# *Traducción dirigida por la Sintaxis*

Beatriz González Hernández, alu3527
Airam Molina Rodríguez, alu3566

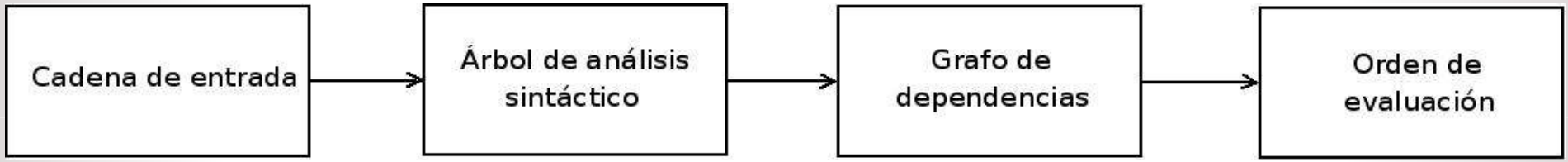
# Índice

- Introducción
- Notaciones:
  - Definición dirigida por sintaxis
    - Gramática con atributos
    - Definición con atributos por la izquierda
  - Esquemas de traducción
- Eliminación de la recursividad por la izquierda
- Diseño de un traductor predictivo
- Árbol sintáctico
  - Grafos dirigos acíclicos
- Preguntas

# Traducción dirigida por sintaxis

- Definición

Es una técnica que asocia información a los símbolos y producciones de la gramática con el objetivo de traducir una entrada en una salida.




-No siempre son necesarias todas las fases

# Atributos

Es información semántica asociada a los símbolos del lenguaje.

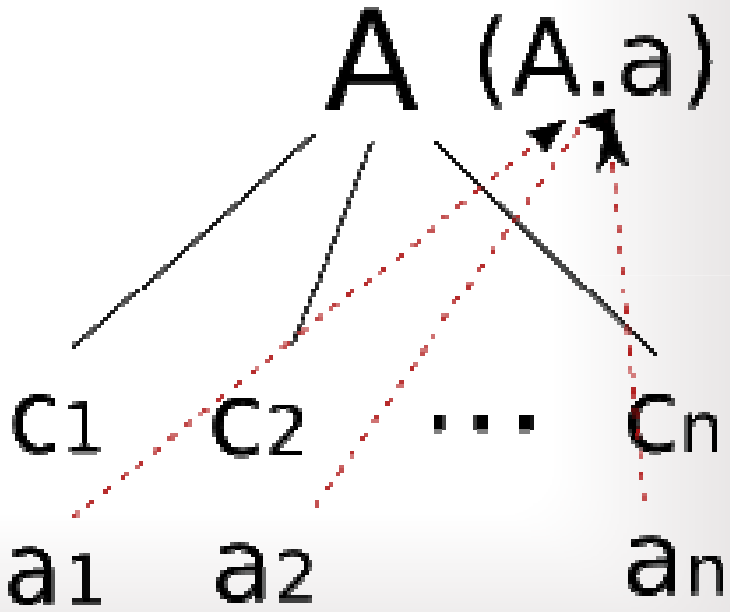
Dos tipos:

- Atributos sintetizados
- Atributos heredados

# Atributos Sintetizados

- Se calculan a partir de los atributos de los nodos hijos.
- Sea $A \rightarrow c_1 c_2 \ldots c_n$

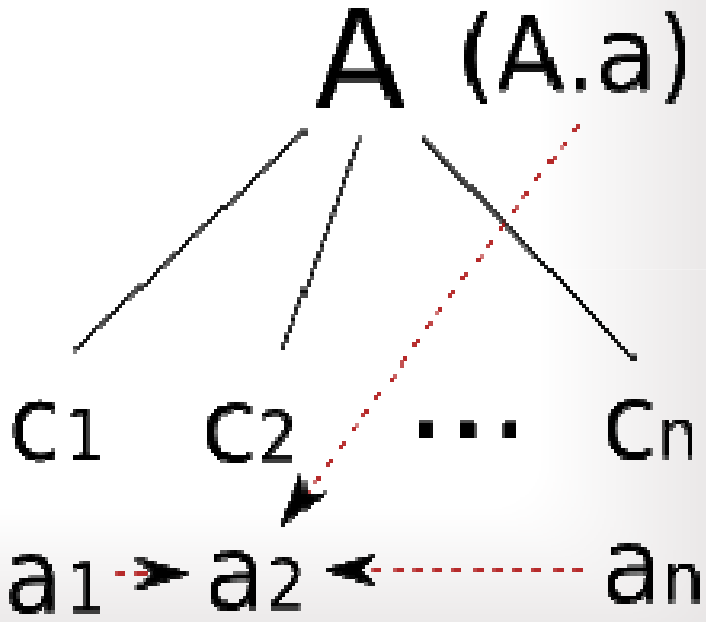
$A.a = f(c_1.a_{i1}, c_2.a_{i2}, c_3.a_{i3}, \ldots, c_n.a_{in})$



# Atributos Heredados

- Se calculan a partir de los atributos del nodo padre y/o hermanos
- Sea $A \rightarrow c_1 c_2 \ldots c_n$

$c.a = f(A.a, c_1.a_{i1}, c_2.a_{i2}, \ldots, c_n.a_{in})$



# Reglas Semánticas

Acciones relacionadas con cada producción que evalúan los atributos y que pueden crear efectos colaterales.

