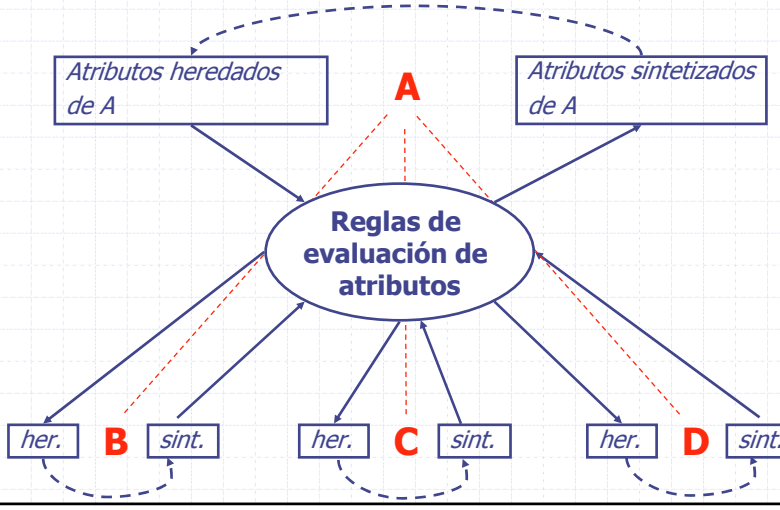
(aparecerá en atributo S.cs)



- Orden topológico de evaluación
  - En general puede haber varios órdenes topológicos

Evaluación

- Diseño estático del orden de recorrido

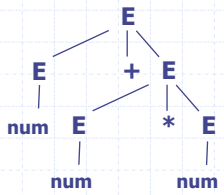


Creación del Árbol Sintáctico Abstracto (ASA)

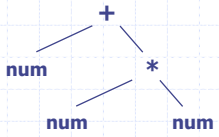
- Si no se puede evaluar la gramática al tiempo que se hace el análisis sintáctico
- Evaluación en varias pasadas:
  - construir el árbol explícitamente (con una DDS)
  - evaluación en función del orden de recorrido (grafo de dependencias)
- Árbol sintáctico abstracto
  - Estructura de datos que condensa un árbol de análisis sintáctico
  - Los operadores y las palabras clave
    - No son hojas
    - Están asociadas con el nodo padre de dichas hojas

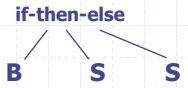
Creación del Árbol Sintáctico Abstracto (ASA)

- Árboles sintácticos (*parse tree*)



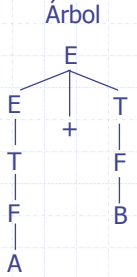
- Árboles de sintaxis abstractos (*sintax tree*)

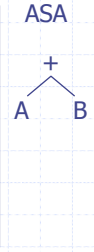


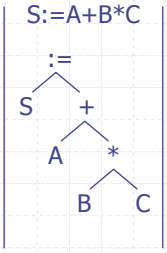


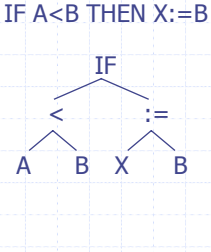
Árboles de Sintaxis Abstracta

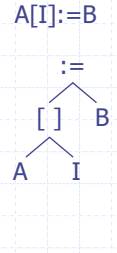
- Son árboles de derivación en los que no existe información superflua
- Cada nodo hoja representa un operando y cada no-hoja un operador
- Ejemplos











Orden de recorrido del árbol

- Orden dinámico: construcción del grafo de dependencias en tiempo de compilación y cálculo del orden topológico
- Orden estático: análisis de la gramática de las dependencias y orden de evaluación
  - Sobre el árbol de atributros
    - Recorrido pre-orden: heredados
    - Recorrido post-orden: sintetizados
    - Recorrido combinados: sintetizados depende heredados
    - Recorridos de varias visitas: heredados dependen sintetizados
  - Sin necesidad de árbol de atributos: gramáticas "de una pasada": S-A y L-A

