Autómatas y Lenguajes

3er curso

1er cuatrimestre

Alfonso Ortega: alfonso.ortega@uam.es

UNIDAD 1: Modelos de cómputo y familias de lenguajes

TEMA 4: Gramáticas de atributos

Tema 4: Gramáticas de atributos

4.1 Conceptos previos
4.2 Gramática de atributos
4.3 Nociones de programación
4.4 Notación definitva para gramáticas de atributos

4.1

Conceptos previos

Introducción

Sintáxis y semántica en los lenguajes formales y de programación

- La distinción entre sintaxis y semántica en los lenguajes formales es un tanto “artificial”.
- “Computacionalmente completo”, concepto asociado, formalmente, a
  - Máquinas de Turing
  - Gramáticas de tipo 0 de Chomsky
- Informalmente,
  - Ordenadores
  - Lenguajes de programación de alto nivel
- Hemos de tener en cuenta que los autómatas a pila (en el nivel independiente del contexto) tuvieron su origen en el ámbito de la especificación formal de los lenguajes de programación de alto nivel y con el objetivo de proporcionar herramientas que permitieran facilitar la construcción de las herramientas informáticas para su gestión (en particular compiladores e intérpretes)

Introducción

Sintáxis y semántica en los lenguajes formales y de programación

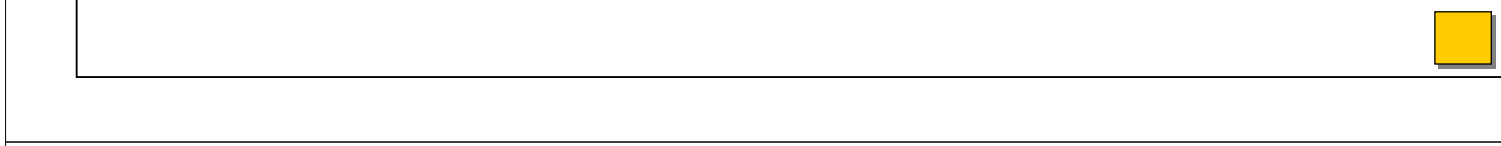
- Una de las herramientas más utilizadas en el diseño de lenguajes de programación son las gramáticas independientes del contexto
- Gramáticas independientes del contexto, asociadas, formalmente, a
  - Autómatas a pila (que no son computacionalmente completos)
- Por lo tanto, hay un subconjunto propio de problemas computables no representables mediante gramáticas independientes del contexto

Introducción

Sintáxis y semántica en los lenguajes formales y de programación

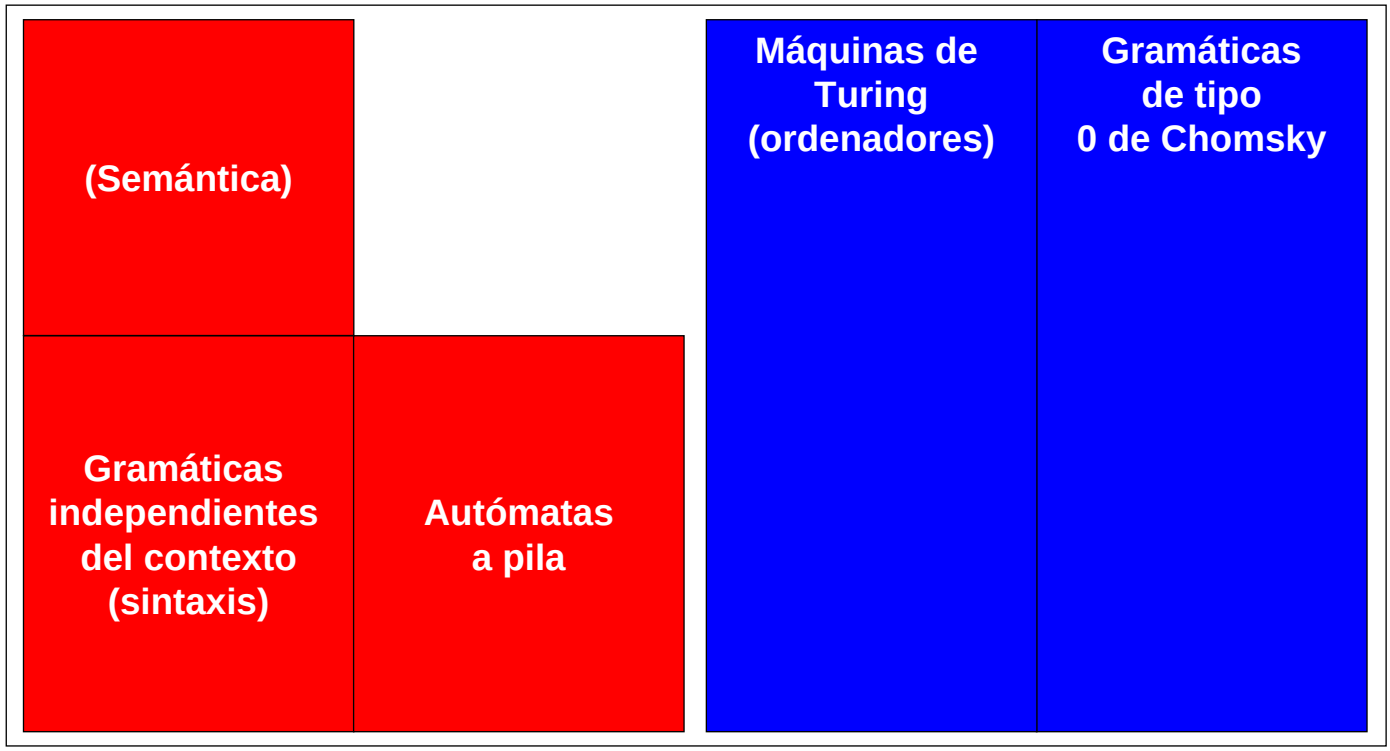
- Tradicionalmente la especificación de la mayoría de los lenguajes de programación se ha dividido en dos partes
  - Una gramática independiente del contexto (la sintaxis)
  - Observaciones más o menos formales para el resto de aspectos (semántica):
    - Restricciones en el uso de variables
      - Necesidad de declaración previa a su uso
      - Uso de apuntadores
    - Restricciones en el uso de tipos de datos
      - Reglas para el cambio de tipo de datos

- El uso de gramáticas independientes del contexto (más “semántica”) en lugar de gramáticas de tipo 0 de Chomsky se debe a la facilidad de manejo de las primeras



Introducción

Sintáxis y semántica en los lenguajes formales y de programación



Introducción

Semántica como propagación de información en los árboles de derivación de GIC

- A partir de los árboles de derivación obtenidos de las gramáticas independientes del contexto, se puede implementar “la semántica” como anotaciones de información en el árbol.
- Como se verá a continuación, al igual que las cadenas se construyen a partir de la aplicación de las reglas de la gramática a los diferentes símbolos no terminales, la información “semántica” de unos símbolos se “calculará” a partir de la información “semántica de otros”
- Veremos a continuación
  - Que esta información se llama “atributos semánticos”
  - El mecanismo de cálculo se denomina “propagación”
  - La propagación debe añadirse a las reglas de producción mediante lo que se conoce como “sistemas de atributos”

Atributo

Introducción

- Para poder realizar un movimiento de información como el que se comenta en la diapositiva previa, es necesario extender la gramática para permitir que los símbolos (tanto terminales como no terminales) tengan información asociada.
- Esta información asociada es lo que se llama **atributo**. Cada símbolo puede tener asociado uno o más atributos.
- La notación que se emplea es similar a la que se usa en Java para acceder a los atributos de un objeto o en C para referirse a una componente de una estructura : nombreSímbolo.nombreAtributo.
- Por ejemplo: si tenemos el símbolo no terminal E, al que queremos asociar el atributo val que guarda su valor, la notación sería:
  - Símbolo: E
  - Atributo: val
  - **Notación: E.val**

Atributo

Ejemplo

- Para tratar correctamente las expresiones, es frecuente que los lenguajes de programación tengan reglas para especificar su tipo (t) y valor (v).
- Por lo que resulta útil asociar a cada símbolo los atributos t y v.
- Considerad la siguiente gramática para expresiones aritméticas:

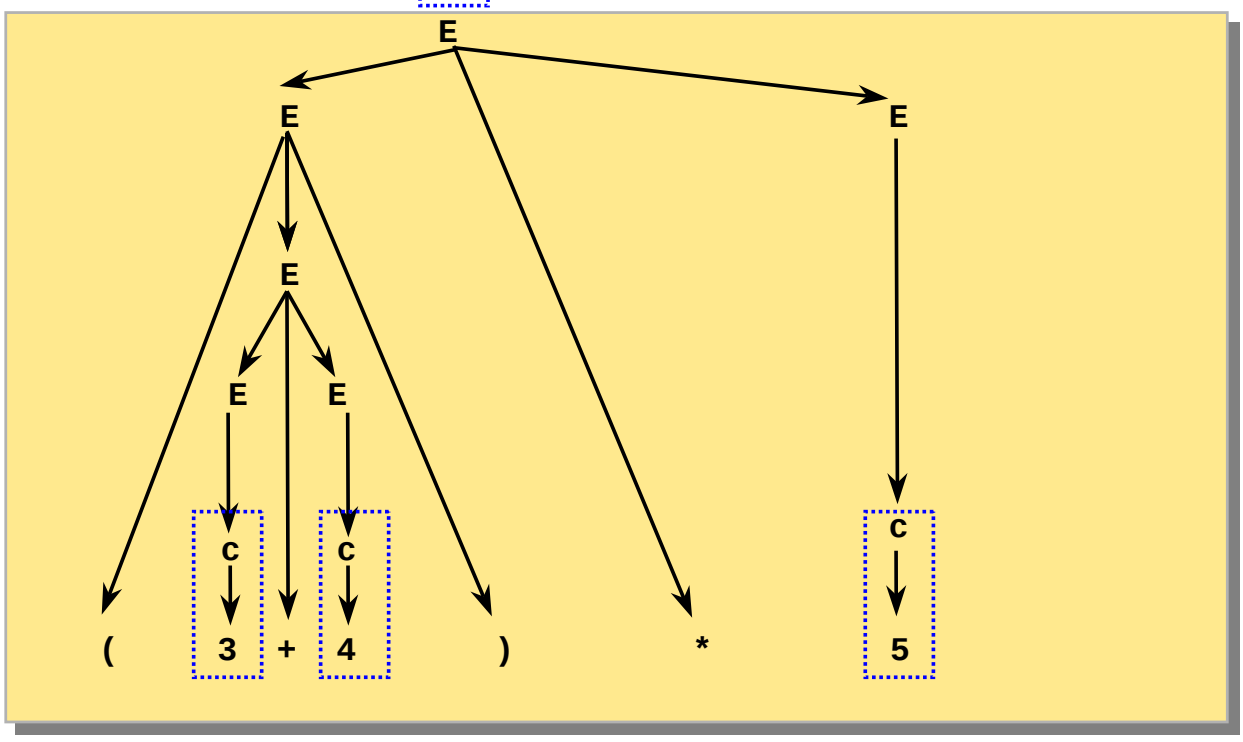
G_E={{+,*,(,),c,i}, {E},
{
E→E+E,
E→E-E,
E→-E,
E→E*E,
E→E/E,
E→E^E,
E→(E),
E→i,
E→c,
},
E}

*(**i** se refiere a un identificador y **c** a una constante numérica)*

Atributo

Ejemplo

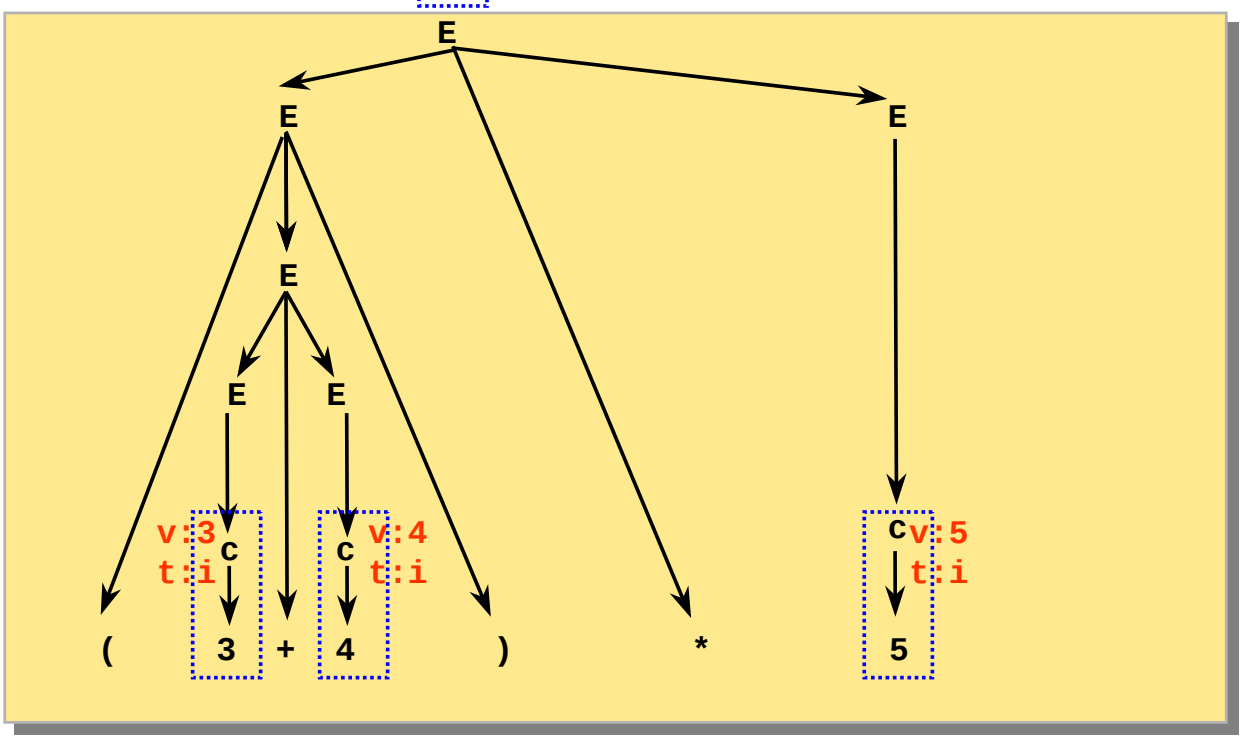
- Expresión: (3+4)*5 puntos de entrada de información semántica al árbol



Atributo

Ejemplo

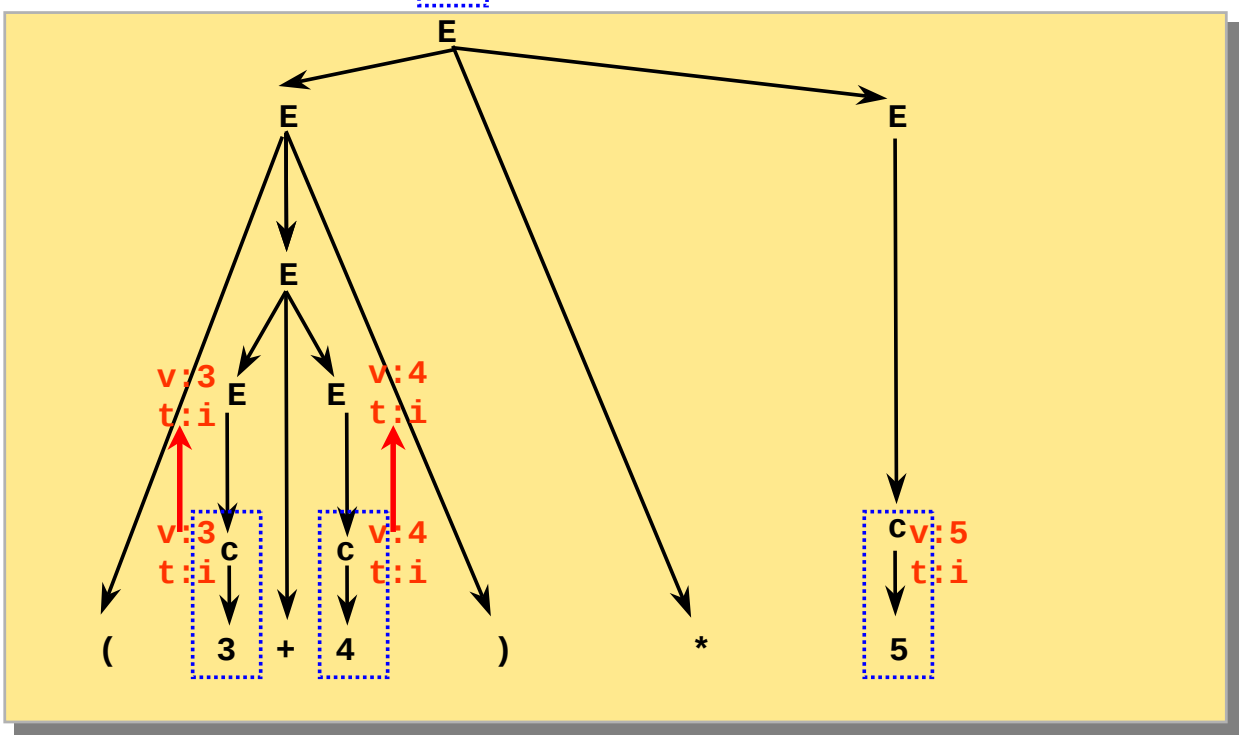
- Expresión: (3+4)*5 puntos de entrada de información semántica al árbol



