8. Gramáticas con Atributos.[i]

Definición Dirigida por la Sintaxis (SDD)

Una **Definición Dirigida por la Sintaxis (SDD)** es una gramática libre de contexto que asocia:

1. A cada símbolo de la gramática, un conjunto de **atributos**.
2. A cada producción, un conjunto de **reglas semánticas** para calcular los valores de los atributos asociados a los símbolos que aparecen en la producción.

Un árbol de análisis sintáctico que muestra los valores de los atributos se llama ***árbol anotado*** o ***árbol decorado***.

Atributos

Un atributo puede ser un valor numérico, de cadena, una tabla de referencia, un objeto.

Si X es un símbolo y a es uno de sus atributos, se escribe $X.a$ para denotar el valor a en un nodo del árbol sintáctico de etiqueta X.

Hay dos tipos de atributos para los **no terminales:**

1. ***Atributos Sintetizados***.
2. ***Atributos Heredados***.

Para los **terminales**, los atributos sólo pueden ser ***Atributos Sintetizados***.

Atributos Sintetizados.

Dada $A \to \alpha$, y sea $A.a = f(\alpha_1.y_1, \alpha_2.y_2, \dots, \alpha_n.y_n)$ una regla semántica para calcular el atributo a, $A.a$ es un ***atributo sintetizado***, ya que depende del valor de los atributos de los símbolos que están a la derecha de la producción (hijos de A en el árbol sintáctico).

Un ***atributo sintetizado*** para el no terminal A en el nodo N del árbol sintáctico está definido por una regla semántica asociada a la producción en ese nodo en términos de valores de atributos de los hijos de N y del mismo N. El No terminal A aparece en la parte izquierda de la producción.

Una **Definición Dirigida por la Sintaxis (SDD)** donde todos los atributos son sintetizados se denomina **S-Atribuida**.

Si además no tiene efectos globales, se denomina ***Gramática con Atributos***.

Las reglas en una ***gramática con atributos*** definen el valor de un atributo exclusivamente en términos de los valores de otros atributos o constantes.

Los efectos globales podrían ser, por ejemplo, la actualización de una variable global, imprimir en alguna salida, manejar una tabla, etc.

Ejemplo:

Esta DDS evalúa las expresiones terminadas por la marca $\mathbf{n}$.

Todos los no terminales tienen un atributo sintetizado llamado $\boldsymbol{val}$.

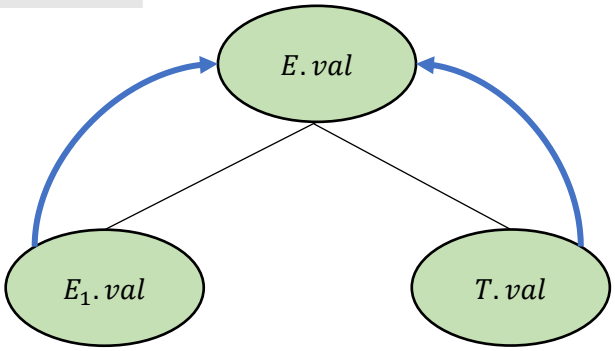
El terminal $\mathbf{digit}$ tienen un atributo sintetizado llamado $\boldsymbol{lexval}$, que es un valor entero retornado por el ***analizador léxico***.

$G =< \{L, E, T, F\}, \{+, *, (,), \mathbf{digit}\}, L, P >$, donde:

$$
\begin{aligned}
P = \{ & \\
& L \to E\ \mathbf{n} \\
& E \to E + T | T \\
& T \to T * F | F \\
& F \to (E) | \mathbf{digit} \\
\} &
\end{aligned}
$$

PRODUCCIÓN	REGLA SEMÁNTICA
1) $L \rightarrow E$ **n**	$L.val = E.val$
2) $E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
3) $E \rightarrow T$	$E.val = T.val$
4) $T \rightarrow T_1 * F$	$T.val = T_1, val \times F.val$
5) $T \rightarrow F$	$T.val = F.val$
6) $F \rightarrow (E)$	$F.val = E.val$
7) $F \rightarrow$ **digit**	$F.val =$ **digit**$.lexval$

