Gramáticas de Grafos y su Aplicación en Inferencia Gramatical

Eric Jeltsch F.*
Depto. Matemáticas,
Area de Computación,
Universidad de La Serena,
La Serena, Chile

RESUMEN: En este trabajo se presentan las gramáticas de grafos, en particular un tipo de gramáticas de grafos, las cuales tienen un contexto libre en el mecanismo de derivación, al igual que las gramáticas de Chomsky , proporcionando de esta manera un simple mecanismo para generar lenguajes de grafos. Para hacer transparente el proceso de generación de grafos, se introducen los grafos decorados, que son grafos equipados con una información adicional, llamada hiperlínea, las cuales al ser reemplazadas en forma iterativa desencadenan procesos generativos. El control de este proceso, así como, cuando y donde se realiza depende fundamentalmente del rótulo y del número de hiperlíneas que el grafo decorado posee. Como aplicación se desarrolla un algoritmo de inferencia gramatical, el cual posee como entrada un conjunto finito de grafos y como salida este tipo particular de gramáticas de grafos. El algoritmo de inferencia está basado esencialmente en iterar las aplicaciones de una operación, la cual descompone de una manera particular las reglas de la gramática. Además es estimado el número de gramáticas inferidas dependiente del tamaño del grafo. Dado que el Algoritmo de Inferencia produce en general un número exponencial de Gramáticas es que para reducir el número de gramáticas inferidas se considerarán medidas semánticas para limitarlas. Una conocida restricción de este tipo son los ejemplos negativos, de los cuales se exige que ellos no puedan ser generados.

Temas Específicos: Gramáticas de Grafos, Inferencia Gramatical.
(*)e-mail: ejeltsch@elqui.cic.userena.cl

1 INTRODUCCION

En muchas Areas de las Ciencias de la Computación es frecuente representar la información por grafos, que son objetos matemáticos muy apropiados para describir conocimientos, información, o más exactamente representar modelos u objetos, tales como diagrama de flujos, sistema de estado de un desarrollo celular etc. Ahora cualquier cambio local que pueda afectar a un 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 grafos que intervienen en este proceso conforman las llamadas Gramáticas de Grafos, las cuales proporcionan una alternativa para representar la información de una forma más general, en vez de utilizar textos, facilitando de esta manera el acceso a un amplio espectro de aplicaciones en diversas Areas, como por ejemplo: Reconocimiento de Formas, Sistema Base de Datos, Inferencia Gramatical, Semántica de Lenguajes, así como en la generación ó la simulación del crecimiento de células o plantas en la Biología. Vea [CER 79], [ENR 83], [ENRR 87] y [EKR 91]. Por otra parte, los procesos de inferencia son un tema de mucha importancia en la informática de hoy en día, pues tienen una gran relevancia en la generación y reconocimiento de modelos. Como meta fundamental este problema se refiere a encontrar una descripción sintáctica, por ejemplo Gramáticas, Autómatas o algún sistema, que permita la generación o el reconocimiento automático de los modelos, ya sean caracteres, árboles, grafos etc. Vea Fu [Fu 82]. La situación básica del problema de inferencia gramatical se refiere a: Dado un conjunto finito S_+ de modelos, buscar una Gramática G desde una clase predefinida $\mathcal{G}$ de Gramáticas, de manera tal que el lenguaje generado $L(G)$ contenga los modelos, e. d. $S_+ \subseteq L(G)$. En [Lu 93] se muestra una gran variedad de algoritmos de inferencia y aplicaciones que se han desarrollado, así como en la dirección electrónica http://bavi.unice.fr/Biblio/Ai/gramatical.inference.html
Dado que los procesos de inferencia forman estructuras desde modelos elementales, los cuales pueden ser considerados como un conjunto de objetos con atributos y porque sobre estos modelos pueden definirse relaciones y acciones, las cuales pueden ser representadas por grafos, en donde los nodos son los objetos y atributos y las líneas, las relaciones y acciones, es que podemos considerar un proceso de inferencia como un sistema de inferencia de Gramáticas de grafos. El algoritmo de inferencia gramatical que aquí se presenta genera un cierto tipo de Gramáticas de grafos, llamadas Gramáticas de reemplazo de hiperlíneas, a partir de un conjunto de modelos que están representados por grafos no-dirigidos y no-rotulados. Las Gramáticas de reemplazo de hiperlíneas tienen una concepcion similar a las Gramáticas definidas por Chomsky –Gramáticas regulares, contexto libre, contexto sensitivo, lema del bombeo–, de manera que cuentan con una sólida base teórica para llevar a cabo su investigación. Vea Habel [Ha 92]. Esencialmente, el proceso de inferencia está basada en la descomposición del lado derecho de las producciones de la Gramática representadas por grafos. Vea Jeltsch [Je 95]. Este trabajo esta organizado de manera que se entregan en los preliminares las herramientas básicas del algoritmo de inferencia. En la sección 3 se muestran las operaciones que intervienen en el algoritmo. En la sección 4, se realiza una estimación del número de gramáticas inferidas, según el algoritmo de inferecia. En la sección 5, se definen las condiciones y propiedades del concepto de *Compatibilidad*, vital para llevar a cabo las restricciones del flujo de gramáticas inferidas, concluyendo en la sección 6 con una discusión, proyecciones y aplicaciones de la investigación realizada.