Atributo

Ejemplo

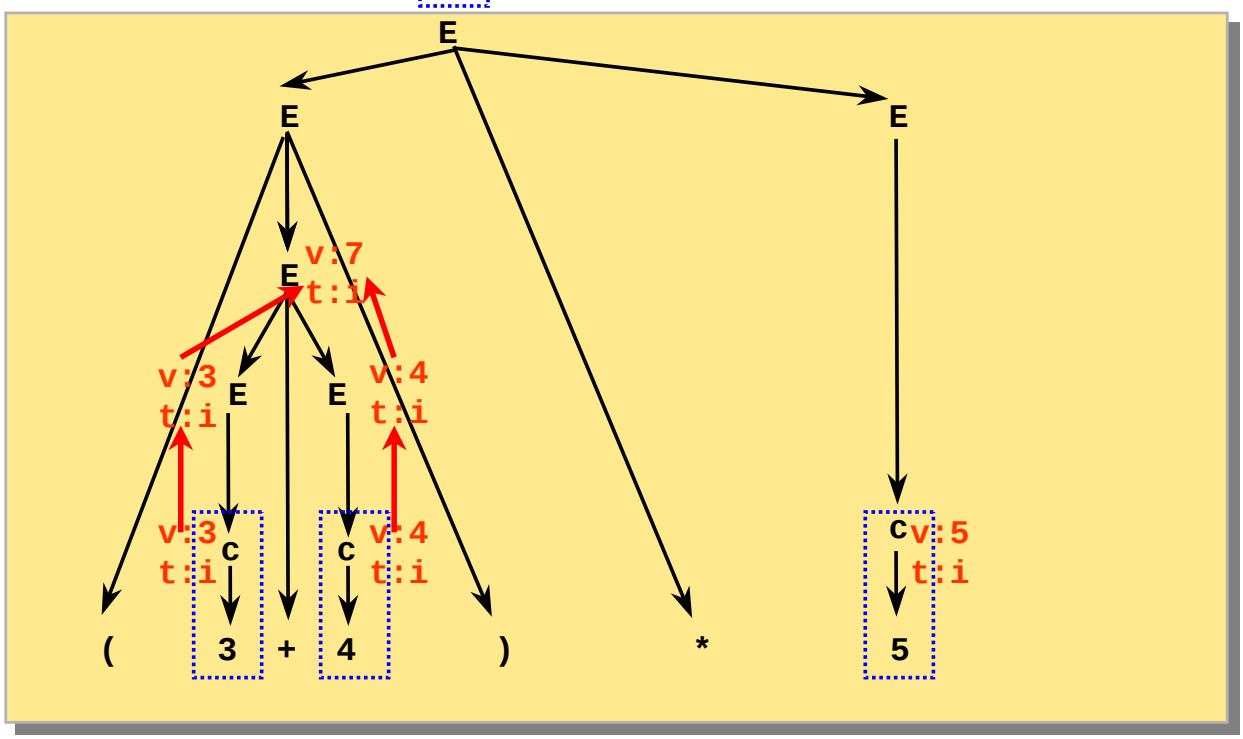
- Expresión: (3+4)*5 puntos de entrada de información semántica al árbol



Atributo

Ejemplo

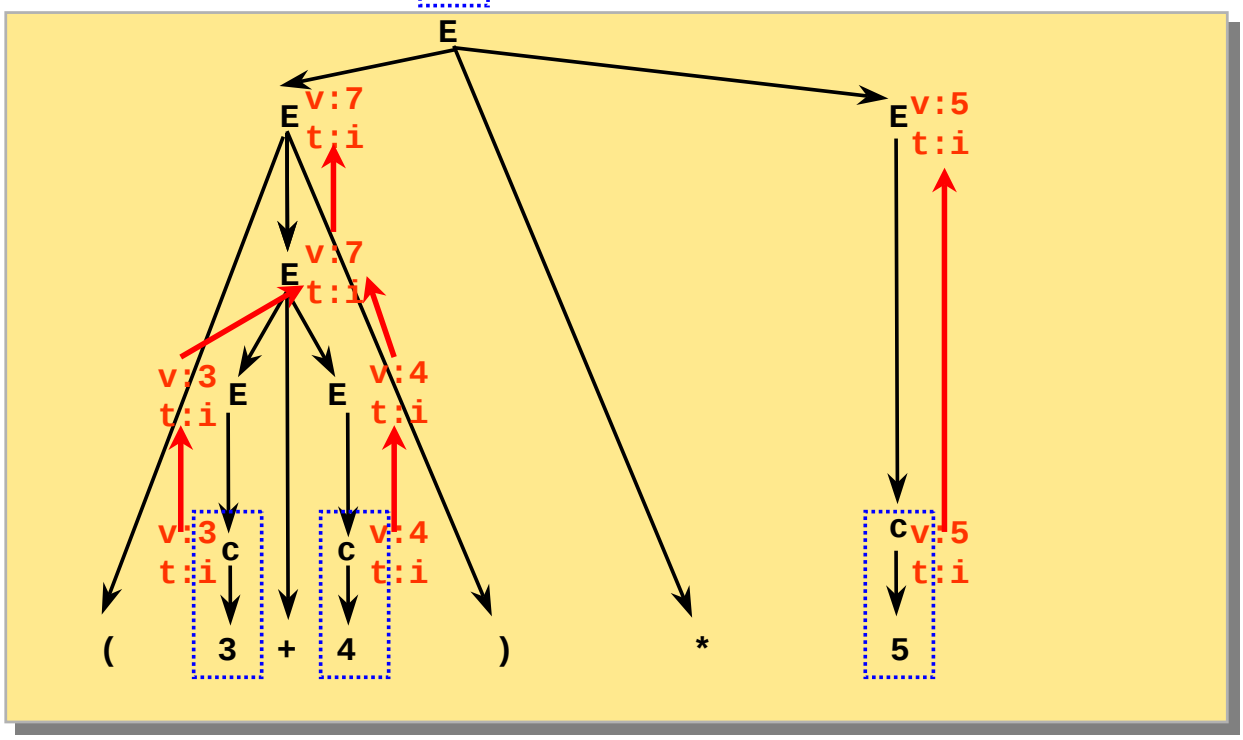
- Expresión: (3+4)*5 puntos de entrada de información semántica al árbol



Atributo

Ejemplo

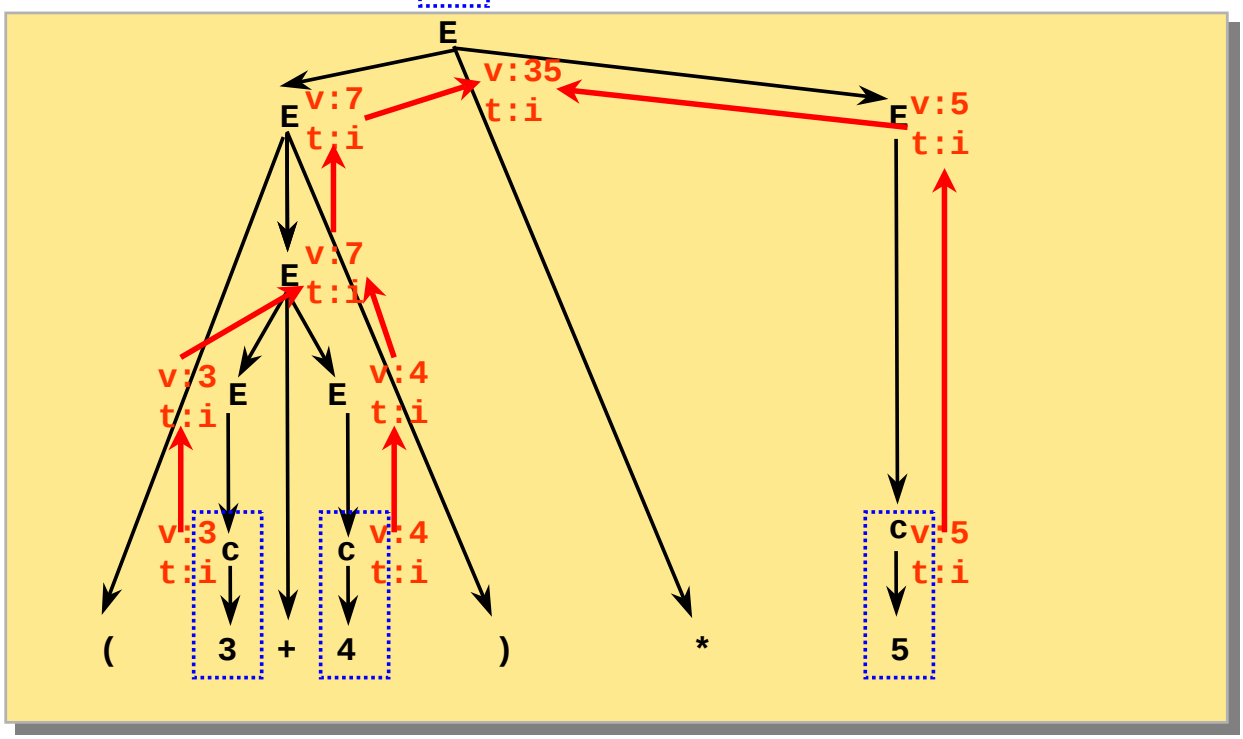
- Expresión: (3+4)*5 puntos de entrada de información semántica al árbol



Atributo

Ejemplo

- Expresión: (3+4)*5 puntos de entrada de información semántica al árbol



Atributo

Análisis del ejemplo

- En este caso, para obtener el valor correcto de toda la expresión, el atributo “valor” tiene que ser propagado hacia arriba. También el atributo “tipo” es propagado desde las hojas hacia el nodo superior.
- Desde el punto de vista de las reglas, es de la parte derecha (hijos en el árbol) desde donde se obtiene la información de la parte izquierda (padre en el árbol). Es decir, por ejemplo, en la regla:

$$E \rightarrow E+E$$

- Se puede añadir la siguiente expresión

$$E \rightarrow E+E \text{ y } E.valor = E.valor+E.valor$$

- Además es necesario poder distinguir entre las dos apariciones de **E** en la parte derecha, por lo que se añaden subíndices. Por ej. en este caso, **i** y **d:**

$$E \rightarrow E_i+E_d \text{ y } E.valor = E_i.valor + E_d.valor$$

- No obstante, no siempre la propagación de atributos es ascendente, como veremos en el siguiente ejemplo también puede ser descendente y horizontal.

Atributo

Ejemplo 2

- Consideremos ahora una gramática para declarar variables (ej. **`int A, B`**):

G_E**`={{int, real, i}, {D,T,L},`**

`{`

`D→TL`

`T→int`

`T→real`

`L→L,i`

`L→i`

`},`

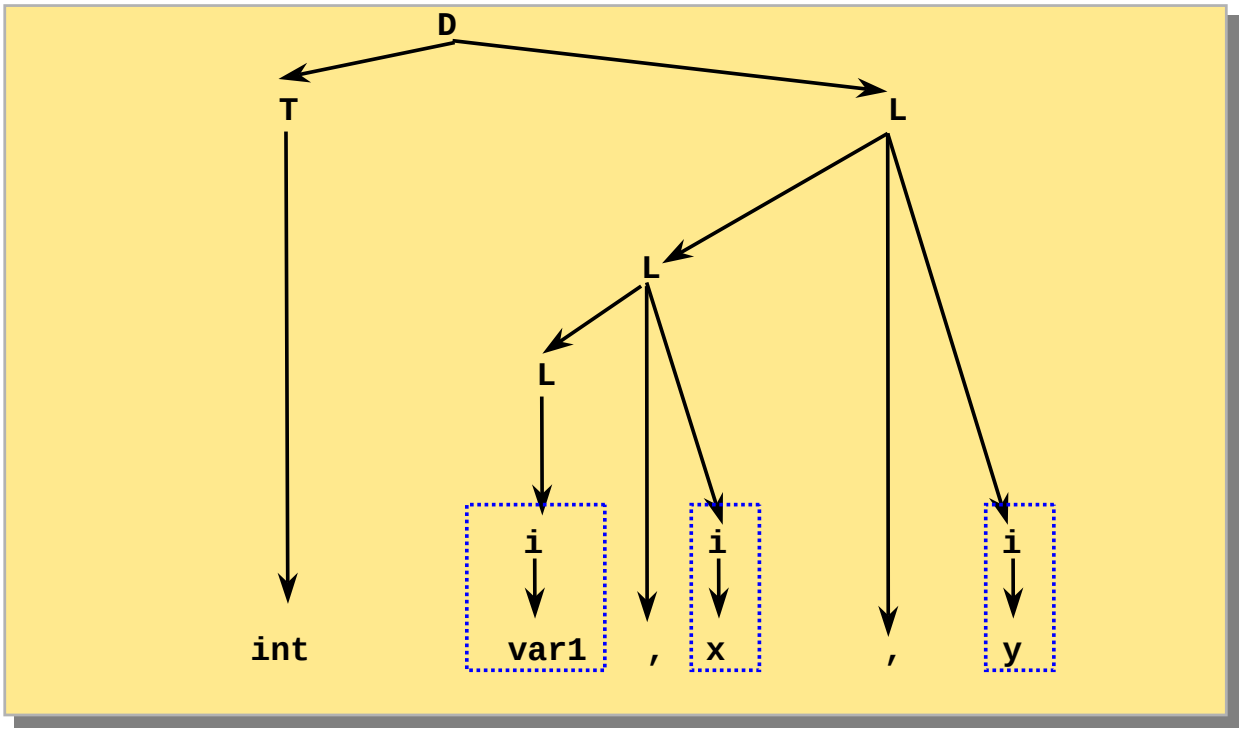
`D}`

- La notación que usaremos en este ejemplo es la siguiente:
  - **`i`** se refiere a identificador.
  - **`D`** representa una declaración.
  - **`T`** indica el tipo de las variables (**`int`** o **`real`**).
  - **`L`** representa una lista de identificadores separados por comas.

