

**Generación de Formas Pictóricas  
sobre un Sistema de Inferencia  
a través de TreeBag.(\*)**

**Mauricio Aguirre, Eric Jeltsch, Gerardo Rosales.**

**Departamento de Matemáticas - Área de Computación  
Universidad de La Serena, Benavente 980,  
La Serena, Chile.**

**{ejeltsch, maguirre, grosales}@dns.uls.cl**

**Resumen:**

*En este trabajo se presenta un sistema generador de árboles con intérprete que está basado en gramáticas de árboles regulares y transductores, el que junto con una álgebra, la cual actúa como intérprete de los árboles generados por el sistema antes mencionado, es capaz de generar formas pictóricas o modelos. Como aplicación, se desarrolla un algoritmo de inferencia, el cual posee como entrada un conjunto finito de árboles y como salida un tipo particular de generador de árboles. El control de este proceso, así como cuándo y dónde se realiza depende fundamentalmente del rótulo, sintaxis y del número de niveles del árbol en cuestión, labor que se realiza esencialmente en forma top-down a partir de la raíz. Hemos utilizado el sistema **TreeBag(TreeBasedGenerator)** para la implementación de algunos ejemplos, con el fin de darle una componente visual a la propuesta.*

**Clasificación:** Gramáticas de árboles, Transductores, Inferencia Gramatical, sistema TreeBag.

---

(\*) Trabajo financiado por Proyecto N° 0220-2-20, DIULS (Dirección de Investigación de la Universidad de La Serena).

## 1 INTRODUCCION

En muchas áreas de las Ciencias de la Computación es frecuente representar la información por árboles o grafos, que son objetos muy apropiados para describir conocimientos, información, o representar modelos. Ahora cualquier cambio local que pueda afectar a un árbol o grafo, puede quedar registrado a través de una regla o producción, de manera que cualquier proceso dinámico que represente cambios de estado, puede representarse a través de un conjunto de reglas. Las reglas y árboles que intervienen en este proceso conforman las llamadas gramáticas de árbol, las cuales proporcionan una alternativa para representar la información de una forma más general, en vez de utilizar cadenas de caracteres, facilitando de esta manera el acceso a un amplio espectro de aplicaciones en diversas áreas, como por ejemplo: Reconocimiento Sintáctico de Formas, Inferencia Gramatical, Semántica de Lenguajes, así como en la generación sistemática y manipulación de diseños gráficos, fractales, o la simulación del crecimiento de células o plantas en la Biología. Vea [Bar88], [PL90], [PJS92] y [FV98].

En particular, tratamos en este trabajo el problema de la *inferencia gramatical*, el cual se refiere a encontrar una descripción sintáctica, por ejemplo Gramáticas, Autómatas, Transductores o algún sistema, en nuestro caso un *Sistema Generador de Árboles con Interpretador* que consta de gramáticas de árbol, un conjunto finito de Transductores y un Álgebra ( en nuestro caso álgebras Collage que actúan como intérprete), permitiendo la generación o el reconocimiento automático de un conjunto finito de modelos en  $S_+$ , que es un conjunto finito de modelos pertenecientes a algún lenguaje, también llamados ejemplos positivos, los que en general pueden ser cadenas de caracteres, árboles, grafos, modelos u otros, en nuestro caso son *árboles*, y posiblemente un conjunto de árboles del complemento de  $S_+$ , también llamados ejemplos negativos. Para mayor información, vea [Web1].

Una gran variedad de algoritmos de inferencia se han desarrollado, por ejemplo, Fu y Booth [FB75] genera gramáticas regulares a partir de modelos codificados como cadena de caracteres. En [CRA76] Cook, Rosenfeld y Aronson introducen operaciones de inferencia que infieren gramáticas de contexto libre a partir de un conjunto finito de cadenas de caracteres. Radhakrishnan y Nagaraja [RN88] desarrolla un método para gramáticas regulares desde ejemplos positivos, los cuales forman una estructura de esqueleto. En [JL87] Jürgensen y Lindenmayer consideran modelos representados por estructuras arbóreas que generan OL-Sistemas. En [Je95] Jeltsch desarrolla un algoritmo de inferencia basado en grafos.

Inspirados por los trabajos antes mencionados, se presenta un algoritmo de inferencia gramatical que en lo sustantivo trata de determinar a partir de la sintaxis de cada uno de los árboles, en un proceso top-down desde la raíz, algunas estructuras que darán origen a las producciones de las gramática de árboles. El control de este proceso, así como cuándo y dónde se realiza depende fundamentalmente del rótulo y del número de niveles del árbol en cuestión. La generación sintáctica de los modelos en  $S_+$ , es obtenida por una gramática de árbol, transductores (eventualmente vacío) y álgebras apropiadas, en nuestro caso álgebras Collage.

Las gramáticas de árbol usadas son regulares, pues tienen una concepción similar a las gramáticas definidas por Chomsky, de manera que resultan ser muy flexibles y con una sólida base teórica, así mismo los transductores tienen un comportamiento similar a los autómatas con salida.