2 PRELIMINARES

Asumimos que todos los modelos a considerar son grafos no-rotulados, no-dirigidos, sin lazos ni líneas múltiples. Para generar un conjunto de grafos, estos son equipados con una información adicional llamada hiperlínea. Una hiperlínea es un item atómico con un rótulo y un número fijo de tentáculos ordenados; ella puede ser adherida a cualquier tipo de estructura que tenga un conjunto de nodos, en el sentido que cada tentáculo es adherido a un nodo. El rol que juega la hiperlínea es reservar un lugar dentro del grafo, en donde la hiperlínea pueda ser reemplazada eventualmente por otra estructura. En la etapa de reemplazo la hiperlínea es borrada y en el lugar reservado es adherida otra estructura, conectada con la estructura original. Para este propósito, se supone que la

estructura de reemplazo tiene una sucesión de nodos externos, los cuales son fusionados a los nodos de los tentáculos ordenados de la hiperlínea, llevando a cabo una identificación de los nodos que intervienen en el proceso de generación. El reemplazo de una hiperlínea en una estructura puede ser realizada en forma iterativa si la estructura adherida está decorada con una hiperlínea compatible con los nodos que intervienen.

Grafos: Un *grafo* (no-dirigido, no-rotulado) es un par $G = (V, E)$, donde V es un conjunto de nodos(o vertices) y E es un conjunto de 2-elementos subconjuntos de V, llamado línea. El conjunto de todos los grafos es denotado por $\mathcal{G}_0$.

Ejemplo 1. El *grafo completo bipartido* $K_{m.n}$ para algún $m, n \in I\!N$ consiste of $m + n$ nodos, tal que cada uno de los m nodos es adyacente a cada uno de los otros n nodos. La siguiente figura muestra los $K_{3,i}$ para $i = 1, 2, 3, 4$.

Grafos Decorados: Sea N un conjunto de *rotulos no-terminales.*
Un *grafo decorado*(sobre N) es un sistema $H = (V, E, Y, att, lab, ext)$, donde $(V, E) \in \mathcal{G}_0$, Y conjunto de *hiperlíneas*, $lab : Y \longrightarrow N$ es una función, llamada *rotulación*, $att : Y \longrightarrow V^*$ es una función llamada *adherencia*, y $ext \in V^*$ es una sucesión de nodos, llamados *nodos externos.*

Observaciones:

1. Las componentes V, E, Y, att, lab y ext de H son también denotadas por V_H, E_H, Y_H, att_H, lab_H, ext_H respectivamente.

2. El largo de ext_H es llamado el *tipo* de H y denotado por $tipo(H)$. Si $tipo(H) = n$, entonces H es llamado $n\text{-}grafo$.

3. El largo de $att_H(y)$ para $y \in Y_H$ es llamado *tipo* de y y denotado por $type(y)$. Si $tipo(y) = n$, y es llamada $n\text{-}hiperlínea$.

4. La clase de todos los grafos decorados sobre N es denotado por $\mathcal{G}(N)$.

5. Una hiperlínea es un item rotulado con un número fijo de tentáculos ordenados. En un grafo decorado H se adhiere esta hiperlínea, de tal manera que cada tentáculo se adhiere a un nodo. Sea $y \in Y_H$, y $att_H(y)$ la sucesión de nodos $v_1 \ldots v_n$, entonces v_i con $i = 1 \ldots n$ es el nodo, que se adhiere al i-tentáculo de y. Esto se denota por $att_H(y)_i{=}v_i$.