Atributo

Ejemplo 2

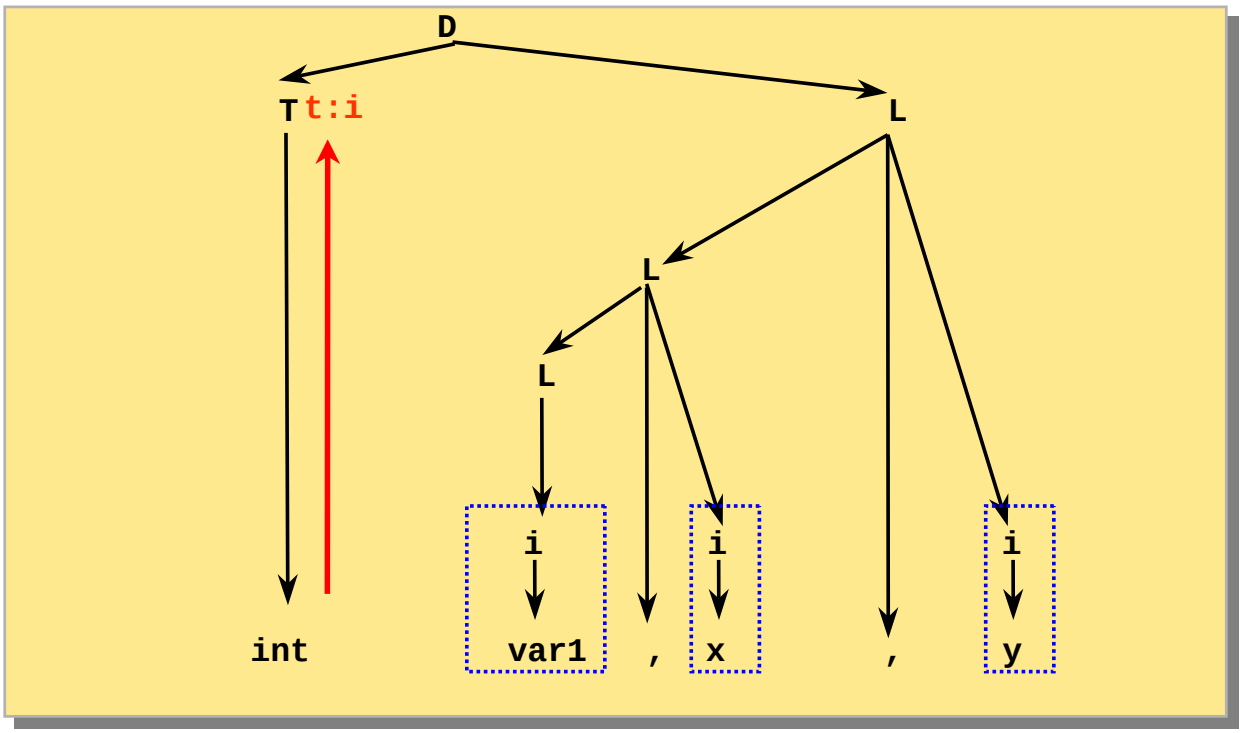
- Si tomamos como ejemplo la declaración: **`int var1, x, y`**



Atributo

Ejemplo 2

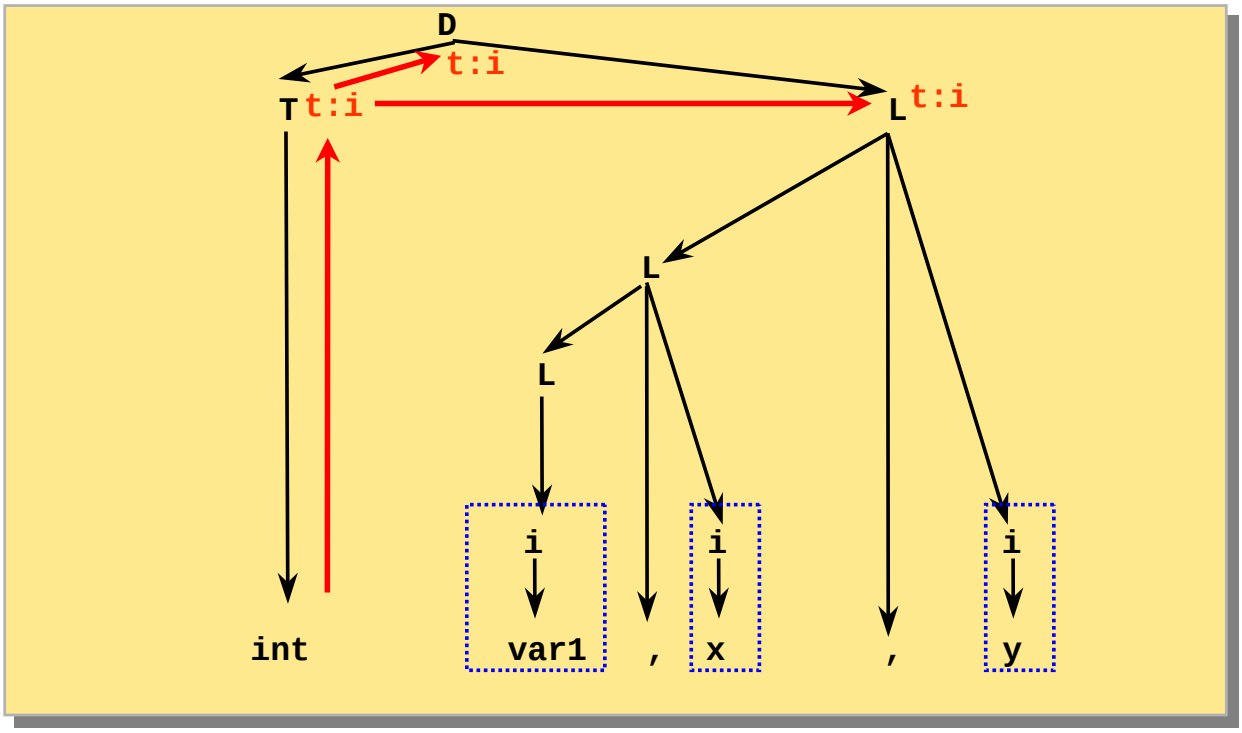
- Si tomamos como ejemplo la declaración: **`int var1, x, y`**



Atributo

Ejemplo 2

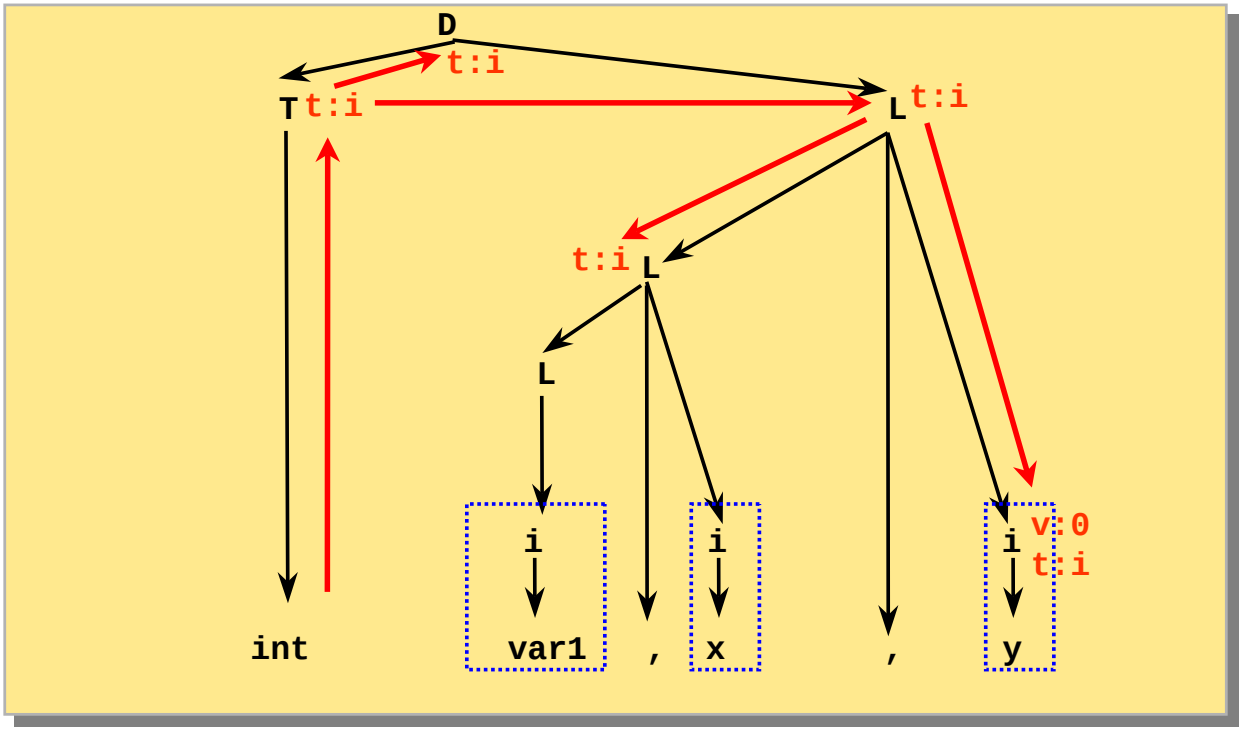
- Si tomamos como ejemplo la declaración: **`int var1, x, y`**



Atributo

Ejemplo 2

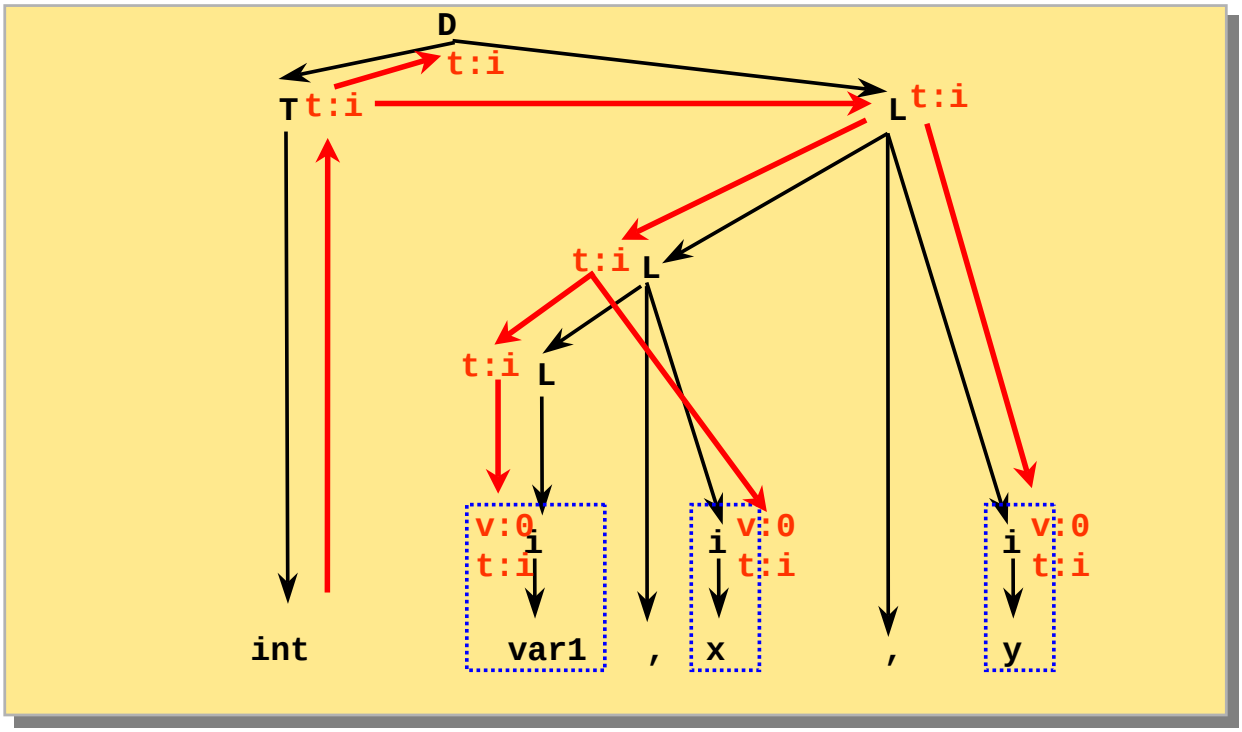
- Si tomamos como ejemplo la declaración: **`int var1, x, y`**



Atributo

Ejemplo 2

- Si tomamos como ejemplo la declaración: **`int var1, x, y`**



Definiciones formales

- Fromalmente:
  - **Atributos**:
    - Es una asociación de un nombre a un dominio de datos (como las variables de los lenguajes de programación.
    - Por ejemplo
      - El atributo de nombre **`valor`** del ejemplo anterior que, informalmente, puede tomar valores numéricos o
      - El atributo de nombre **`tipo`** que podría tomar como valor los diferentes tipos de datos permitidos en las variables
  - **Propagación (de atributos)**:
    - Llamaremos **propagación** al proceso de cálculo de los atributos en función de los valores de los atributos de los que dependen

Tipos de atributos

- Se pueden distinguir dos tipos de atributos conforme a la propagación de sus valoes:
  - **Atributos sintetizados**:
    - Los atributos de un nodo se calculan a partir de los atributos de sus nodos hijo. Esto es, los atributos de los símbolos de la parte izquierda se calculan a partir de los atributos de los símbolos de la parte derecha de la regla.
    - Por ejemplo, el atributo valor es sintetizado ya que
      - $E \rightarrow E_i + E_d$ { **`E.valor = `**E_i**`.valor + `**E_d**`.valor }`
  - **Atributos heredados**:
    - Los atributos de un nodo se calculan a partir de los atributos de su nodo padre o nodos hermanos en el árbol. Esto es, atributos de los símbolos que aparecen en la parte derecha necesitan para su cálculo de los atributos de los símbolos de la parte derecha de la regla o de su padre.
    - Por ejemplo, el atributo tipo es heredado ya que
      - **`D→TL { …; L.tipo = T.tipo; … }`**

4.2

Gramática de atributos

Gramática de atributos

Definición

- Una **gramática de atributos** es una extensión de un gramática independiente del contexto en la que se permite que:
  - Cada símbolo tenga asociado 1 o más atributos con un nombre y un tipo.
  - Cada regla de producción tenga asociada 1 o más acciones semánticas, cada una de las cuales especifica cómo cada atributo se puede calcular a partir de los atributos de otro símbolos de la regla de producción.
  - Existan datos globales, accesibles desde cualquiera de sus reglas, pero no asociadas a ningún símbolo concreto.
- Por lo tanto, al aplicar una gramática de atributos e ir construyendo el árbol de derivación no sólo se aplican las reglas de producción sino también las reglas semánticas asociadas. El **árbol resultante queda anotado** mostrando los valores de los atributos en cada nodo.
- Los **atributos** de los símbolos podrían ser considerados **información local de las reglas**.

Gramática de atributos

Ejemplo de atributos asociados a los símbolos

- En las gramáticas de atributos, por tanto, cada símbolo tiene asociada una lista de atributos (símbolos nuevos) como valores semánticos.
  - En el ejemplo de las expresiones, los alfabetos de terminales y no terminales quedan

    {+,*,(,),c{valor,tipo},i{valor,tipo}}, {E{valor,tipo}}
  - En el ejemplo de las declaraciones, los alfabetos de terminales y no terminales quedan

{{int{tipo}, real{tipo}, i{tipo}}, {D{tipo},T{tipo},L{tipo}}

Gramática de atributos

Ejemplo 1 expresado como gramática de atributos

- Si consideramos esta versión simplificada de la gramática que genera expresiones aritméticas de los últimos ejemplos:

E→E+E
E→E-E
E→-E
E→(E)
E→c

Gramática de atributos

Ejemplo 1 expresado como gramática de atributos

- En primer lugar, para definir la gramática de atributos hay que distinguir los símbolos que aparecen en la misma regla de producción puesto que tienen que ser únicos:

$E \rightarrow E_1 + E_2$
$E \rightarrow E_1 - E_2$
$E \rightarrow -E_1$
$E \rightarrow (E_1)$
$E \rightarrow c$

Gramática de atributos

Ejemplo 1 expresado como gramática de atributos

- En segundo lugar, se asocia cada símbolo con los atributos. En este caso, ya los teníamos definidos como **valor** (que representaremos mediante su inicial **v**) y **tipo** (que representaremos mediante su inicial **t**).

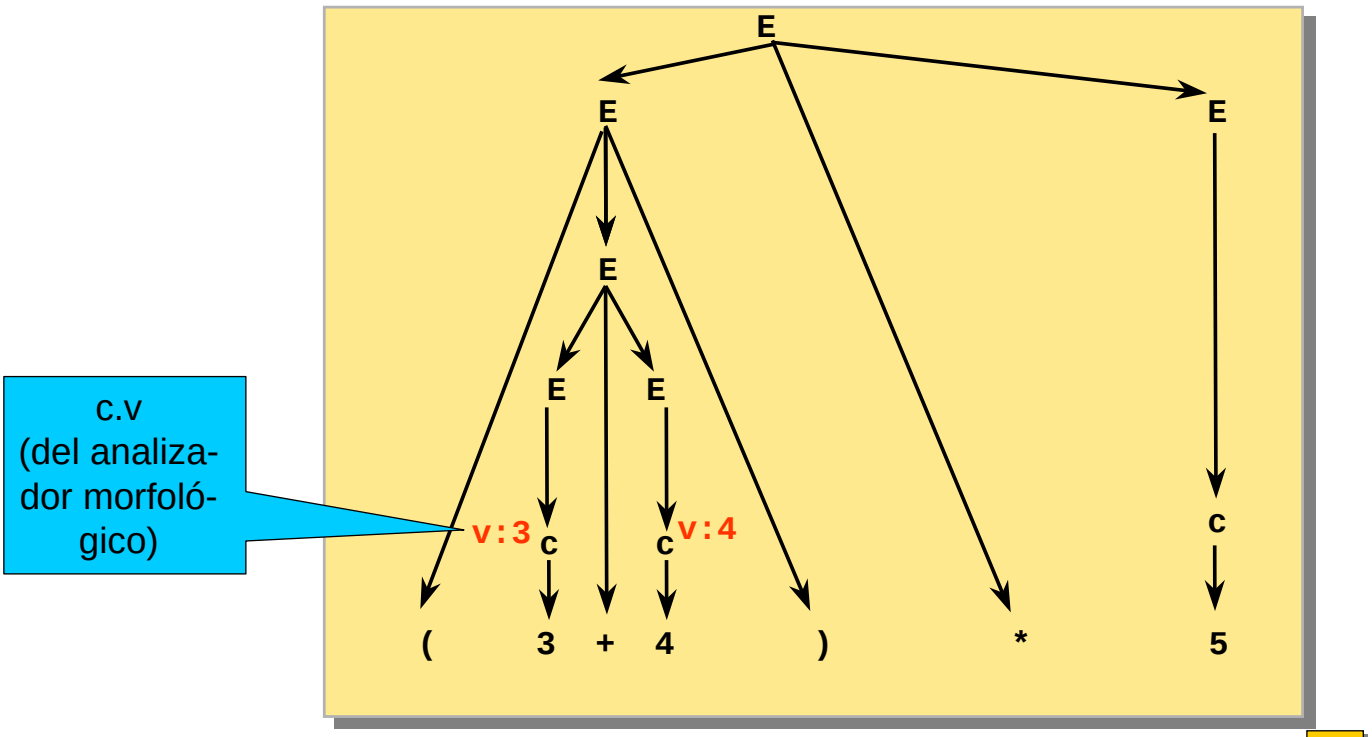
- En tercer lugar, añadimos las reglas semánticas:

$E \rightarrow E_1 + E_2$	{	E.v:= E_1.v + E_2.v }
$E \rightarrow E_1 - E_2$	{	E.v:= E_1.v - E_2.v }
$E \rightarrow -E_1$	{	E.v:= -E_1.v }
$E \rightarrow (E_1)$	{	E.v:= E_1.v }
$E \rightarrow c$	{	E.v:= c.v }

Gramática de atributos

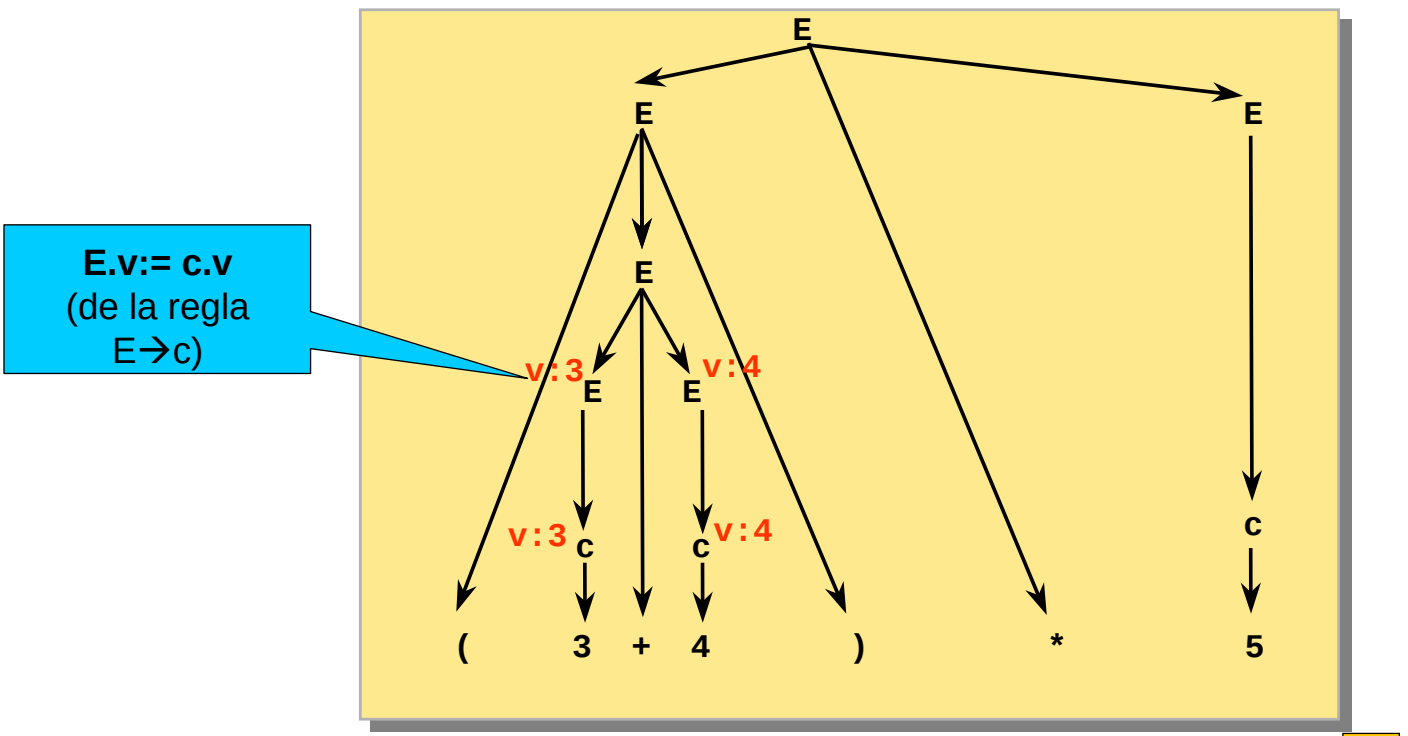
Ejemplo 1

Si analizamos la misma expresión, (3+4)*5



Gramática de atributos

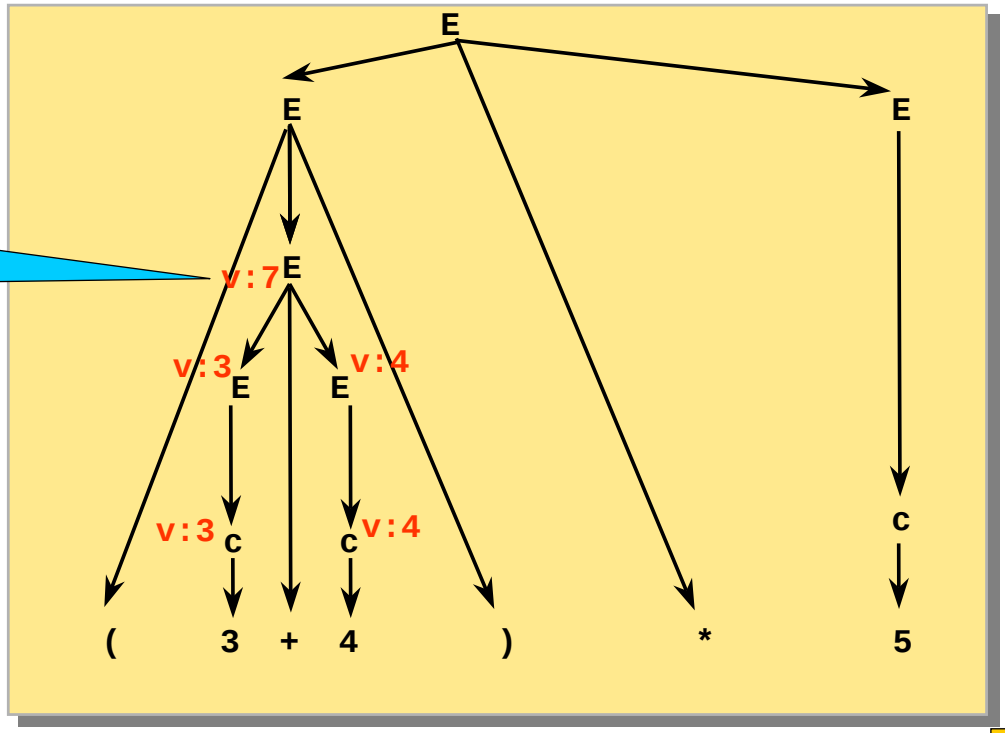
Ejemplo 1



Gramática de atributos

Ejemplo 1

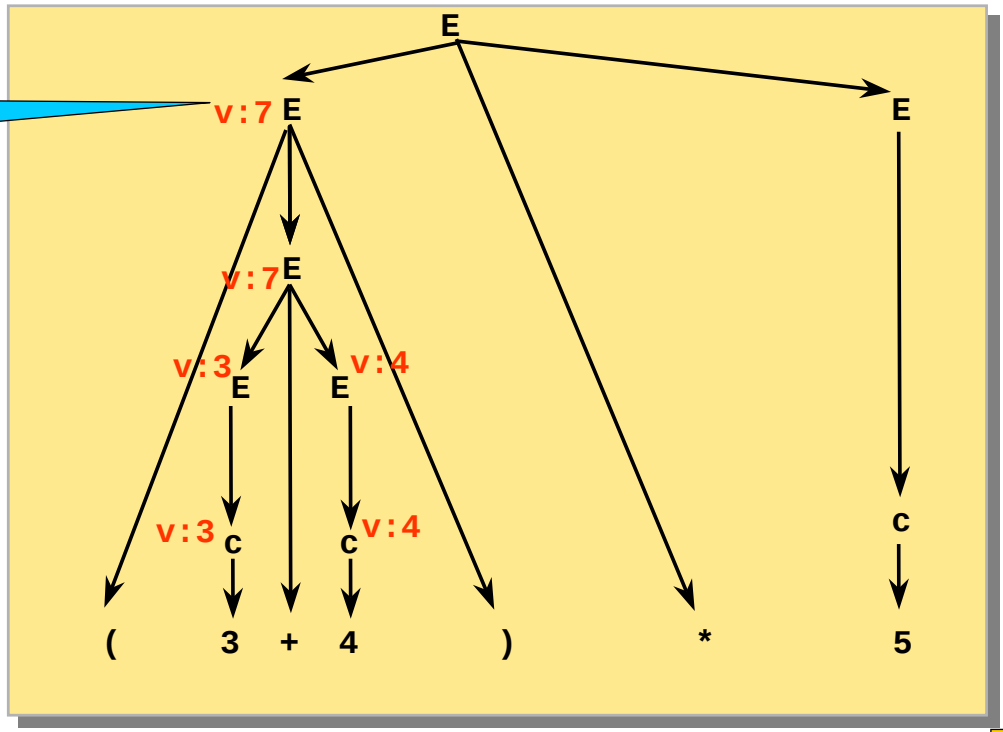
E.v:= E_1.v+E_2.v



Gramática de atributos

Ejemplo 1

E.v:= E_1.v

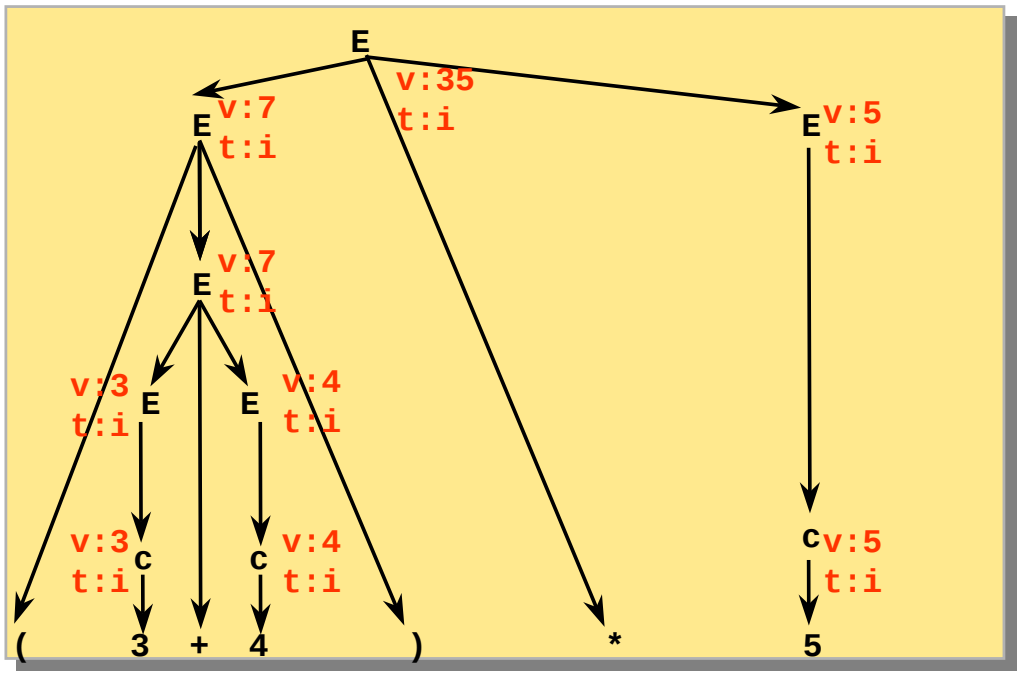


Gramática de atributos

Árbol de análisis anotado

Definición:

- El árbol resultado del análisis se puede comprobar como está anotado en cada nodo con los valores de los atributos.



Gramática de atributos

Ejemplo 2

- Si se quiere ahora añadir las reglas de declaración de identificadores vistas anteriormente

G_E={{int, real, i}, {D,T,L},
{
D→TL
T→int
T→real
L→L,i
L→i
},
D}

- En la que **i** se refiere a identificadores

Gramática de atributos

Ejemplo 2

- La gestión del tipo de los identificadores podría incluir la siguiente gramática

G_E={ {{int{tipo}(1), real{tipo}(1), i{tipo}},
{D{tipo},T{tipo},L{tipo}},
{
D→TL {L.tipo:=T.tipo},
T→int {T.tipo:=entero},
T→real {T.tipo:=real},
$L \rightarrow L_d$,i {L_d.tipo:=L.tipo; i.tipo := L.tipo},
L→i {i.tipo := L.tipo}
},
D}

  - (1) Estos valores deben ser proporcionados como entrada

Gramática de atributos

Ejemplo 2

- Se ha indicado que se puede añadir acciones semánticas cuyo efecto sea distinto que calcular el valor de otros atributos semánticos. Cuando esto ocurre (la acción tiene un **efecto lateral**, porque suele modificarse de forma permanente cierta información global a toda la gramática como efecto lateral de la acción)
- Una situación muy frecuente se encuentra cuando se construyen compiladores.
  - Es habitual necesitar información de los indentificadores declarados que es utilizada cuando son utilizados más adelante en el programa.
  - Habitualmente los compiladores tienen un diccionario (llamado tabla de símbolos) que es una estructura de datos donde se puede guardar esa información para recuperarla posteriormente.
  - En este ejemplo se muestra esta posibilidad.

Gramática de atributos

Ejemplo 2

- Se podría añadir acciones para modificar la tabla de símbolos cuando se determina el tipo de un identificador (en las últimas reglas):

```
G_E={   {{int{tipo}, real{tipo}, i{tipo}}(1),
        {D{tipo},T{tipo},L{tipo}},
    {
        D→TL {L.tipo:=T.tipo},
        T→int {T.tipo:=entero},
        T→real {T.tipo:=real},
        L→L_d,i {L_d.tipo:=L.tipo, i.tipo := L.tipo},
        L→i {i.tipo := L.tipo}
    },
    D}
```


Gramática de atributos

Ejemplo 2

- Supongamos que la acción deseada es realizada por la función
  añade_tipo(TablaSímbolos* pT, Identificador i, Tipo tipo)
- Donde:
  - **pT** es un apuntador a la tabla
  - **i** es el token devuelto por el analizador morfológico (suponemos que con la suficiente información como para acceder a su posición en la tabla de símbolos)

```
G_E={   {{int{tipo}, real{tipo}, i{tipo}}(1),
        {D{tipo},T{tipo},L{tipo}},
    {
        D→TL {L.tipo:=T.tipo},
        T→int {T.tipo:=entero},
        T→real {T.tipo:=real},
        L→L_d,i {L_d.tipo:=L.tipo, i.tipo := L.tipo,
                 añade_tipo(&T,i,L.tipo)},
        L→i {i.tipo := L.tipo, añade_tipo(&T,i,L.tipo)}
    },
    D}
```

Gramática de atributos

Ejemplo 3

- Se puede considerar ahora una gramática un poquito más complicada que incorpora las otras dos anteriores para permitir un sublenguaje de programación que admite declaración de variables y asignación de expresiones a las mismas

G_E={{int{tipo}, real{tipo}, i{tipo,valor}, +, *, (,)},
{D{tipo},T{tipo},L{tipo},E{valor,tipo},A{tipo},P},
{ P→DA,
A→i:=E {"i∈T"∧E.tipo=i.tipo⇒A.tipo:=i.tipo;
añade_valor(&T,i,E.valor);}
D→TL {L.tipo:=T.tipo, D.tipo:=L.tipo},
T→int {T.tipo:=entero},
T→real {T.tipo:=real},
$L→L_d$,i {L_d.tipo:=L.tipo,añade_tipo(&T,i,L.tipo)},
L→i {añade_tipo(&T,i,L.tipo)},
$E→E_1+E_2$ {E.v:= E_1.v + E_2.v }
$E→E_1-E_2$ {E.v:= E_1.v - E_2.v }
$E→-E_1$ {E.v:= -E_1.v }
$E→(E_1)$ {E.v:= E_1.v }
E→i {E.v:= valor_de(i,T) }
E→c {E.v:= c.v }},
P}

Gramática de atributos

Ejemplo 3

- Observe que hay una pareja de acciones semánticas con las que se mantiene el valor de los identificadores.
- En la siguiente regla (asignación de una expresión a un valor) se guarda en la tabla de símbolos el valor de la expresión como valor del identificador con la función `añade_valor`