Gramáticas Evaluables durante Análisis Sintáctico

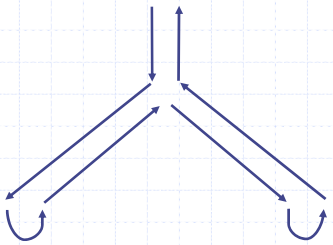
- Gramáticas de **Atributos Sintetizados** (*S-A Grammars*)
  - Sólo existen atributos sintetizados

    A.s, p:A->$X_1X_2...X_n$:
    - A.s=f(X_1.s, X_2.s, …, X_n.s)
  - Información fluye hacia arriba
- Gramáticas de **Atributos por la Izquierda** (*L-A Grammars*)
  - En una gramática de atributos por la izquierda, todo atributo heredado de cualquier producción:

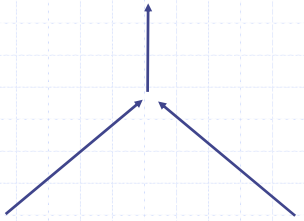
    X_i.h, en $1 \leq i \leq n$ en p:A->$X_1X_2...X_n$ depende sólo de:
    - 1. Atributos de símbolos por la izquierda: X_j, $1 \leq j < i$
    - 2. Atributos heredados de A
  - La información nunca fluye de derecha a izquierda
- Permiten evaluación durante el Análisis Sintáctico
  - Notación: Esquemas de Traducción

Gramáticas L-atribuídas, S-atribuídas

- Evaluación durante el Análisis Sintáctico
  - No se precisa un árbol explícito (toda la información necesaria en el nodo recorrido)



L-atribuída

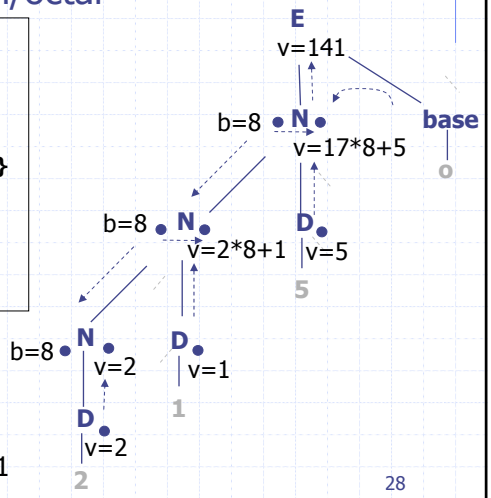


S-atribuída

Ejemplo gramática No L-Atribuída

Valor númerico decimal/octal

Producción	Acciones semánticas
E→ N base	{N.base=base.b E.val=N.val}
N→ N D	{N_1.base=N_0.base N_0.val = N_1.val*N_1.base +D.val}
N→ D	{N.val = D.val}
D→"0"	{D.val = 0}
D→"1"	{D.val = 1}
...	
D→ "9"	{D.val = 9}



Sentencia: 215o; Resultado: 141

Evaluación de gramáticas L-atribuídas, S-atribuídas en An. Sint.

- Gramáticas de **Atributos Sintetizados (S-A)**
  - Analizadores descendentes:
    - Las LL(1) pueden evaluarse con analizadores LL(1) y descenso recursivo
    - Si tienen recursividad a izquierda, puede extenderse la transformación de gramática a atributos, y utilizar un analizador descendente
  - Analizadores ascendentes
    - Las LL(1) y las LR(1) pueden evaluarse con analizadores LR(1). Uso de una pila semántica

Evaluación de gramáticas L-atribuídas, S-atribuídas en An. Sint.

- Gramáticas de **Atributos por la Izquierda (L-A)**
  - Analizadores descendentes
    - Pueden evaluar todas las LL(1) con atributos por la izquierda
    - Transformación de rec. izda. no válida con atributos heredados
  - Analizadores ascendentes
    - En principio no son adecuados para calcular atributos heredados (?)
    - Transformaciones posibles: inclusión de marcadores para insertar acciones intermedias y eliminación de atributos heredados
      - Válido para todos las gramáticas LL(1), y para algunas LR(1)
    - Alternativa: transformación de la gramática

Esquema de Traducción (ETDSs)

- Es una notación para hacer un traductor
- Las acciones semánticas se intercalan con los símbolos del consecuente de la producción
  - X::=ab {accion();} b
- Orden de evaluación fijo
- Dos tipos
  - ETDS sólo con atributos sintetizados
    - Acciones al final de la producción
  - ETDS con atributos sintetizados y heredados
    - Atributos heredados de un símbolo del consecuente
    - Atributos sintetizados del antecedente

