UNA VISUALIZACIÓN DE UN SISTEMA DE INFERENCIA A TRAVÉS DE TREEBAG

A VISUALIZATION OF INFERENCE SYSTEM THROUGH TREEBAG

Mauricio Aguirre[1] **Eric Jeltsch**[1] **Gerardo Rosales**[1]

Recibido 31 de mayo de 2006, aceptado 20 de marzo de 2007
Received: May 31, 2006 Accepted: March 20, 2007

RESUMEN

En este trabajo se desarrolla un algoritmo de inferencia gramatical, el cual posee como entrada un conjunto finito de árboles y como salida *top-down tree generator* (td generador), que consiste de una gramática de árbol regular y un conjunto finito de transductores. Se presenta también el sistema de *software TreeBag* (***Tree Ba****sed* ***G****enerador*), herramienta útil para la generación y visualización de objetos de todo tipo: grafos, árboles, string, etc.

Palabras clave: Gramática de árboles, transductores, inferencia gramatical, sistema *TreeBag.*

ABSTRACT

*In this paper, a grammatical inference algorithm is developed with finite sets of sample trees as inputs and top-down tree generator (td generator) a consisting of a regular tree grammars and a finite sequence of transducers as outputs. A software system called TreeBag (**Tree Ba**sed **G**enerator) is presented, its aim being is allowing for the generation and visualization of objects of all kinds such as: graph, trees, string, etc.*

Keywords: Tree grammars, transducers, grammatical inference, TreeBag system.

INTRODUCCIÓN

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 o, en la Bioinformática, a través de la aplicación de la computación a secuencias biológicas, tales como DNA o proteínas. Vea [1, 14, 13, 7 y 21].

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 Intérprete 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 [18].

[1] Universidad de La Serena, Av. Cisternas 1200, La Serena, Chile. ejeltsch@userena.cl

Una gran variedad de algoritmos de inferencia se han desarrollado, por ejemplo, Fu y Booth [6] genera gramáticas regulares a partir de modelos codificados como cadena de caracteres. En [2] 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 [15] desarrolla un método para gramáticas regulares desde ejemplos positivos, los cuales forman una estructura de esqueleto. En [12] Jürgensen y Lindenmayer consideran modelos representados por estructuras árboreas que generan OL-Sistemas. En [11] Kreowski y Jeltsch desarrollan un algoritmo de inferencia basado en grafos. En [22, 23] Jeltsch, Rosales y Aguirre presentan TreeBaG, un sistema para generar y visualizar formas pictóricas.

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 [4].

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; asimismo, los transductores tienen un comportamiento similar a los autómatas con salida.

Este trabajo está organizado de manera que se entregan en los preliminares los conceptos teóricos fundamentales, tales como gramáticas de árboles regulares y los transductores. En la sección "Sistema de generador de árboles" se define td-generador que es el sistema generador. En la sección siguiente se muestran las distintas etapas que componen el algoritmo de inferencia. En la sección "Visualización sobre el sistema TreeBag" se introduce el sistema TreeBaG 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.
- El despliegue de los valores que adquieren los árboles en la pantalla.

Todo esto, con el objetivo de brindar una visualización del algoritmo y de las etapas de derivación.

PRELIMINARES

En esta sección se entregan notaciones y conceptos teóricos que son utilizados. Para mayor información vea [8, 5 y 9].

Signaturas y árboles

El conjunto de los números naturales incluido el 0 es denotado por $\mathbb{N}$. El conjunto de todas las sucesiones (llamadas cadenas o palabras) sobre un conjunto S es denotado por S^*. S^+ es $S^* - \{\lambda\}$, donde λ 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 Σ 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 $t_1, \ldots, t_n$ los cuales son denotado por $f[t_1, \ldots, t_n]$. 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. Σ^n con $n \in \mathbb{N}$ denota el conjunto de todos los símbolos en Σ, cuyo rango es *n*. En el siguiente ejemplo, hemos considerado la expresión infija 11 + fac[3+2], la cual origina un árbol **t,** tal como se muestra en figura 1, sobre la signatura Σ el que 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 Σ una signatura entonces el

conjunto $T_\Sigma(T)$ denota el conjunto de árboles sobre Σ con subárboles en T. El conjunto $T_\Sigma(\phi)$ de árboles sobre Σ es denotado por T_Σ. Σ(T), denota el conjunto de los árboles $f[t_1, \ldots, t_n]$, para $f \in \Sigma^n$ y $t_1, \ldots, t_n \in T$.

