11-8-2016

Material de apoyo

Clase No. 1, 2 y 3

Tutor académico: Jordy Alexander Gonzalez Catalán

ORGANIZACIÓN LENGUAJES Y COMPILADORES 2

Clase No. 1 y 2

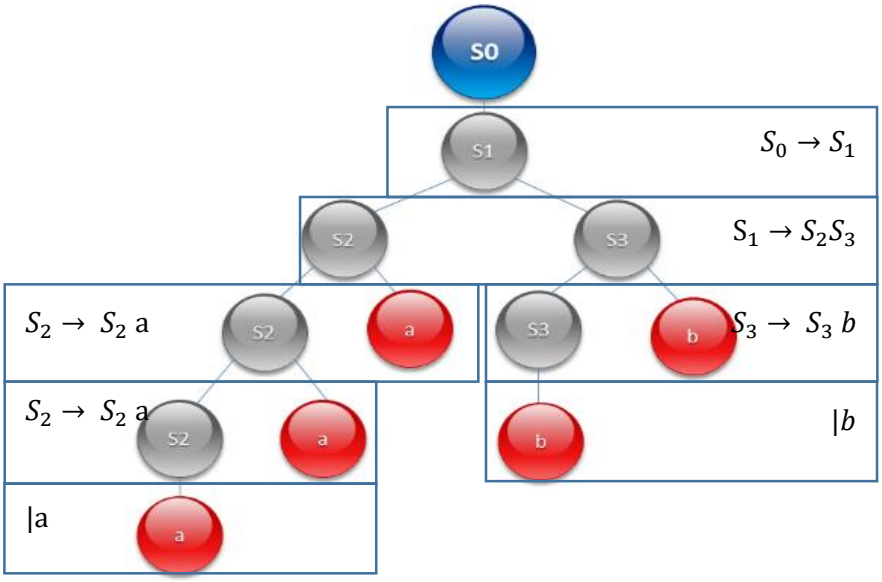
1 GRAMATICAS (RECURSIVIDAD Y AMBIGÜEDAD)

Ejemplo No. 1 (recursividad por la izquierda)

Escriba una gramática que reconozca $a^n b^{\mathrm{m}}\ n \geq 1;\ \mathrm{m} \geq 0$. Donde n y m es el número de veces que puede aparecer a y b respectivamente. Cadena entrada: **aaabb**

Gramática:

$S_0 \rightarrow S_1$
$S_1 \rightarrow S_2 S_3$
$\quad |S_2$
$S_2 \rightarrow S_2$ a
$\quad |$a
$S_3 \rightarrow S_3\ b$
$\quad |$b



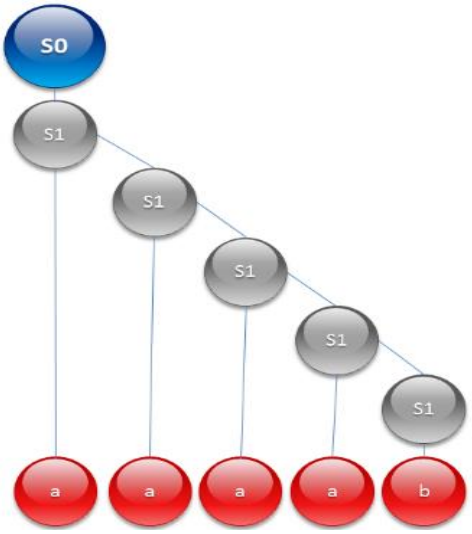
Nótese que la gramática es recursiva por la izquierda en la producción S_2 y S_3. Hay que resaltar que la construcción del árbol se da con derivación por la izquierda es decir se el árbol crece de izquierda a derecha.

Ejemplo No. 2 (recursividad por la derecha)

Escriba una gramática que reconozca a^nb $\ n \geq 1$; cadena de entrada: **aaaab**

Gramática:

$S_0 \rightarrow S_1$
$S_1 \rightarrow a\ S_1$
$\quad |b$

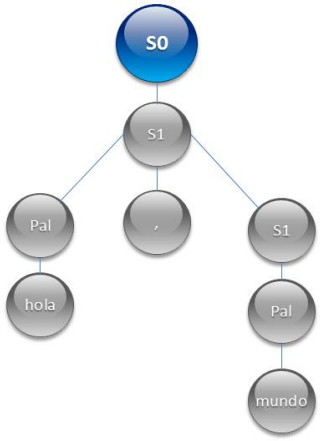


Ejemplo No. 3 (varias soluciones)

Escriba una gramática que reconozca una lista de palabras separadas por comas y acepte la cadena: **hola, mundo**