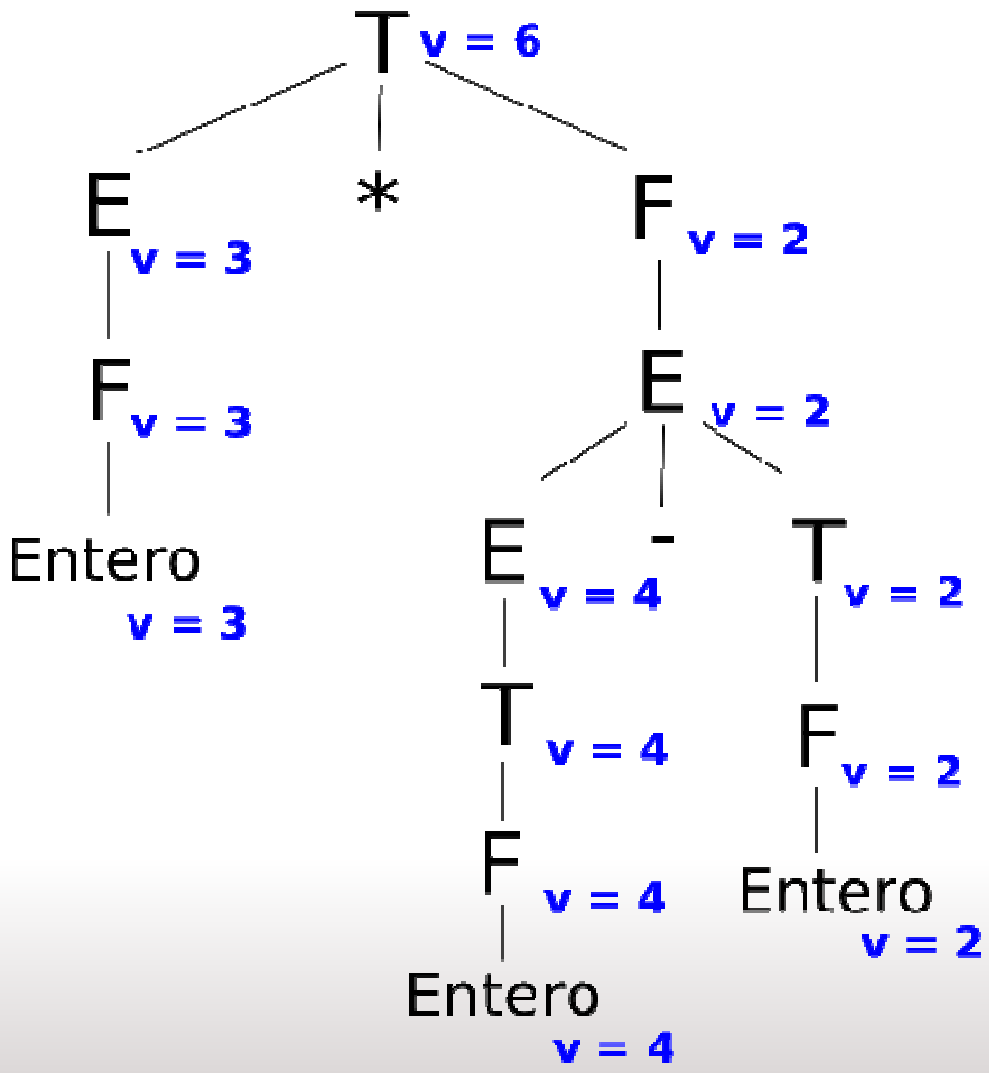
- Son fragmentos de código

$E \rightarrow E_1 + T$ { E.val = E1.val + T.val }{ print(E.val) }

$T \rightarrow num$ { T.val = num.val }

# Árboles decorados

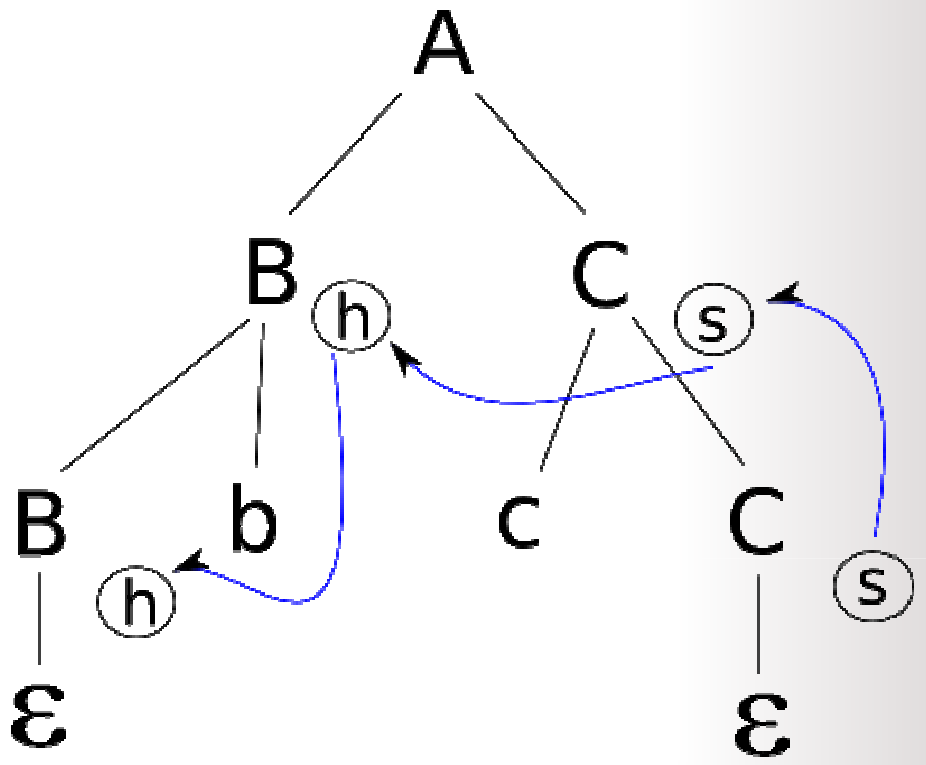
AAS que muestra los valores de los atributos de cada nodo.