Esquema de Traducción (ETDSs)

- Restricciones sobre un ETDS
  - Un atributo heredado para un símbolo en el lado derecho de una producción se calculará en una acción antes que dicho símbolo
    - $A \rightarrow \alpha_1 \; B \; \alpha_2$ {B.h=f(...)}
  - Una acción no debe referirse a un atributo sintetizado de un símbolo que esté a la derecha de ésta
    - $A \rightarrow \alpha_1 \alpha_2 \; \{\alpha_2.s = f(\alpha_3.s)\} \; \alpha_3$
  - Un atributo sintetizado para el no terminal de la izquierda sólo puede calcularse después de que se hayan calculado todos los atributos a los que hace referencia. (Típicamente al final)
    - $A \rightarrow \alpha_1 \alpha_2 \; \{A.s = f(\alpha_1.s, \alpha_2.s, \alpha_3.s)\} \; \alpha_3$

Ejemplo ETDS

El siguiente ETDS no cumple los criterios:

S::= AA	{A_1.her=1; A_2.her=2}
A::= a	{imprimir(A.her)}

Modificación:

S::=	{A_1.her=1;}
A	{A_2.her=1;}
A	
A::= a	{imprimir(A.her)}

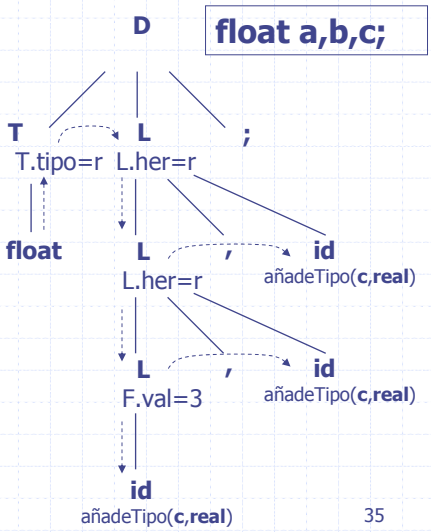
Ejemplo ETDS

- Ejemplo
  - Atributo heredado: información de tipo

Producción	Reglas semánticas
D→T	*{L.her := T.tipo}*
L	
T→**int**	*{T.tipo := integer}*
T→**float**	*{T.tipo := real}*
L→	*{L_1.her := L_0.her}*
L, id	*{añadetipo (id.lex, L.her)}*
L→**id**	*{añadetipo (**id**.lex, L.her)}*

Ejemplo ETDS

Ejemplo con atributos heredados: flujo horizontal y descendente



Evaluación Descendente de Gramáticas con Atributos por la Izquierda

- Con gramáticas L-atribuidas se puede efectuar la evaluación durante el análisis sintáctico: orden de evaluación de atributos es el de "creación" de nodos en el árbol sintáctico
- No se precisa crear árbol, se utiliza un recorrido recursivo como en el analizador descendente recursivo
- Evaluación: cada función asociada a cada no terminal recibe como argumentos los atributos heredados y devuelve los atributos sintetizados (es importante el orden)

```
Evaluar(Nodo: T, T.h): devuelve T.s
    Para cada hijo de T: Ci
        calcula atributos heredados Ci.h=g(Cj.h,Cj.s,T.h)
        Ci.s=evaluar(Ci, Ci.h)
    devolver T.s=f(Cj.h,Cj.s)
```

Evaluación Descendente

Eliminar recursión por la izquierda en ETDS

- Cada producción recursiva se elimina con la transformación:

A->AY → A->XA'
A->X → A'->YA'
A'->λ

- Extensión para su aplicación a **atributos sintetizados**

A->AY {A_0.a=g(A_1.a, Y.y)}
A->X {A.a=f(X.x)}

A->X {A'.h=f(X.x)}
A' {A.a=A'.s}
A'->Y {A'_1.h=g(A'_0.h,Y.y)}
A' {A'_0.s=A'_1.s}
A'->λ {A'.s=A'.h}