A→i:=E {"i∈T"∧E.tipo=i.tipo⇒A.tipo:=i.tipo;
añade_valor(&T,i,E.valor);}

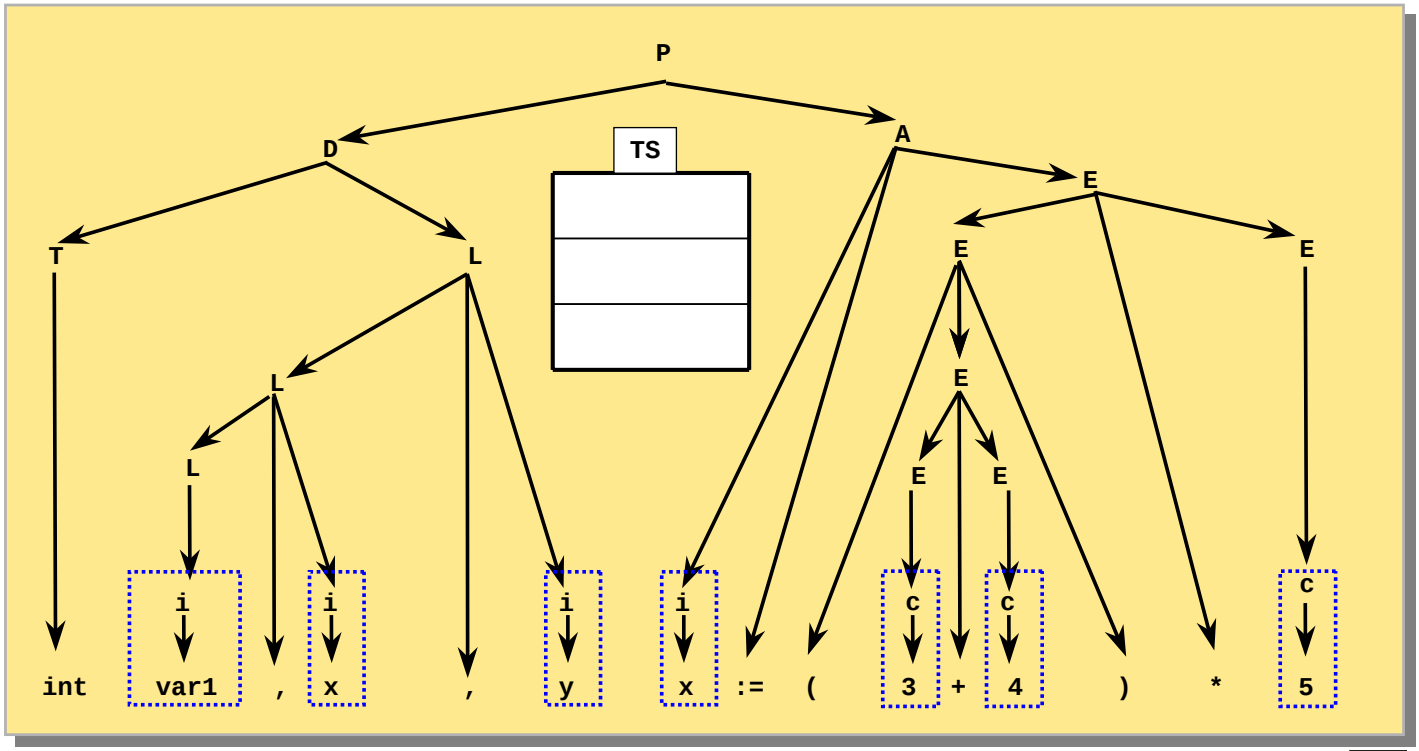
- Y en la siguiente regla (una expresión es un identificador) se recupera como valor de la expresión el del identificador guardado en la tabla de símbolos mediante la función `valor_de`