6. Otro caso particular es un grafo decorado con una hiperlínea simple, en donde los nodos externos coinciden con los nodos que se adhieren, es decir $H = (V, \emptyset, \{y\}, lab, att, ext)$ con $att(y) = ext$. Si $V = \{1, \ldots, n\}$, $lab(y) = A$ y $att(y) = ext{=}1 \ldots n$, H se llama $n-$ *polipo* y se denota por $(A, n)^{\bullet}$.

Ejemplo 2. La siguiente figura muestra un grafo decorado G que consiste de un grafo junto con cuatro hiperlíneas.

Las hiperlíneas son rotuladas A, B, C y D, el orden de los tentáculos está indicado por sus números. En forma similar se proporciona el número de nodos externos indicando el orden de ext. El tipo de las hiperlíneas rotuladas por A, B y C es 2, mientras que la hiperlínea rotulada por D es de tipo 3.

Reemplazo de hiperlínea: Sea $H \in \mathcal{G}(N)$ y $B = \{y_1, \ldots, y_n\} \subseteq Y_H$ conjunto de hiperlíneas ha ser reemplazadas. Sea $repl : B \longrightarrow \mathcal{G}(N)$ una función con $type(y_i) = type(repl(y_i))$ para todo $i = 1, \ldots, n$. Entonces el *reemplazo* de B en H a través de $repl$ produce el grafo decorado:

$$REPLACE(H, repl) = ((H - B) + \sum_{i=1}^{n} repl(y_i)/(att = ext))$$

con $att = att(y_1) \ldots att(y_n)$ y $ext = ext_{repl(y_i)} \ldots ext_{repl(y_n)}$.

Ejemplo 3. Considerando ahora el grafo decorado G dado en Ejemplo 2. Si reemplazamos la hiperlínea rotulada con D por G mismo, obtenemos

Regla: Una *regla de reemplazo de hiperlínea* (sobre N) es un par $p = (A, R)$ con $A \in N$ y $R \in \mathcal{G}(N)$. A es llamado el lado izquierdo de p y denotado por $lhs(p)$. R es llamado el lado derecho de p y denotado por $rhs(p)$. Sea $H \in \mathcal{G}(N)$, $B \subseteq Y_H$ y P un conjunto finito de reglas. Una función $b : B \longrightarrow P$ es llamada una *base* en H, si $lab_H(y) = lhs(base(y))$ y $type(y) = type(rhs(b(y)))$ para todo $y \in B$.

Derivación: Sean $H, H' \in \mathcal{G}(N)$, y $b : B \longrightarrow P$ una base en H. Entonces H *deriva directamente* H' *a través de* b si $H' = REPLACE(H, repl)$ con $repl : B \longrightarrow \mathcal{G}(N)$, $repl(y) = rhs(b(y))$ para todo $y \in B$. Para tales efectos denotamos $H \underset{b}{\Longrightarrow} H'$ o $H \underset{P}{\Longrightarrow} H'$. Si en particular $B = \{y\}$, y $b(y) = p$, entonces podemos escribir $H \underset{y,p}{\Longrightarrow} H'$ o $H \underset{p}{\Longrightarrow} H'$.
Una sucesión de derivaciones de la forma $H = H_0 \underset{P}{\Longrightarrow} H_1 \underset{P}{\Longrightarrow} \ldots \underset{P}{\Longrightarrow} H_k = H'$ es llamada una derivación de largo k, se denota por $H \overset{*}{\underset{P}{\Longrightarrow}} H'$, si k no juega ningún rol.

Gramáticas de Grafos: Una *Gramática de reemplazo de hiperlínea* es un sistema $GRH = (N, P, Z)$, donde N es un conjunto de rotulos no-terminales, P un conjunto de reglas de reemplazo de hiperlíneas (sobre N) y $Z \in \mathcal{G}_O(N)$, llamado *axioma.*

Lenguajes de Grafos Dada una Gramática de reemplazo de hiperlínea $GRH = (N, P, Z)$, el *lenguaje de grafos generados* $L(GRH)$ consiste de todos los grafos derivables desde Z.

$$L(GRH) = \{G \in \mathcal{G}_0 \mid Z \overset{*}{\underset{P}{\Longrightarrow}} G\}.$$

Ejemplo 4. El conjunto $\{K_{3,n} \mid n \geq 1\}$ puede ser generado por la Gramática de reemplazo de hiperlínea $GRH = (\{D, K\}, \{p_1, p_2, p_3\}, Z)$ donde el axioma Z y las tres reglas están dadas por la siguiente figura:

La siguiente figura ilustra la situación donde el rótulo K es omitido y se aplica $i-$ veces la regla p_1 para luego ir generando los elementos $K_{3.i}$ al aplicar p_2 y p_3, para todo $i \in I\!N$.