Regla 2, No terminal E:



El siguiente programa Yacc se corresponde con la gramática con atributos anterior. Pero para $L \rightarrow E$, imprime el valor $E.val$ (efecto global), en lugar de definir $L.val = E.val$

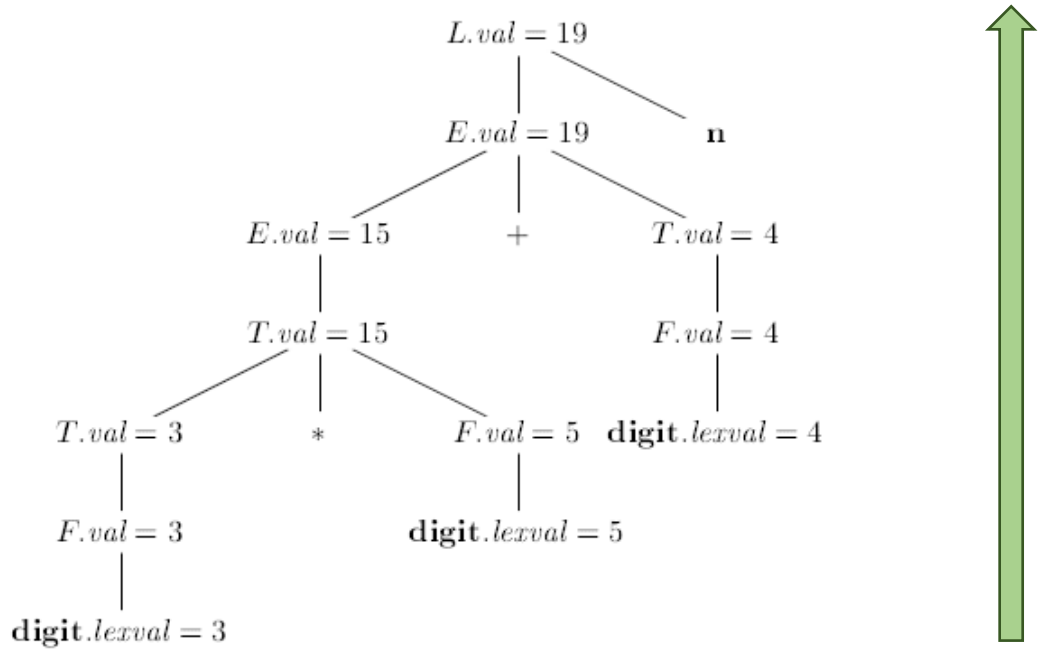
```
%{
#include <ctype.h>
%}

%token DIGIT

%%
line   : expr '\n'        { printf("%d\n", $1); }
       ;
expr   : expr '+' term    { $$ = $1 + $3; }
       | term
       ;
term   : term '*' factor  { $$ = $1 * $3; }
       | factor
       ;
factor : '(' expr ')'     { $$ = $2; }
       | DIGIT
       ;
%%
yylex() {
    int c;
    c = getchar();
    if (isdigit(c)) {
        yylval = c-'0';
        return DIGIT;
    }
    return c;
}
```

Si todos los atributos son sintetizados, se pueden evaluar todos los hijos de un nodo antes de evaluar el de él mismo. Entonces se puede evaluar en forma ascendente

El **árbol decorado** para la cadena de entrada $3 * 5 + 4$ **n** es el siguiente: (se asume que los valores $\boldsymbol{lexval}$ para los terminales los proveyó el analizador léxico)



Atributos Heredados.

Dada $A \to \alpha$, y sea $\alpha_i.a = f(\alpha_1.y_1, \alpha_2.y_2, \ldots, \alpha_n.y_n, A.b)$ una regla semántica para calcular el atributo a, $\alpha_i.a$ es un ***atributo heredado***, ya que depende del valor de cualquiera de los atributos de los símbolos en la producción (hermanos o padre en el árbol sintáctico).