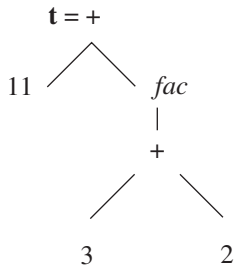

Figura 1. Expresión infija de **t**.

Sustitución e iteración

Consideremos la signatura infinita $X = \{x_1^{(0)}, x_2^{(0)}, \ldots\}$ como una signatura de símbolos distintos llamados *variables* y denotamos por X_n el subconjunto $\{x_1, \ldots, x_n\}$, 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 Σ, Σ 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 $t_1, \ldots, t_n$, denotamos **t** $[[t_1 \ldots t_n]]$ 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** $[[t_1 \ldots t_n]] = t_i$ y si **t** = $f[s_1, \ldots, s_k]$ con $f \notin X_n$ entonces **t** $[[t_1 \ldots t_n]] = f[s_1[[t_1 \ldots t_n]], \ldots, s_k[[t_1 \ldots t_n]]]$.

Una 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 $l \to r$. Consideremos algún $n \in \mathbb{N}$, tal que X_n contiene todas las variables que ocurren en *l.* Entonces, ρ determina la relación binaria $\to_\rho$ de árboles tal que **t** $\to_\rho$ **t'** si **t** puede ser escrito como t_0 $[[\, l\, [[t_1 \ldots t_n]]\,]]$ para un árbol t_0 que contiene x_1 exactamente una vez, y **t'** igual a t_0 $[[\, r\, [[t_1 \ldots t_n]]\,]]$. Si P es el conjunto de reglas o producciones, $\to_P$ denota la unión de todas las $\to_\rho$ tal que $\rho \in P$. Como es usual, **t** $\to_R$ **t'** se llama *etapa de derivación* y una sucesión:

$$t_0 \to_P t_1 \to_P \ldots \to_P t_k \ (k \geq 0)$$

es una *derivación,* también denotada por

$$t_0 \to {}^*_P\, t_k.$$

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

Álgebras

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

Si **t** = $f[t_1, \ldots, t_n]$ entonces $val_\Delta(\mathbf{t}) = f_\Delta(val_\Delta(t_1), \ldots, val_\Delta(t_n))$.

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

Note que, si $S \to {}^*_P\, \mathbf{t}$ es una derivación en una gramática regular de árboles entonces se tiene que **t** $\in T_\Sigma(T)$. 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 **g** = ({ A, B}, Σ, P, A), una gramática regular de árbol, con signatura $\Sigma = \{ a^{(2)}, b^{(2)}, \#^{(0)} \}$, y producciones P = { A → a[A, B], A → a[B, A], A → #, B → b[B, B], B → #}.

L(**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(**g**) si y solo si es igual a # o lee $a[t_1, t_2]$, donde uno de t_1, t_2 está en L(**g**) y el otro está en el árbol sobre $\{ b^{(2)}, \#^{(0)} \}$.

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 [9].

Un *transductor de árboles* es una relación binaria $\tau \subseteq T_\Sigma \times T_{\Sigma'}$, donde Σ and Σ' 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 Tatcher [16] y [17], 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 **td** = (Σ, Σ', Q, R, q_0), donde

i) Σ, Σ'son signaturas finitas,
ii) Q es una signatura finita de (*estados)*, de rango 1, tal que $Q \cap (\Sigma \cup \Sigma') = \emptyset$,
iii) $R \subseteq \Gamma(\Sigma(X)) \times T_{\Sigma'}(\Gamma(X))$ es un conjunto finito de reglas.
iv) $q_0 \in \Gamma$ es un *estado inicial*.

La transformación de árboles realizados por td (también denotado por td) es el conjunto de pares (**t**, **t'**) $\in T_\Sigma \times T_{\Sigma'}$, tal que $q_0[\mathbf{t}] \rightarrow^*_R \mathbf{t'}$. 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: Sea **td** = (Σ, Σ', $\{q_0, q_1, q_2\}$, R, q_0), donde $\Sigma = \{ a^{(2)}, b^{(2)}, \#^{(0)} \}$, $\Sigma' = \Sigma \cup \{o^{(2)}\}$ y R consiste de las reglas,

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

En efecto, **td** transforma todo árbol $\mathbf{t} \in T_\Sigma(T)$, 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.