3 UN ALGORITMO INFERENCIA GRAMATICAL

En esta sección desarrollamos un procedimiento no-determinístico que infiere Gramáticas de reemplazo de hiperlíneas a partir de un conjunto finito de grafos, de manera que el lenguaje de la Gramática inferida contiene todos los grafos. El Algoritmo está basado en dos operaciones. El proceso de inferencia no-determinístico parte con la operación $INIT$ para luego repetir un número arbitrario de etapas $DECOMPOSE$.

(1) La operación $INIT$ transforma un conjunto finito de grafos en Gramáticas de reemplazo de hiperlínea considerando los modelos como lado derecho de una producción, cuyo lado izquierdo consta solamente de un literal. La Gramática $INIT$ genera obviamente los modelos de entrada.

(2) La operación $DECOMPOSE$ transforma Gramáticas de reemplazo en Gramáticas de reemplazo usando una descomposición del lado derecho de las producciones en forma disjunta.

Construcción:
Sea S_+ un conjunto finito de grafos. Sea N conjunto de rótulos con $S \in N$. Entonces la *Gramática inicial* de S_+ es:

$$INIT(S_+) = (N, \{(S, M) \mid M \in S_+\}, (S, 0)^\bullet).$$

Observación:

$$L(INIT(S_+)) = S_+.$$

Ejemplo 6.
Si los cuatro grafos mostrados en Ejemplo 1 son escogidos como modelos, obtenemos la siguiente Gramática inicial:

$$INIT(\{K_{3.1}, K_{3.2}, K_{3.3}, K_{3.4}\}) = (N, \{(S, K_{3.i}) \mid i = 1, 2, 3, 4\}, (S, 0)^\bullet).$$

En la siguiente etapa definimos la operación $DECOMPOSE$, esencial en nuestro algoritmo, la cual descompone el lado derecho de la producción.

Construcción:
Sea $GRH = (N, P, Z)$ una Gramática de reemplazo de hiperlínea y $(A, R) \in P$, $R_0 \in \mathcal{G}(N)$, $B \subseteq Y_{R_0}$, y sea $repl : B \longrightarrow \mathcal{G}(N)$ una función con $type(b) = type(repl(b))$ para todo $b \in B$ tal que $R = REPLACE(R_0, repl)$. Entonces obtenemos la Gramática:

$$DECOMPOSE(GRH) = (N, \overline{P}, Z)$$

con $\overline{P} = (P - \{(A, R)\}) \bigcup \{(A, R_0)\} \bigcup \{(lab_{R_0}(b), repl(b)) \mid b \in B\}$.

Observación:

$$L(HRG) \subseteq L(DECOMPOSE(HRG)).$$

Ejemplo 7.
Consideremos las producciones $r_i = (S, K_{3.i})$ para i=1,2,3,4 en Ejemplo 6 La producción r_2 puede ser descompuesta en las producciones $r_{21} = (S, H_1)$ y $r_{22} = (D, H_2)$, donde H_1 y H_2 son dada en la siguiente figura.

La regla r_{22} puede ser descompuesta otra vez en las reglas $r_{221} = (D, H_3)$ y $r_{222} = (D, H_4)$, donde H_3 y H_4 están dada en la siguiente figura.

Por lo tanto las reglas de la Gramática inferida $\overline{HRG}$ son r_1, r_{21}, r_{221}, r_{222}, r_3, y r_4. En esta Gramática tenemos por ejemplo la siguiente derivación:

El algoritmo de inferencia basado en $INIT$ y $DECOMPOSE$ trabaja como sigue: Para un conjunto finito S_+ de grafos como entrada, $INIT(S_+)$ escoje un conjunto de rótulos N en donde el grafo de partida $S \in N$, y luego aplica una sucesión arbitraria de operaciones $DECOMPOSE$, escogiendo una descomposición apropiada de producciones. Todas las Gramáticas de reemplazo de hiperlíneas que se pueden obtener de esta manera son denotadas por $INFERENCIA(S_+)$.