Este trabajo esta organizado de manera que se entregan en los preliminares los conceptos teóricos fundamentales, tales como gramáticas de árboles regulares y los transductores, en sección 3 se define td-generator que es el sistema generador, en sección 4 se muestran las distintas etapas que componen el algoritmo de inferencia, concluyendo en la sección 5 con una discusión,

proyecciones y aplicaciones de la propuesta. Con el fin de incorporar una componente visual a la investigación teórica que aquí se propone, hemos utilizado el sistema computacional TreeBag. Algunas de las ventajas del mencionado sistema se refieren a que es un sistema construido enteramente en Java, y en el cual se pueden combinar en forma activa cuatro diferentes tipos de componentes que son de nuestro interés: Gramáticas de árboles que generan árboles, transductores de árboles que transforman una entrada de árboles en una salida de árboles, Álgebras que interpretan árboles como expresiones sobre algún dominio y por último el despliegue de los valores que adquieren los árboles en la pantalla, como se muestra en Ejemplo6. Cabe mencionar que estamos trabajando en la implementación del algoritmo de inferencia de tal manera que sea capaz de interactuar con TreeBag o actuar como un sistema independiente. Para mayor información, vea [Web2].

## 2 PRELIMINARES

En esta sección se entregan notaciones y conceptos teóricos que serán utilizados en este trabajo. Para mayor información vea [GS84, Dre96, GS97] y las referencias citadas.

### Signaturas y Árboles.

El conjunto de los números naturales incluido el 0 es denotado por  $\mathbb{N}$ . Para todo  $n \in \mathbb{N}$ ,  $[n]$  denota el conjunto  $[1, \dots, n]$ . El conjunto de todas las sucesiones ( llamadas cadenas o palabras ) sobre un conjunto  $S$  es denotado por  $S^*$ .  $S^+$  es  $S^* - \{\lambda\}$ , donde  $\lambda$  denota la sucesión vacía. Para todo  $n \in \mathbb{N}$ , el conjunto de todas las sucesiones de longitud  $n$  en  $S^*$  es denotada por  $S^n$ .

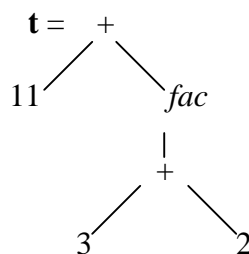
Un símbolo con *rango* es un par  $(f, n)$  que consiste de un símbolo  $f$ , y un número  $n \in \mathbb{N}$ , llamado rango. El símbolo con rango  $(f, n)$  es denotado como  $f^{(n)}$  o simplemente  $f$ , y llamado *símbolo*. Notar sin embargo que  $f^{(m)}$  y  $f^{(n)}$  son diferentes para  $m \neq n$ . Una *signatura* es un conjunto  $\Sigma$  de símbolos, denotado por  $f : n$ .

*Ejemplo:*  $\Sigma = \{+:2, fac:1\} \cup \{c:0 \mid c \in \mathbb{N}\}$  es una signatura que contiene los símbolos  $+$  y  $fac$  de rango 2 y 1, respectivamente, en unión de todos los números naturales, considerados como símbolos de rango 0.

Un *árbol (rotulado)*  $t$  es un par que consta de un símbolo raíz  $f$  y  $n$  subárboles  $t_1, \dots, t_n$  los cuales son denotado por  $f[t_1, \dots, t_n]$ . Para  $n=0$ , se puede abreviar  $f[]$  como  $f$ . Notar que, por esta convención todo símbolo de rango 0 es identificado con el nodo simple del árbol cuya raíz es rotulada con este símbolo. Notar la relación entre símbolo de rango  $n$  y los  $n$ -hijos que posee el árbol.  $\Sigma^n$  con  $n \in \mathbb{N}$ , denota el conjunto de todos los símbolos en  $\Sigma$ , cuyo rango es  $n$ .

En el siguiente ejemplo, hemos considerado la expresión infija  $11 + fac[3+2]$ , la cual origina un árbol  $t$  sobre la signatura  $\Sigma$ , el cual podría ser denotado por  $+[11, fac+[3,2]]$ , usual en lenguajes funcionales.

*Ejemplo:*



Una *signatura* o árbol se dice *monádico* si ningún símbolo de rango 2 o mayor ocurre en él. La *altura* de un árbol  $t$  es la máxima longitud de un camino desde la raíz a la hoja. Entonces, si  $t$  es un símbolo de rango 0 entonces la altura es 0. Por otra parte, si  $m$  es la altura máxima de sus subárboles entonces la altura de  $t$  es  $m + 1$ .

Si  $T$  es el conjunto de árboles y  $\Sigma$  una *signatura* entonces el conjunto  $T_\Sigma(T)$  denota el conjunto de árboles sobre  $\Sigma$  con subárboles en  $T$ . El conjunto  $T_\Sigma(\emptyset)$  de árboles sobre  $\Sigma$  es denotado por  $T_\Sigma$ .

$\Sigma(T)$ , denota el conjunto de los árboles  $f[t_1, \dots, t_n]$ , tal que  $f \in \Sigma^n$  y  $t_1, \dots, t_n \in T$ , para todo  $i \in [n]$ .

### **Sustitución e Iteración.**

Consideremos una *signatura* infinita  $X = \{x_1^{(0)}, x_2^{(0)}, \dots\}$  como una *signatura* de símbolos distintos llamados *variables* y denotamos por  $X_n$  el subconjunto  $\{x_1, \dots, x_n\}$ , para todo  $n \in \mathbb{N}$ . Para evitar confusiones las variables son consideradas como símbolos especiales que tienen rango 0 y no son admitidas que ocurran en una *signatura* común. Para una *signatura*  $\Sigma$ ,  $\Sigma$  y  $X$  son disjuntos, entonces  $T_\Sigma(X)$  es el conjunto de todos los árboles sobre  $\Sigma \cup X$ .