Un ***atributo heredado*** para el no terminal B en el nodo N del árbol sintáctico está definido por una regla semántica asociada a la producción en el nodo padre de N, en términos de valores de los atributos del padre de N, de N, y de los hermanos de N. El no terminal B aparece en el cuerpo de la producción.

Ejemplo:
Esta gramática no tiene recursividad a izquierda, y es apta para un análisis descendente.
$G =< \{T, T', F\}, \{*, \mathbf{digit}\}, T, P >$, donde:

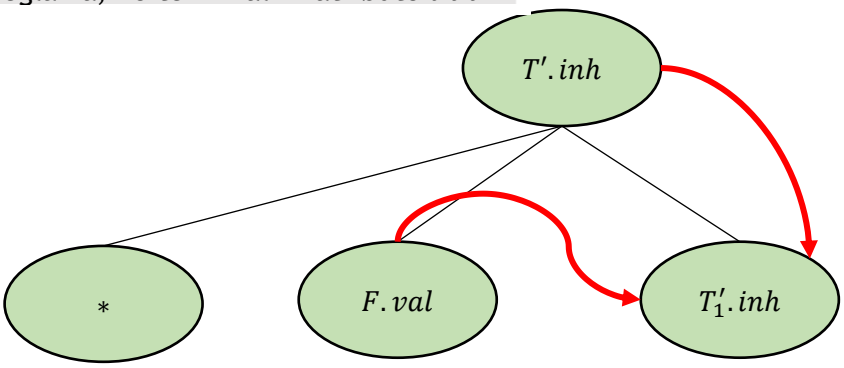
$$
\begin{aligned}
P = \{ \\
&T \to FT' \\
&T' \to * FT'_1 \\
&T' \to \lambda \\
&F \to \mathbf{digit} \\
\}
\end{aligned}
$$

PRODUCCIÓN	REGLA SEMÁNTICA
1) $T \to FT'$	$T'.inh = F.val$ $T.val = T'.syn$
2) $T' \to * FT'_1$	$T'_1.inh = T'.inh \times F.val$ $T'.syn = T'_1.syn$
3) $T' \to \lambda$	$T'.syn = T'.inh$
4) $F \to \mathbf{digit}$	$F.val = \mathbf{digit}.lexval$

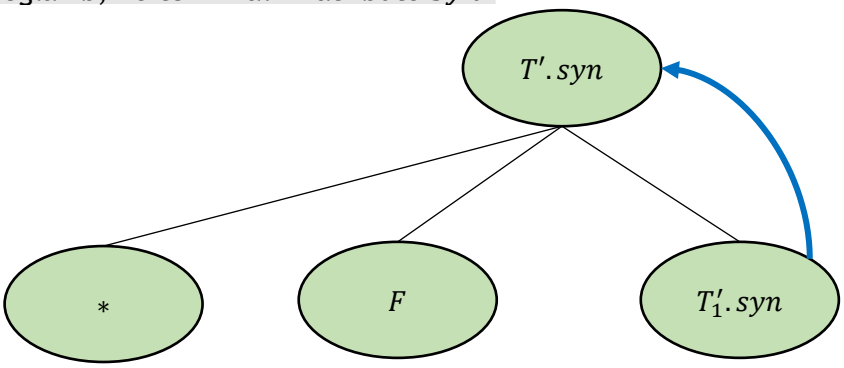
Los no terminales T y F tienen un atributo sintetizado llamado $\boldsymbol{val}$.

El no terminal T' tiene dos atributos, uno sintetizado ($\boldsymbol{syn}$) y otro heredado ($\boldsymbol{inh}$).