SISTEMA GENERADOR E INTÉRPRETE DE ÁRBOLES

Ahora es posible definir la noción de *árbol generador top-down*, que es el tipo de sistema que se usará a través del trabajo. Un *árbol generador top-down* 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 Σ- álgebra P al generador top-down de árbol cuya signatura de salida es un subconjunto de Σ 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 (*árbol generador top-down*). Un *árbol generador top-down* (**td**-generador) G es un par $G = (g, td_1 \ldots td_n)$ que consiste de una gramática de árbol regular *g* y una sucesión finita de transductores $td_1 \ldots td_n$ (para algún $n \in N$), de manera que

(i) g y $td_1 \ldots td_n$ 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 $L(G) = td_n(\ldots(td_1(L(g)))\ldots)$.

En lo sucesivo, la gramática de árbol regular g de un **td** generador *G* es denotada por g_G y la composición $td_n \ldots td_1$ es denotada por τ_G .

Ejemplo 3: Sea G = (**g**, **td**) td-generador, donde **g** = ({A, B}, Σ, 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 Ejemplo 1. Consideremos **td** = (Σ, Σ',$\{q_0, q_1, q_2\}$, 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 Ejemplo 2. En efecto, **td** transforma todo árbol $\mathbf{t} \in T_\Sigma(T)$, 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.

ALGORITMO DE INFERENCIA

En esta sección construimos un algoritmo de inferencia, al cual se le incorpora la componente de ***td**-generador con intérprete* (similar a la definida en [3]), que es un sistema generador de árboles, cuyos intérpretes son álgebras, en este caso álgebras Collage (introducidas por Habel y Kreowski en [10]). Todo esto, con el objetivo de lograr una visualización a través del sistema TreeBaG.(vea [19]). 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.

Entrada: $S_+ = \{ t_1, t_2, t_3, \ldots, t_n\}$, conjunto finito de árboles.

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

El algoritmo de inferencia consiste en:

Etapa 1: Construcción de la gramática regular de árboles g:

a) Representar cada árbol t_i del conjunto S_+ a través de su forma sintáctica que consiste de al menos una subestructura que eventualmente se puede 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_+.

b) Se procede luego a excluir los subárboles que mantienen una subestructura repetitiva e identificar símbolos no-terminales a nodos y construir una gramática canónica de árbol g_i para t_i, para todo i. Entonces, la gramática de árbol regular inferida para el conjunto S_+ es,

$$g_{total} = \cup\, g_i, \text{ para todo } t_i \in S_+ .$$

Etapa 2: Construcción del td-generador, G = (g_{total}, λ).

Etapa 3: Construcción de un td-generador con intérprete, cuya ejecución es realizada por el sistemaTreeBaG.

Ejemplo 4: Sea $S_+ = \{ t_1, t_2, t_3, t_4 \}$ conjunto finito de ejemplos, que poseen una forma estructural que eventualmente se pueden repetir, tal como se muestra en este ejemplo, en donde encontramos solamente 2, y que son los que 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.

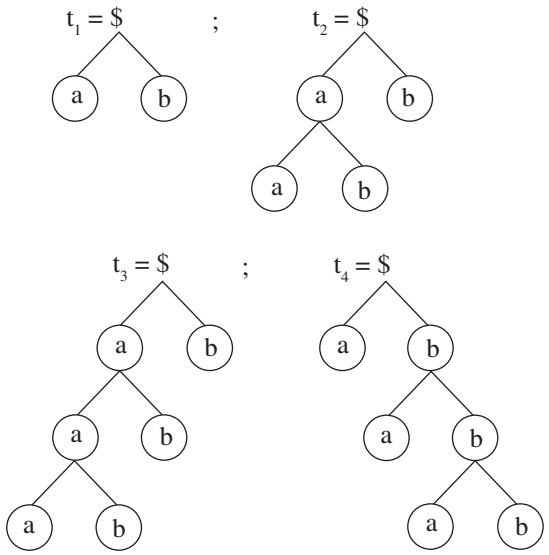

Figura 2. Sintaxis de los árboles t_1, t_2, t_3, t_4.