Para un árbol  $t$ , con subárboles  $t_1, \dots, t_n$ , denotamos  $t[[t_1 \dots t_n]]$  al árbol obtenido por la sustitución de  $x_i$  por  $t_i$  en  $t$  con  $i \in [n]$ . Es decir, si  $t = x_i$  para algún  $i \in [n]$  entonces  $t[[t_1 \dots t_n]] = t_i$  y si  $t = f[s_1, \dots, s_k]$  con  $f \notin X_n$  entonces  $t[[t_1 \dots t_n]] = f[s_1[[t_1 \dots t_n]], \dots, s_k[[t_1 \dots t_n]]]$ .

Una regla o producción de reemplazo es un par  $\rho = (l, r)$  de árboles, llamado lado izquierdo y derecho de la producción respectivamente tal que  $l$  está en  $X$  y toda variable en  $r$  también ocurre en  $l$ , usualmente denotada por  $l \rightarrow r$ . Consideremos algún  $n \in \mathbb{N}$ , tal que  $X_n$  contiene todas las variables que ocurren en  $l$ . Entonces,  $\rho$  determina la relación binaria  $\rightarrow_\rho$  de árboles tal que  $t \rightarrow_\rho t'$  si  $t$  puede ser escrito como  $t_0[[l[[t_1 \dots t_n]]]]$  para un árbol  $t_0$  que contiene  $x_1$  exactamente una vez, y  $t'$  igual a  $t_0[[r[[t_1 \dots t_n]]]]$ . Si  $P$  es el conjunto de reglas o producciones,  $\rightarrow_P$  denota la unión de todas las  $\rightarrow_\rho$  tal que  $\rho \in P$ . Como es usual,  $t \rightarrow_R t'$  se llama *etapa de derivación* y una sucesión  $t_0 \rightarrow_P t_1 \rightarrow_P \dots \rightarrow_P t_k$  ( $k \geq 0$ ) de etapas de derivación, es una *derivación*, también denotada por  $t_0 \rightarrow^*_P t_k$ .

Un conjunto  $L$  de árboles es llamado *lenguaje de árbol* si  $L \subseteq T_\Sigma$ , para alguna *signatura* finita  $\Sigma$ .

### **Álgebras.**

Si  $\Sigma$  es una *signatura*, una  $\Sigma$ -álgebra es un par  $\Delta = (A, (f_\Delta)_{f \in \Sigma})$ , donde  $A$  es un conjunto, llamado el dominio de  $A$ , y para toda  $f^{(n)} \in \Sigma$ ,  $f_\Delta: A^n \rightarrow A$  es una función llamada *interpretación* de  $f$  en  $\Delta$ . El *valor*  $val_\Delta(t)$  de  $t \in T_\Sigma$  con respecto a una  $\Sigma$ -álgebra  $\Delta$  está definida mediante:

Si  $t = f[t_1, \dots, t_n]$  entonces  $val_\Delta(t) = f_\Delta(val_\Delta(t_1), \dots, val_\Delta(t_n))$ .

**Definición** (*gramática de árbol regular*). Una *gramática de árbol regular*  $g$  es una 4-upla  $g = (N, \Sigma, P, S)$ , donde  $N$  es un conjunto finito de símbolos llamados *no-terminales*,  $\Sigma$  es una *signatura* finita tal que  $A : 0 \notin \Sigma$ , para todo  $A \in N$ ,  $P$  es un conjunto finito de *producciones* de la forma  $A \rightarrow t$ , donde  $A \in N$  y  $t \in T_\Sigma(N)$ , y  $S \in N$  es el *no-terminal inicial* (o *axioma*). El lenguaje generado por  $g$  es  $L(g) = \{t \in T_\Sigma \mid S \rightarrow^*_P t\}$ .

Note que, si  $S \rightarrow^*_P t$  es una derivación en una *gramática regular* de árboles entonces se tiene que  $t \in T_\Sigma(N)$ . Esto es, que la forma sentencial es similar a las gramáticas de string de caracteres regular, en donde el lado derecho debería ser ya sea un terminal o un terminal junto con un no terminal. Consideremos como ejemplo, la siguiente *gramática regular* de árboles.

### Ejemplo 1:

Sea  $\mathbf{g} = (\{A, B\}, \Sigma, P, A)$ , una gramática regular de árbol, con signatura  $\Sigma = \{a^{(2)}, b^{(2)}, \#^{(0)}\}$ , y producciones  $P = \{A \rightarrow a[A, B], A \rightarrow a[B, A], A \rightarrow \#, B \rightarrow b[B, B], B \rightarrow \#\}$ .  $L(\mathbf{g})$  consiste de todos los árboles en el conjunto  $T_\Sigma(T)$ , que contiene un único camino  $a$ -rotulado. Más preciso, un árbol pertenece a  $L(\mathbf{g})$  si y solo si es igual a  $\#$  ó lee  $a[t_1, t_2]$ , donde uno de  $t_1, t_2$  está en  $L(\mathbf{g})$  y el otro está en el árbol sobre  $\{b^{(2)}, \#^{(0)}\}$ . Una muestra de la derivación es mostrada en Fig.1.