# Grafo de dependencias

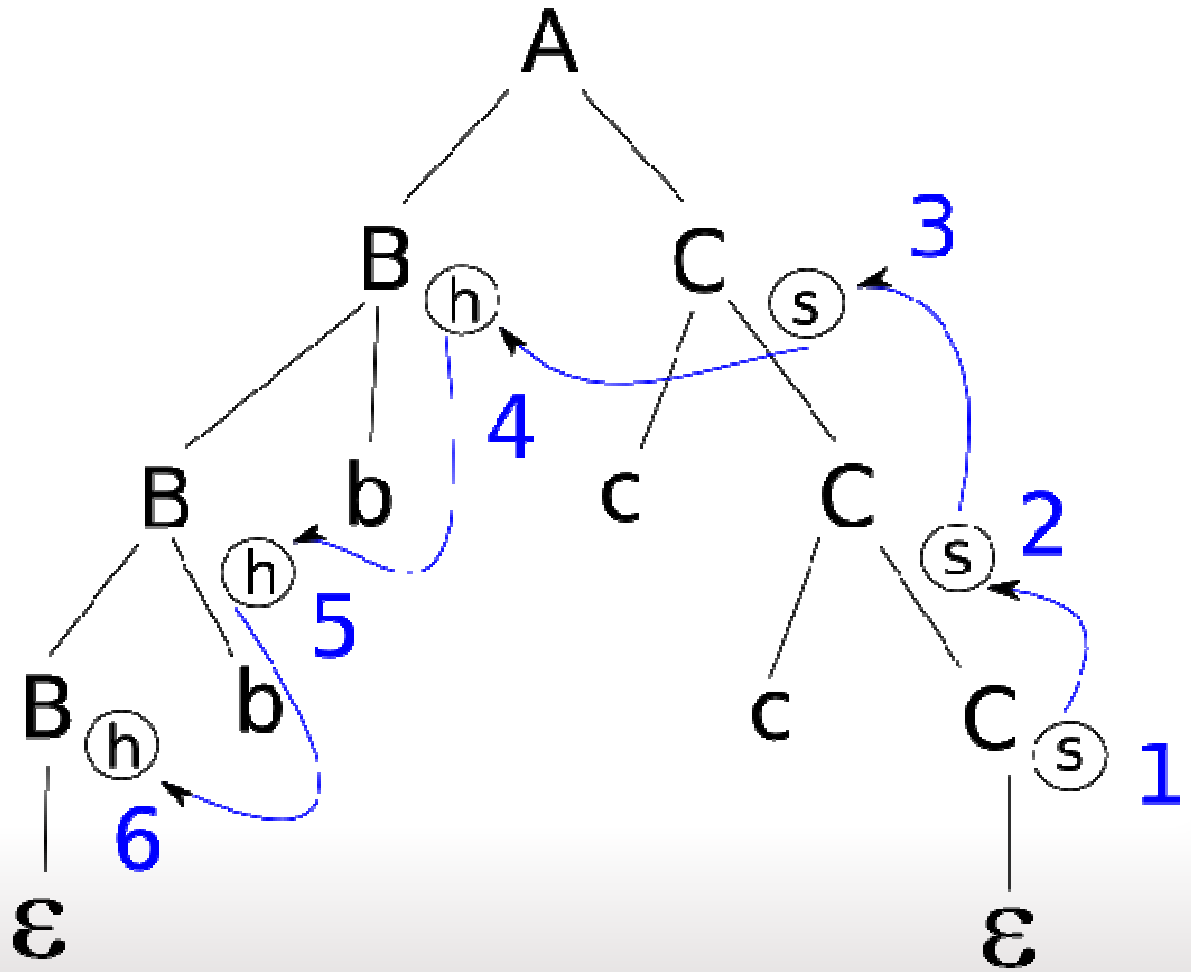
Es aquel grafo en el que se muestran las dependencias de los atributos

| *Producción* | *Reglas Semánticas* |
|---|---|
| A → B C | B.h := C.s |
| B → B1 b | If (B.h >= 0) then<br>B1.h := B.h – 1;<br>Endif |
| B → ε | If (B.h = 0) then<br>write ('Ok')<br>else write ('Error'); |
| C → c C1 | C.s := C1.s + 1 |
| C → ε | C.s := 0 |



# Orden de Evaluación

Orden en el que se evalúan los atributos



# Ejemplo

Traducir ingredientes en un perrito caliente:

PERRITO → pan salchicha EXTRAS { Abrir(pan.attr); Colocar(salchicha.attr);
Extra = EXTRAS.attr; n = EXTRAS.n;
while (n > 0) {Colocar(Extra[n]); n--;} }

EXTRAS → nextra EXTRAS1 { Extra = EXTRA1.attr; n = EXTRA1.n ;
Extra[n] = nextra.attr; n++;
EXTRA.atr = Extra; EXTRA.n = n}

EXTRA → ε {EXTRA.n = 0}

# Notaciones

- Definición Dirigida por Sintaxis (DDS)
  - Alto Nivel
  - Ocultan detalles de la implementación
  - No impone orden de las reglas semánticas
- Esquemas de Traducción (ETDS)
  - Bajo Nivel
  - Muestran detalles de la implementación
  - Impone orden de las reglas semánticas

# Definición Dirigida por Sintaxis

Gramática ampliada a la que se le añade atributos y un conjunto de reglas semánticas

| ***Producción*** | ***Reglas Semánticas*** |
|---|---|
| Asignación -> **num =** Suma | If (suma.tipo = = num.tipo) {<br>num.valor = Suma.valor;<br>} else Error_Semántico; |
| Suma -> **num1 + num2;** | If (num1.tipo = = num2.tipo) {<br>Suma.tipo = num1.tipo;<br>Suma.valor = num1.valor +<br>num2.valor;<br>} else Error_Semántico; |

# Definición Dirigida por Sintaxis

Definición formal:

- Cada producción $A \rightarrow \alpha$ tiene asociado un conjunto de reglas semánticas de la forma $b=(c_1, c_2, \ldots, c_k)$
  - b es un atributo sintetizado de A y $c_1, c_2, \ldots, c_k$ son atributos que pertenecen a los símbolos de la producción
  - b es atributo heredado de uno de los símbolos gramaticales del lado derecho de la producción, y $c_1, c_2, \ldots, c_k$ son atributos que pertenecen a los símbolos gramaticales de la producción

Se asume que los terminales y el inicial sólo tienen atributos sintetizados.