- A' tendrá un par de atributos (s,h) por cada sintetizado a original A.a

Evaluación Descendente

Ejemplo

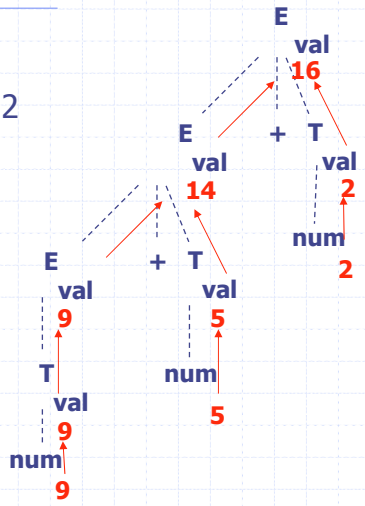
E::= E + T {E_0 .val =E_1 .val +T.val}
E::= T {E.val =T.val}
T::= num {T.val =num.val}

Transformación:

E::= T {R.h =T.val}
E' {E.valor=E'.s}
E'::= +
T {E'_1.h =T.val+ E'_0.h}
E' {E'_0.s = E'_1.h}
E'::= λ {E'.s =E'.h}
T::= num {T.val =num.val}

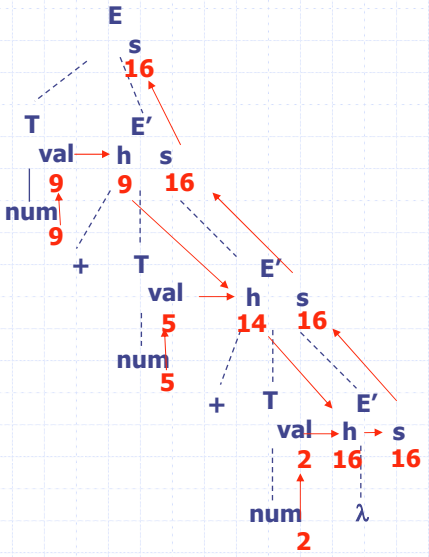
Gramática Original

9+5+2



Con Evaluación Descendente

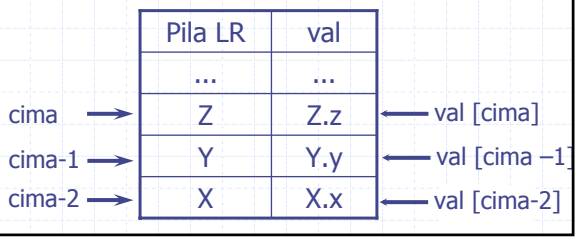
9+5+2



Evaluación Ascendente de Gramáticas con Atributos Sintetizados

- Evaluación muy frecuente (herramientas LALR:yacc/bison, jcup)
- Los atributos sintetizados se pueden evaluar con un analizador sintáctico ascendente según la entrada es analizada
- La pila de estados LR tiene asociados los símbolos gramaticales. Basta crear otra pila con los valores de los atributos sintetizados asociados a éstos, **"pila semántica": val**
- Con cada reducción se calculan los nuevos atributos sintetizados accediendo en la pila a los atributos de símbolos gramaticales del lado derecho de la producción

Producción	**Acciones semánticas**
A::= XYZ	A.s=f(X.x,Y.y,Z.z)



Evaluación Ascendente de Definiciones con Atributos Sintetizados

- Ejemplo:

Acciones ejecutadas tras cada reducción

Producción	**Fragmento de Código**
S→E **n**	*print (val [cima])*
$E \rightarrow E_1 + T$	*ncima=cima-2* *val* [*ncima*] := *val* [*cima*-2] + *val* [*cima*] *cima=ncima*
E→T	
$T \rightarrow T_1 * F$	*ncima=cima-2* *val* [*ncima*] := *val* [*cima*-2] × *val* [*cima*] *ncima=cima*
T→F	
F→(E)	*ncima=cima-2* *val* [*ncima*] := *val* [*cima*-1] *cima=ncima*
F→**dígito**	

Evaluación Ascendente de Gramáticas LA

- Todas las gramáticas con atributos por la izquierda anteriores eran LL(1). Se pueden hacer transformaciones y extender a analizadores ascendentes para cubrir muchas gramáticas LR(1) con atributos por la izquierda
  - Extensión con símbolos marcadores
    - Eliminación de acciones intercaladas
    - Herencia de atributos en la pila del analizador sintáctico (simulación de la evaluación de atributos heredados)