Inferencia Gramatical:
Sea S_+ un conjunto finito de grafos y $GRH \in INFERENCIA(S_+)$ construída a partir de $INIT(S_+)$, seguida por una secuencia de operaciones $DECOMPOSE$. Entonces:

$$S_+ \subseteq L(GRH).$$

Ejemplo 8. Resumiendo, obtenemos la siguiente situación: Inicializamos el proceso de inferencia por los grafos $K_{3,1}$, $K_{3,2}$, $K_{3,3}$ y $K_{3.4}$(vea Ejemplo 1 y 6), luego se realizan dos operaciones $DECOMPOSE$ (Ejemplo 7), produciendo así una Gramática, la cual genera el conjunto de los grafos bipartitos $K_{3,n}$ para $n \geq 1$ y es de similar simplicidad como la dada en Ejemplo 4.

4 Estimación del número de gramáticas

En esta sección será estimado el número de gramáticas inferidas dependiente del tamaño del grafo. Dado que en general el lado derecho de las producciones pueden ser descompuesto de varias maneras en subgrafos disjuntos, es que el algoritmo de inferencia es altamente no-determinístico, razón por la cual el número de gramáticas inferidas puede crecer en forma exponencial según el tamaño del grafo. Por tal motivo a través de este análisis, queda expresada la necesidad de acotar las posibilidades de descomposición. Una de las medidas sintácticas posibles de considerar son por ejemplo, evitar descomposiciones triviales ó determinadas descomposiciones que en nada constribuyen, ó considerar hiperlíneas de algún cierto tipo. Por lo pronto, consideremos los *grafos estrella* $K_{1.n}$, y su respectiva descomposición en dos componentes. El número de descomposiciones se pueden dejar expresadas dependiendo del tipo de hiperlíneas utilizados en la descomposición, la cual será introducida como tipo de descomposición.

4.1 Definición(Reemplazo de Hiperlíneas)

Sea $H \in \mathcal{H}_N$ y $B = \{y_1, \ldots, y_n\} \subseteq Y_H$, conjunto de hiperlíneas ha ser reemplazadas. Sea $repl : B \longrightarrow \mathcal{H}_N$ una función con $type(y_i) = type(repl(y_i))$ para todo $i = 1, \ldots, n$. Entonces el *reemplazo de B en H* a través de $repl$ genera el grafo decorado:

$$X = REPLACE(H, repl) = ((H - B) + \sum_{i=1}^{n} repl(y_i))/(att = ext)$$

con $att = att(y_1).....att(y_n)$ y $ext = ext_{repl(y_1)}.....ext_{repl(y_n)}$.

Observación

(1) $REPLACE(H, repl)$ denota el reemplazo conforme a $repl$ desde el grafo H. Si $B = \{b\}$ y $repl(b) = H'$, escribimos $H \otimes_b H'$ y llamamos al resultado *composición de H y H' en b*. La construcción de $H \otimes_b H'$ es lograda por el reemplazo de la hiperlínea b desde H por H'. La hiperlínea $y \in Y_H$ es llamada *tipo de descomposición*

4.2 Lema

Los grafos estrella $K_{1.n}$ para $n \geq 0$ ocupan (al menos) $2 + k(n - k + 2)$ descomposiciones del Tipo k para $0 \leq k \leq n$, en donde respectivamente una de las componentes de al menos dos descomposiciones aparecen los grafos estrella $K_{1.n-1}$ como subgrafo.

Demostración:

$K_{1.n}$ tiene dos descomposiciones triviales de Tipo 0:
$(K_{1.n} + (S,0)^\bullet) \otimes \emptyset$ y $(S,0)^\bullet \otimes K_{1.n}$. (Designemos por $\emptyset$ el grafo vacío)
las descomposiciones de Tipo 1 tienen la siguiente forma:

para $i = 0, \ldots, n$.

Las descomposiciones de Tipo 2 tienen la siguiente forma:

para $i = 0, \ldots, n-1$.

El caso general para $0 \leq k \leq n$ está representado como sigue:

para $i = 0, \ldots, n - (k-1)$.