# Gramática con Atributos

Es una DDS en la que las funciones en las reglas semánticas no pueden tener efectos colaterales

# Definición con Atributos por la Izquierda (DAI)

DDS en la que la información que utiliza las reglas semánticas está disponible en el momento de su ejecución.

Toda definición con atributos sintetizados es una definición con atributos por la izquierda.

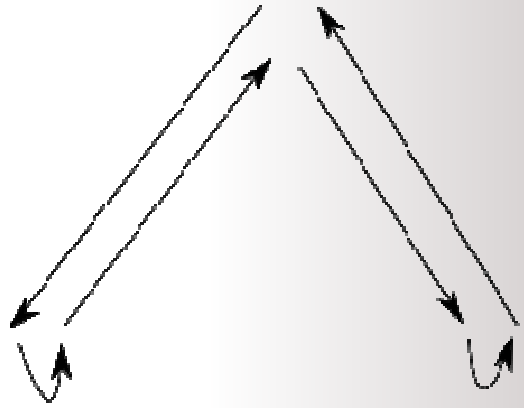
# Definición con Atributos por la Izquierda (DAI)

Definición formal:

- Una DDS es una DAI si para $X_j$, 1<=j<=n, para cada atributo heredado del lado derecho de $A \rightarrow X_1X_2...X_n$ depende sólo de:
  - Los atributos de los símbolos X1X2...Xj-1 a la izquierda de Xj en la producción
  - Los atributos heredados de A

# Definición con Atributos por la Izquierda (DAI)

- Heredan del nodo "padre"
  - Ej: **A → XYZ { Y.valor = A.valor }**



- Heredan de hermanos a su "izquierda"
  - Ej: **A → XYZ { Y.valor = X.valor }**

- Heredan de otros atributos del mismo símbolo
  - Ej: **A → XYZ { Y.valor = float(Y.int_value)*2 }**

# Esquemas de Traducción (ETDS)

Es una gramática independiente del contexto en la que se asocian atributos a los símbolos gramaticales y se insertan reglas semánticas entre llaves {} en la parte derecha de las producciones.

E → T { E'.th := T.ts } E' { E.ts := E'.ts }

E' → **op** T { E'1.th := E'.th || T.ts || **op**.lexema}

E'1 { E'.ts := E'1.ts }

E' → ε {E'.ts := E'.th }

T → **num** { T.ts := **num**.lexema }

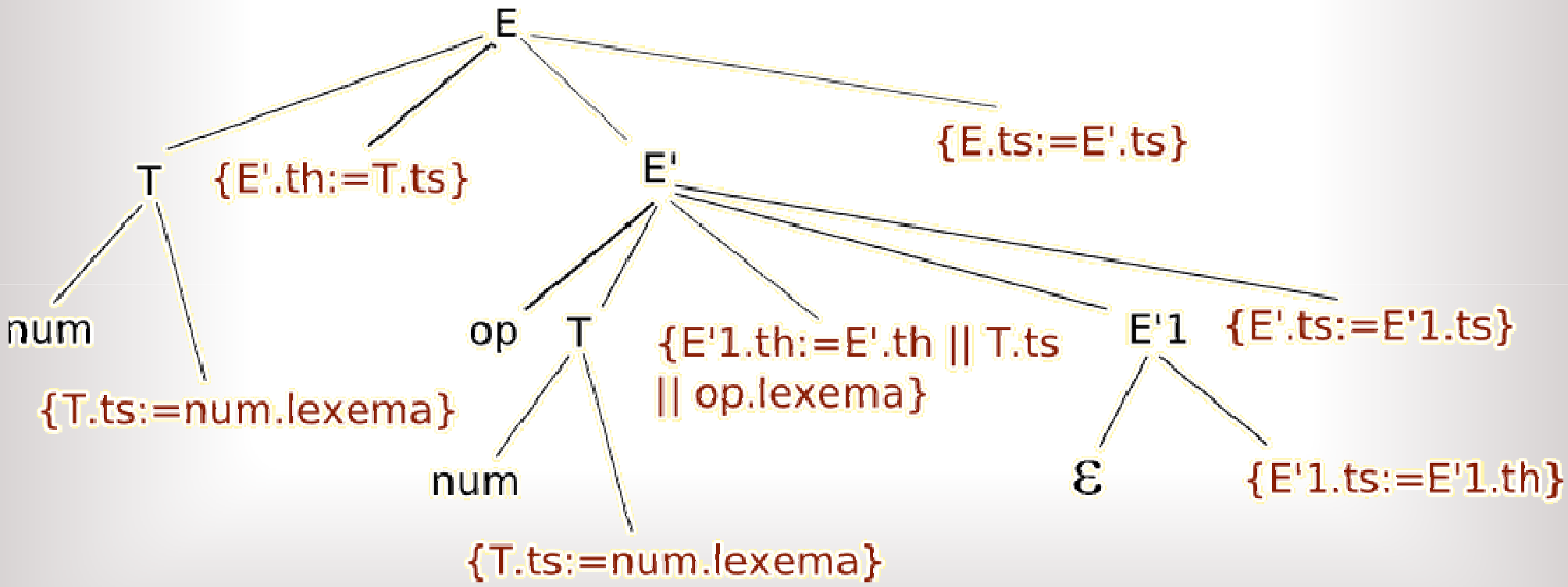
Se consideran las acciones como si fueran símbolos terminales

E → T { E'.th := T.ts } E' { E.ts := E'.ts }

E' → **op** T { E'1.th := E'.th || T.ts || **op**.lexema}
E'1 { E'.ts := E'1.ts }