Evaluación Ascendente: Eliminación de acciones intercaladas en un ETDS

- Cuando no hay atributos heredados, la transformación es inmediata:
- $A \rightarrow X_1 \ldots X_{j-1}$ {accion} X_j ⇒ $\begin{cases} A \rightarrow X_1 \ldots X_{j-1} M X_j \\ M \rightarrow \lambda \ \{accion\} \end{cases}$

- Ej:

E::=T E′
E′::=+ T {*escribe +*} E′
E′::= - T {*escribe -*} E′
E′::=λ
T::= num {*escribe* ***num.lex***}

⇒

E::=T E′
E′::= + T M E′
E′::= - T N E′
E′::=λ
T::= num {*escribe* ***num.lex***}
M::=λ {*escribe* ***+***}
N::=λ {*escribe -* }

Evaluación Ascendente: Herencia de atributos heredados en la pila semántica

- Se introducen marcadores por cada símbolo con atributo heredado:

  $A \to X_1 \dots X_n$ ⇒ $A \to (M_1) X_1 \dots M_n X_n$ (si todo X_i tiene atrib heredado)

- Todo atributo heredado está en el sintetizado del marcador

  $A \to X_1 \dots X_{j-1}$ { $X_j.h = f(A.x, X_1.x, \dots, X_{j-1}.x)$ }
  X_j

  pasa a:

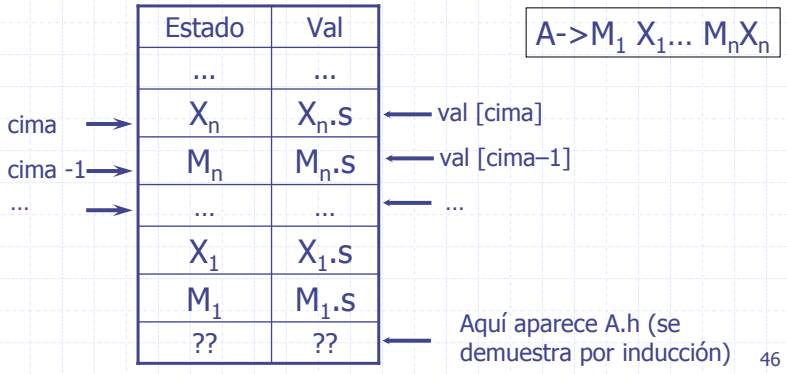
  $A \to X_1 \dots X_{j-1} M_j X_j$

  $M_j \to \lambda$ { $M_j.s = f(A.x, X_1.x, \dots, X_{j-1}.x)$ }

  - desaparecen las acciones con atributos heredados (su valor está en el atributo del marcador a la izquierda)
  - el cálculo de los nuevos marcadores es en realidad acceder a posiciones conocidas de la pila (cada marcador está en una sola regla)

Herencia de atributos heredados en pila semántica con analizador ascendente

- En la pila sólo hay atributos sintetizados
- Todos los atributos heredados están en posiciones predecibles en la pila (incluyendo el de la parte izquierda)



Ejemplo de herencia de atributos heredados en pila

- Ejemplo 2
  - Atributo heredado: información de tipo

Producción	Reglas semánticas
D→T M L	*{}*
M→ λ	*{tipo_act=T.tipo}* *OJO: no cumple DDS*
T→**int**	*{T.tipo := integer}*
T→**float**	*{T.tipo := real}*
L→ L, id	*{añadetipo (id.lex, L_0.her)}*
L→**id**	*{añadetipo (**id**.lex, L.her)}*

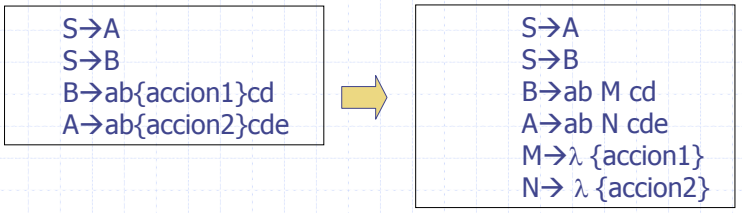
Ejemplo 2 de herencia de atributos heredados en pila