VISUALIZACIÓN SOBRE EL SISTEMA TREEBAG

Se utiliza el sistema TreeBaG para interpretar un tipo particular de generador de árbol top-down, llamado *td-generador*, el cual está compuesto de una gramática de árbol regular y un transductor de árbol. El objetivo final es visualizar el proceso generativo de la gramática de árbol, en donde los árboles son interpretados primeramente por un álgebra Collage2D, generándose formas pictóricas o modelos de un determinado dominio. Vea [22], para mayor detalle. La versión utilizada del sistema TreeBaG es la 1.1, para mayor detalle vea [23].

Ejemplo 5: 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 uno 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 generativas que dan origen a la gramática de árbol regular. Basado en algoritmo de inferencia descrito se obtiene la gramática regular g definida, 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-generador es $G = (g_{total}, \lambda)$, de donde L(G) contiene al menos las formas pictóricas de S_+.

Visualización vía álgebra Collage2D

Para su visualización, definamos un álgebra Collage capaz de interpretar los árboles primitivos en formas geométricas, tal como lo muestra la figura 3. Al elemento H de la gramática le hemos asociado la forma gráfica básica que es un triángulo inicialmente de color verde, lo mismo sucede con los elementos S y A pero con color inicial rojo y azul respectivamente. Los elementos propios del álgebra t_1, t_2 y t_3 realizan las transformaciones geométricas en 2 dimensiones. Se define que el elemento F está compuesto por las transformaciones t_1, t_2 y t_3; éstas serán aplicadas en cada inferencia a los elementos que componen F en la gramática. Por último, se define una transformación de color para el elemento G de la gramática.

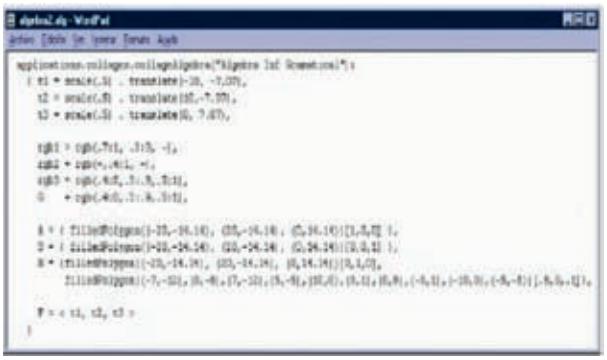

Figura 3. Álgebra collage asociada a la gramática inferida.

En la figura 5 se muestra el proceso generativo de la gramática inferida mediante la representación del árbol de derivación y el álgebra Collage2D. En la primera de ellas (esquina superior izquierda) podemos ver la representación del símbolo inicial **S**. En la segunda (a la derecha) se visualiza un árbol primitivo por la aplicación de la producción **S → F[S,S,S]** donde, el triángulo inferior-izquierdo corresponde al hijo izquierdo de **F** en el árbol de derivación, el triángulo inferior-derecho corresponde al hijo medio de **F** y el triángulo superior con el hijo derecho de F. En la tercera (esquina superior derecha) vemos que la rama izquierda de la raíz **F** del árbol tiene como hijo a **H** por la producción **S → H**, **H** se definió en la gramática como elemento terminal de rango 0, por lo que ya no se sigue ramificando, no así con los hijos medio **F** y derecho **G**.

La figura 4 muestra el proceso tras varias iteraciones.

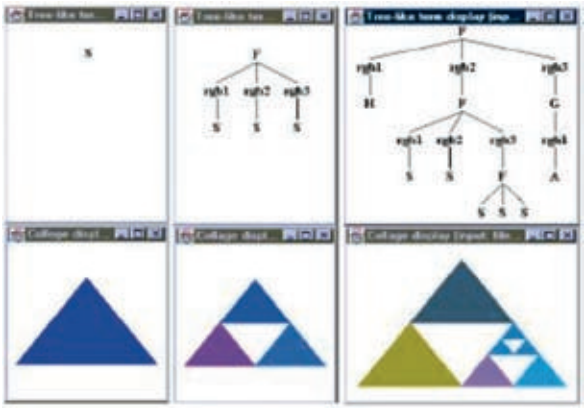

Figura 4. Visualización de un proceso generativo de árboles, vía álgebra Collage2D.