E' → ε {E'.ts := E'.th }

T → **num** { T.ts := **num**.lexema }



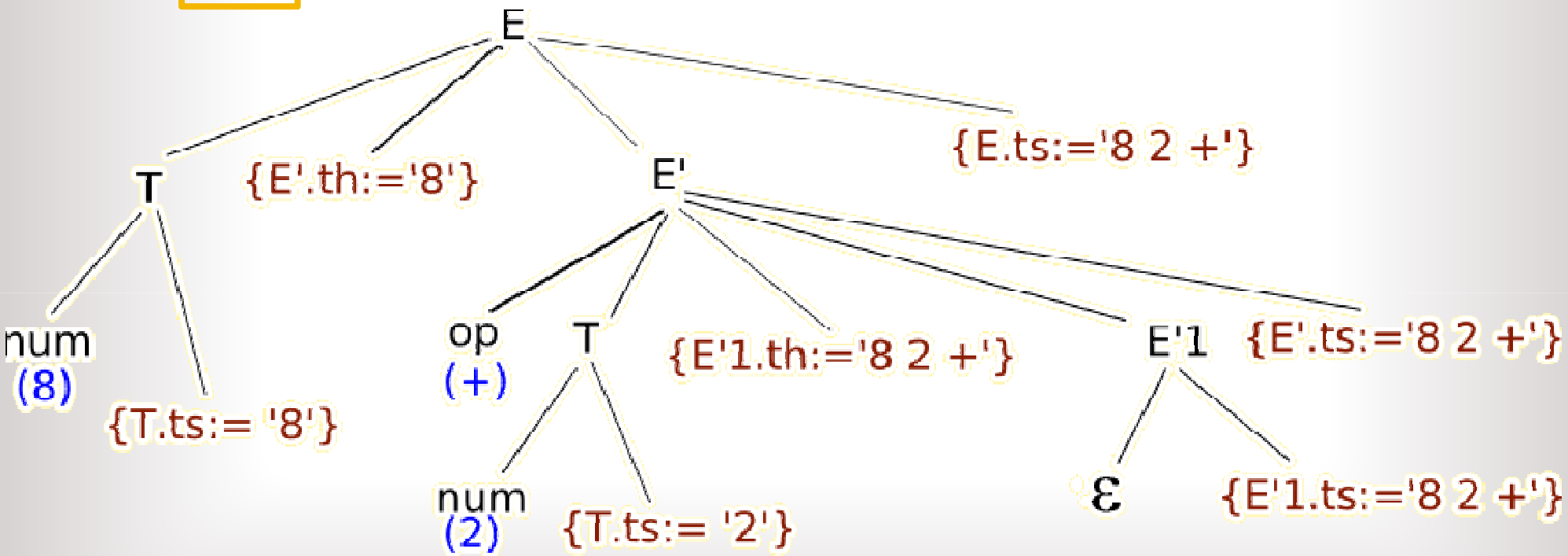
$E \rightarrow T \{ E'.th := T.ts \} E' \{ E.ts := E'.ts \}$

$E' \rightarrow \textbf{op}\ T \{ E'_1.th := E'.th \,||\, T.ts \,||\, \textbf{op}.lexema \}$
$E'_1 \{ E'.ts := E'_1.ts \}$

$E' \rightarrow \varepsilon \{ E'.ts := E'.th \}$

$T \rightarrow \textbf{num} \{ T.ts := \textbf{num}.lexema \}$

cadena: 8 + 2



# Restricciones de Diseño de un ETDS de una sola pasada

- Si sólo tenemos atributos sintetizados:
  - Las reglas semánticas se colocan al final del lado derecho de la producción
- Si además tenemos heredados:
  - Un atributo heredado para un símbolo en el lado derecho de una producción debe calcularse en una acción antes que dicho símbolo.
  - Una acción no debe referirse a un atributo sintetizado de un símbolo que esté a la derecha de la acción.
  - Un atributo sintetizado para un NO terminal de la izquierda solo puede calcularse después de que se hayan calculado todos los atributos a los que hace referencia. (La acción se sitúa al final del lado derecho de la producción).

# Pasos para el diseño de un ETDS

1. Definir qué atributos son necesarios
2. Añadir a la gramática las acciones semánticas que calculen los valores de los atributos
3. Ver de qué tipo es cada atributo
4. Comprobar que se cumplen las restricciones para el diseño de una ETDS de una sola pasada

# Eliminación de Recursividad por la Izquierda

- Esquema general
  - Dadas las producciones:

    $A \rightarrow A_1\ Y\ \{A.a := g(A_1.a, Y.y)\}$

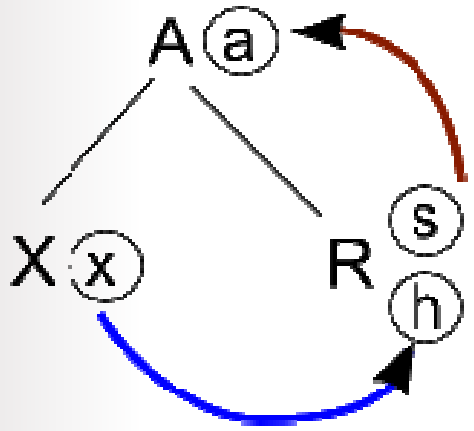
    $A \rightarrow X\ \{A.a := f(X.x)\}$
  - Transformación:

    $A \rightarrow X\ \{R.h := f(X.x)\}\ R\ \{A.a := R.s\}$

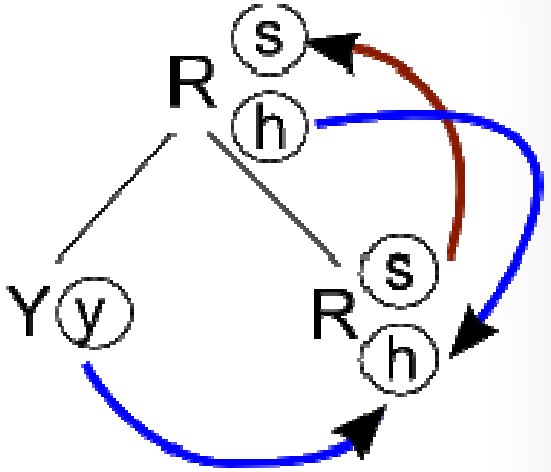
    $R \rightarrow Y\ \{R_1.h := g(R.h, Y.y)\}\ R_1\ \{R.s := R_1.s\}$