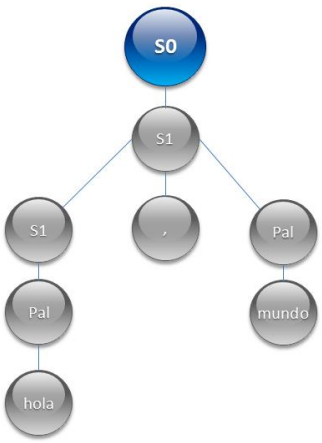
Forma No. 1 (recursiva por la derecha)

$$S_0 \rightarrow S_1$$
$$S_1 \rightarrow Pal\ ,S_1$$
$$\mid Pal$$



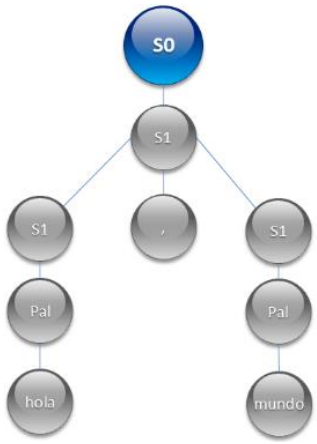
Forma No. 2 (recursiva por la izquierda)

$$S_0 \rightarrow S_1$$
$$S_1 \rightarrow S_1\ ,Pal$$
$$\mid Pal$$



Forma No. 3 (Ambigua)

$$S_0 \rightarrow S_1$$
$$S_1 \rightarrow S_1\ ,S_1$$
$$\mid Pal$$



Nótese que hay 3 formas para representar lo mismo, las 3 aceptan la cadena de entrada por lo cual surge la duda ¿Qué gramática uso?, pues simplemente se toma la gramática más eficiente las cuales son las gramáticas recursivas por la izquierda ya que la síntesis se realiza al principio por lo tanto no se guarda todas las palabras porque de una vez se van usando.

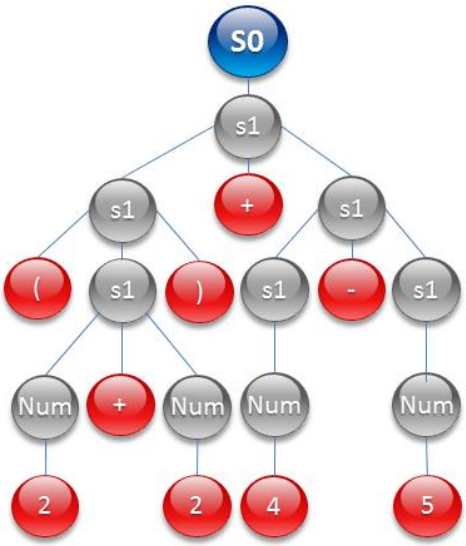
Ejemplo No. 4 (Ambigüedad)

Escriba una gramática que acepte la siguiente entrada: (2+2)+4-5

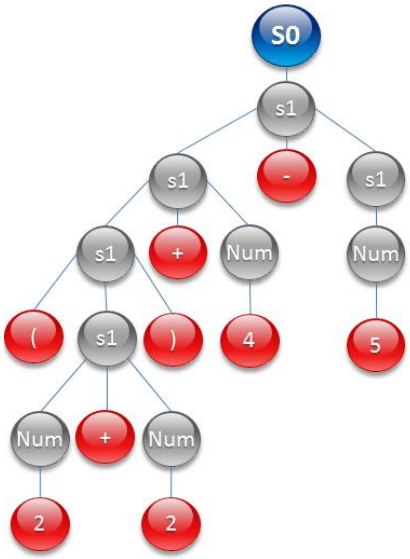
$S_0 \rightarrow S_1$
$S_1 \rightarrow S_1 + S_1$
$\quad |S_1 - S_1$
$\quad |(S_1)$
$\quad |\text{Num}$

¿Por qué se dice que la gramática anterior es ambigua?, simplemente porque se crea más de un árbol de derivación para una misma cadena de entrada, tal y como se muestra a continuación.

Árbol No. 1



Árbol No. 2



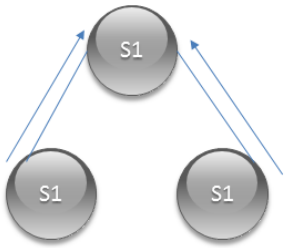
Tips para hacer gramáticas.

- Abstraer la gramática y visualizarla como un árbol por tanto una gramática tiene que ir de forma descendente donde los no terminales serán los nodos padres y los terminales serán los hijos.
- No toda recursividad es aceptada por el analizador, nunca hacer este tipo de recursividad: (recordemos que la gramática se transformara en un árbol y no puede estar en el hijo S1 y regresar al padre S0)
  $S_0 \rightarrow S_1$
  $S_1 \rightarrow \text{Num}\ S_0$
  $\quad |\text{Num}$
- En la teoría no es óptimo o recomendable hacer gramáticas ambiguas ya que estás generan más de un árbol de derivación y no tiene presidencia de operadores.
- En la teoría es más óptimo buscar la recursividad por la izquierda.
- Para representar a los no terminales se puede usar cualquier letra E, F, T,.. etc. O estados S1, S2, S3,… etc.