$K_{1.n-1}$ ocupa también 2 descomposiciones triviales y para $i = 0, \ldots, n - (k-1)$ cada k descomposiciones posibles del Tipo k. En general se tienen (al menos) $2 + k(n - (k-1) + 1) = 2 + k(n - k + 2)$ descomposiciones del Tipo k. En especial existen (al menos) dos descomposiciones, junto a las cuales aparecen los grafos estrellas $K_{1.n-1}$ como subgrafo en una componente.

4.3 Teorema

Si $K_{1.n} \in S_+$, entonces el conjunto $INFERENCE(S_+)$ contiene al menos 2^n gramáticas.

Demostración:

Según el lema anterior se originan al menos 2 gramáticas por la aplicación del paso $DECOMPOSE$, cuyo lado derecho contiene a $K_{1.n-1}$ como subgrafo. Por inducción se dejan inferir 2^{n-1} gramáticas, de tal manera que se originan en total 2^n gramáticas.

5 RESTRICCIONES AL ALGORITMO DE INFERENCIA

El Algoritmo de Inferencia dado en los preliminares produce en general un número exponencial de Gramáticas, pues existen grafos que permiten varias descomposiciones. Entonces para reducir el número de gramáticas inferidas se considerarán medidas semánticas para limitarlas. Una conocida restricción de este tipo son los ejemplos negativos, de los cuales se exige que ellos no puedan ser generados. Si los grafos actuan como modelos de entrada es apropiado considerar propiedades de la teoría de grafos para formular otras exigencias. Por ejemplo se podría exigir que todos los grafos generados sean planar y conexos pero no 3-coloreado, o que todo grafo generado que sea conexo contenga un camino Hamiltoniano. Estas exigencias pueden ser formuladas, sino para todo grafo generado, al menos para un número finito de ellos. Ahora, para que este tipo de exigencias sean asimiladas por los lenguajes generados por las gramáticas de reemplazo de hiperlíneas es necesario pensar en las siguientes situaciones :

1. Todo $M \in L(G)$ deben ser de tal manera.
2. Ningún $M \in L(G)$ debe ser de tal manera.
3. Deben existir en $L(G)$ grafos que satisfagan tal, o cual propiedad.
4. Casi todo $M \in L(G)$ deben ser de tal y tal manera.
5. A lo más, finitos $M \in L(G)$ pueden ser de tal o cual manera.

En este contexto la Teoría de los Grafos y las Gramáticas de Grafos permiten a través del concepto de Compatibilidad dar una solución a este problema. La reducción del conjunto de gramáticas inferidas a través de exigencias significa no admitir gramáticas que generen grafos con propiedades no deseadas. Es decir, en otras palabras, si se le antepone una exigencia al lenguaje generado, como por ejemplo, "Todo grafo generado debe ser planar y conexo, pero no 3-coloreado", entonces el algoritmo de inferencia debería solamente suministrar gramáticas, cuyo lenguaje satisfaga los requerimientos exigidos.

5.1 Definición (Compatibilidad)

1. Sea $\mathcal{C}$ una clase de gramáticas de hiperlíneas, I un conjunto finito de indices, $PROP$ un predicado decidible, que está definido sobre el par (H, i) con $H \in \mathcal{G}$ y $i \in I$, y $PROP'$ un predicado que está definido sobre la tripleta $(R, assign, i)$, donde $R \in \mathcal{G}$, $assign : Y_R \to I$ función, e $i \in I$.

   Luego $PROP$ es llamado $\mathcal{C}, PROP'$-*Compatible*, si para toda $GRH = (N, P, Z) \in \mathcal{C}$ y toda función $(A, n)^\bullet \Longrightarrow R \stackrel{*}{\Longrightarrow} H$ con $A \in N$, $n \in I\!N$ y $H \in \mathcal{G}$, y todo $i \in I$, se tiene lo siguiente:

   $PROP(H, i)$ es válido, si y solo si existe una función $assign : Y_R \to I$, donde $PROP'(R, assign, i)$ y $PROP(H(y), assign(y))$ son válidos, para todo $y \in Y_R$.

2. Un predicado $PROP_0$, definido sobre $\mathcal{G}$ es llamado $\mathcal{C}$-*Compatible*, si existen los predicados $PROP$, $PROP'$ y un $i_0 \in I$, tal que $PROP_0 = PROP(-, i_0)$ y $PROP$ es $\mathcal{C}$, $PROP'$-*Compatible*.