    $R \rightarrow \varepsilon\ \{R.s := R.h\}$

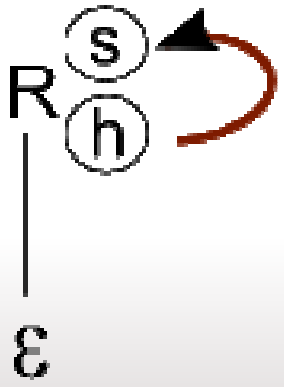
A -> X {R.h := f(X.x)} R {A.a := R.s}



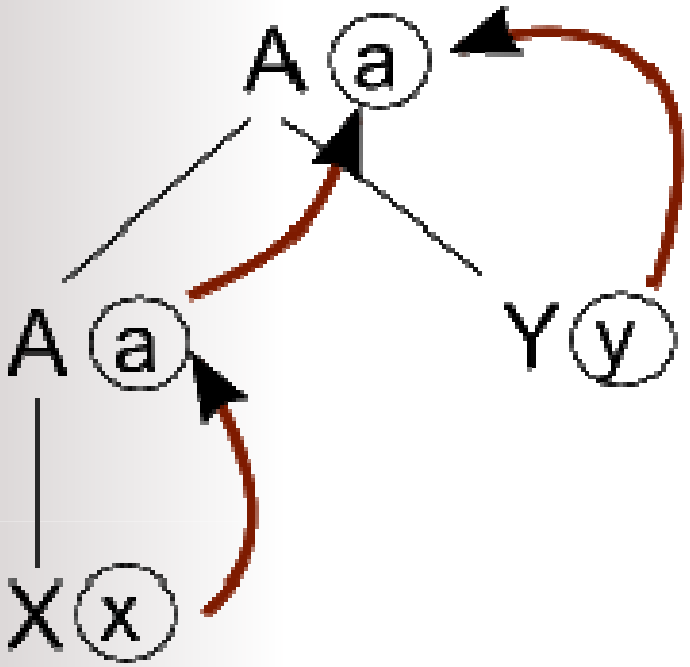

R -> Y {R1.h := g(R.h, Y.y)} R1 {R.s := R1.s}



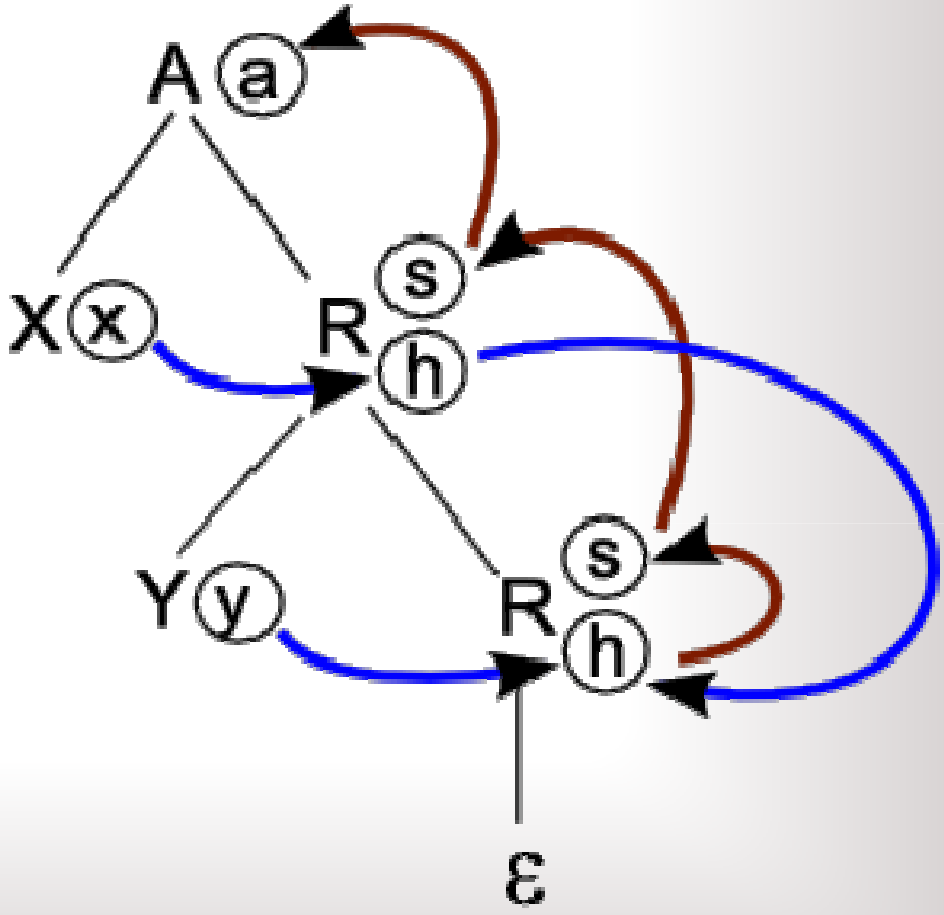

R -> ε {R.s := R.h}



A -> A1 Y {A.a := g(A1.a, Y.y)}
A -> X {A.a := f(X.x)}




A -> X {R.h := f(X.x)} R {A.a := R.s}
R -> Y {R1.h := g(R.h, Y.y)} R1 {R.s := R1.s}
R -> ε {R.s := R.h}



# Eliminación de Recursividad por la Izquierda

- Pasos a realizar:
  - Identificar:

    A, A1, Y, g(A1.a, Y.y), A.a, X, f(X.x)
  - Sustituir en el esquema general

- Ejemplo:

  $E \rightarrow E1 + T \{E.val := E1.val + T.val\}$

  $E \rightarrow T \{E.val := T.val\}$

  $T \rightarrow num \{T.val := num.val\}$

E -> E1 + T {E.val := E1.val + T.val}
E -> T {E.val := T.val}
T -> num {T.val := num.val}

Identificar:

A ➡ E  A1 ➡ E1  Y ➡ +T  g(A.a,Y.y) ➡ E1.val+T.val

X ➡ T  f(X.x)➡ T.val

Sustituir:

E → T {R.h:=T.val} R {E.val:=R.s}

R → +T{R1.h:=R.h+T.val} R1 {R.s:=R1.s}

R → ε {R.s:=R.h}

# Diseño de un Traductor Predictivo

Modificación del analizador sintáctico predictivo.