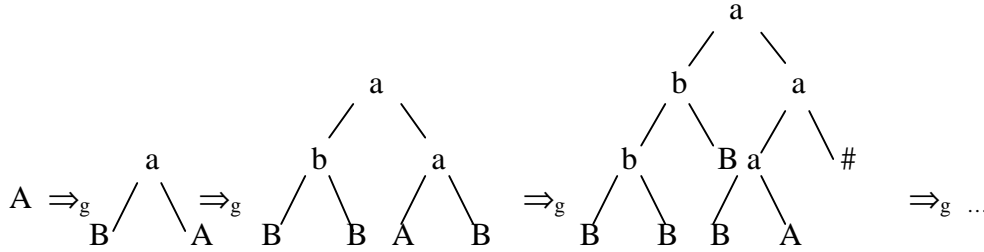


Fig. 1 Una derivación para la gramática de árbol regular.

**Observación:** Cabe hacer notar que existe una estrecha relación entre el conjunto de árboles de derivación de una gramática de contexto libre (Chomsky) basado en cadenas de caracteres y las gramáticas de árbol regulares, sin embargo esto no debería sorprendernos ya que todo lenguaje de árbol regular es una proyección de un lenguaje de árbol de derivación de una gramática de contexto libre basado en cadenas de caracteres. Por otra parte, el conjunto de árboles de derivación de una gramática de contexto libre pertenece a un lenguaje de árbol regular. Estas observaciones son caracterizaciones de los lenguajes de contexto libre y aparecen formalizadas en [GS97, Prop14.2 y 14.3].

Un *transductor de árboles* es una relación binaria  $\tau \subseteq T_\Sigma \times T_{\Sigma'}$ , donde  $\Sigma$  and  $\Sigma'$  son signaturas de entrada y salida, respectivamente. En particular, los llamados transductores de árboles top-down, y los transductores de árboles bottom-up que fueron introducidos por Rounds y Thatcher [Rou70,Tha70], son de gran importancia en este trabajo. Formalmente, digamos que los transductores de árbol top-down serán principalmente usados como generador de árboles, es decir estamos más interesados en el conjunto  $L(\tau)$  de árboles de salida más que de relación entrada-salida de árboles.

**Definición** (*transductor de árbol top-down*). Un *transductor de árbol top-down* es una 5-upla  $\mathbf{td} = (\Sigma, \Sigma', \Gamma, R, \gamma_0)$ , donde

- $\Sigma, \Sigma'$  son signaturas finitas,
- $\Gamma$  es un conjunto finito de símbolos (*estados*), tal que  $\gamma : 1 \notin \Sigma \cup \Sigma'$  para todo  $\gamma \in \Gamma$ ,
- $\gamma_0 \in \Gamma$  es un *estado inicial*, y
- $R \subseteq \Gamma(\Sigma(X)) \times T_{\Sigma'}(\Gamma(X))$  es un conjunto finito de reglas de sustitución de árboles lineal izquierdo.

La transformación de árboles realizados por  $\mathbf{td}$  (también denotado por  $\mathbf{td}$ ) es el conjunto de pares  $(\mathbf{t}, \mathbf{t}') \in T_\Sigma \times T_{\Sigma'}$ , tal que  $\gamma_0[\mathbf{t}] \rightarrow_{\mathbf{td}}^* \mathbf{t}'$ .

Debido a la forma especial de las producciones, un transductor de árboles top-down transforma entrada de árboles en árboles de salida leyendo los símbolos de arriba hacia abajo (top-down).

### Ejemplo 2:

Aquí se muestra la relación existente entre las gramáticas de cadenas de contexto libre y su árbol de derivación con las gramáticas de árbol y transductores. Sea  $\{ (, ), +, \times, a, b, \dots, z \}$  un alfabeto. Definamos una gramática de contexto libre de cadena de caracteres  $G_1$  por las siguientes reglas, en donde  $E$  es el axioma:

$$E \rightarrow ME', E' \rightarrow +E | \lambda, M \rightarrow FM', M' \rightarrow \times M | \lambda, F \rightarrow I | (E) \text{ e } I \rightarrow a | b | c | \dots | z.$$

Consideremos la palabra  $u = (a + b) \times c$ , cuyo árbol asociado es  $\times ( + (a, b), c)$ , definido sobre el alfabeto rankeado  $\{ +^{(2)}, \times^{(2)}, a^{(0)}, b^{(0)}, c^{(0)} \}$ . La siguiente transformación asocia a un árbol sintáctico  $t$  la transformación llevada a cabo por el transductor.

$$I(x) \rightarrow x, \quad M(x, M'(\lambda)) \rightarrow x, \quad M(x, M'(\times, y)) \rightarrow \times(x, y), \quad F((, x, )) \rightarrow x, \quad F(x) \rightarrow x, \\ E(x, E'(\lambda)) \rightarrow x, \quad E(x, E'(+, y)) \rightarrow +(x, y)$$

El conjunto de árboles de derivación obtenidos de  $G_1$  son asociados con un árbol y computado por un transductor de árbol top-down  $A'$ . Los estados de  $A'$  son  $q_E, q_F$ , y  $q_M$ , siendo el estado inicial  $q_E$ . El conjunto de reglas es:

$$q_E(x) \rightarrow E(q_M(x), E'(\lambda)), \quad q_M(x) \rightarrow M(q_F(x), M'(\lambda)), \quad q_F(x) \rightarrow F((, q_E(x), )), \quad q_F(b) \rightarrow F(I(b)) \\ q_F(a) \rightarrow F(I(a)), \quad q_F(c) \rightarrow F(I(c)), \quad q_E(+ (x, y)) \rightarrow E(q_M(x), E'(+, q_E(y))) \\ q_M(\times(x, y)) \rightarrow M(q_F(x), M'(\times, q_M(y))).$$

La Fig.2 muestra la ejecución del transductor  $A'$ , transformando un árbol abstracto  $+(a, \times(b, c))$  en un árbol sintáctico  $t'$  de la palabra  $a + b \times c$ .