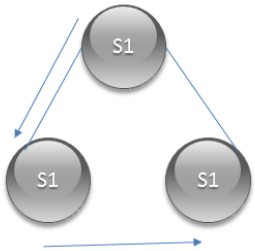
2 DEFINICIÓN DIRIGIDA POR LA SINTAXIS

Atributos heredados y sintetizados

Atributos sintetizados: los valores se pasan de las hojas a la raíz



Atributos heredados: los valores pueden pasarse de padre e hijos y entre hermanos



Antes de hacer definición dirigida por la sintaxis, es necesario aumentar la gramática.

$$S_1 \rightarrow S_1 + S_1 \mid S_1 - S_1 \mid (S_1) \mid \text{Num}$$

$$S_0 \rightarrow S_1$$
$$S_1 \rightarrow S_1 + S_1 \mid S_1 - S_1 \mid (S_1) \mid \text{Num}$$

$$S_0 \rightarrow S_1\{\text{print("resultado")}\}$$
$$S_1 \rightarrow S_1 + S_1 \mid S_1 - S_1 \mid (S_1) \mid \text{Num}$$

Esto se hace para sintetizar los valores e imprimirlos en la raíz o la producción inicial.

Ejemplo No. 1

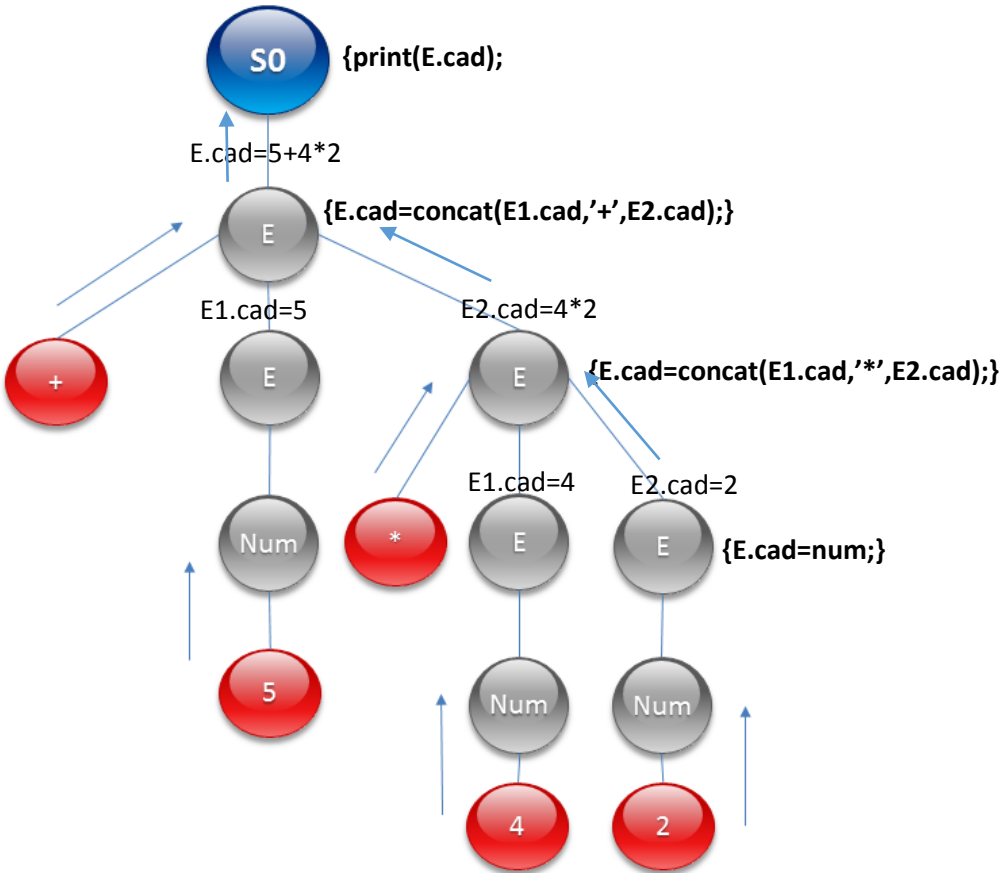
Escriba una gramática que acepte una expresión en notación prefija y retorne notación infija.