E→i {E.v:= valor_de(i,T) }

Definiciones dirigidas por la sintaxis

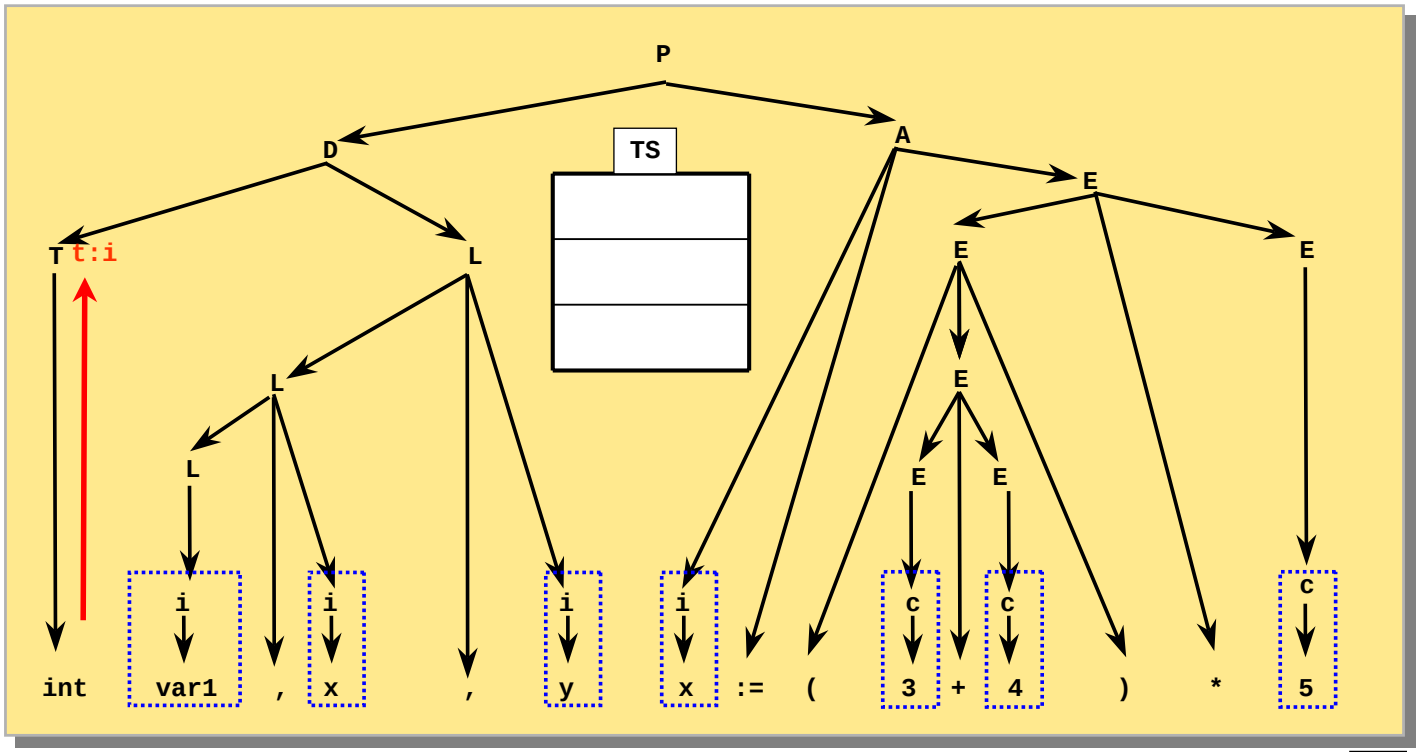
Ejemplos

- A continuación se muestra el tratamiento de **`int var1, x, y  x:=(3+4)*5`**



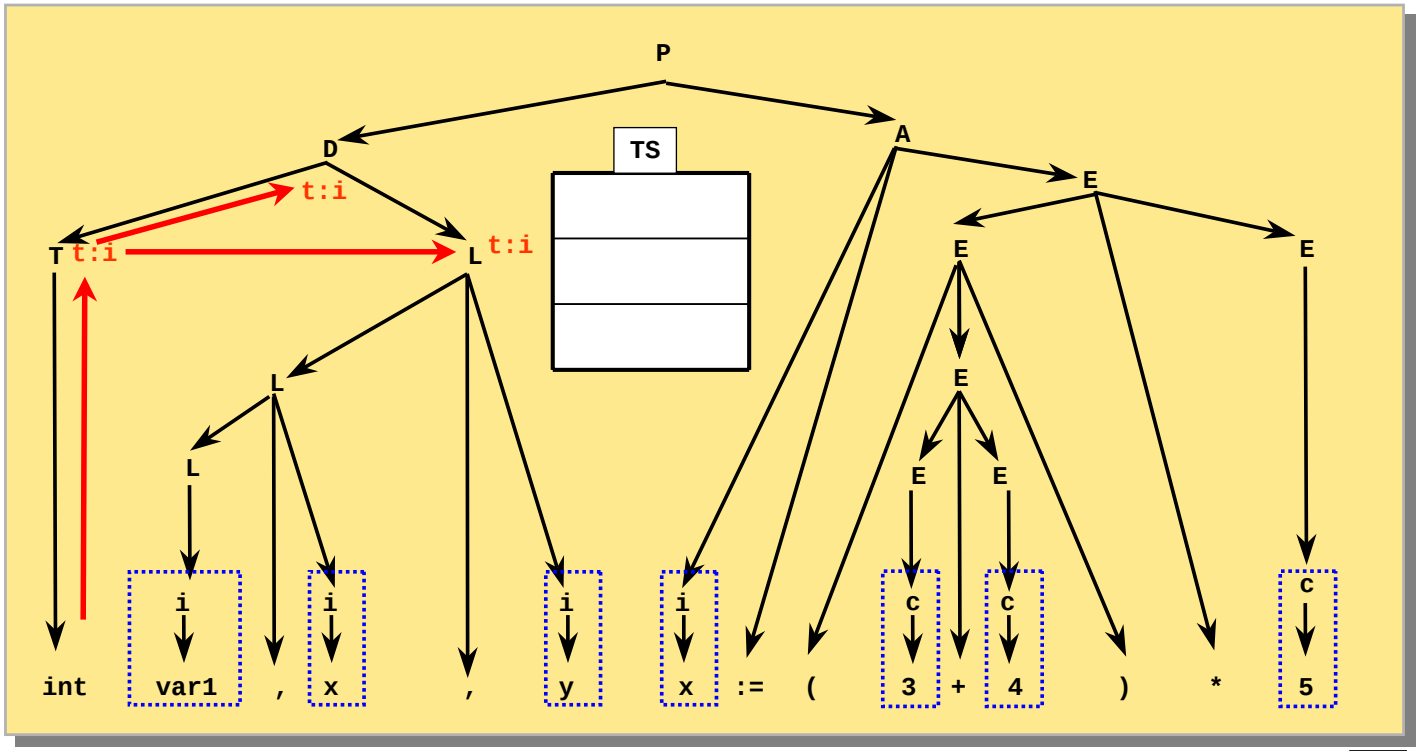
Definiciones dirigidas por la sintaxis

Ejemplos



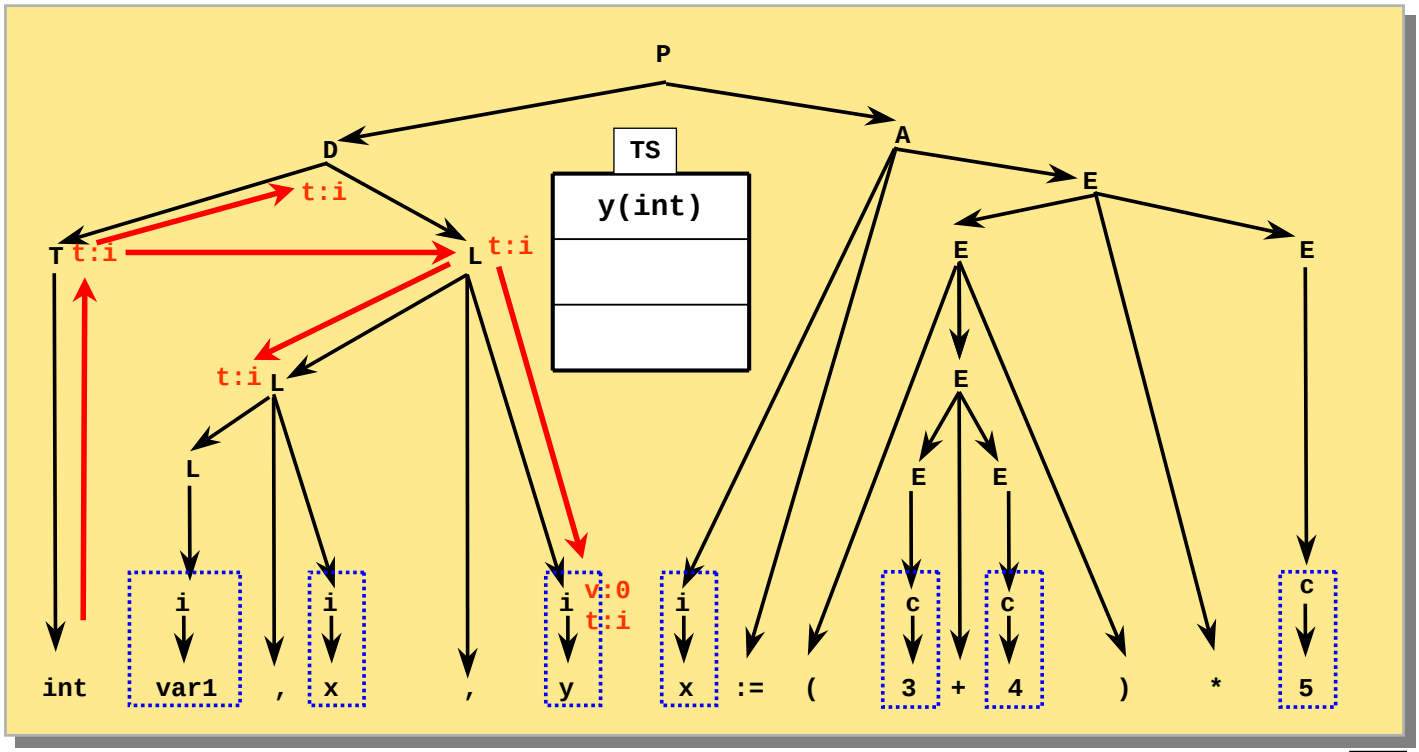
Definiciones dirigidas por la sintaxis

Ejemplos



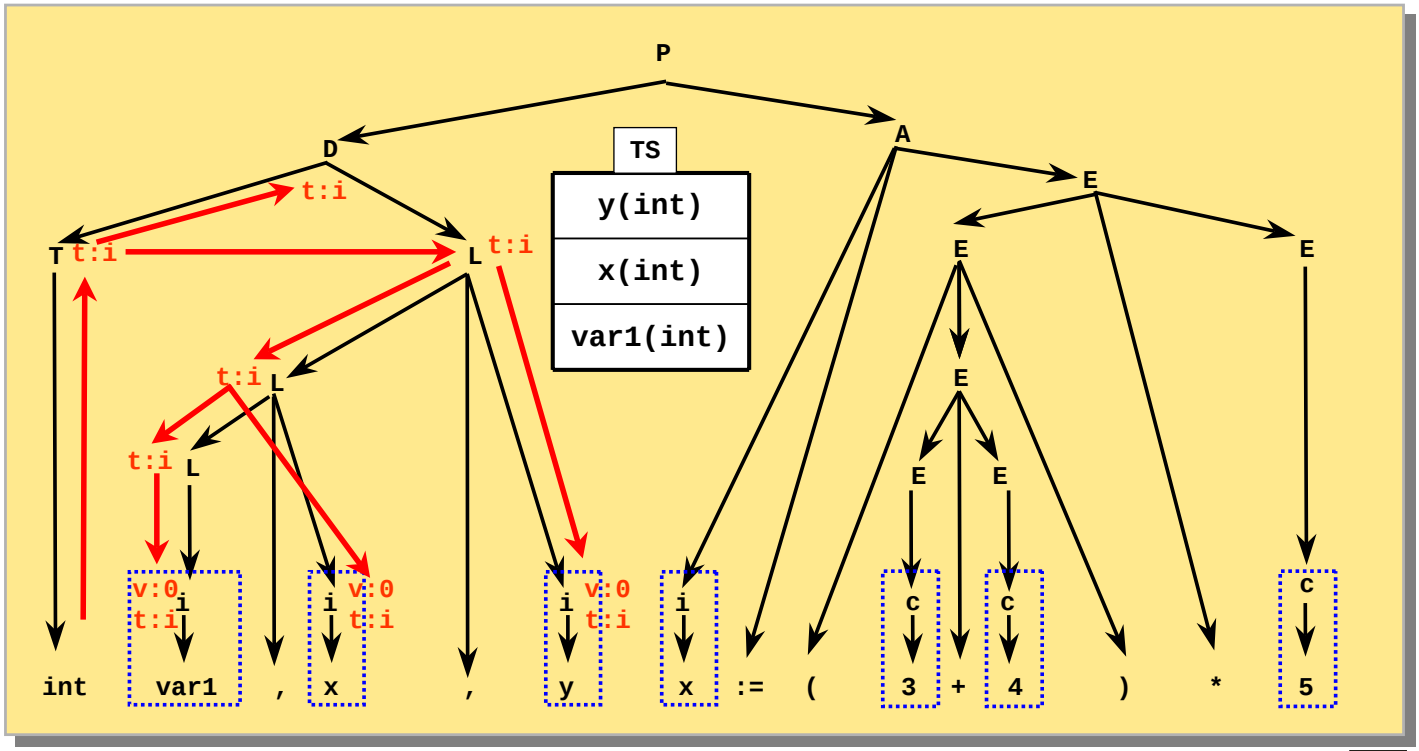
Definiciones dirigidas por la sintaxis

Ejemplos



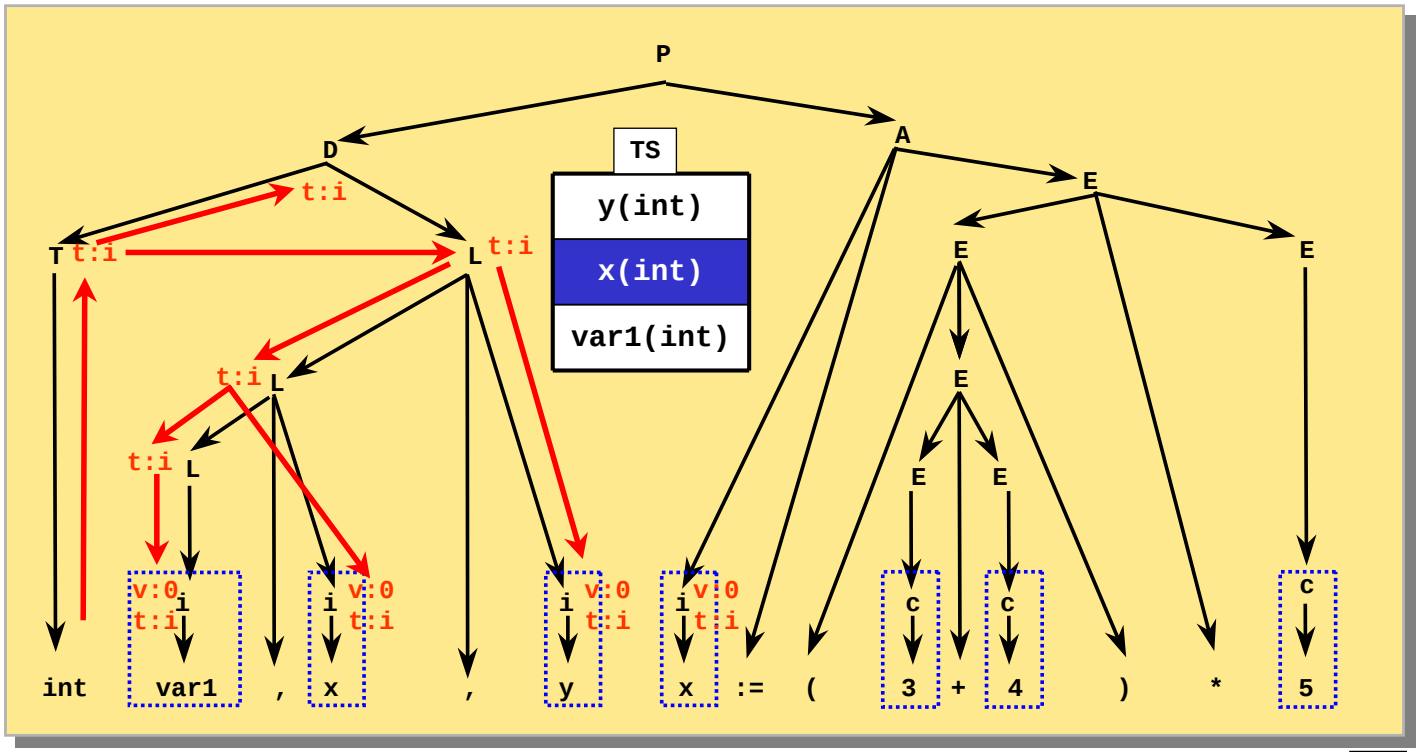
Definiciones dirigidas por la sintaxis

Ejemplos



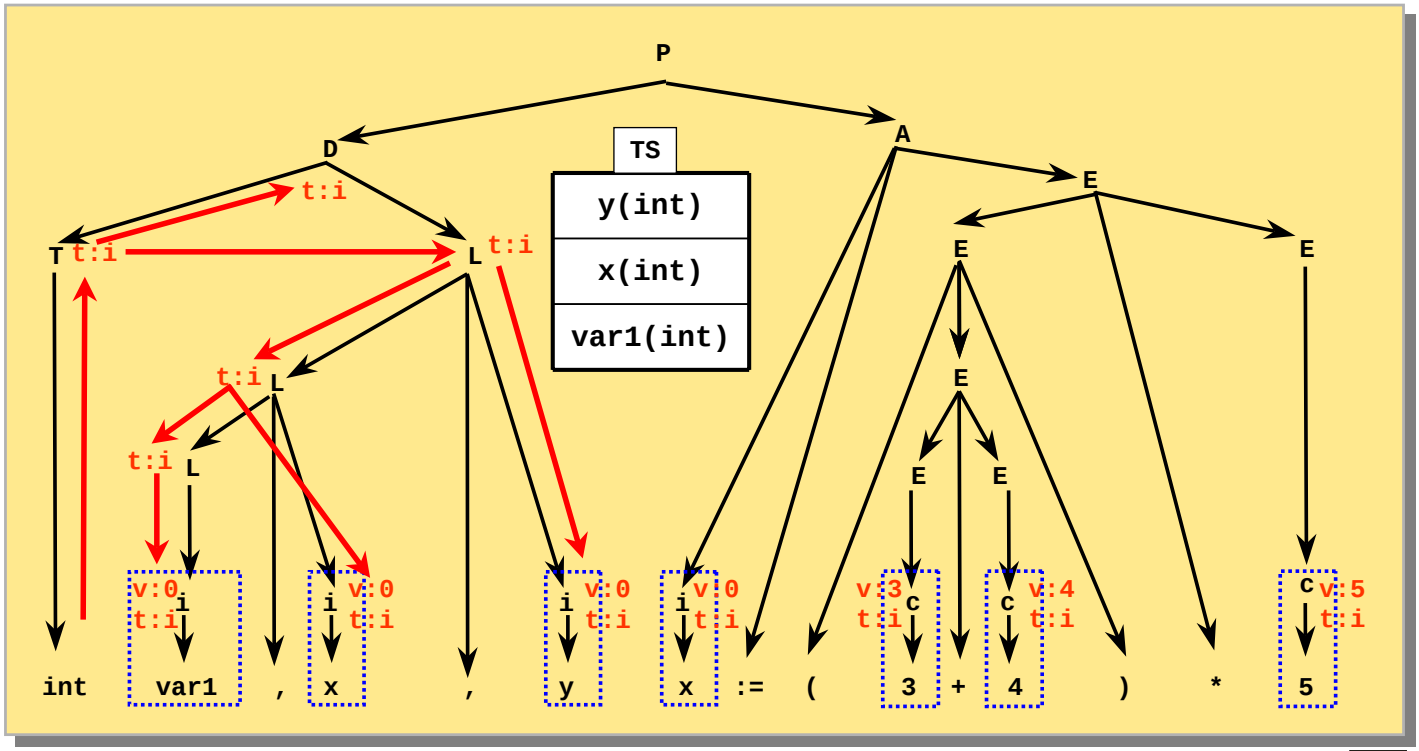
Definiciones dirigidas por la sintaxis

Ejemplos



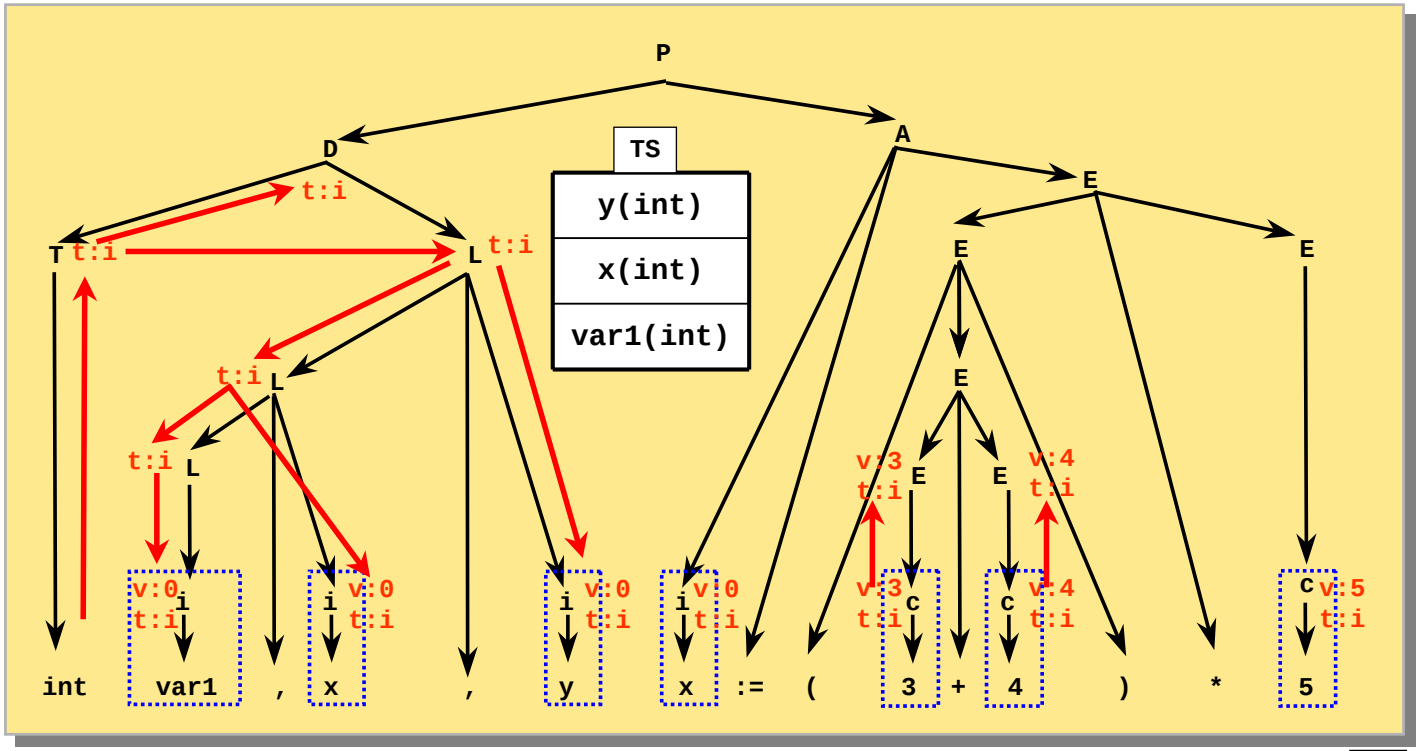
Definiciones dirigidas por la sintaxis

Ejemplos



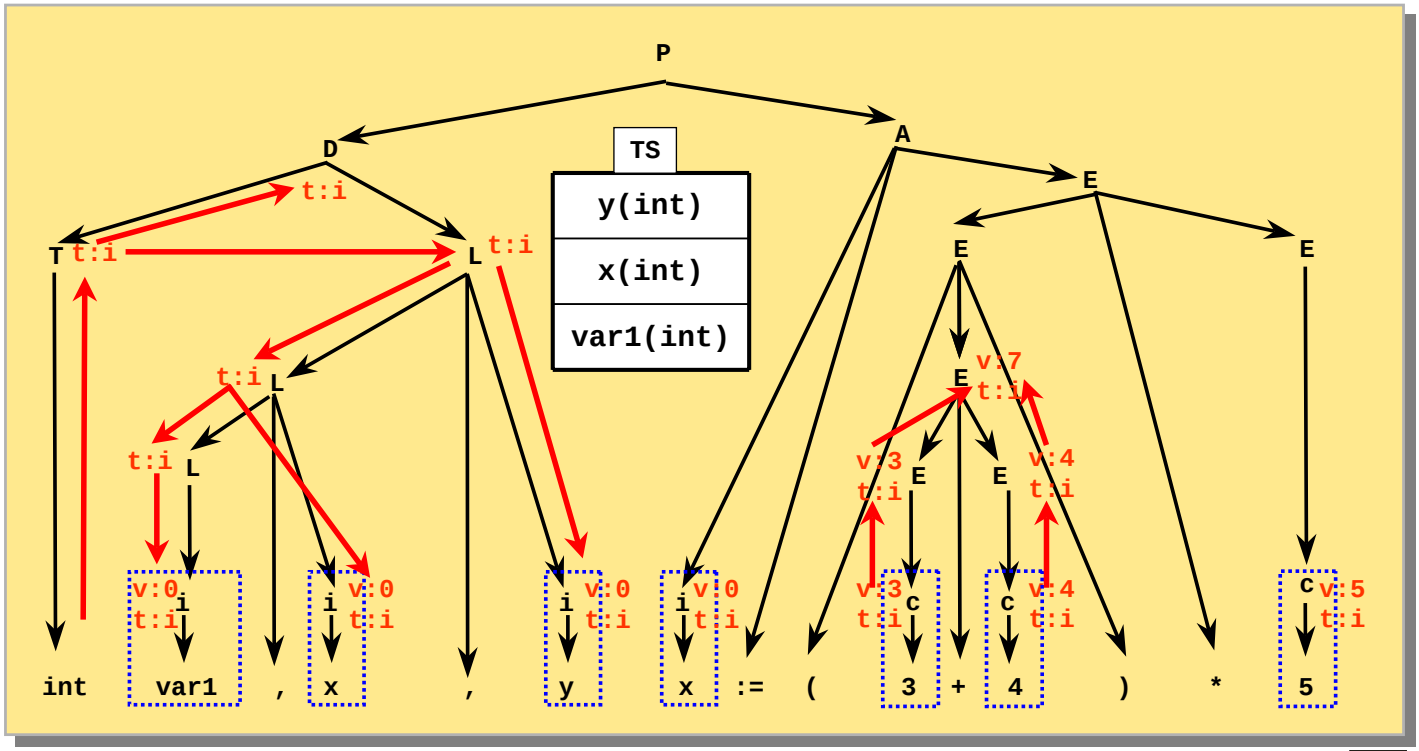
Definiciones dirigidas por la sintaxis

Ejemplos



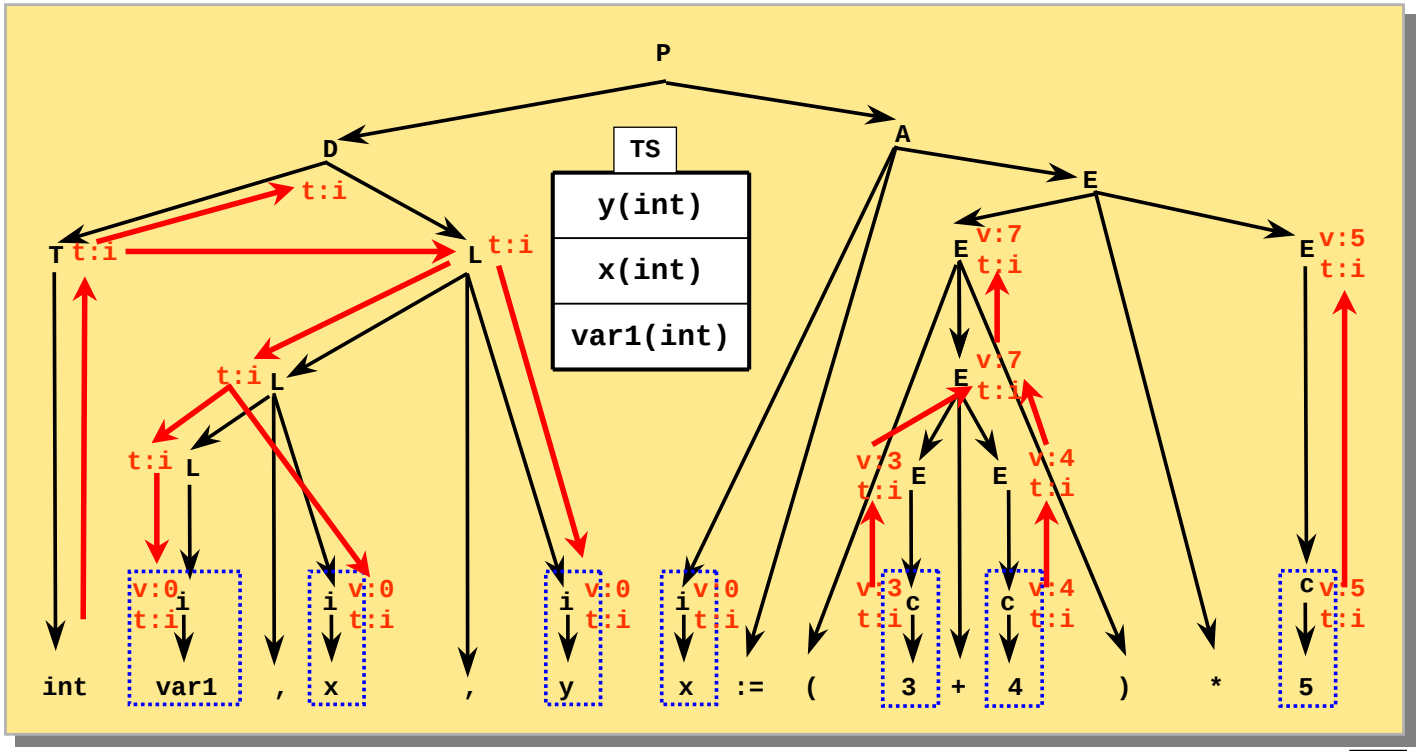
Definiciones dirigidas por la sintaxis

Ejemplos



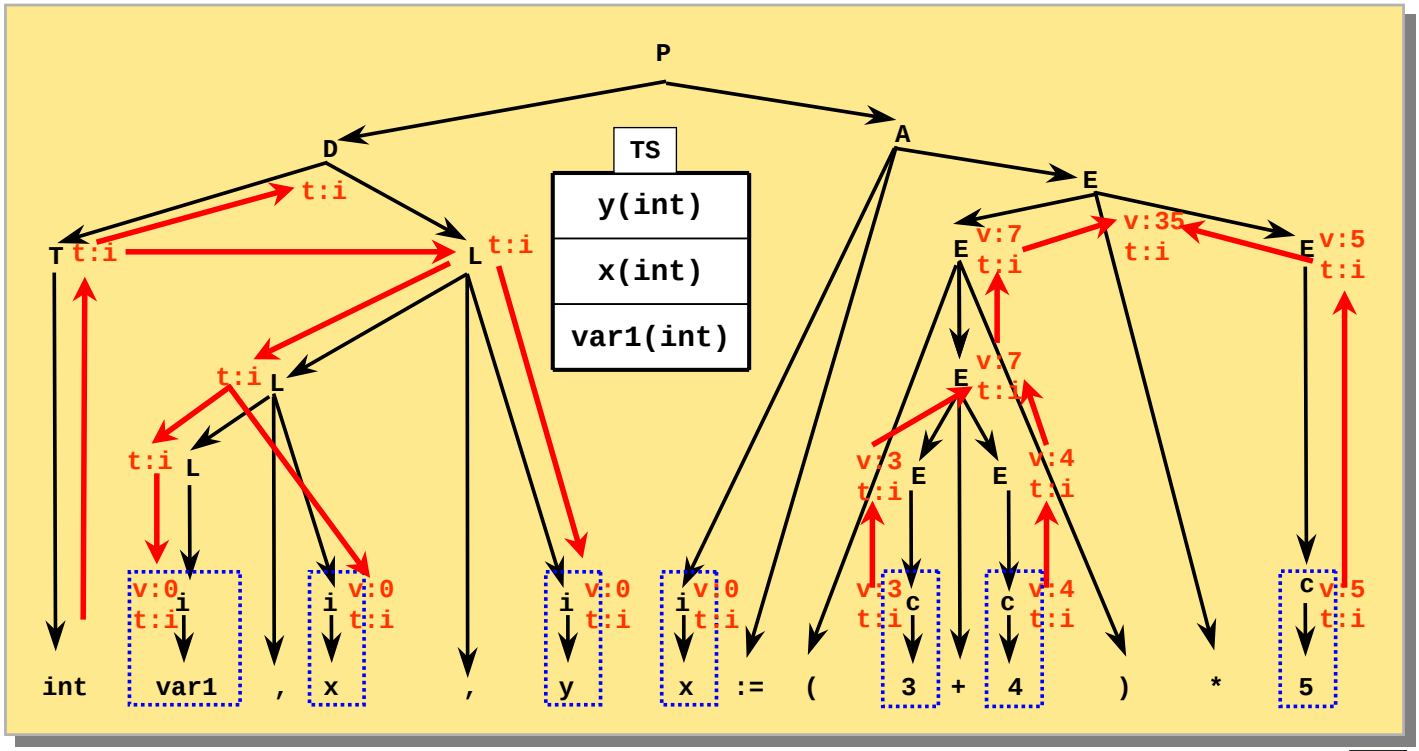
Definiciones dirigidas por la sintaxis

Ejemplos



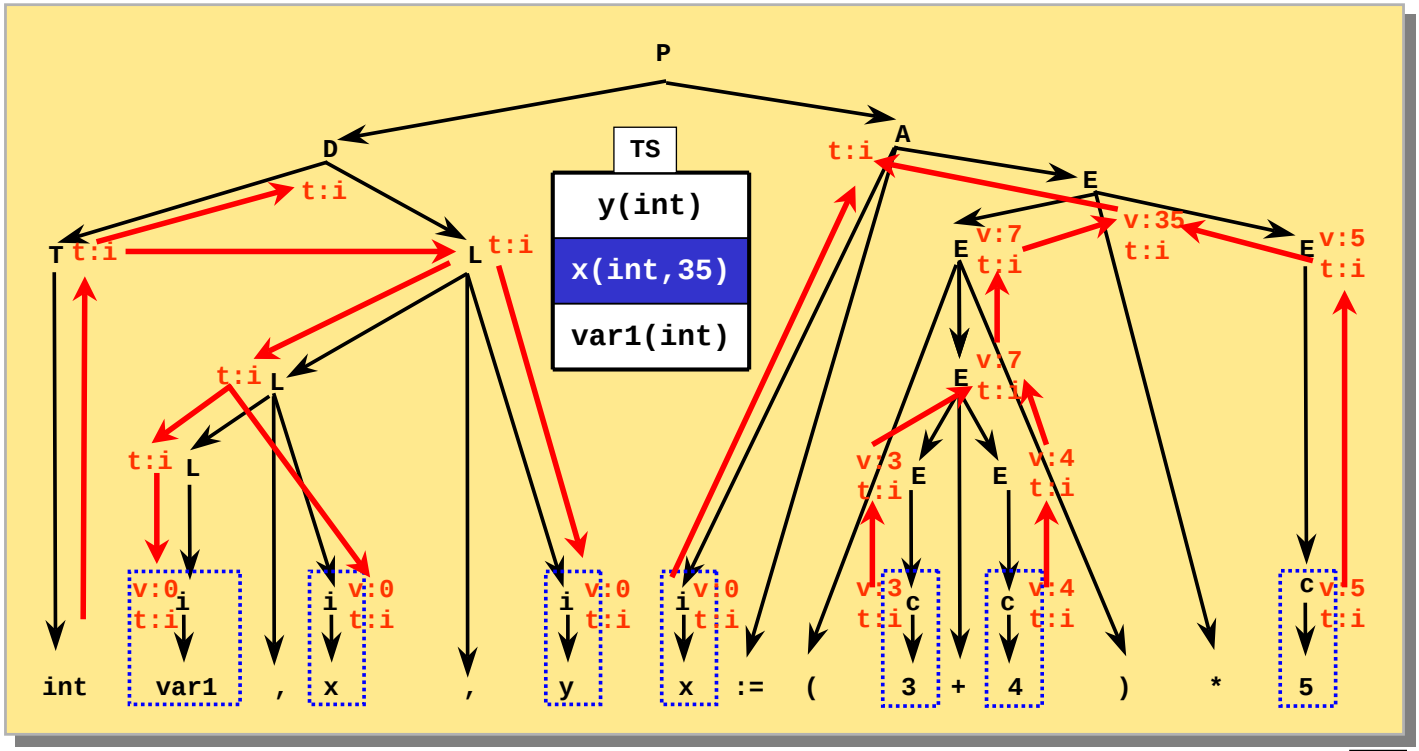
Definiciones dirigidas por la sintaxis

Ejemplos



Definiciones dirigidas por la sintaxis

Ejemplos



Gramática de atributos

Reglas semánticas y tipos de atributos

- Una regla semántica para un atributo sintetizado siempre tendrá el mismo símbolo en la parte izquierda de la regla de producción que en la regla semántica. Por ejemplo:

  Expr :- $Expr_1$ ’+’ $Expr_2$
  { **Expr**.v = $Expr_1$.v + $Expr_2$.v }

- Una regla semántica para un atributo heredado siempre tendrá un símbolo de la parte derecha de la regla de producción en la parte izquierda de la regla semántica. Por ejemplo:

  L :- $\mathbf{L_1}$, id { $\mathbf{L_1}$.t = L.t }

Gramática de atributos

Definición formal

- Una gramática de atributos es una quíntupla:

$$(\Sigma_T, \Sigma_N, S, P, K)$$

- Donde:
  - Se mantiene la estructura de las gramáticas independientes del contexto.
  - A cada símbolo de la gramática le acompaña la lista de sus atributos semánticos entre paréntesis. En los símbolos sin atributos semánticos se omitirán los paréntesis.

Gramática de atributos

Definición formal

- Las reglas de producción se modifican para distinguir distintas apariciones del mismo símbolo.
- A cada regla de producción le acompaña su acción semántica
  - Las instrucciones de la acción se escriben entre llaves.
  - Para referirse a un atributo semántico de un símbolo dentro de las acciones semánticas, se escribirá el nombre del atributo tras el del símbolo, separados por un punto '.'
  - Si alguna regla no necesita realizar ninguna acción semántica, se escribirá '{}'.
- La información global se añade como nueva componente adicional, al final de la gramática de atributos (K).

Gramática de atributos

Ejemplos

- A continuación se muestra la definición formal de la gramática del ejemplo de expresiones aritméticas:

{ Σ_T={+, *, (,), c(valor,tipo), i(valor,tipo)},
Σ_N={E(valor,tipo)},
E,
P={
E→E_i+E_d
{ E.valor = E_i.valor+E_d.valor;
E.tipo = E_i.tipo;},
E→-E_d,
{ E.valor = -E_d.valor;
E.tipo = E_d.tipo; },
E→E_i*E_d
{ E.valor = E_i.valor*E_d.valor;
E.tipo = E_i.tipo;},

Gramática de atributos

Ejemplos

```
E→(Ed)
 {   E.valor = Ed.valor;
     E.tipo = Ed.tipo},
E→i
 {    E.v:= valor_de(i,T),
      E.tipo = i.tipo},
E→c
 {   E.valor = c.valor,
     E.tipo = c.tipo}
},
K={T}}
```


Gramática de atributos

Ejemplos

- A continuación se muestra la definición formal de la gramática del ejemplo de declaración de identificadores (en la que se omite la gestión de la tabla de símbolos para proporcionar un ejemplo sin información constante):