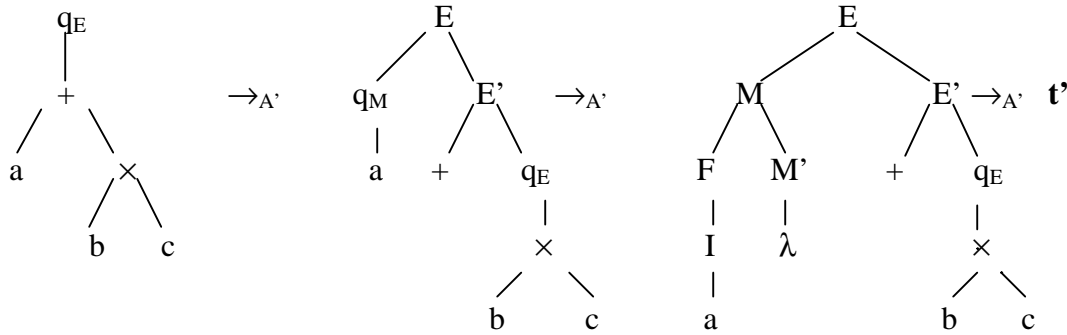


Fig.2 Ejemplo de ejecución de  $A'$ .

### Ejemplo 3:

Sea  $td = (\Sigma, \Sigma', \{q_0, q_1, q_2\}, R, q_0)$ , donde  $\Sigma = \{ a^{(2)}, b^{(2)}, \#^{(0)} \}$ ,  $\Sigma' = \Sigma \cup \{ o^{(2)} \}$  y  $R$  consiste de las reglas,

$$q_0 \# \rightarrow o[\#, \#], \quad q_0 c \rightarrow o[ c [q_1 x_1, q_1 x_2], c [q_2 x_2, q_2 x_1] ] \text{ para } c \in \{ a, b \}, \\ q_0 c \rightarrow o[ c [q_2 x_2, q_2 x_1], c [q_1 x_1, q_1 x_2] ] \text{ para } c \in \{ a, b \}, \quad q \# \rightarrow \# \text{ para } q \in \{ q_1, q_2 \}, \\ q_1 c \rightarrow c [q_1 x_1, q_1 x_2] \text{ para } c \in \{ a, b \} \text{ y } q_2 c \rightarrow c [q_2 x_2, q_2 x_1] \text{ para } c \in \{ a, b \}.$$

En efecto,  $td$  transforma todo árbol  $t \in T_\Sigma(T)$ , en un árbol  $o[t, t']$  y  $o[t', t]$ , tal que intuitivamente,  $t'$  es obtenido por el reflejo de  $t$  basado en el eje vertical( donde  $q_1$  es el estado que produce  $t$  en la salida y  $q_2$  produce la imagen refleja de  $t$  al revertir el orden de los dos subárboles en todo nodo que no sea hoja).

Entonces  $L(G)$  es el conjunto de todos los árboles  $o[t, t']$  tal que  $t$  contiene un camino de  $a$ 's desde la raíz hasta alguna hoja rotulada por  $\#$ , y todo otro nodo binario de  $t$  ha sido rotulado por  $b$ 's, y  $t'$  es reflejo de  $t$  sobre el eje vertical, como puede verse en la Fig.3.