Regla 2a, no terminal T' atributo ***inh***

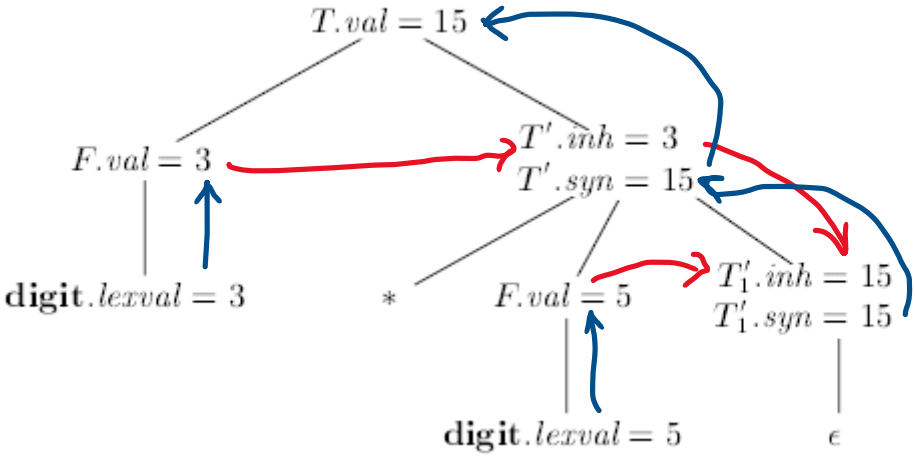


Regla 2b, no terminal T' atributo ***syn***



El terminal **digit** tienen un atributo sintetizado llamado ***lexval***, que es un valor entero retornado por el ***analizador léxico***.

El **árbol decorado** para la cadena de entrada $3 * 5$ es el siguiente: (se asume que los valores ***lexval*** para los terminales los proveyó el analizador léxico)



Orden de evaluación de una DDS

Un **Grafo de Dependencias** permite determinar el ***orden de evaluacion*** de los atributos para un arbol sintáctico dado.
Un arco de un atributo a otro significa que el valor de uno es necesario para calcular el otro, y expresa una restricción dada por las reglas semánticas.

Construcción del Grafo de Dependencias:

```
Para cada nodo X en el árbol sintáctico:
      Para cada atributo a de X:
            Construir un nodo con etiqueta X.a
```

```
Para cada nodo X en el árbol sintático:
      Para cada regla semántica v:= f(v1, v2, .., vk) asociada con la
      producción que dio lugar a X y sus hijos en el árbol:
            Para i desde 1 hasta k:
                  Construir un arco dirigido desde vi hasta v.
```

Ejemplo:

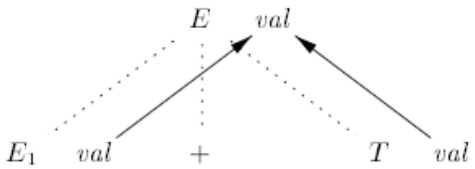
Para la gramática del ejemplo anterior,

$$
\begin{aligned}
&P = \{\\
&L \rightarrow E\ \mathbf{n}\\
&E \rightarrow E + T | T\\
&T \rightarrow T * F | F\\
&F \rightarrow (E) | \mathbf{digit}\\
&\}
\end{aligned}
$$

Considerando la regla:

PRODUCCIÓN	REGLA SEMÁNTICA
$E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$

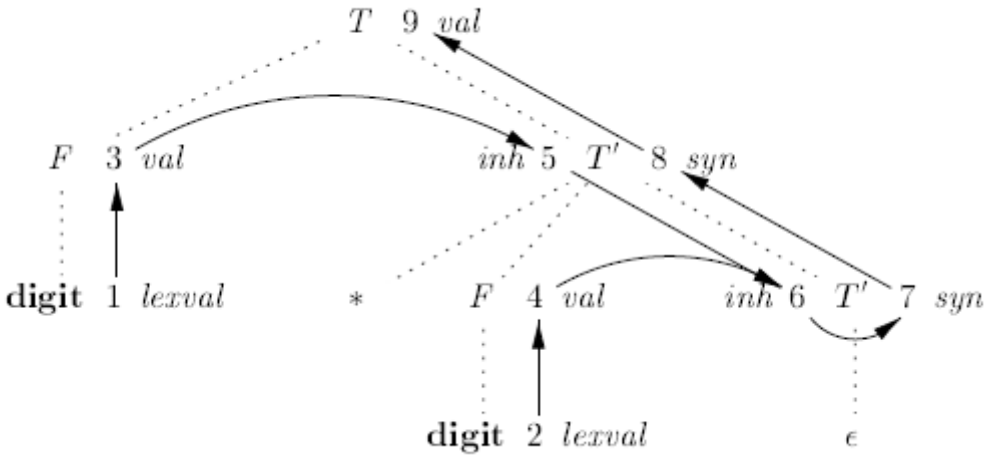
La parte del grafo de dependencias que usa esta producción, se verá así:



Donde las líneas punteadas corresponden al árbol sintáctico y las flechas al grafo de dependencias.
El grafo de dependencias para la gramática

$$
\begin{aligned}
&P = \{\\
&T \rightarrow FT'\\
&T' \rightarrow * FT_1'\\
&T' \rightarrow \lambda\\
&F \rightarrow \mathbf{digit}\\
&\}
\end{aligned}
$$

Siendo el árbol sintáctico el que corresponde a la cadena $3 * 5$, y los números que aparecen indican el orden en que se evaluarán los atributos.



Orden topológico del grafo:-

Si un grafo de dependencias tiene un arco **del nodo M al nodo N**, entonces los atributos en M deben evaluarse **antes** que los atributos en N.

La secuencia de evaluación de los nodos es $N_1, N_2, \dots N_k$, si $\forall i < j$, hay un arco de N_i a N_j

Si hubiera ciclos no hay orden topológico del grafo y por lo tanto tampoco hay forma de evaluar la Definición Dirigida por la Sintaxis para ese árbol.

Definiciones S-Atribuidas.

Una DDS es ***S-atribuida*** si todos los atributos son sintetizados.

Cuando una DDS es S-atribuida, se pueden evaluar todos los atributos en cualquier orden de abajo hacia arriba de los nodos en el árbol. Lo más simple es evaluar efectuando un recorrido post-orden transversal del árbol sintáctico, aplicando la función postorder a la raíz:

```
Postorder(N){
Para cada hijo C de N, desde la izquierda:
      Postorder(C)
Evaluar atributos asociados a N.
```

Como este orden es el mismo que usa un analizador LR para hacer las reducciones, al evaluar atributos sintetizados los puede almacenar en la pila mientras efectúa el análisis sintáctico, sin tener que crear otros nodos explícitamente.

Definiciones L-Atribuidas.

Una DDS es ***L-atribuida*** si los atributos son: 1.Sintetizados 2.Heredados, pero con reglas semánticas limitadas.(Para que los arcos en el grafo de dependencias vayan de izquierda a derecha – ***Left to right*** -, nunca de derecha a izquierda)

Si hay una producción $A \to X_1 X_2 \dots X_n$, y hay un atributo $X_i.a$ que se calcula por una regla semántica asociada a esta producción, entonces la regla semántica puede usar:

(a) Atributos heredados asociados a A.
(b) Atributos heredados o sintetizados asociados con los símbolos $X_1, X_2, \dots X_{i-1}$ situados a la izquierda de X_i
(c) Atributos heredados o sintetizados asociados a X_i, pero sólo de forma tal que no se formen ciclos en el grafo de dependencia con los atributos de este X_i

Ejemplo:
En la gramática del ejemplo anterior:

$$
\begin{aligned}
&P = \{\\
&T \to FT'\\
&T' \to * FT'_1\\
&T' \to \lambda\\
&F \to \textbf{digit}\\
&\}
\end{aligned}
$$

PRODUCCIÓN	REGLA SEMÁNTICA
1) $T \to FT'$	$T'.inh = F.val$ $T.val = T'.syn$
2) $T' \to * FT'_1$	$T'_1.inh = T'.inh \times F.val$ $T'.syn = T'_1.syn$

Para la producción $T \to FT'$, $T'.inh$ es atributo heredado, asociado a F, que está a la izquierda de T' en la producción (cumple (b))

Para la producción $T' \rightarrow * FT'_1$, $T'_1.inh$ es atributo heredado, asociado a T', que es la parte izquierda de la producción (cumple(a)), y asociado a F, que está a la izquierda de T' en la producción (cumple(b))