```
{ ΣT={int(tipo), real(tipo), i(tipo)},
     ΣN={D(tipo),T(tipo),L(tipo)},
  D,
  P={  D→TL
       {      L.tipo = T.tipo;
              D.tipo = T.tipo;},
       T→int {T.tipo = entero;},
       T→real {T.tipo = real;},
       L→Ld,i
       {      Ld.tipo = L.tipo;
              i.tipo = L.tipo;},
       L→i {i.tipo = L.tipo;}
   },
  K=Φ}
```

4.3

Nociones de programación

Gramática de atributos: nociones de programación

Introducción

- Las gramáticas de atributos (la demostración queda fuera del ámbito del curso) son computacionalmente completas por lo que pueden ser consideradas un lenguaje de programación en sí mismas.
- A continuación se van a analizar algunos ejemplos que pueden ilustrar las peculiaridades de este nuevo lenguaje de programación y, de cuyas soluciones, pueden extraerse técnicas aplicables a otros problemas.

Gramática de atributos: nociones de programación

Lenguaje de paréntesis

- Algunos de los ejemplos consideran el **lenguaje de los paréntesis**:

> Cadenas compuestas exclusivamente por paréntesis balanceados y todos incluidos en un paréntesis exterior.

- Las siguientes cadenas pertenecen a este lenguaje:

```
( ( ) ( ) )
(   (  ( )( ( ) )( )  )   )
```

- Las siguientes cadenas no pertenecen a este lenguaje:

```
( ( )
( ) ( )
```


Gramática de atributos: nociones de programación

Gramática independiente del contexto para el lenguaje de paréntesis

- Puede considerarse la siguiente gramática independiente del contexto de partida para este lenguaje

{
Σ_T={ (,) },
Σ_N={ <lista>, <lista_interna>},
<lista>,
P={
<lista>→(<lista_interna>),
<lista_interna>→<lista_interna>(<lista_interna>),
<lista_interna>→λ
}
}

Gramática de atributos: nociones de programación

Diseño de gramáticas de atributos: ejemplo 1

- Se desea diseñar una gramática de atributos que sea capaz de calcular la **profundidad de una palabra** cualquiera del lenguaje de paréntesis.
- Por ejemplo:
  - ( ), tiene profundidad 1
  - ***(*** ( ) ***(( )))***, tiene profundidad 3, debido a la sublista derecha

Gramática de atributos: nociones de programación

Diseño de gramáticas de atributos: ejemplo 1

- La siguiente gramática de atributos puede resolver el problema:

{

Σ_T={ (,) },
Σ_N={ <lista>(entero profundidad;),
<lista_interna>(entero profundidad)},
<lista>,
P={ <lista>→(<lista_interna>)
{<lista>.profundidad= <lista_interna>.profundidad + 1;
IMPRIMIR("PROFUNDIDAD:“,<lista>.profundidad);},

Gramática de atributos: nociones de programación

Diseño de gramáticas de atributos: ejemplo 1

- La siguiente gramática de atributos puede resolver el problema:

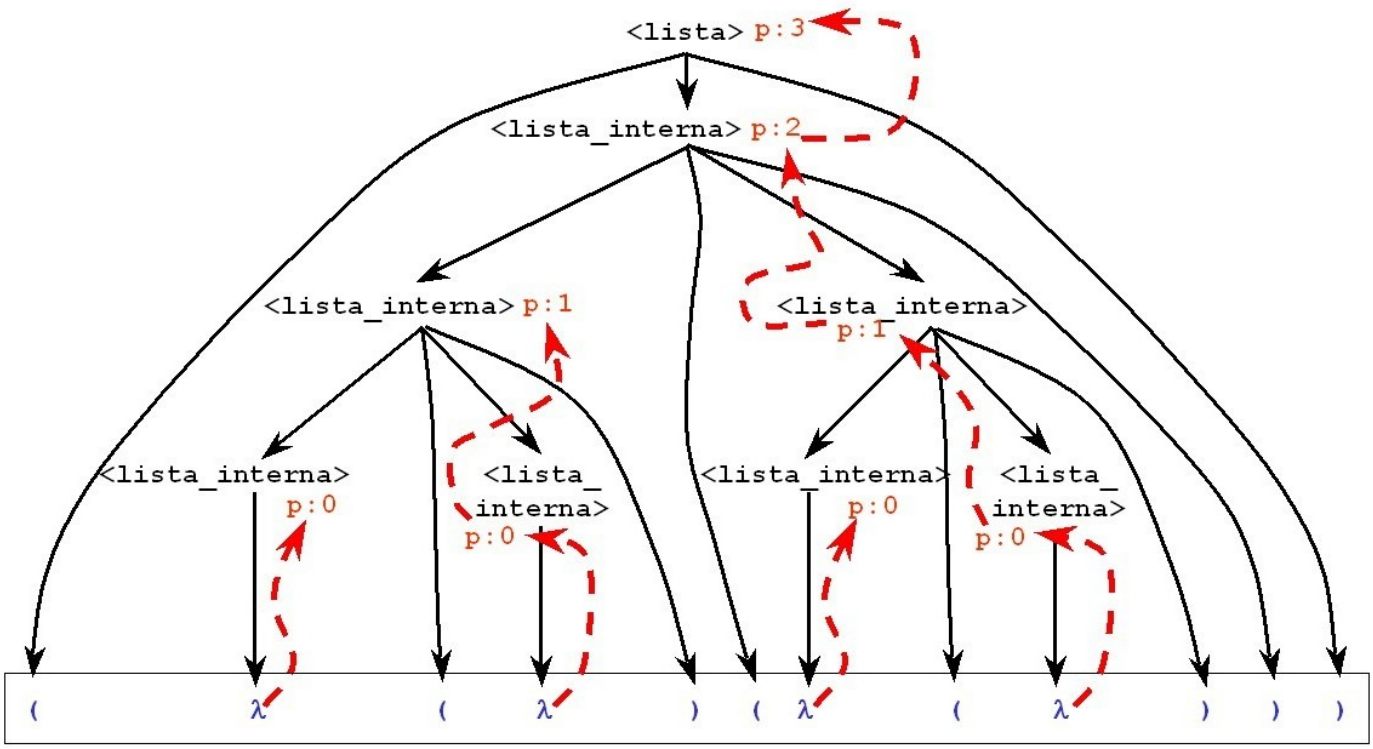
<lista_interna>→<lista_interna>¹ (<lista_interna>²)
{SI(<lista_interna>¹.profundidad > <lista_interna>².profundidad)
<lista_interna>.profundidad=<lista_interna>¹.profundidad ;
EN OTRO CASO
<lista_interna>.profundidad=<lista_interna>².profundidad+1;},

<lista_interna>::=λ
{ <lista_interna>.profundidad = 0;}
},
K=Φ}

Gramática de atributos: nociones de programación

Diseño de gramáticas de atributos: ejemplo 1

- A continuación se muestra gráficamente un ejemplo:



Gramática de atributos: nociones de programación

Ejemplo 1: conclusiones

- Obsérvese que **sólo** ha sido necesario utilizar **atributos sintetizados**.
- La razón es que **cada fragmento** de la entrada (cada lista) **puede calcular toda su información** (profundidad) **sin consultar otras partes** de la cadena que estén a su derecha o a su izquierda.