Un ejemplo de una expresión escrita en prefija seria: +5*42 esto traducido a infijo seria: 5+4*2, es decir el operador va al principio de un número o de otra expresión, dado que en infijo el operador siempre viene en medio de dos números una producción gramatical para el operador + seria $E \rightarrow E + E$, esta misma producción en prefijo seria $E \rightarrow +E\ E$ por lo tanto la gramática quedaría de la siguiente forma:

$S \rightarrow \mathrm{E}$
$\mathrm{E} \rightarrow +\mathrm{EE}$
$\quad\quad | * \mathrm{EE}$
$\quad\quad |\mathrm{num}$

Definición

$S \rightarrow \mathrm{E}\{\mathrm{print}(E.cad);\}$
$\mathrm{E} \rightarrow +\mathrm{EE}\ \{\mathrm{E.cad} = \mathrm{concat}(\mathrm{E1.cad}, ' + ', \mathrm{E2.cad});\}$
$\quad\quad | * \mathrm{EE}\{\mathrm{E.cad} = \mathrm{concat}(\mathrm{E1.cad}, '*', \mathrm{E2.cad});\}$
$\quad\quad |\mathrm{num}\{\mathrm{E.cad} = \mathrm{num};\}$



Nótese que se está usando el atributo "cad" esto no es una variable global es un atributo asociado a un no terminal. Un no terminal puede tener varios atributos, también se utiliza una función "cancat" que concatena a lo máximo 3 valores. En la práctica dicha función (función concat) debe existir es decir se tiene que programar esta función de tal forma que concatene los valores ingresados y retorne lo concatenado.

Clase No. 3

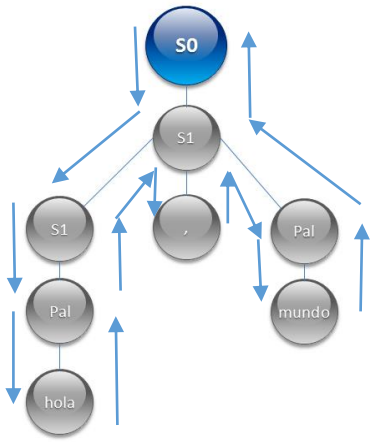
3 ATRIBUTOS HEREDADOS

Para los atributos heredados hay que tener en cuenta lo siguiente:

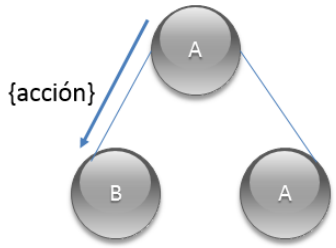
1. Gramáticas: para heredar atributos es recomendable realizar una gramática recursiva por la derecha pero NO ES LEY se puede heredar atributos con gramáticas recursivas por la izquierda (a menos que se esté utilizando un analizador LL(k)).
2. Recorrido de un árbol de análisis sintáctico.

Ejemplo: Dada la gramática que acepta una lista de palabras, realizar el árbol de análisis sintáctico con y marcar la forma en que se recorre el árbol. Entrada: **hola, mundo**

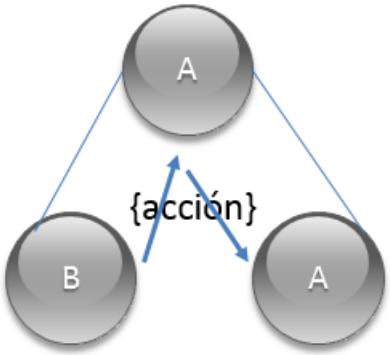
$$S_0 \rightarrow S_1$$
$$S_1 \rightarrow S_1\,, Pal$$
$$\mid Pal$$



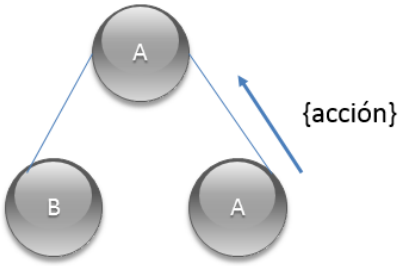
3. Esquema de traducción: en esquema de traducción se puede realizar acciones en la gramática al principio, medio o al final de los no terminales. por ejemplo: **T->** {acción}**F**{acción} **T´**{acción}.
4. No se puede heredar y sintetizar al mismo tiempo se debe visualizar el recorrido del árbol para saber cuándo heredar o sintetizar.
5. ¿Dónde poner la acción?: independientemente de lo que se esté haciendo se coloca las acciones en las siguientes circunstancias.
    a. Para la siguiente gramática: **A->**{Acción}**B A**