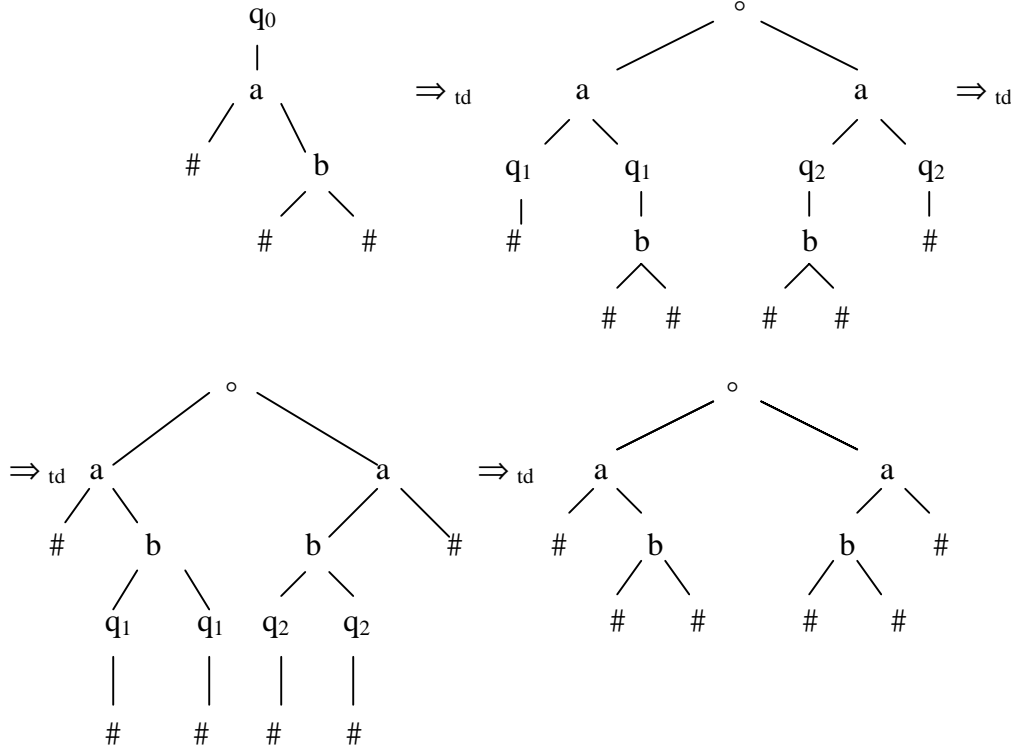


Fig.3 Una derivación por un transductor.

### 3 SISTEMA GENERADOR E INTERPRETE DE ARBOLES.

Ahora es posible definir la noción de *generador top-down de árbol*, que es el tipo de sistema que se usará a través del trabajo. Un *generador top-down de árbol* consiste de una *gramática regular de árboles* y una *sucesión finita de transductores top-down de árboles*, el cual es capaz de generar un lenguaje de árboles, que son obtenidos por el lenguaje generado por las gramáticas regulares de árboles y la aplicación sistemática de los transductores de árboles. Ahora, si le incorporamos una  $\Sigma$ -álgebra  $P$  al generador top-down de árbol cuya *signatura de salida* es un subconjunto de  $\Sigma$  entonces el par  $(g, P)$  es llamado *generador de formas basados en árboles con intérprete*. Se podría identificar  $(g, P)$  con  $g$  y denotar  $L_P(g)$  por  $L(g)$ .

**Definición** (*generador top-down de árboles*). Un *generador top-down de árboles* (**td**-generador) es un par  $G = (g, td_1 \dots td_n)$  que consiste de una gramática de árbol regular y una sucesión finita de  $td$  transductores  $td_1 \dots td_n$  (para algún  $n \in \mathbb{N}$ ), de manera que

- (i)  $g$  y  $td_1 \dots td_n$  son  $\subseteq N \times \Sigma(N)$  y que
- (ii) para todo  $i \in [n]$ , la *signatura entrada* de  $td_i$  coincide con la *signatura de salida* de  $td_{i-1}$ , para todo  $i > 1$  y de  $g$  para  $i=1$ .

$G$  genera el lenguaje  $L(G) = td_n(\dots(td_1(L(g)))) \dots$ . En lo sucesivo, la gramática de árbol regular  $g$  de un  $td$  generador  $G$  es denotada por  $g_G$  y la composición  $td_n \circ \dots \circ td_1$  es denotada por  $\tau_G$ .

#### Ejemplo 4:

Sea  $G = (g, td)$  *td*-generador, donde  $g = (\{A, B\}, \Sigma, P, A)$  es la gramática regular de árbol, y signatura  $\Sigma = \{a^{(2)}, b^{(2)}, \#^{(0)}\}$  y  $P = \{A \rightarrow a[A, B], A \rightarrow a[B, A], A \rightarrow \#, B \rightarrow b[B, B], B \rightarrow \#\}$ .  $L(g)$  consiste de todos los árboles en el conjunto  $T_\Sigma(T)$ , que contiene un único camino *a*-rotulado, como lo muestra el Ejemplo1. Por otra parte, consideremos  $td = (\Sigma, \Sigma', \{q_0, q_1, q_2\}, R, q_0)$ , donde  $\Sigma = \{a^{(2)}, b^{(2)}, \#^{(0)}\}$  es como en la gramática  $g$  anterior.  $\Sigma' = \Sigma \cup \{o^{(2)}\}$  y  $R$  las reglas dadas en Ejemplo3. En efecto,  $td$  transforma todo árbol  $t \in T_\Sigma(T)$ , en un árbol  $o[t, t']$  y  $o[t', t]$ , tal que intuitivamente,  $t'$  es obtenido por el reflejo de  $t$  basado en el eje vertical( donde  $q_1$  es el estado que produce  $t$  en la salida y  $q_2$  produce la imagen refleja de  $t$  al revertir el orden de los dos subárboles en todo nodo que no sea hoja). Luego,  $L(G)$  es el conjunto de los árboles  $o[t, t']$  tal que  $t$  contiene un camino de *a*'s desde la raíz hasta alguna hoja rotulada por  $\#$ , y todo otro nodo binario de  $t$  ha sido rotulado por *b*'s, y  $t'$  es reflejo de  $t$  sobre el eje vertical.

#### 4 ALGORITMO DE INFERENCIA

En esta sección nos basamos en los *td-generador con intérprete*, que es un sistema generador de árboles los cuales son interpretados por álgebras Collage, cuya visualización es realizada por el sistema TreeBag. La variante en este sentido radica en que tratar los modelos a nivel de árbol resulta ser más ventajosas que analizar conceptualmente la forma pictórica. Más aún, al considerar este sistema se tiene la ventaja, (vea [Dre00]) que las gramáticas regulares de árbol son equivalentes a las gramáticas Collage introducidas por Habel y Kreowski en [HK91]. El algoritmo de inferencia está basado en la generación de *td-generador*, resultando en forma natural la extensión a *td-generador con intérprete* al considerar el sistema TreeBag. Cabe mencionar que al considerar álgebras Collage como *intérprete* en nada restringe los alcances de las definiciones básicas previas. El algoritmo de inferencia que se propone es un mecanismo no-determinístico que infiere un *td-generador* a partir de un conjunto finito  $S_+$  de árboles, de manera que el lenguaje *td-generador* contenga los modelos.