Ejemplo:

Si se tiene una DDS con esta producción y reglas, no puede ser L-atribuida:

PRODUCCIÓN	REGLA SEMÁNTICA
$A \rightarrow BC$	$A.s = B.b$ $B.b = f(C.c, A.s)$

La primera regla define $A.s$ como un atributo sintetizado, que depende de $B.b$, es decir de un atributo de un hijo de A en el árbol.
La segunda regla define $B.b$, que depende de atributos de C y de A, pero C está a la derecha de B en $A \rightarrow BC$, por lo que no cumple (b). Tampoco cumple (c) porque se forma un ciclo.

Esquemas de Traducción Dirigidos por la Sintaxis.

Los **Esquemas de Traducción Dirigidos por la Sintaxis** – ETDS (SDT en inglés), son una notación complementaria a las Definiciones Dirigidas por la Sintaxis.

UN ETDS es una gramática libre de contexto con fragmentos de programa en los cuerpos de las producciones, es decir ***acciones semánticas*** que pueden aparece en cualquier posición dentro del cuerpo de la producción.

Por convención, las ***acciones semánticas*** se escriben entre llaves.

Hay dos grupos de ETDS:

1. ***Esquemas de Traducción Postfija*** (para Definiciones S-atribuidas).
2. ***Esquemas de Traducción para Definiciones L-atribuidas***.

Esquemas de Traducción Postfijos.

Cuando la DDS es S-atribuida, se puede construir un ETDS en el cual cada acción se ubica al final de la producción y se ejecuta cuando se hace una **reducción** en el análisis sintáctico LR.

Pueden implementarse durante el análisis LR ejecutando las acciones cuando ocurren las reducciones. Para eso, los atributos de cada símbolo pueden ponerse también en la pila, en un lugar donde puedan encontrarse durante la reducción.

Así, si se tienen los atributos $X.x, Y.y, Z.z$ y $A.a = f(X.x, Y.y, Z.z)$, para la producción $A \rightarrow XYZ$, en el momento que se hace la reducción XYZ,por A, se calcula $A.a$.

Ejemplo:

Para la gramática del ejemplo anterior,

$$
\begin{aligned}
&P = \{\\
&L \rightarrow E\ \mathbf{n}\\
&E \rightarrow E + T | T\\
&T \rightarrow T * F | F\\
&F \rightarrow (E) | \mathbf{digit}\\
&\}
\end{aligned}
$$

El ETDS resulta:

$$
\begin{aligned}
&L \rightarrow E\mathbf{n} && \{\boldsymbol{print}(\boldsymbol{E.val});\}\\
&E \rightarrow E_1 + T && \{E.val = E_1.val + T.val;\}\\
&E \rightarrow T && \{E.val = T.val;\}\\
&T \rightarrow T_1 * F && \{T.val = T_1.val \times F.val;\}\\
&T \rightarrow F && \{T.val = F.val;\}\\
&F \rightarrow (E) && \{F.val = E.val;\}\\
&F \rightarrow \mathbf{digit} && \{F.val = \mathbf{digito}.lexval;\}
\end{aligned}
$$

Esquemas de Traducción Para Definiciones L-Atribuidas.

Las reglas para convertir una DDS L-atribuida en un ETDS son:

1. Colocar la acción que calcula los atributos heredados para un no terminal A inmediatamente antes de la ocurrencia de A en el cuerpo de la producción. Si varios atributos heredados para A dependen unos de otros en forma **acíclica**, ordenar la evaluación de los atributos para que aquellos que se necesiten primero se calculen antes.
2. Ubicar las acciones que calculan un atributo sintetizado para la parte izquierda de una producción, al final del cuerpo de esa producción.

Ejemplo:

Los lenguajes tipo TEX para escribir fórmulas matemáticas tienen formas de definir subíndices, subíndices de subíndices, etc. Ignorando superíndices, fracciones y otras estructuras.