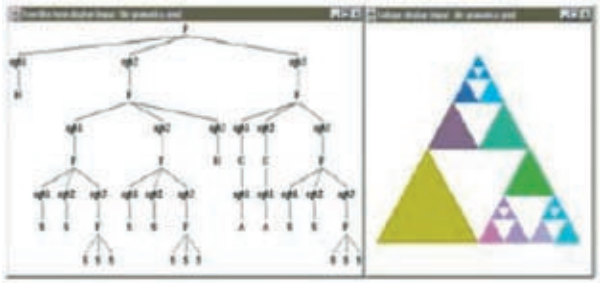

Figura 5. Proceso de inferencia tras varias iteraciones, interpretados por álgebra Collage2D.

Visualización vía transductores

El proceso de inferencia no ha considerado transductores sino que hemos aplicado solamente las gramáticas de árboles y la interpretación vía álgebra Collage2D, sin embargo el proceso de inferencia es posible de

generalizarlo si consideramos el potencial de TreeBaG en donde hemos utilizado en el proceso de derivación de árboles un transductor intermedio λ y un álgebra Collage arbitraria, generándose lo que muestra la figura 6, en donde a) muestra la visualización tras la interpretación de la regla de producción **S → F[S,S,S]** de la gramática g del ejemplo, en b) se visualiza la transformación de la regla anterior por el transductor λ al aplicar **P[F[x1,x2,x3]] → st2[P[x1],P[x3]]** descrita por la flecha, en c) se interpreta *cuadro* por un álgebra Collage.

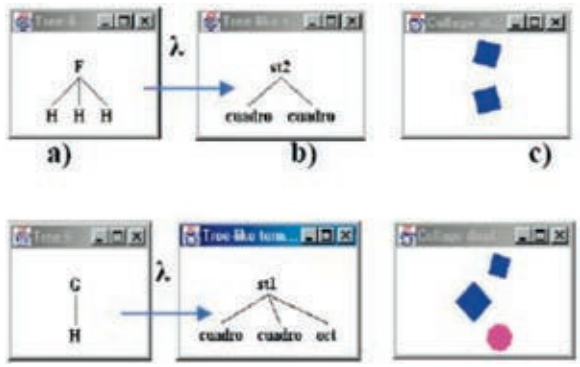

Figura 6. Transformación e Interpretación de un árbol.

Análogamente, en d) considera la producción **S → G[A]**, en e) se transforma mediante la regla **P[G[x1]] → st1[P[x1],P[x1],P[x1]]** descrita por la flecha en un árbol con 3 nodos hijos, y en f) se muestra la visualización del árbol a través de un álgebra collage arbitraria. Finalmente, luego de varias iteraciones se transforma la forma pictórica (figura 6), en otra forma pictórica, pero ésta afectada por el transductor λ, según muestra la figura 7.

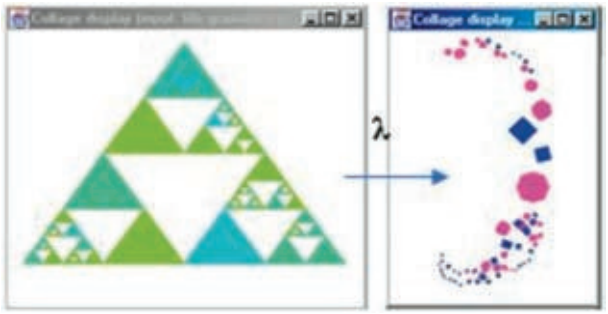

Figura 7. Transformación del triángulo de Sierpinsky mediante el transductor λ.

Visualización vía álgebra Collage3D y transductores

En el sistema TreeBaG es posible definir un álgebra collage3D, y así sea capaz de interpretar los árboles primitivos en formas geométricas tridimensionales. Para mostrar el proceso generativo de la gramática inferida mediante la representación del árbol de derivación y el álgebra collage3D, es que se ha considerado la figura 8. En la primera de ellas (esquina superior izquierda) podemos ver la representación del símbolo inicial e inmediatamente abajo se visualiza una **esfera** que es el álgebra asociada a la forma, que actúa como la interpretación del símbolo inicial. En la segunda (a la derecha) se tiene un árbol primitivo, luego de haber aplicado la regla **S→F[S,S,S]** e inmediatamente abajo la interpretación, que corresponde a la segmentación de la esfera. En la tercera (esquina superior derecha) vemos que la rama izquierda a partir de la raíz **F** del árbol tiene como hijo a **H** por la producción **S → H**, **H** se definió en la gramática como elemento terminal de rango 0, por lo que ya no es posible su ramificación, no así con sus hijos **F** y **G**.