b. Para la siguiente gramática: **A->B** {Acción} **A**



a. Para la siguiente gramática: **A->B A** {Acción}



3.1 EJEMPLO

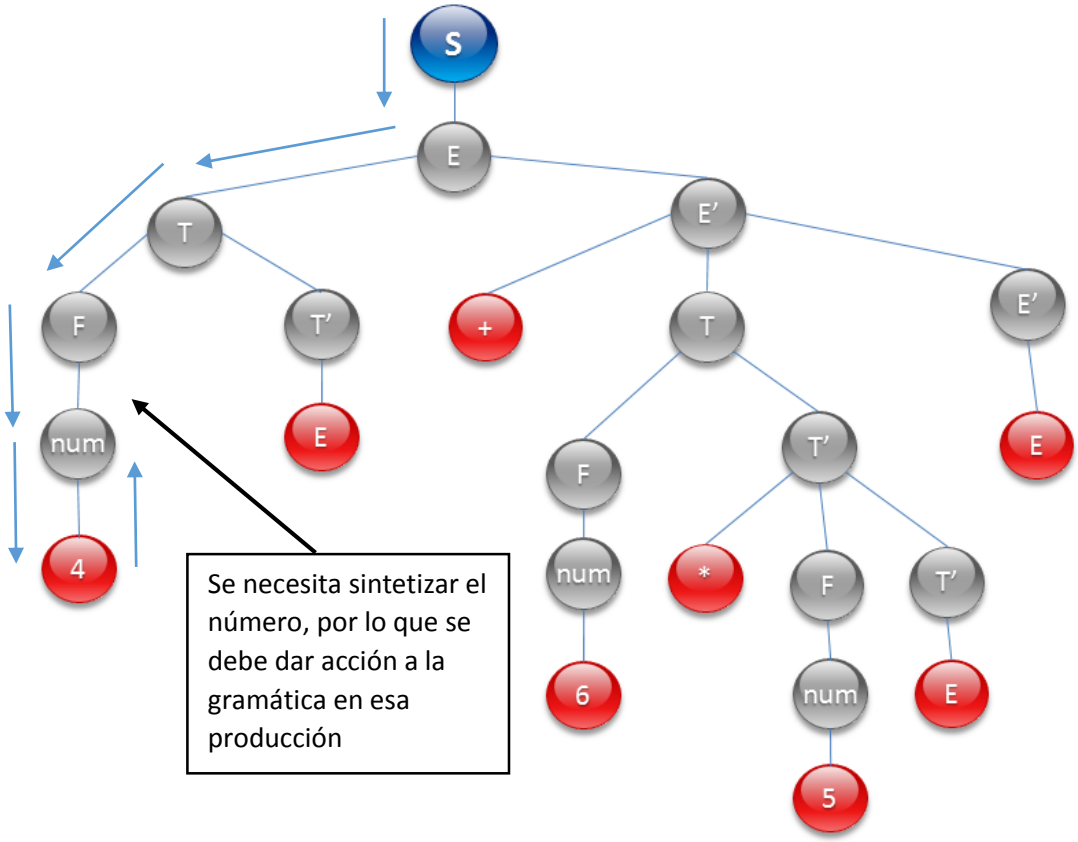
Calculadora con atributos heredados.

1. Paso No. 1: crear la gramática.

$S \rightarrow E$
$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E'$
$\quad | \in$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T'$
$\quad | \in$
$F \rightarrow num$

2. Cada nodo posee su atributo normal y su atributo de herencia por ejemplo: para el no terminal “E”, tendrá los atributos E.v y Eh.v (el primero servirá para sintetizar y el segundo para heredar)

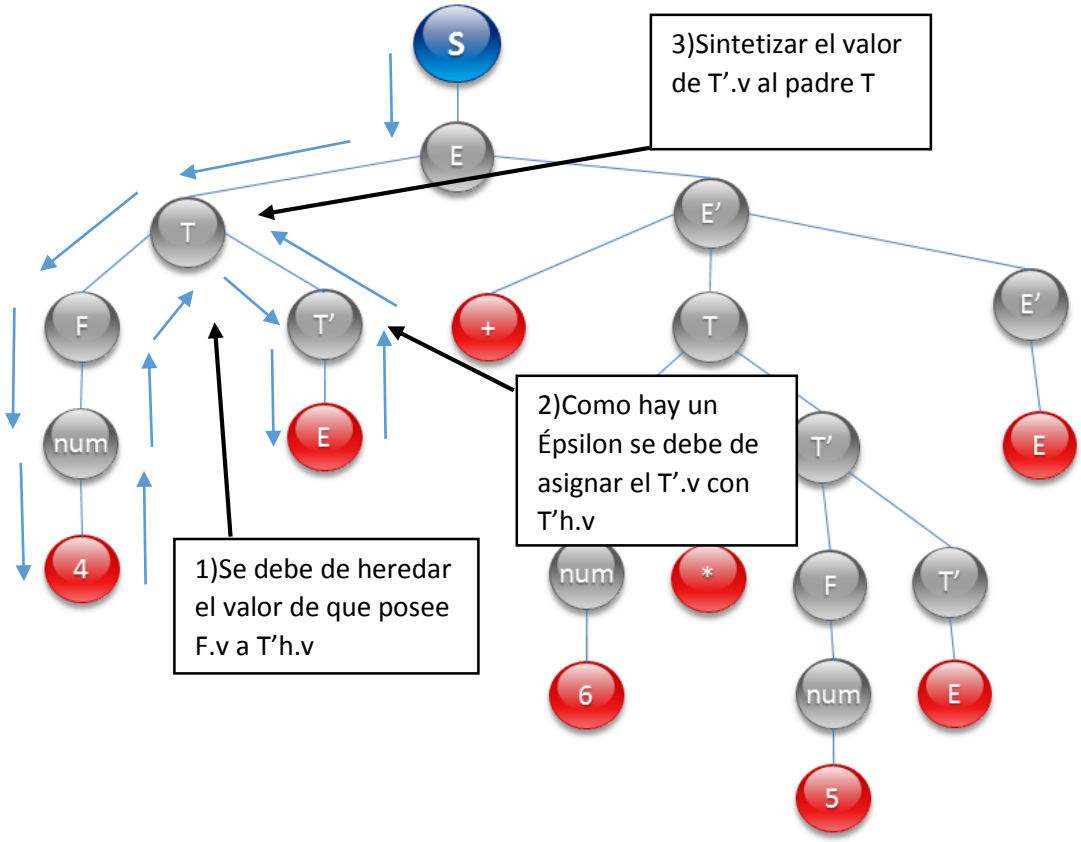
3. Paso No. 2: crear el árbol (para una entrada que abarque la mayoría de producciones, en este caso 4+6*5) y recorrerlo