Por ejemplo, uno puede escribir `a sub i sub j` para indicar a_{i_j}

Una gramática simple para estas cajas de texto (Boxes) es:

$$G =< \{B\}, \{\textbf{sub}, (,), \textbf{text}\}, B, P >$$
$$P = \{$$
$$B \rightarrow BB | B\ \textbf{sub}\ B | (B) | \textbf{text}$$
$$\}$$

Es decir:

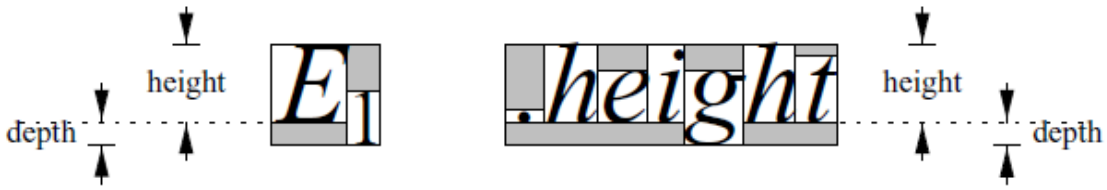
1. Una caja de texto está a continuación de otra.
2. Una caja de texto puede tener otra de subíndice.
3. Una caja de texto puede estar entre paréntesis.
4. Una caja de texto puede ser sólo texto (secuencia de símbolos).

Esta gramática es ambigua, pero se puede llegar a analizar en forma ascendente si se hace que el subíndice tenga precedencia sobre la concatenación.

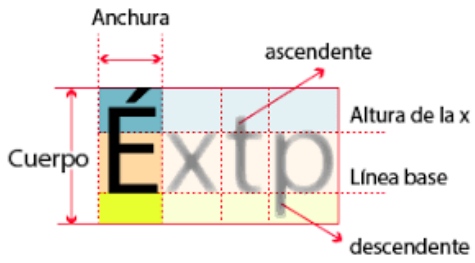
Por ejemplo, si se tienen las cajas para el texto $\boldsymbol{E_1}$ y para $\boldsymbol{.height,}$ se pueden concatenar para formar $\boldsymbol{E_1.height}$.

A su vez, la caja de texto $\boldsymbol{E_1}$ se formó a partir de una caja de texto para E que se concatenó con el subíndice 1, que es otra caja de texto, pero de tamaño un 30% más pequeña.

Por otro lado, puede verse $\boldsymbol{.height}$ como la concatenación de las cajas correspondientes a cada símbolo o letra.



Los valores asociados a las cajas de texto son los que se ven en la siguiente imagen:



a) Tamaño en puntos ***point size***. Es el tamaño del ***cuerpo*** de la caja. Si una caja tiene tamaño de punto $\boldsymbol{p}$, entonces, la caja de subíndice, tiene que tener tamaño de punto $\mathbf{0.7}\boldsymbol{p}$. Se usará el atributo $.\boldsymbol{ps}$.
b) Línea de base, ***baseline***. Corresponde a la línea donde se apoyan la mayoría de las letras. Algunas letras, como la "g", se extienden por debajo de esa línea. Para el subíndice, también está más abajo.
c) Altura de la caja, ***height***. Distancia desde la parte superior de la caja, hasta la línea de base. Se usará el atributo $.\boldsymbol{ht}$.
d) Profundidad de la caja, ***depth***. Distancia desde la línea de base hasta la parte inferior de la caja. Se usará el atributo $.\boldsymbol{dp}$.

Para calcular los valores anteriores, la DDS es:

PRODUCCIÓN	REGLA SEMÁNTICA
1. $S \rightarrow B$	$B.ps = 10$
2. $B \rightarrow B_1 B_2$	$B_1.ps = B.ps$ $B_2.ps = B.ps$ $B.ht = \text{MAX}(B_1.ht, B_2.ht)$ $B.dp = \text{MAX}(B_1.dp, B_2.dp)$
3. $B \rightarrow B_1 \mathbf{sub} B_2$	$B_1.ps = B.ps$ $B_2.ps = 0.7 \times B.ps$ $B.ht = \text{MAX}(B_1.ht, B_2.ht - 0.25 \times B.ps)$ $B.dp = \text{MAX}\ (B_1.dp, B_2.dp + 0.25 \times B.ps)$
4. $B \rightarrow (B_1)$	$B_1.ps = B.ps$ $B.ht = B_1.ht$ $B.dp = B_1.dp$
5. $B \rightarrow \mathbf{text}$	$B.ht = getHt(B.ps, \mathbf{text}.lexval)$ $B.dp = getDp(B.ps, \mathbf{text}.lexval)$

La producción 1, asigna el valor inicial 10 a la caja.