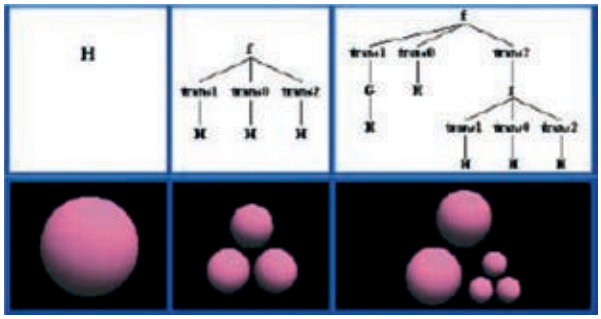

Figura 8. Visualización de un proceso generativo de árboles, vía álgebra Collage3D y transductores.

A continuación, en la figura 9 se ha realizado el proceso de inferencia tras varias iteraciones.

Notar que al proceso de inferencia se le puede integrar transductores tal como se hizo antes, obteniéndose con ello gramáticas de árboles y sus derivaciones en conjunto con su interpretación vía álgebras Collage3D. En general, al proceso de inferencia es posible de generalizarlo en el sentido de aplicar en el proceso de derivación de árboles un transductor intermedio λ y un álgebra Collage3D arbitraria. La figura 10 muestra la transformación de una esfera segmentada en estrellas.

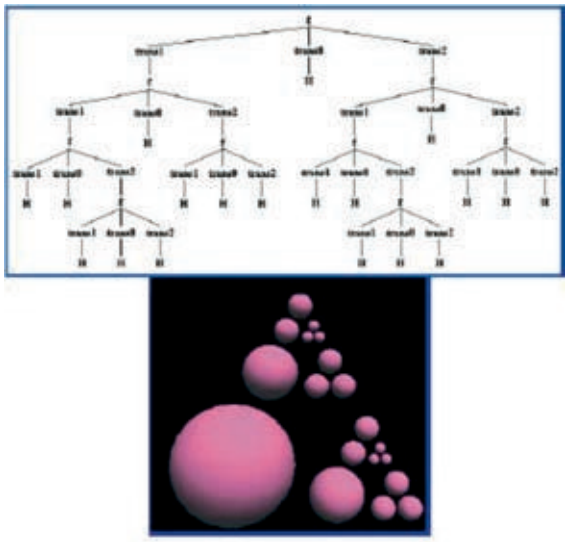

Figura 9. Proceso de inferencia tras varias iteraciones, interpretados por álgebra Collage3D y Transductores.

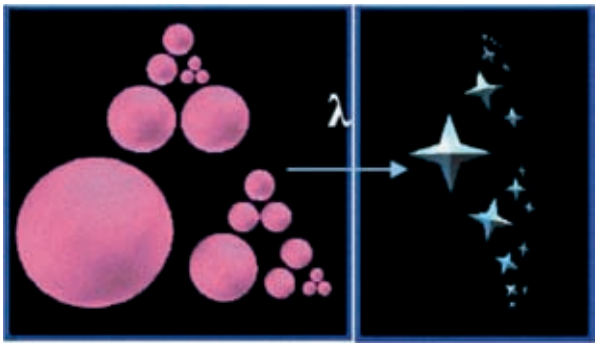

Figura 10. Transformación de la pirámide en estrellas mediante el transductor λ.

CONCLUSIONES

En este trabajo se muestran los resultados preliminares de un mecanismo capaz de generar las formas o modelos de un cierto lenguaje (lenguaje de árboles), a través de la generación de las mismas por las gramáticas de árboles y transductores con la interpretación de un álgebra apropiada. 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 dentro del sistema TreeBaG de un tipo de algoritmo de inferencia; sin embargo, surgen interrogantes, tales como: ¿Cuántas muestras o 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?

Por otra parte, sabemos que en la Bioinformática los métodos de Reconocimiento de Patrones, como sugiere su nombre, se basan en el supuesto de que alguna característica subyacente de una secuencia proteica, o de una estructura de proteína, puede emplearse para identificar caracteres semejantes en proteínas emparentadas. En otras palabras, si se mantiene o conserva parte de una secuencia o estructura, esta característica puede emplearse para determinar nuevos miembros de la familia. Y es en este sentido en donde TreeBaG puede ser de gran ayuda, para su visualización.