- Implementación con la "pila semántica" ***val***:

Reglas	**Código**
D::=TML	*{ncima=cima-2;* *cima=ncima;}*
M::=	*{ncima=cima+1;* *val[ncima]=val[cima];* *cima=ncima;}*
T::=int	*{val[cima]=integer;}*
T::=float	*{val[cima]=float;}*
L::=id	*{añadeTipo(id.lex,val[cima-1]);*
L::=L,id	*{ncima=cima-2* *añadeTipo(id.lex,val[cima-3]);* *cima=ncima;}*

Evaluación Ascendente de Gramáticas LA

- Dos limitaciones en los ETD y analizadores LR:
  - Hay que poder acceder a posiciones debajo de la pila
  - La inclusión de marcadores puede llevar a gramáticas no LR (1)



LALR(1) No LR(1)

Eliminación de atributos heredados con modificación de gramática

- (Knuth,1968): Toda gramática con atributos heredados se puede modificar a gramática con sintetizados
- Ej.: Gramática no L-atribuida (Pascal)

producción	**Acciones semánticas**
S::= var L : T;	L.th=T.ts
L::= L, id	L_1.th= L_0.th añadirTipo(id.valex, L_0.th)
L::=id	añadirTipo(id.valex, L.th)
T::= real	T.ts = "float"
T::= integer	T.ts = "int"

- Atributos:
  - L: L.th (tipo heredado), T: T.ts (tipo sintetizado)

Ejemplo modificación gramática

- gramática SA para hacer lo mismo (Pascal):

Producción	Acciones semánticas
S::= var L ;	
L::= id , L	añadirTipo(id.lex, L_1.ts) L_0.ts=L_1.ts
L::= id : T	añadirTipo(id.lex, T.ts) L.ts=T.ts
T::= real	T.ts = "float"
T::= integer	T.ts = "int"

- Atributos:
  - L: L.ts (tipo sintetizado), T: T.ts (tipo sintetizado)

Eliminación de atributos heredados con modificación de gramática

- Ej.: Gramática L-atribuida (C)

producción	Acciones semánticas
S::= T L;	L.th=T.ts
L::= L, id	L_1.th= L_0.th añadirTipo(id.valex, L_0.th)
L::=id	añadirTipo(id.valex, L.th)
T::= float	T.ts = "real"
T::= int	T.ts = "int"

- Atributos:
  - L: L.th (tipo heredado), T: T.ts (tipo sintetizado)

Versión gramática SA?

Ejemplo Análisis Semántico

Valor numérico decimal/octal

Gramática NO L-A

E::=N base
N::= N D | D
base::=o | λ

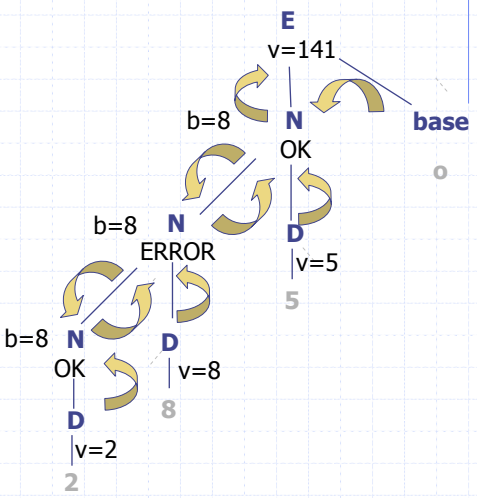
3 atributos:

base, valor, posicion

Acc. sem.: comprobar

Sentencia: 285o

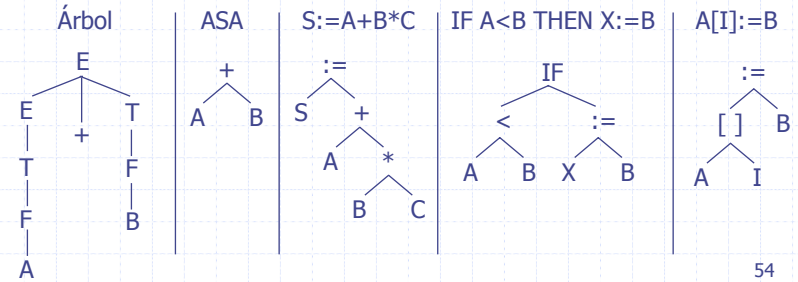
Resultado: error digito 2



Versión gramática SA?

