Tema 6. Gramáticas de atributos

1. Gramáticas de atributos.
2. Orden de evaluación.
3. Definiciones S-atribuidas.
4. Definiciones L-atribuidas.

1. Introducción

Atributos y reglas semánticas

Especificación léxica --> Expresiones regulares

Especificación sintáctica --> Gramáticas independientes del contexto

Especificación semántica --> Gramáticas de atributos

- Asociamos atributos a los símbolos de la gramática.
- Los valores de los atributos se calculan mediante reglas semánticas asociadas a la reglas de producción.

Definición y Ejemplo 1

Sea G=(N,T,P,S) una GIC. Se define:

Conjunto de atributos de $X \in (N \cup \Sigma)$: $A(X) = \{ a_1, a_2, \ldots a_n \}$
y se representan por $X.a_1, X.a_2, \ldots X.a_n$

Conjunto de acciones semánticas de la regla ***r*** : $X_0 \rightarrow X_1 X_2 \ldots X_n \in P$:

$R(r) = \{ (X_i.a := f(X_r.b, \ldots X_s.c)$
$\quad | \quad a \in A(X_i), b \in A(X_r), .., c \in A(X_s)$ con $0 \leq i, r, \ldots, s \leq n \}$

$E' \rightarrow E \$$	*{ printf("%d", E.valor) }*
$E \rightarrow E_1 + T$	*{ E.valor := E_1.valor + T.valor }*
| T	*{ E.valor := T.valor }*
$T \rightarrow T_1 * F$	*{ T.valor := T_1.valor * F.valor }*
| F	{ T.valor := F.valor }
$F \rightarrow (E)$	*{ F.valor := E.valor }*
| num	*{ F.valor := num.lexval }*

Definiciones

Árbol sintáctico anotado: Árbol sintáctico de una frase en el que en cada nodo aparecen los valores de los atributos de ese nodo.

Tipos de Atributos

$X_i.a$ será un atributo sintetizado sii todas las acciones semánticas que lo definen son de la forma:

r : Xo → X1 X2 ... Xn $\{ (Xo.a := f(Xi1.bj1, Xi2.bj2, ..., Xip.bjp) \}$

$X_i.a$ será un atributo heredado sii todas las acciones semánticas que lo definen son de la forma:

r : Xo → X1 X2 ... Xn $\{ (X_{i1}.a := f(Xo.bjo, Xi1.bj1, ..., Xip.bjp) \}$

con i1 <> o

Atributos sintetizados y heredados

$D \rightarrow T\ L$	{ L.tipo := T.tipo }
$T \rightarrow$ **int**	{ T.tipo := tentero }
$T \rightarrow$ **float**	{ T.tipo := treal }
$L \rightarrow L_1$, **id**	{ L_1.tipo := L.tipo ; *InsertaTDS* (id.nom, L.tipo) }
$L \rightarrow$ **id**	{ *InsertaTDS*(id.nom, L.tipo) }

Cadena ω= int id,id

Ejemplo 2: Contar a's y b's

S → (A)	*{ printf ("Num.a: %d Num.b: %d", A.as ,A.bs) }*
A → A_1 , D	*{ A.as = A_1.as+D.as ; A.bs= A_1.bs+D.bs }*
A → D	*{ A.as = D.as ; A.bs= D.bs }*
D → a	*{ D.as = 1 ; D.bs= 0 }*
D → b	*{ D.as = 0 ; D.bs= 1 }*
D → (A)	*{ D.as = A.as ; D.bs= A.bs }*

Ejemplo 3: Convertir de binario a decimal

S → L	*{ printf (L.**v**) }*
L → L_1 B	*{ L.**v** := L_1.**v** * 2 + B.**v** ;}*
L → B	*{ L.**v** := B.**v** }*
B → **0**	*{B.**v** := 0 }*
B → **1**	*{B.**v** := 1 }*

Ejemplo 4: Convertir a base 10

S → **h** L	*{ L.**base** := 16; printf (L.**v**) }*
S → **b** L	*{ L.**base** := 2; printf (L.**v**) }*
S → **o** L	*{ L.**base** := 8; printf (L.**v**) }*
L → L_1 B	*{ L.**v** := L_1.**v** * L.**base** + B.**v** ; L_1.**base** := L.**base**}*
L → B	*{ L.**v** := B.**v** }*
B → **0**	*{B.**v** := 0 }*
B → **1**	*{B.**v** := 1 }*
B → **2**	*{B.**v** := 2 }*
…..	…..
B → **f**	*{B.**v** := 15 }*

Ejemplo 4: Convertir a base 10

Ejemplo con comprobación de números erróneos

S → **h** L	*{ L.**base** := 16; printf (L.**v**) }*
S → **b** L	*{ L.**base** := 2; printf (L.**v**) }*
S → **o** L	*{ L.**base** := 8; printf (L.**v**) }*
L → L_1 B	*{ L.**v** := L1.**v** * L.**base** + B.**v** ; L1.**base** := L.**base** ;*
	*si B.v >=L.**base** ent yyerror ("digito erróneo"); }*
L → B	*{ L.**v** := B.**v** ; si B.v >=L.**base** ent yyerror ("digito erróneo"); }*
B → **0**	*{B.**v** := 0; }*
B → **1**	*{B.**v** := 1; }*
B → **2**	*{B.**v** := 2; }*
.....	.....
B → **f**	*{B.**v** := 15; }*

2. Orden de evaluación

Orden de evaluación

Si un atributo c de un nodo del árbol anotado depende de un atributo b, la regla semántica que define al nodo c debe ser evaluada después de la regla semántica que define a b.