Finalmente, sigue siendo nuestro interés en construir nuevos ejemplos, álgebras y gramáticas como las aquí presentadas, ya sea en 2D como en 3D, con el fin de obtener nuevas interpretaciones.

AGRADECIMIENTOS

Este trabajo ha sido íntegramente financiado por Proyecto Nº 0220-2-20, DIULS (Dirección de Investigación de la Universidad de La Serena).

REFERENCIAS

[1] Michael Barnsley. Fractals Everywhere, Academic Press, Boston. 1988.

[2] C. Cook, A. Rosenfeld, A. Aronson. "Grammatical Inference by Hill Climbing", Informational Sciences Vol. 10, pp. 59-80. 1976.

[3] Frank Drewes. Tree-Based Picture Generation. Theo. Comp. Sc. Vol. 246, pp. 1-51. 2000.

[4] Frank Drewes, A. Habel, H-J. Kreowski y S. Taubenberger. "Generating self affine fractals by Collage Grammars. Theoretical Computer Science", Nº 187, pp. 145-159. 1995.

[5] F. Drewes. "Computation by tree transductions". Doctoral dissertation. University of Bremen". Germany. 1996.

[6] K.S. Fu, T. K. Booth: Grammatical Inference Introduction an Survey Part I y II, IEEE-Trans. Syst. Man and Cyber. Nº 5, pp. 95-111 y 409-423. 1975.

[7] Zoltán Fülöp, Heiko Vogler. Syntax-Directed Semantics:Formal Models Based on Tree Transducers. Springer. 1998.

[8] F. Gécseg, M. Steinby. Tree Automata. Akadémiai Kiadó, Budapest. 1984.

[9] F. Gécseg, M. Steinby. Tree Languages. In G. Rozenberg and A. Salomaa, editores, Handbook of Formal Languages. Vol. III: Beyond Words, Cap. I, pp. 1-68. Springer. 1997.

[10] A. Habel, H.-J. Kreowski. Collage Grammars, Lect. Not. Comp. Sci. 532, pp. 411-429. 1991.

[11] H.-J. Kreowski, Eric Jeltsch. Grammatical Inference based on hyperedge replacement. Lect. Not. Comp. Sci. 532, 461-474. 1991.

[12] H. Jürgensen, A. Lindenmayer. Inference Algorithms for Developmental Systems with Cell Lineages. Bulletin of Mathematical Biology. Vol. 49 Nº 1, pp. 93-123. 1987.

[13] Heinz-Otto Peitgen, Hartmut Jürgens y Dietmar Saupe. Chaos and Fractals. New Frontiers of Science. Springer-Verlag, New York, EE.UU. 1992.

[14] P. Prusinkiewicz, A. Lindenmayer. The Algorithmic Beauty of Plants. Springer-Verlag, New York, EE.UU. 1990.

[15] V. Radhakrishnan, G. Nagaraja. Inference of Even Linear Grammar and Its Application to Picture Description Languages, Pattern Recognition, Vol. 21 Nº 1, pp. 55-62. 1988.

[16] W. C. Rounds. Mapping and Grammars on Trees. Mathematical Systems Theory. Vol. 4, pp. 257-287. 1970.

[17] James W. Thatcher. Generalized sequential machine maps. Journal of Computer and System Sciences. Vol. 4, pp. 339-367. 1970.

[18] Homepage of the Grammatical Induction Community. http://eurise.univ-st-etienne.fr/gi/

[19] The TreeBag Homepage. http://www.informatik.uni-bremen.de/theorie/treebag/

[20] Theodore W. Hong and Keith L. Clark Using Grammatical Inference to Automate Information Extraction from the {Web}. Lecture Notes in Computer Science, Vol. 2168, pp. 216-220. 2001.

[21] D. López, A. Caro, M. Vásquez de Parga, B. Calles, J. Sempere, T. Pérez, J. Ruiz, P. García. Detection of functional Motifs in Biosequences: A grammatical approach. Proceeding 5th. Annual Spanish Bioinformatics Conference. Barcelona, España. 2004.

[22] E. Jeltsch, M. Aguirre, G. Rosales. Generación de formas pictóricas sobre un sistema de Inferencia a través de TreeBag. Actas del X Encuentro Chileno de Computación. Copiapó, Chile. 2002.

[23] E. Jeltsch, M. Aguirre, G. Rosales. Aplicaciones en TreeBag. Actas del IV Congreso de Educación Superior en Computación. Copiapó, Chile. 2002.