**Input:**  $S_+ = \{t_1, t_2, t_3, \dots, t_n\}$ , conjunto finito de árboles.

**Salida:**  $G = (g, td)$ , *td-generador*, en donde  $S_+ \subseteq L(G)$ .

El algoritmo de inferencia consiste principalmente de las siguientes etapas:

**Etapas:** Construcción de la gramática regular de árboles  $g$ :

- Representar cada árbol  $t_i$  del conjunto  $S_+$  a través de su forma sintáctica que llamaremos  $\alpha_i$  en donde cualquiera de los subárboles  $\alpha_1, \alpha_2, \dots, \alpha_n$  consisten de al menos una subestructuras que eventualmente se pueden repetir. Cada una de las subestructuras se forma partiendo desde la raíz en top-down determinando subárboles con profundidad 1, para todo árbol del conjunto  $S_+$ .
- Se procede a excluir los subárboles que mantienen una subestructura repetitiva.
- Identificar símbolos no-terminales a nodos y construir una gramática canónica de árbol  $g_i$  para  $t_i$ , para todo  $i$ .
- La gramática de árbol regular inferida para el conjunto completo de ejemplos en  $S_+$  será entonces:

$$g_{total} = \bigcup_{t_i \in S_+} g_i,$$

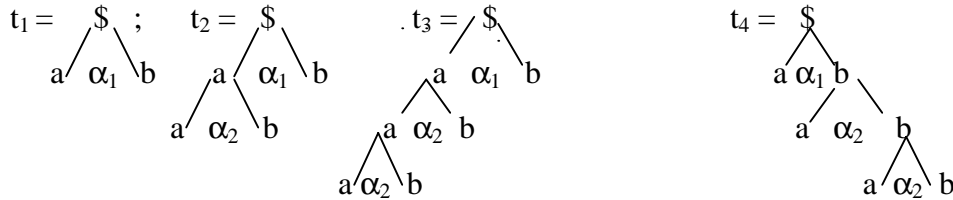


**Etap2:** Construcción del td-generator,  $G = (g_{total}, \lambda)$ .

**Etap3:** Construcción de un td-generator con intérprete, cuya ejecución es realizada por el sistema TreeBag, a lo que se obtiene una visualización de los árboles interpretados por el álgebra.

### Ejemplo 5:

Sea  $S_+ = \{ t_1, t_2, t_3, t_4 \}$  conjunto finito de ejemplos, cuya sintaxis es,



en donde la forma estructural  $\alpha_i$  consiste de subestructuras que eventualmente se pueden repetir, podemos encontrar solamente 2 en nuestros modelos, como se visualiza en la figura anterior. Notar  $\alpha_1$  y  $\alpha_2$  son modelos distintos, pues se generan a partir de raíces distinta, en un caso son  $a$ 's y en otro son  $b$ 's. Luego las reglas o producciones que forman parte de  $P$  son las siguientes:

$$S \rightarrow \$[A, B], A \rightarrow a[A, B], B \rightarrow b[A, B], A \rightarrow a, B \rightarrow b$$

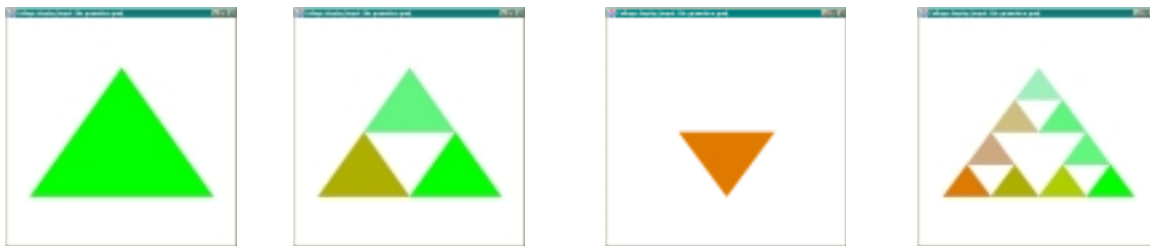
La gramática regular de árbol es  $g = (\{S, A, B\}, \{a^{(2)}, b^{(2)}\}, P, S)$ , generando el conjunto de todos los árboles binarios con  $a$  sobre la rama izquierda y  $b$  sobre la rama derecha.