Métodos para la evaluación semántica:

- Mediante grafo de dependencias
- Dirigidos por la sintaxis

Orden de evaluación

Grafo de dependencias: Grafo dirigido acíclico con:

- *Un nodo para cada atributo*
- *Un arco $b \rightarrow c$ si el atributo c depende del atributo b*

- Cualquier orden topológico del grafo de dependencias proporciona un orden de evaluación de las reglas semánticas válido.

Métodos dirigidos por la sintaxis:

Se restringe la clase de gramáticas atribuidas que puede ser utilizada.

Gramáticas S y L atribuidas

Definiciones

Notaciones para asociar reglas semánticas con producciones:

- *Gramática de atributos:*

  Gramática más un conjunto de atributos y de reglas semánticas asociadas a cada regla.

- *Esquema de traducción dirigida por la sintaxis (ETDS):*

  Gramática más un conjunto de atributos y de reglas semánticas asociadas a cada regla. Estas reglas pueden aparecer entre los símbolos de la parte derecha de las producciones.

3. Definiciones S-atribuidas

Gramática S-atribuida

Definición atribuida que solo usa atributos sintetizados.

Orden de evaluación:
Se puede anotar el árbol sintáctico de forma ascendente: desde las hojas hasta la raíz.

- Las definiciones S-atribuidas pueden transformarse en esquemas de traducción sin más que poner todas las acciones semánticas al final de las reglas de producción.

Ejemplo: S-atribuida y análisis LR

Gramática de atributos que pase a **base 10** un número **binario**

(1) S →L *{ printf (L.v) }*

(2) L → L_1 B *{ L.v := L_1.v * 2 + B.v }*

(3) L → B *{ L.v := B.v }*

(4) B → 0 *{B.v := 0 }*

(5) B → 1 *{B.v := 1 }*

Cadena ω= 101

	0	1	$	S	L	B
0	d-4	d-5		1	2	3
1			Acep			
2	d-4	d-5	r-1			6
3	r-3	r-3	r-3			
4	r-4	r-4	r-4			
5	r-5	r-5	r-5			
6	r-2	r-2	r-2			

Ejemplo 5: Calculadora

S→ **(print** E **)**

E→ **(Op** E **)** | **(Op** E E **)** | **num**

Op→ **+** | **-** | ***** | **/**

ETDS que calcule el valor numérico de una expresiones y la imprima.

Ejemplo: (print (/ 20 (* 2 (- 5)))) → Imprimirá –2

Solución ejemplo 5 Calculadora

S -> **(print** E **)**	print (E.valor);
E -> **(** **op** E_1 **)**	**si** (op.o = mas) **ent** E.valor := E_1.valor **sino si** (op.o = menos) **ent** E.valor := - E_1.valor **sino** yyerror("Operador no válido");
E -> **(** **op** E_1 E_2 **)**	**si** (op.o = mas) **ent** E.valor := E_1.valor + E_2.valor **sino si** (op.o = menos) **ent** E.valor := E_1.valor - E_2.valor **sino si** (op.o = por) **ent** E.valor := E_1.valor * E_2.valor **sino si** (op.o = divi) **ent** E.valor := E_1.valor / E_2.valor
E -> **num**	E.valor = num.lexval;
op -> **+**	op.o := mas
op -> **-**	op.o := menos
op -> *****	op.o := por
op -> **/**	op.o := divi

4. Definiciones L-atribuidas

Definición L-atribuida

Definición atribuida en la que cada atributo heredado de X_j ($1 \leq j \leq n$), en la producción $A \rightarrow X_1\ X_2\ \ldots\ X_n$ depende solo de:

a) Los atributos de los símbolos $X_1, X_2, \ldots X_{j-1}$ a la izquierda de X_j

b) Los atributos heredados de A

$D \rightarrow T\ L$	{ L.tipo := T.tipo }
$T \rightarrow$ **int**	{ T.tipo := tentero }
$T \rightarrow$ **float**	{ T.tipo := treal }
$L \rightarrow L_1$, **id**	{ L_1.tipo := L.tipo ; InsertaTDS (id.nom, L.tipo) }
$L \rightarrow$ **id**	{ InsertaTDS(id.nom, L.tipo) }

Orden de evaluación ETDS

```
Procedure Primero_en_Profundidad (N: nodo):
   begin  {N → X1 , X2 , .... Xp }
        For i := 1 to p do
           begin
                evaluar atributos heredados de Xi ;
                Primero_en_Profundidad (Xi) ;
           end ;
        evaluar atributos sintetizados de N;
   end
```

Esquema de Traducción Dirigido por Sintaxis (ETDS)

⇨ En un ETDS las acciones semánticas no pueden usar atributos de símbolos que aparezcan a su derecha:

- Se debe usar el algoritmo Primero_en_Profundidad para determinar el orden en el que deben aparecer las reglas entre los símbolos de la producción.