En la práctica resultan ser *Compatible* diversas propiedades de los grafos a través del proceso de derivación de las gramáticas de hiperlíneas. Por ejemplo: Conexo, Planar, Existencia de caminos Eulerianos y Hamiltonianos, Coloreados con k colores para un $k \in I\!N$, y otras. Además las propiedades compatibles son cerradas respecto a las operaciones Booleanas, de manera que muchas otras propiedades son factibles de derivar a partir de la Negación, Conjunción y Disjunción.

5.2 Teorema

Sean $PROP_1$, $PROP_2$ predicados $\mathcal{C}$-compatibles para una clase $\mathcal{C} \subseteq \mathcal{GRH}$.
Entonces son $(PROP_1 \wedge PROP_2)$, $(PROP_1 \vee PROP_2)$ y $(\neg PROP_1)$ $\mathcal{C}$-compatible.

Observación

Si $\mathcal{C} \subseteq \mathcal{GRH}$ y $PROP$ un predicado $\mathcal{C}$-compatible. Entonces $PROP$ es también $\mathcal{C}'$-compatible para toda clase $\mathcal{C}' \subseteq \mathcal{C}$.

Otros resultados sobre Compatibilidad pueden ser encontrados en los trabajos de Habel [Ha 92].

Notación

Sea $\mathcal{L}$ una clase de lenguajes y $PROP$ un predicado $\mathcal{C}$-compatible.

1. Con $L(PROP)$ es descrito el conjunto de todos los grafos, los cuales poseen la propiedad $PROP$.
2. $\mathcal{L} \bigcap PROP = \{L \bigcap L(PROP) \mid L \in \mathcal{L}\}$.
3. $\mathcal{L} - PROP = \{L - L(PROP) \mid L \in \mathcal{L}\}$.
4. Sea $\mathcal{C}' \subseteq \mathcal{GRH}$, $\mathcal{L}(\mathcal{C}') = \{L(GRH) \mid GRH \in \mathcal{C}'\}$.

En lo que continúa se muestran algunas propiedades decidibles válidas para las gramáticas de hiperlíneas.

5.3 Problema de Existencia

"Existe un algoritmo, que para GRH arbitrario como entrada pueda decidir $L(GRH) = \emptyset$, ó no ?".

5.4 Problema de Pertenencia

" Existe un algoritmo, que teniendo como entrada GRH y grafos arbitrarios $H \in \mathcal{G}$ pueda decidir si $H \in L(GRH)$, ó no ? ".

5.5 Problema de Finitud

"Existe un algoritmo, que para un GRH arbitrario como entrada pueda decidir si el lenguaje $L(GRH)$ contiene un número finito o infinito (salvo isomorfismo)?".

Habel [Ha 92] ha demostrado que para cada predicado $PROP$ $\mathcal{C}$-*Compatible* y $L \in \mathcal{L}(\mathcal{C})$, se tiene que también $L \cap L(PROP) \in \mathcal{L}(\mathcal{HEG})$ y $L \cap L(\neg PROP) \in \mathcal{L}(\mathcal{GRH})$. Con la solución a los problemas antes mencionados, se tiene la validez del siguiente Teorema.

5.6 Teorema

Sean $PROP$ y $PROP'$ predicados $\mathcal{C}$-Compatible referida a una clase $\mathcal{C}$ de gramáticas de hiperlíneas. Entonces para todo $GRH \in \mathcal{C}$ las siguientes preguntas son decidibles.

1. Existen $H \in L(GRH)$ con la propiedad $PROP$?
2. Posee todo $H \in L(GRH)$ la propiedad $PROP$?
3. Posee todo $H \in L(GRH)$, salvo un número finito de exepciones, la propiedad $PROP$?
4. Posee solamente un número finito de $H \in L(GRH)$ la propiedad $PROP$?
5. Posee todo $H \in L(GRH)$ con la propiedad $PROP$ la propiedad $PROP'$?
6. En el caso que todo $H \in L(GRH)$ posea la propiedad $PROP$, entonces todos ellos tienen la propiedad $PROP'$?

5.7 Definición (Requerimientos y Objeciones)

Sea $INFERENCE$ el conjunto de todas las gramáticas inferidas según el algoritmo de inferencia. Sea $COMPATIBLE$ el conjunto de todas las propiedades $INFERENCE$-Compatible.