Gramática de atributos: nociones de programación

Diseño de gramáticas de atributos: ejemplo 2

- Se desea diseñar una gramática de atributos que sea capaz de calcular **el número de listas de una palabra** cualquiera del lenguaje de paréntesis.
- Por ejemplo:
  - ( ), contiene una lista
  - (( ) (( ))) contiene 4 listas
  - (( ) ( ) ( ) ( )) contiene 5 listas
  - ((((( ))) ( ))) contiene 6 listas

Gramática de atributos: nociones de programación

Diseño de gramáticas de atributos: ejemplo 2

- La siguiente gramática podría solucionar el ejmplo:

```
{ ΣT={ (, ) },
  ΣN={ <lista>(entero num_listas_total;),
       <lista_interna>(    entero num_listas_antes;
                           entero num_listas_despues)},
  <lista>,
  P={  <lista>→(  <lista_interna> )
       { <lista_interna>.num_listas_antes = 1;
         <lista>.num_listas_total=
              <lista_interna>.num_listas_despues;
         IMPRIMIR("ELEMENTOS“,<lista>.num_listas_total);},
```


Gramática de atributos: nociones de programación

Diseño de gramáticas de atributos: ejemplo 2

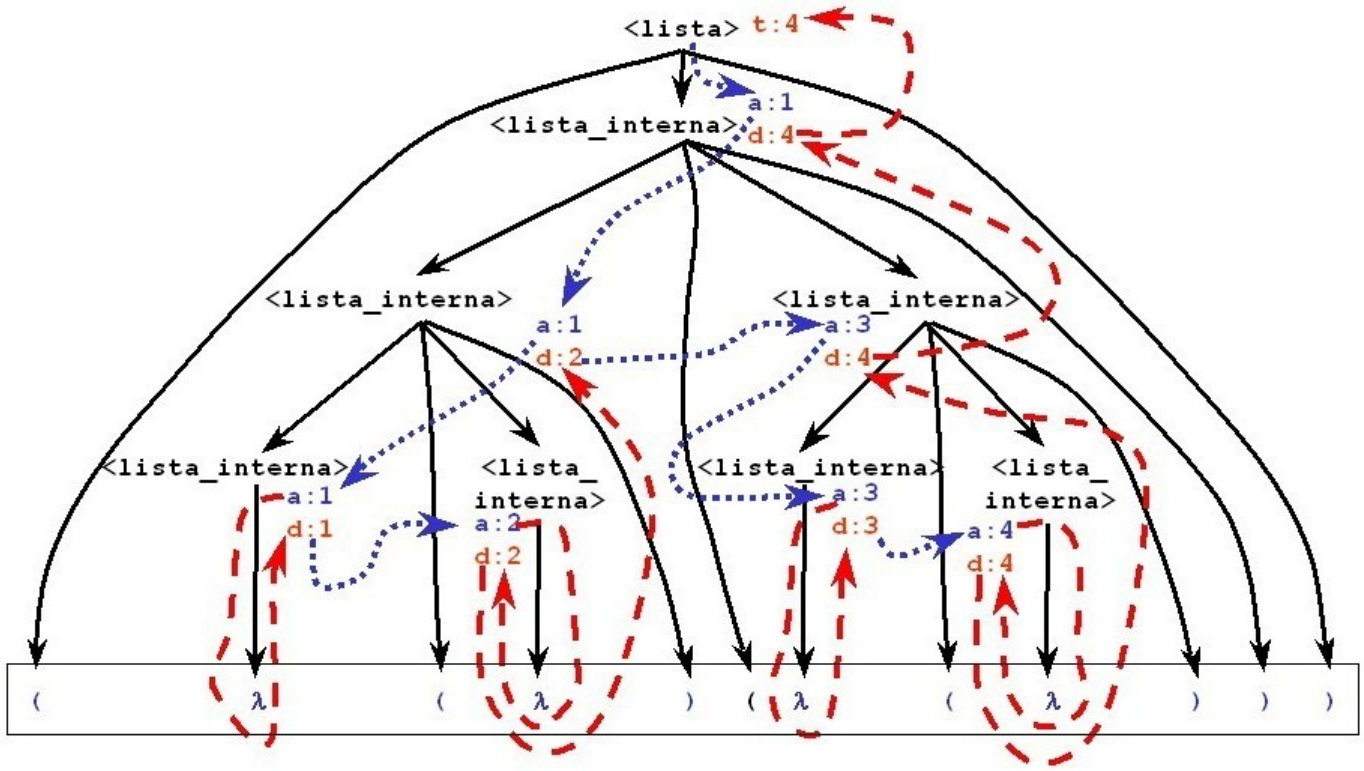
```
       <lista_interna>→<lista_interna>¹(<lista_interna>²)
       {
         <lista_interna>¹.num_listas_antes=
                         <lista_interna>.num_listas_antes;
         <lista_interna>².num_listas_antes =
                          <lista_interna>¹.num_listas_despues+1;
         <lista_interna>.num_listas_despues =
                     <lista_interna>².num_listas_despues;
       },

       <lista_interna>→λ
       {
        <lista_interna>.num_listas_despues =
                     <lista_interna>.num_listas_antes;
       }},
  K=Φ}
```

Gramática de atributos: nociones de programación

Diseño de gramáticas de atributos: ejemplo 2

- A continuación se muestra gráficamente un ejemplo de su funcionamiento:



Gramática de atributos: nociones de programación

Ejemplo 2: conclusiones

- Obsérvese que **se ha necesitado herencia**.
- La razón es que **el número de listas total tiene que acumular el de todas las partes de la cadena** en el **sentido de lectura** (de izquierda a derecha)

Gramática de atributos: nociones de programación

Diseño de gramáticas de atributos: ejemplo 3

- La siguiente gramática también soluciona el último ejemplo

```
{ Σ_T={ (, ) },
  Σ_N={<lista>(), <lista_interna>()},
  <lista>,
  P={  <lista>→( <lista_interna> )
       {      num_elementos = num_elementos + 1;
              IMPRIMIR("HAY ",num_elementos, " LISTAS" );},

       <lista_interna>→<lista_interna> ( <lista_interna> )
       {      num_elementos++;},

       <lista_interna>→λ{ }},
  K={entero num_elementos=0;}
}
```


Gramática de atributos: nociones de programación

Conclusiones a los ejemplos

- Obsérvese que **se ha simplificado la gramática mediante la sustitución de un atributo heredado por la manipulación de información global**.
- Esta técnica es generalizable.

Notación definitiva para las gramáticas de atributos

Gramáticas de atributos

- Uno de los órdenes de recorrido de los árboles en general y también en particular de los árboles de derivación más **interesantes** es el **recorrido en profundidad por la izquierda con retroseguimiento**.
  - Permite todos los tipos de interés de gestión de información:
    - **Global**.
    - Atributos **sintetizados**
    - Atributos **heredados**
      - De **padres** a hijos
      - De hermanos desde la **izquierda**.

Notación definitiva para las gramáticas de atributos

Gramáticas de atributos

- Por otro lado, **algunas construcciones** (en especial el uso de información **global**) puede dificultar la comprensión de las gramáticas de atributos
- Recuerde la gramática de uno de los ejemplos del apartado dedicado a las “nociones de programación”

```
{ Σ_T={ (, ) },
  Σ_N={<lista>(), <lista_interna>()},
  <lista>,
  P={  <lista>→( <lista_interna> )
       {      num_elementos = num_elementos + 1;
              IMPRIMIR(“HAY ”,num_elementos, “ LISTAS” );},

       <lista_interna>→<lista_interna> ( <lista_interna> )
       {      num_elementos++;},

       <lista_interna>→λ{ }},
  K={entero num_elementos=0;}
}
```

4.4

Notación definitiva para gramáticas de atributos

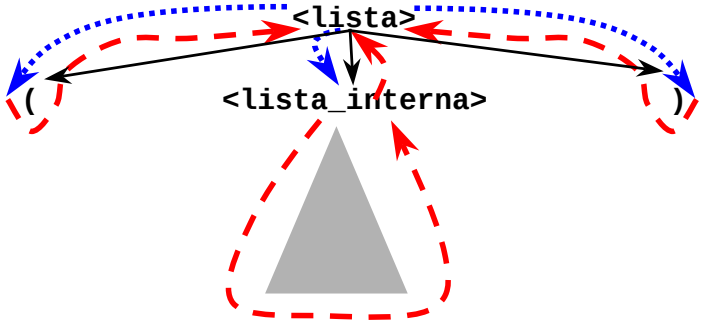
Notación definitiva para las gramáticas de atributos

Indicación expresa del momento de ejecución de las acciones

- Suponga para la evaluación de sus atributos en la cadena ejemplo

  (( ) (( )))

- El nodo asociado con la regla que imprime el mensaje está en la raíz del árbol.



- Es visitado en múltiples ocasiones como muestra la figura. En cuál de ellas debe ejecutarse la impresión del mensaje.
  - En el momento de crear los hijos se mostraría el valor inicial del contador (0)
  - Tras la finalización del último hijo se mostraría el valor final (4).
  - Cualquier otro instante mostrará un valor intermedio.

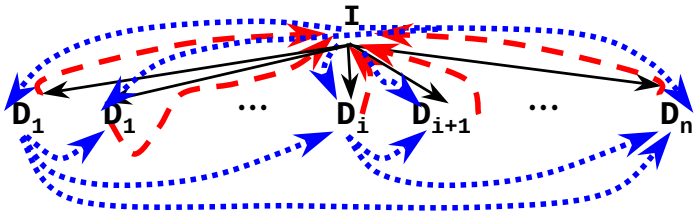
Notación definitiva para las gramáticas de atributos

Indicación expresa del momento de ejecución de las acciones

- En este problema parece claro que el instante correcto es el segundo. Sin embargo, queda claro que **en otros problemas puede ser necesario otro momento** y, sobre todo, que **la notación actual no permite distinguirlos**.
- Para solucionar este problema **se extenderá la notación de las acciones semánticas** presentada hasta este momento de la siguiente manera:
  - En general las reglas de una gramática de atributos tendrán la siguiente forma

$$I \rightarrow D_1 \ \ldots \ D_i \ \ldots \ D_n$$

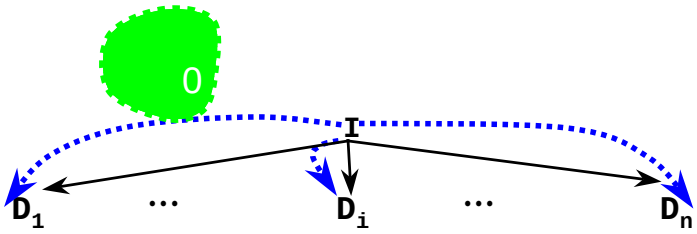
  - Y distinguiremos (no es la única opción) los siguientes posibles instantes para la ejecución de las acciones semánticas basándonos en las veces en las que cada nodo es visitado por el recorrido más potente en profundidad por la izquierda con retroseguimiento
  - La siguiente gráfica muestra todas las posibles propagaciones de la regla.



Notación definitiva para las gramáticas de atributos

Indicación expresa del momento de ejecución de las acciones

- **Instante 0**
  - Creación de los nodos hijos (parte derecha) del de la parte izquierda.

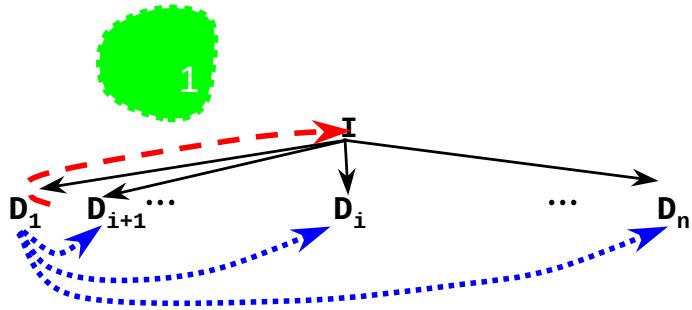


  - Obsérvese que en este instante sólo se puede ejecutar
    - Herencia de atributos de los hijos desde el padre
    - Acciones sobre información global.

Notación definitiva para las gramáticas de atributos

Indicación expresa del momento de ejecución de las acciones

- **Instante 1**
  - Cuando se ha terminado con el proceso del árbol del primer símbolo de la parte derecha.

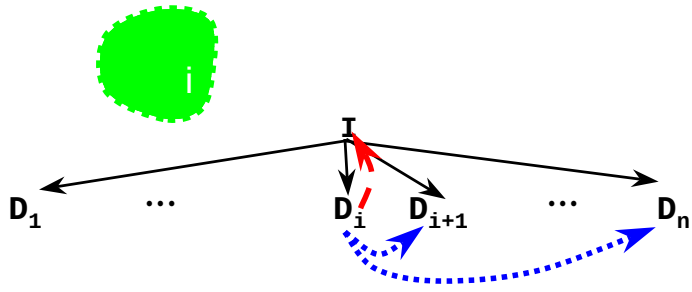


  - Obsérvese que en este instante sólo se puede ejecutar
    - Herencia de atributos de los hermanos a la derecha del símbolo D_1 a partir de los atributos de D_1
    - Síntesis de atributos del padre (**I**) que dependan de D_1.
    - Acciones sobre información global.

Notación definitiva para las gramáticas de atributos

Indicación expresa del momento de ejecución de las acciones

- **Instante i** (i de 1 a n, donde n es el número de símbolos de la parte derecha)
  - Cuando se ha terminado con el proceso de los árboles de todos los símbolos de la parte derecha desde el primero a la izquierda hasta el i-ésimo incluído.

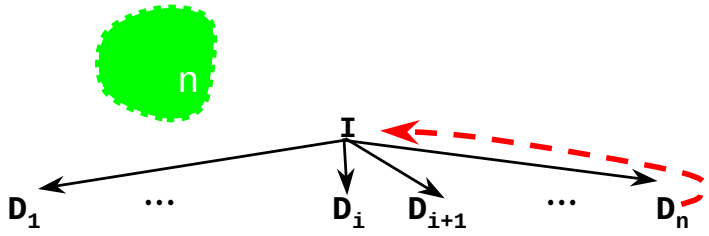


  - Obsérvese que en este instante sólo se puede ejecutar
    - Herencia de atributos de los hermanos a la derecha del símbolo D_i a partir de los atributos de D_i (o de D_i junto con otros símbolos anteriores por la izquierda)
    - Síntesis de atributos del padre (**I**) que dependan de D_i sólo o con otros valores a la izquierda de D_i.
    - Acciones sobre información global.

Notación definitiva para las gramáticas de atributos

Indicación expresa del momento de ejecución de las acciones

- **Instante n**
  - Cuando se ha terminado con el proceso de los árboles de todos los símbolos de la parte derecha desde el primero a la izquierda hasta el último.



  - Obsérvese que en este instante sólo se puede ejecutar
    - Síntesis de atributos del padre (**I**) que dependan de D_n sólo o con otros valores de nodos a la izquierda de D_n.
    - Acciones sobre información global.
    - Observe también que se podría considerar retrasar la síntesis de los atributos del padre (I) a este instante

Notación definitiva para las gramáticas de atributos

Indicación expresa del momento de ejecución de las acciones

- **Notación para las acciones de las reglas.**
  - Se dividirá en bloques el cuerpo de las acciones semánticas.
  - Cada bloque estará identificado con uno de los números descritos en las trasparencias anteriores y tendrá ese significado.
  - Por lo tanto, la componente **P** (producciones) de la notación explicada anteriormente se extiende de la siguiente manera

```
P={  ...
     I→ D1 ... Di ... Dn
     0:{   }
     1:{   }
     ...
     i:{   }
     ...
     n:{   },

     ...
   }
```

Notación definitiva para las gramáticas de atributos

Indicación expresa del momento de ejecución de las acciones

- **Restricciones del modelo.**
  - Y para considerar que las acciones son correctas las acciones semánticas y los atributos deben cumplir las siguientes restricciones:
    - **Cada símbolo** de la gramática tendrá una **lista de atributos sintetizados** y **otra de heredados** (potencialmente vacías) que **deben mantenerse en todas las reglas**.
    - Los **bloques se ejecutan en el orden** en el que el **recorrido en profundidad por la izquierda con retroseguimiento** los ejecutaría.
    - **Dentro** de cada **bloque** las acciones se ejecutan **en el orden en el que aparecen**.
    - **Cada acción** debe ejecutarse **solamente una vez**.

- **Ejemplos**
  - Se verán cuando se solucionen los ejercicios.

Notación definitiva para las gramáticas de atributos

Indicación expresa del momento de ejecución de las acciones: observaciones

- Es importante realizar las siguientes reflexiones:
  - Esta información es redundante para aquellas acciones que implementan
    - Síntesis
    - Herencia
  - Ya que la dependencia de sus atributos induce un orden que siempre es compatible con el descrito en la extensión de la notación.
  - Esta información puede ser imprescindible para saber cuándo se deben ejecutar las acciones que manipulan información global para asegurar la corrección de la gramática.
  - En estas trasparencias aparecen ejemplos de gramáticas de atributos en los que esta extensión a la notación no se utiliza ya que el contexto y las acciones dejan claro el momento en que deben ejecutarse.