Agregando acción:

$S \rightarrow E$
$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E'$
$\quad\quad | \in$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T'$
$\quad\quad | \in$
$F \rightarrow |num\{\boldsymbol{F.v = atoi(num)};\}$

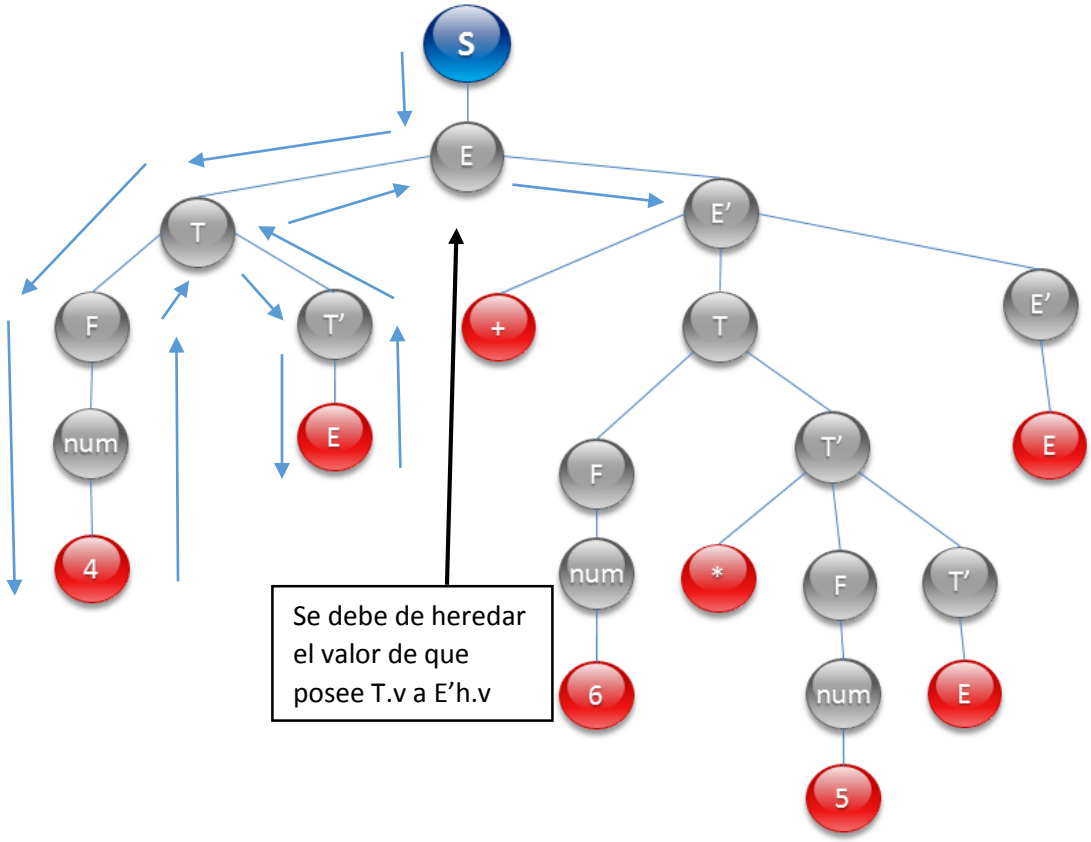
4. Se continúa el proceso de recorrido.



Agregando acción:

$S \rightarrow E$
$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E'$
$\quad\quad | \in$
$T \rightarrow F\{\boldsymbol{T'h.v = F.v};\}T'\{\boldsymbol{T.v = T'.v};\}$
$T' \rightarrow *\ F\ T'$
$\quad\quad | \in \{\boldsymbol{T'.v = T'h.v};\}$
$F \rightarrow num\{\boldsymbol{F.v = atoi(num)};\}$