1. Para cada no terminal A, se construye una función que tenga un parámetro por cada atributo heredado de A, que devolverá como resultado el valor del atributo sintetizado de A.
2. El código del no terminal A decide qué producción utilizar basándose en el símbolo en curso de entrada.
3. Se consideran los componentes léxicos, no terminales y acciones del lado derecho de la producción de derecha a izquierda:

# Diseño de un Traductor Predictivo

3.1. Si es un componente léxico X con atributo sintetizado x se guarda el valor de x en la variable declarada para X.x

3.2. Si es un no terminal B se incluye la sentencia c = B(b1, b2,..., bk) donde b1, b2, ..., bk son las variables que corresponden a los atributos heredados del símbolo B y "c" es la variable que corresponde al atributo sintetizado de B.

3.3. Si es una acción semántica se copia literalmente el código y se adapta para que utilice los atributos correctos.

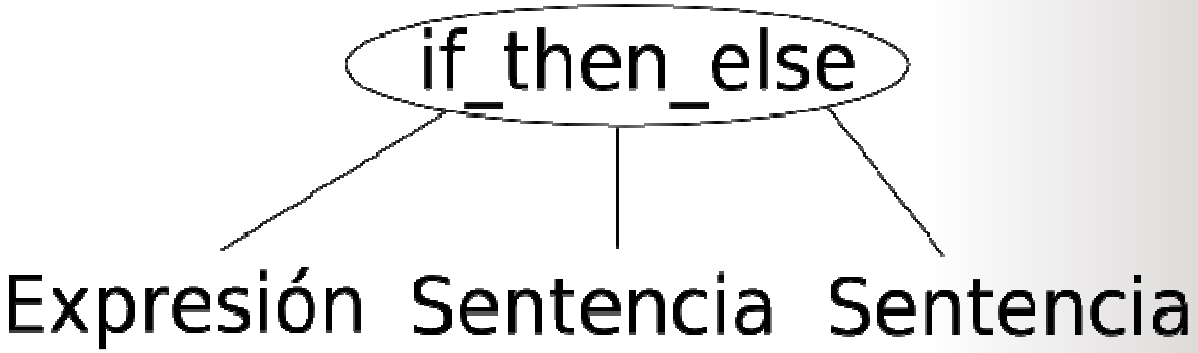
# Diseño de un Traductor Predictivo

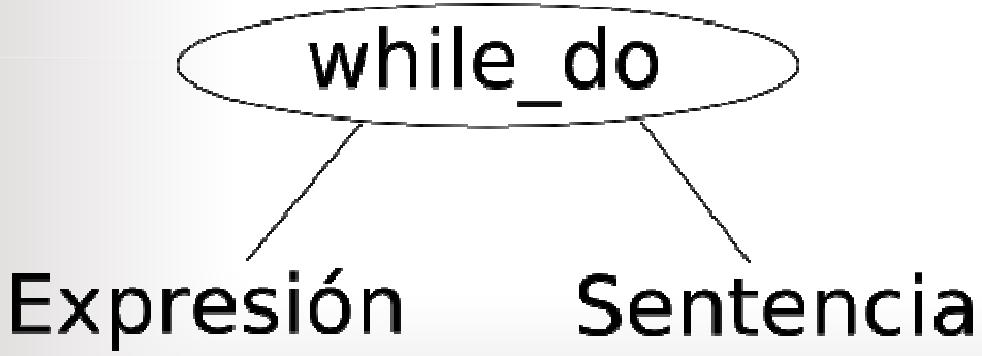
```
function R (h: nodo árbol): nodo árbol;
var apn, h1, s1, s : nodo árbol;
   lexemaopsuma : char;
begin
   if (lookahead = opsuma) then begin
      lexemaopsuma := valex;
      Match (opsuma);
      apn := T;
      h1 := haznodo (lexemaopsuma, h, apn);
      s1 := R (h1);
      s := s1;
   end
   else s := h;
   return s;
end;
```

$R \rightarrow$ **opsuma**
T { R1.h := haznodo (**opsuma**.lexema, R.h, T.apn) }
R1 { R.s := R1.s }
$R \rightarrow \varepsilon$ { R.s := R.h }