Ejemplo:

D → T { *L.tipo := T.tipo* } L

T → **int** { *T.tipo := tentero* }

T → **float** { *T.tipo := treal* }

L → { *L1.tipo := L.tipo* } L1, **id** {*InsertaTDS (id.nom, L.tipo)* }

L → **id** { *InsertaTDS(id.nom, L.tipo)* }

Propiedades de los ETDS

(derivadas del algoritmo Primero_en_Profundidad)

- Un atributo heredado para un símbolo del lado derecho de una producción debe calcularse en una acción antes de dicho símbolo.
- Una acción no puede usar un atributo sintetizado de un símbolo a la derecha de dicha acción.
- Una acción para calcular un atributo sintetizado del no-terminal del lado izquierdo solo puede ejecutarse después de las reglas que calculan todos los atributos que usa.

Ejemplo 6: Calculadora detecta errores

S→ (print E)

E→ (Op E) | (Op E E) | num

Op→ + | − | * | /

*Detectar divisiones por cero e informar del **número del operador** en el que se ha producido.*

Ej.: (print (+ (* 2 10) (/ 5 (- 3 3)))) -> Error: Div por cero en op 3

Ejemplo 6: Calculadora detecta errores

S -> **(print** E **)**	E.cont := 0 print (E.valor);
E -> **(** op E_1 **)**	E_1.cont := E.cont+1 **si** (op.o == mas) **ent** E.valor := E_1.valor **sino si** (op.o == menos) **ent** E.valor := - E_1.valor **sino** MemError("Operador no válido"); E.sintcont := E_1.sintcont
E -> **(** op E_1 E_2 **)**	E_1.cont := E.cont+1 ; E_2.cont := E_1.sintcont **si** (op.o == mas) **ent** E.valor := E_1.valor + E_2.valor **sino si** (op.o == menos) **ent** E.valor := E_1.valor - E_2.valor **sino si** (op.o == por) **ent** E.valor := E_1.valor * E_2.valor **sino si** (op.o == divi) **si (E_2.valor == 0) { yyerror("División por cero en op num.", E.cont +1);** E.valor := 0; } **sino** E.valor := E_1.valor / E_2.valor E.sintcont := E2.sintcont
E -> **num**	E.valor = num.lexval; E.sintcont := E.cont
op -> **+**	op.o := mas
op -> **-**	op.o := menos
op -> *****	op.o := por
op -> **/**	op.o := divi

Ejemplo 7

S → A

A → **(num** A A **)** | **(num)**

- ETDS que obtenga el mínimo nivel de profundidad del número de mayor valor.
- *Nivel de profundidad* de un nodo es el número **mínimo** de ramas de la raíz al nodo.

  Ej. (2 (5) (8)) Nivel de prof. del mayor número = 2

Solución con atributos sintetizados

Ejemplo 8

La siguiente gramática reconoce líneas formadas por palabras. Las palabras pueden ser correctas o incorrectas. Escribe un ETDS que cada vez que se reconozca una palabra **incorrecta** muestre el **número de línea** en la que se encuentra.

D -> L **newline** RD

RD -> L **newline** RD

| ε

L -> P RL

RL -> P RL

| ε

P -> **correcta**

| **incorrecta**

Solución ejemplo 8

D -> *{ L.nlin= 1 ; }*
  L **newline** *{ RD.nlin= 2 ; }*
  RD

RD -> *{ L.nlin= RD.nlin ; }*
  L **newline** *{ RD_1.nlin= RD.nlin + 1; }*
  RD_1
 | ε

L -> *{ P.nlin = L.nlin ; }*
  P *{ RL.nlin= L.nlin ; }*
  RL

RL -> *{ P.nlin = RL.nlin ; }*
  P *{ RL_1.nlin= RL.nlin ; }*
  RL_1
 | ε

P -> **incorrecta** *{ printf("Error en línea %d", P.nlin); }*
 | **correcta**

Ejemplo 9

S-> **find2 (** LE **)**
LE -> LE , E | E
E -> **cte** | **ctr** | **op**

Construye un ETDS que detecte la primera aparición de dos "cte" seguidas en la lista y devuelva en un atributo de S la posición que ocupa. Si no hay dos apariciones seguidas de "cte" devolverá 0.

Ej.

find2 (cte op cte cte op cte cte)

Devolverá 3

Solución ejemplo 9

S-> **find2 (** LE **)** *{ S.sol = LE.sol ; }*

LE -> LE_1 , E *{ LE.pos = LE_1.pos + 1 ;*
*__si__ (LE_1.cte == 1 **and** E.cte == 1) LE.sol = LE.pos ;*
__sino__ { LE.cte = E.cte ; LE.sol = 0 ;}
}
| E *{ LE.cte = E.cte ; LE.pos = 1; }*

E -> cte *{ E.cte = 1 ; }*
| ctr *{ E.cte = 0 ; }*
| op *{ E.cte = 0 ; }*