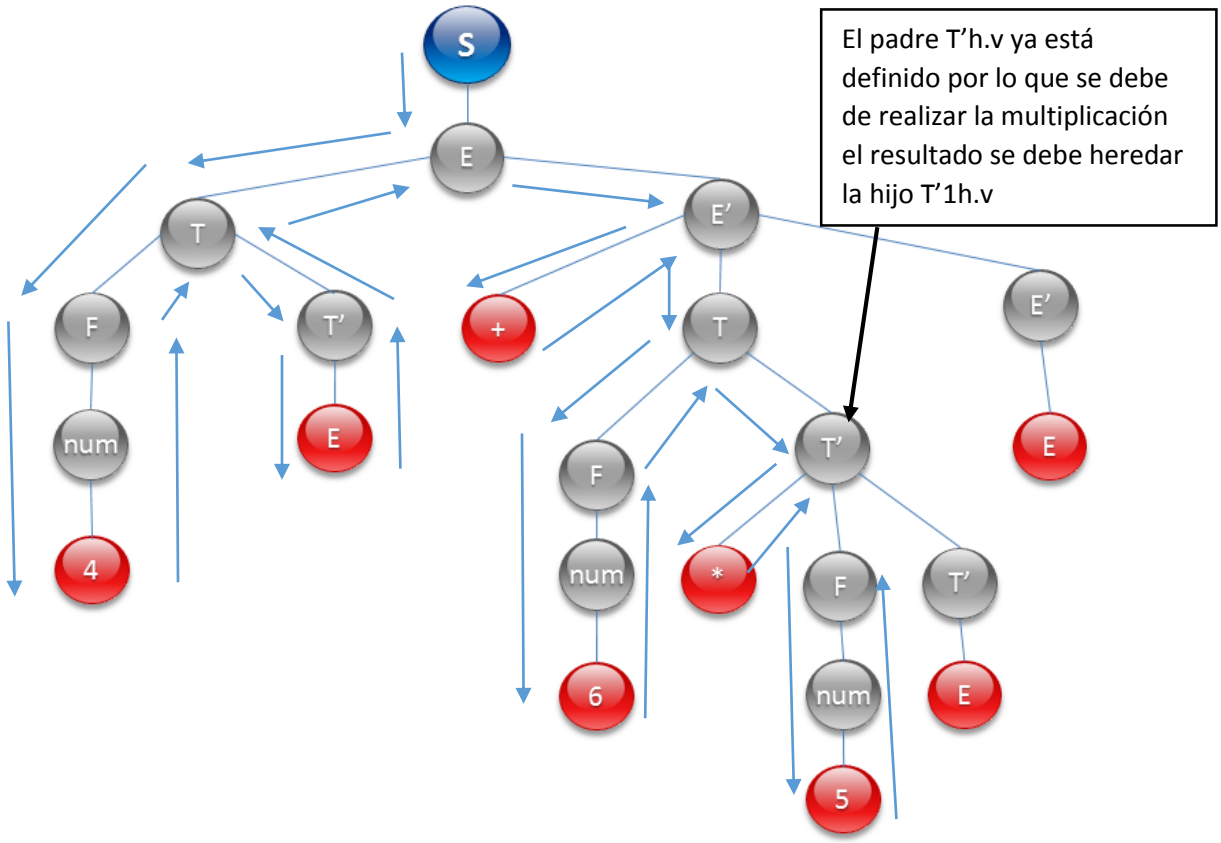
5. Se continúa el proceso de recorrido.



Agregando acción:

$S \rightarrow E$
$E \rightarrow T\{\boldsymbol{E'h.v = T.v}\}E'$
$E' \rightarrow + T\ E'$
$\quad | \in$
$T \rightarrow F\{\boldsymbol{T'h.v = F.v;}\}T'\{\boldsymbol{T.v = T'.v;}\}$
$T' \rightarrow * F\ T'$
$\quad | \in \{\boldsymbol{T'.v = T'h.v;}\}$
$F \rightarrow num\{\boldsymbol{F.v = atoi(num);}\}$