### Ejemplo 6:

Sea  $S_+ = \{ F[S, S, S], G[A], S[H], G[G[A]], G[G[G[A]]], A[H], F[S, S, G[F'[S, S, S]]], F[S, S, G[F'[H, H, H]]], G[G[G[F'[S, S, S]]], S[G[F'[S, S, S]]] \}$  un conjunto finito de ejemplos, formas o modelos, extraídos de un lenguaje de árbol. Basado en nuestro algoritmo de inferencia podemos encontrar las gramáticas canónicas que describen cada una de los árboles dependiendo de la forma estructural de cada uno de ellos, las que en algunos casos se pueden repetir. Analizando cada una de las gramáticas, podemos aún eliminar e identificar no terminales, con el fin de encontrar las producciones generativos que dan origen a la gramática de árbol regular. En nuestro caso hemos obtenido la gramática regular  $g = (\{S, A\}, \{F^{(3)}, F'^{(3)}, G^{(1)}, H^{(0)}\}, P, S)$ , en donde las producciones son:

$$P = \{ S \rightarrow F[S, S, S], S \rightarrow G[A], S \rightarrow H, A \rightarrow F'[S, S, S], A \rightarrow G[A], A \rightarrow H \}.$$

El td-generator es  $G = (g_{total}, \lambda)$ , de donde  $L(G)$  contiene al menos las formas pictóricas de  $S_+$ , es decir  $S_+ \subseteq L(G)$ . La etapa 3 se realiza a través del sistema TreeBag, generándose las siguientes figuras.



## 5 CONCLUSION

En este trabajo se muestran los resultados preliminares de la construcción de un mecanismo capaz de generar las formas o modelos de un cierto lenguaje, (lenguaje de árboles o lenguaje Collage) a través de la generación de las mismas por las gramáticas de árboles y transductores ( o gramáticas Collage) con la interpretación de un álgebra apropiada. Ahora, sabiendo que los lenguajes de árbol sobre alfabeto con rango son cerrados bajo unión, complemento e intersección, hecho que es de suma importancia para los algoritmo de inferencia, se puede determinar el complemento relativo de un lenguaje  $L_1$  con respecto a otro lenguaje  $L_2$ , construyendo  $L_1 - L_2 = \neg(\neg L_1 \cup L_2)$ .

Al mismo tiempo es posible visualizar los modelos asociados a los árboles a través del sistema TreeBag, hecho que se logra por la interpretación de los mismos realizados por el álgebra Collage, pudiendo utilizarse otras álgebras apropiadas, obteniéndose así una forma de inferir gramáticas a nivel de gramáticas Collage que no es más que la inferencia de las gramáticas de árboles que subyace a los modelos.

Las proyecciones futuras tienen varias directrices, una de ellas es la implementación de este algoritmo de inferencia, por otro lado surgen interrogantes, tales como. ¿Cuántas muestras u ejemplos son necesarios para determinar una gramática reveladora?, ¿Qué ocurre si ahora el sistema es dinámico?, es decir, si se tienen  $n$ -ejemplos en  $S_+$  a lo cual podemos construir una gramática  $G$  asociada, pero que pasa si llega un ejemplo  $n+1$ , ¿Cuál es  $G$ ?

**Reconocimientos:** Agradecemos a Sr. Mauro San Martín por las sugerencias y comentarios.

## Referencias

- [Bar88] Michael Barnsley. *Fractals Everywhere*, Academic Press, Boston, 1988.
- [CRA76]C. Cook, A. Rosenfeld, A. Aronson. *Grammatical Inference by Hill Climbing*, Informational Sciences 10, 59–80, 1976.
- [Dre00] Frank Drewes. *Tree-Based Picture Generation*. Theo.Comp. Sc. 246: 1-51, 2000.
- [Dre96] Frank Drewes. *Computation by tree transductions*. Doctoral dissertation, University of Bremen, Germany, 1996.
- [FB75] K.S. Fu, T. K. Booth: *Grammatical Inference Introduction an Survey Part I y II*, IEEE-Trans. Syst. Man and Cyber. 5, 95-111 y 409-423, 1975.
- [FV98] Zoltán Fülöp, Heiko Vogler. *Syntax-Directed Semantics: Formal Models Based on Tree Transducers*. Springer, 1998.
- [GS84] F. Gécseg, M. Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.
- [GS97] F. Gécseg, M. Steinby. *Tree Languages*. In G. Rozenberg and A. Salomaa, editores, Handbook of Formal Languages. Vol. III: Beyond Words, Cap.I, pag. 1-68. Springer, 1997.
- [HK91]A. Habel, H.-J. Kreowski. *Collage Grammars*, Lect. Not. Comp. Sci. 532, 411-429, 1991.
- [Je95] Eric Jeltsch. *Un Algoritmo para inferir Gramáticas de Grafos*, Actas del III Encuentro Chileno de Computación, 1995.
- [JL87] H. Jürgensen, A. Lindenmayer. *Inference Algorithms for Developmental Systems with Cell Lineages*, Bulletin of Mathematical Biology 49, Nr.1, 93-123, 1987.
- [PJS92] Heinz-Otto Peitgen, Hartmut Jürgens y Dietmar Saupe. *Chaos and Fractals*. New Frontiers of Science. Springer-Verlag, New York, 1992.
- [PL90] P. Prusinkiewicz , A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

- [RN88] V. Radhakrishnan, G. Nagaraja. *Inference of Even Linear Grammar and Its Application to Picture Description Languages*, Pattern Recognition, Vol 21, No.1, 55-62, 1988.
- [Rou70] W. C. Rounds. *Mapping and Grammars on Trees*. Mathematical Systems Theory, 4:257-287, 1970.
- [Tha70] James W. Thatcher. *Generalized sequential machine maps*. Journal of Computer and System Sciences, 4:339-367, 1970.
- [Web1] <http://eurise.univ-st-etienne.fr/gi/gi.html>.
- [Web2] <http://www.informatik.uni-bremen.de/theorie/treebag/>).