# Árbol Sintáctico

- Forma condensada de un AAS
- Tipo de Código Intermedio



# Árbol Sintáctico

Podemos utilizar Traducción Dirigida por Sintaxis para crear árboles sintácticos

- Construcción

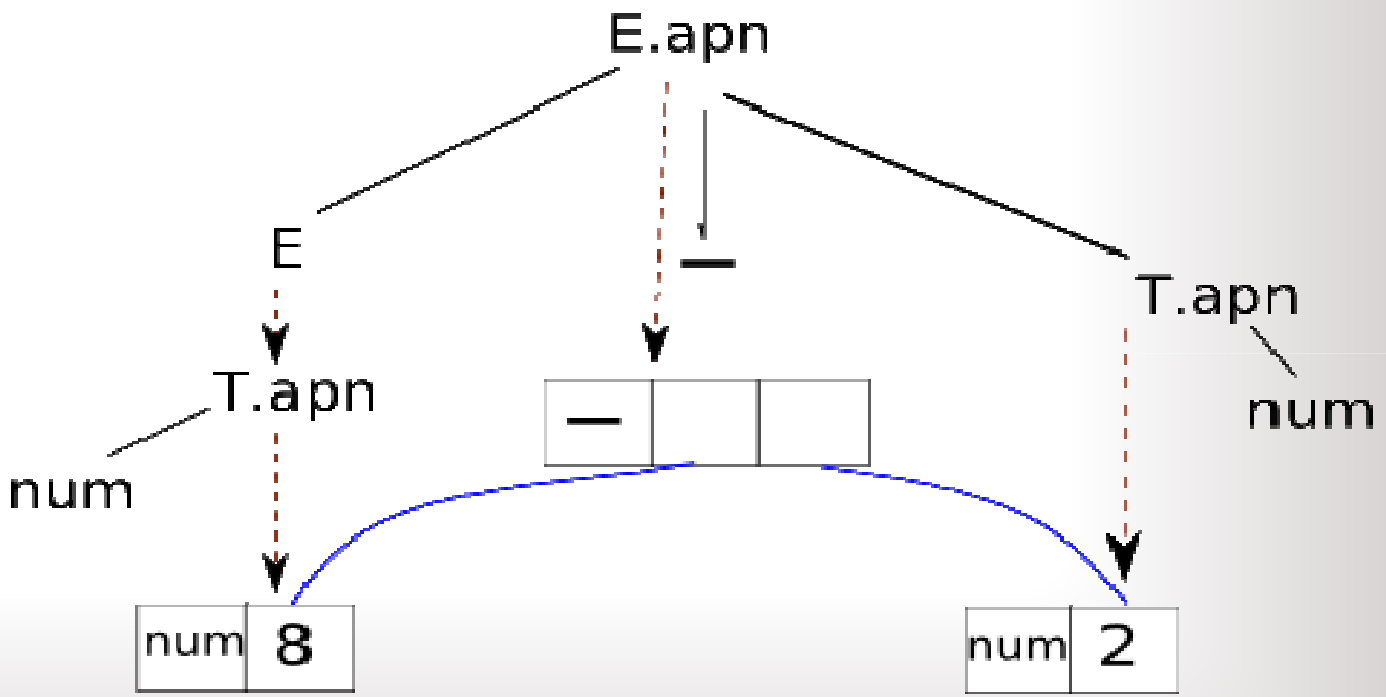
Se construyen subárboles para las subexpresiones creando un nodo (implementado como un registro) para cada operador y cada operando

# Árbol Sintáctico

- Los nodos para los operadores contienen un campo para identificarlo y el resto de campos son punteros a los nodos de los operandos
- Los nodos de los operandos tienen un campo para identificarlos y otro que almacena su valor
- Se utilizan funciones para crear los nodos

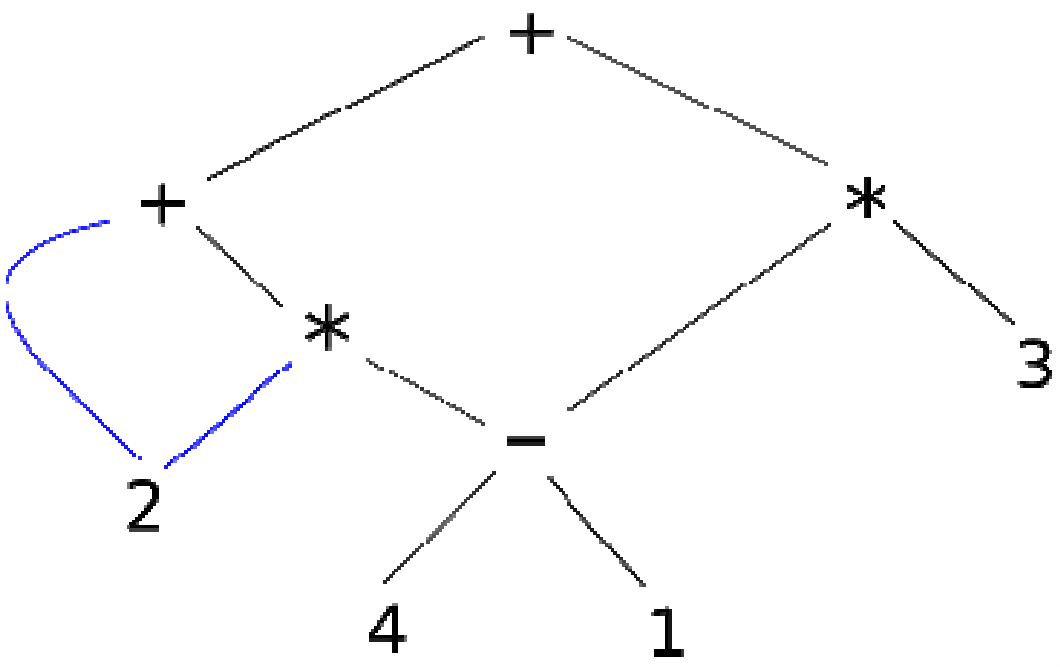
# Árbol Sintáctico

| *Producción* | *Reglas Semánticas* |
|---|---|
| E → E1 - T | { E.apn := haznodo ('-', E1.apn, T.apn } |
| E → T | { E.apn := T.apn } |
| T → **num** | { T.apn := hazhoja (**num**, **num**.val) } |



# Grafo Dirigido Acíclico para expresiones

Es un AS en el que un nodo que representa una subexpresión puede tener más de un padre



# Grafo Dirigido Acíclico para expresiones

- La Definición Dirigida por Sintaxis del Grafo Acíclico es igual a la del Árbol Sintáctico

- Se modifica la función que crea los nodos del AS para comprobar si el nodo ya existe. En ese caso se devuelve un puntero a dicho nodo. En caso contrario se crea el nuevo nodo

# Preguntas

Preguntas, dudas, comentarios…

# Bibliografía


- Alfred V. Aho, Ravi Sethi, Jffrey D. Ullman

  *Compiladores. Principios, técnicas y herramientas. Addison-Wesley*

  Iberoamericana 1990.

- Alicia Garrido, José M. Iñesta, Francisco Moreno, Juan A. Pérez *Diseño de Compiladores. Editorial: Publicaciones* Universidad de Alicante. 2002

# Final de la presentación

## Gracias por su atención

Beatriz González Hernández, alu3527
Airam Molina Rodríguez, alu3566