La producción 2, maneja la concatenación. Copia el tamaño a las cajas que estén debajo en el árbol sintáctico (es decir, las cajas B_1y B_2 heredan el tamaño de la caja más grande) Las alturas y profundidades, en cambio, se calculan desde abajo, tomando el máximo.

La producción 3, maneja la asignación de subíndice. Se asume que la caja de subíndice es el 70% de la caja "padre" (En la práctica, después de algunos niveles, los tamaños de subíndice ya no se hacen más pequeños). Y se asume que la línea base cae un 25% respecto del tamaño.

La producción 4, copia atributos cuando se usan los paréntesis.

La producción 5, maneja los nodos hoja que representan cajas de texto, asumiendo que se tienen funciones getHt y getDp que toman la altura y la profundidad apropiadas.

El ETDS que corresponde es:

1. $S \rightarrow B$ $\{B.ps = 10;\}$
2. $B \rightarrow B_1\ B_2$ $\{B_1.ps = B.ps;\}$ $\{B_2.ps = B.ps\}$ $\{B.ht = \text{MAX}\ (B_1.ht, B_2.ht);$ $B.dp = \text{MAX}(B_1.dp, B_2.dp)\ ;\}$
3. $B \rightarrow B_1\mathbf{sub}\ B_2$ $\{B_1.ps = B.ps;\}$ $\{B_2.ps = 0.7 \times B.ps\}$ $\{B.ht = \text{MAX}(B_1.ht, B_2.ht - 0.25 \times B.ps);$ $B.dp = \text{MAX}(B_1.dp, B_2.dp + 0.25 \times B.ps)\ ;\}$
4. $B \rightarrow (\ B_1)$ $\{B_1.ps = B.ps;\}$ $\{B.ht = B_1.ht;$ $B.dp = B_1.dp;\}$

5. $B \rightarrow$ **text** $\{B.ht = getHt(B.ps, \textbf{text}.lexval);$
$B.dp = getDp(B.ps, \textbf{text}.lexval);\}$

Para que sea legible, cada producción se dividió en más de un renglón. Pero por ejemplo la producción 4 sería:

$\boldsymbol{B} \rightarrow ($ $\{B_1.ps = B.ps;\}$ $\boldsymbol{B_1})$ $\{B.ht = B_1.ht; B.dp = B_1.dp;\}$

Implementación de DDS L-atribuidas

En los dos primeros métodos, se hace la traducción recorriendo un árbol sintáctico:

1. **Construir el árbol sintáctico y completarlo con las anotaciones (árbol decorado).**
   Esto es lo que se vio previamente, donde hay que establecer un orden de traducción, por lo que si se presentan circuitos, no se podrá resolver.
2. **Construir el árbol sintáctico, agregar acciones, y ejecutar las acciones en preorden**
   Consiste en agregar acciones a las producciones, basadas en reglas semánticas.
   Serviría para cualquier definición L-atribuida.
   Luego el algoritmo sería, para cada nodo b:

```
Visitar(b):
      Para cada hijo a de b:
         Evaluar los atributos heredados de a (de izquierda a derecha)
         Visitar (a)
      Evaluar los atributos sintetizados de b.
```

3. **Usar un analizador descendente recursivo.**
   La función para el no terminal A recibe los atributos heredados de A como argumentos, y retorna los atributos sintetizados de A.

---

[i] Bibliografía usada:
Aho, Sethi y Ullman. Compilers: Principles, Techniques and Tools. Second Editions.
IMPORTANTE: Evitar la edición en castellano porque tiene algunas erratas y expresiones mal traducidas.