Árboles de Sintaxis Abstracta

- Son árboles de derivación en los que no existe información superflua
- Cada nodo hoja representa un operando y cada no-hoja un operador
- Ejemplos



Creación del Árbol Sintáctico Abstracto (ASA)

- Construcción
  - Un nodo para cada operador y cada operando
  - Los hijos de un nodo operador son las raíces de los nodos que representan las subexpresiones que constituyen los operandos de dicho operador
- Funciones auxiliares
  - hazNodo (operador, izquierda, derecha)
  - hazHoja (id, entrada)
  - hazHoja (num, val)

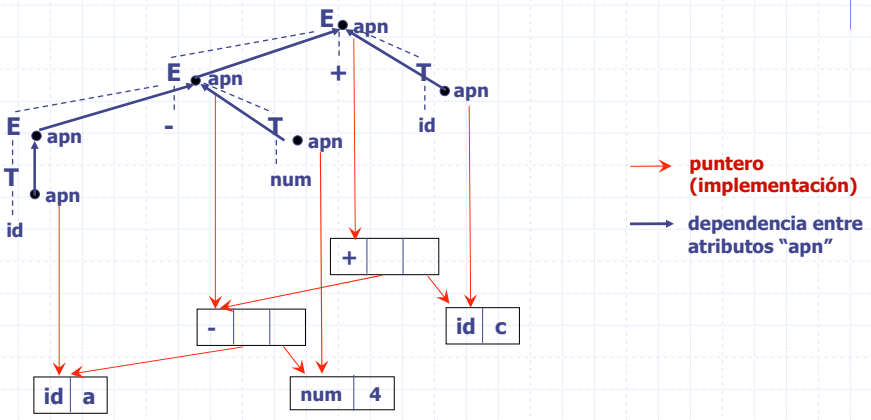
Creación del Árbol Sintáctico Abstracto (ASA)

- DDS para construir árboles sintácticos
  - Definición con atributos sintetizados para construir un árbol sintáctico para una expresión con operadores + y –
- Utiliza
  - Creación de nodos: HazNodo y HazHoja
  - Atributo sintetizado **apn**: conserva los punteros creados en llamadas a las funciones

PRODUCCIÓN	REGLAS SEMÁNTICAS
$E \rightarrow E_1 + T$	E.apn := hazNodo ('+', E_1.apn, T.apn)
$E \rightarrow E_1 - T$	E.apn := hazNodo ('-', E_1.apn, T.apn)
$E \rightarrow T$	E.apn := T.apn
$T \rightarrow (E)$	T.apn := E.apn
$T \rightarrow id$	T.apn := hazHoja (id, id.entrada)
$T \rightarrow num$	T.apn := hazHoja (num, num.val)

Creación del Árbol Sintáctico Abstracto (ASA)

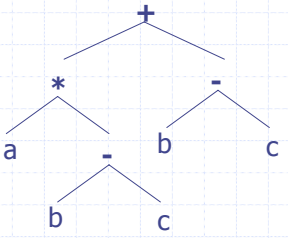
- Ej. (construcción ascendente): **a – 4 + c**



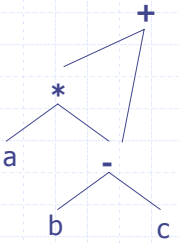
Grafos Acíclicos para expresiones

- Grafos de expresiones
  - Se identifican subexpresiones comunes (búsqueda en árbol)
  - Cada nueva ocurrencia de la subexpresión
    - No genera un nuevo subárbol
    - Referencia al subárbol ya existente o lo crea si no existe
- Ej.: a * (b – c) + (b –c)

árbol sintáctico



grafo acíclico



Utilización frecuente de atributos en compiladores convencionales

- Verificación de tipos: atributos sintetizados/heredados
- Traducción de expresiones aritméticas y booleanas: atributos sintetizados
- Traducción de estructuras de flujo de control: atributos heredados