6. Se continúa el proceso de recorrido.



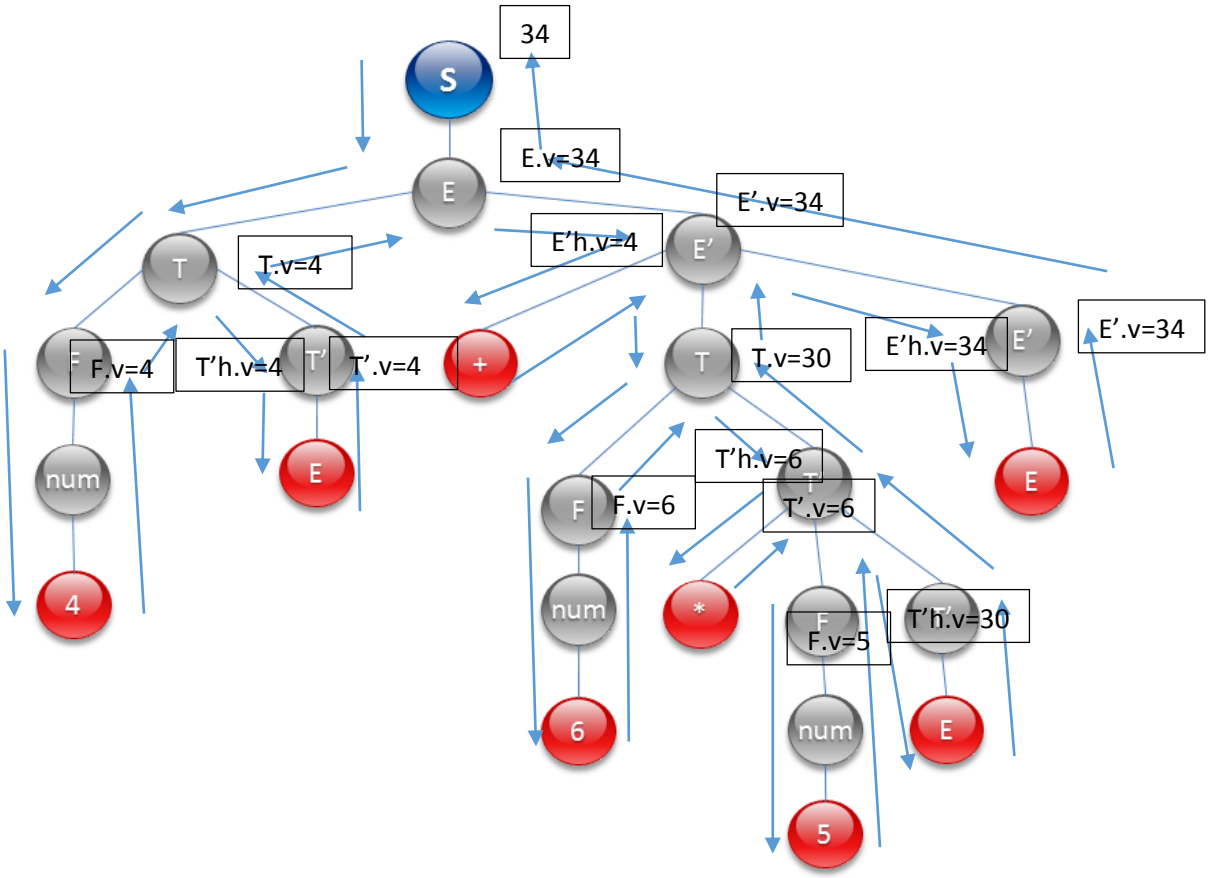
Agregando acción:

$S \rightarrow E$
$E \rightarrow T\{\boldsymbol{E'h.v = T.v};\}\, E'$
$E' \rightarrow + T\ E'$
$\quad\quad | \in$
$T \rightarrow F\{\boldsymbol{T'h.v = F.v};\}T'\{\boldsymbol{T.v = T'.v};\}$
$T' \rightarrow * F\ \{\boldsymbol{T'_1h.v = T'h.v * F.v};\}T'$
$\quad\quad | \in \{\boldsymbol{T'.v = T'h.v};\}$
$F \rightarrow num\{\boldsymbol{F.v = atoi(num)};\}$

7. Se continúa con el mismo proceso hasta obtener

Agregando acción:

$S \rightarrow E\{\boldsymbol{print}(\boldsymbol{E.v});\}$
$E \rightarrow T\{\boldsymbol{E'h.v} = \boldsymbol{T.v};\}E'\{\boldsymbol{E.v} = \boldsymbol{E'.v};\}$
$E' \rightarrow +T\{\boldsymbol{E'_1 h.v} = \boldsymbol{E'h.v} + \boldsymbol{T.v};\}E'\{\boldsymbol{E'.v} = \boldsymbol{E'_1.v};\}$
$\quad | \in \{\boldsymbol{E'.v} = \boldsymbol{E'h.v};\}$
$T \rightarrow F\{\boldsymbol{T'h.v} = \boldsymbol{F.v};\}T'\{\boldsymbol{T.v} = \boldsymbol{T'.v};\}$
$T' \rightarrow *F\{\boldsymbol{T'_1 h.v} = \boldsymbol{T'h.v} * \boldsymbol{F.v};\}T'\{\boldsymbol{T'.v} = \boldsymbol{T'_1.v};\}$
$\quad | \in \{\boldsymbol{T'.v} = \boldsymbol{T'h.v};\}$
$F \rightarrow num\{\boldsymbol{F.v} = \boldsymbol{atoi}(\boldsymbol{num});\}$



4 CONCLUSIONES

1. En la definición dirigida por la sintaxis las acciones van al final de una producción: ejemplo A->B C {Acciones}. La DDS es para la sintetizar atributos.
2. En el esquema de traducción las acciones van adelante, en medio o al final de las producciones A->{Acciones} B {Acciones} C {Acciones}. El EDT es para heredar atributos.
3. La traducción dirigida por la sintaxis engloba la DDS y EDT. Por lo que se puede tener atributos heredados y sintetizados.