1. Una lista REQ de preguntas de la forma:

   - Existe $H \in L(HEG)$ con la propiedad $PROP$?
   - Posee todo $H \in L(HEG)$ la propiedad $PROP$?
   - Posee todo $H \in L(HEG)$ la propiedad $PROP$?, salvo un número finito de exepciones.
   - Posee solamente un número finito $H \in L(HEG)$ la propiedad $PROP$?
   - Posee todo $H \in L(HEG)$ con la propiedad $PROP$ también la propiedad $PROP'$?
   - Caso que todo $H \in L(HEG)$ posea la propiedad $PROP$, entonces todos ellos poseen la propiedad $PROP'$?

   para $PROP$, $PROP' \in COMPATIBLE$ será llamada *Lista de Requerimientos* .

2. Una lista similar OBJ, es llamada *Lista de Objeciones.*
3. Se dice que $GRH \in \mathcal{GRH}$ *considera los requerimientos* REQ, si cada pregunta desde REQ es respondida afirmativamente con SI para GRH.
4. Se dice que $GRH \in \mathcal{GRH}$ *considera las objeciones* OBJ, si cada pregunta desde OBJ es respondida negativamente con NO para GRH.
5. El conjunto de todas las gramáticas de hiperlíneas que consideran, tanto los REQ como también las OBJ son descritas por $\mathcal{GRH}_{REQ,OBJ}$.

Por lo anterior, podemos ahora exigirle al algoritmo de inferencia junto al conjunto de modelos de entrada, ciertos requerimientos u objeciones, con el propósito de que las gramáticas inferidas generen lenguajes, cuyos elementos satisfagan los requerimientos previstos.

Para atacar este probleme se necesita en principio condicionar las operaciones que intervienen en una base de inferencia. Esta idea es justamente la que se expresa a continuación.

5.8 Construcción (Inferencia Condicionada)

Sea $\mathcal{B} = (\mathcal{T}, \mathcal{I})$ una Base de Inferencia, $I \in \mathcal{I}$ y $T \in \mathcal{T}$. Sea REQ y OBJ Listas de Requerimientos y Objeciones respectivamente.

1. Se denota por $I_{REQ,OBJ}$ la *Inicialización Condicionada*, la cual está definida por

$$I_{REQ,OBJ}(S_+) = I(S_+) \cap \mathcal{GRH}_{REQ,OBJ},$$

   para todo $S_+ \in \mathcal{P}(\mathcal{G})$.

2. Con $T_{REQ,OBJ}$ es denotada la *Transformación Condicionada*, la cual está definida por

$$T_{REQ,OBJ}(GRH) = T(GRH) \cap \mathcal{GRH}_{REQ,OBJ},$$

   para todo $GRH \in \mathcal{GRH}$.

3. Sea $\mathcal{I}_{REQ,OBJ}$ el conjunto de todas las Inicializaciones Condicionadas y $\mathcal{T}_{REQ,OBJ}$ conjunto de todas las Transformaciones Condicionadas. Luego denotamos por $\mathcal{B}_{REQ,OBJ} = (\mathcal{T}_{REQ,OBJ}, \mathcal{I}_{REQ,OBJ})$ la *Base de Inferencia Condicionada.*

5.9 Teorema

Sea $\mathcal{B}$ Base de Inferencia. Sean REQ y OBJ las Lista de Requerimientos y Objeciones. Entonces $\mathcal{B}_{REQ,OBJ}$ es una restricción de $\mathcal{B}$.

Demostración:

$INFERENCE_{\mathcal{B}_{REQ,OBJ}}(S_+) = \mathcal{T}^*_{REQ,OBJ}(\mathcal{I}_{REQ,OBJ}(S_+)) \subseteq \mathcal{T}^*(\mathcal{I}(S_+)) = INFERENCE_{\mathcal{B}}(S_+)$, pues según definición vale para todo $S_+ \in \mathcal{P}(\mathcal{G})$, $I \in \mathcal{I}$, $GRH \in \mathcal{GRH}$ y $T \in \mathcal{T}$:
$I_{REQ,OBJ}(S_+) \subseteq I(S_+)$ y $T_{REQ,OBJ}(GRH) \subseteq T(GRH)$.

5.10 Ejemplo

Dadas las condiciones de compatibilidad es posible exigir que, todo grafo generado $H \in L(GRH)$ sea planar y conexo, pero no 3-coloreado. En efecto, sean $PLANAR$, $CONEXO$ y $COLORES_3$ los predicados definidos desde $\mathcal{G}$, los cuales son: $PLANAR(H)$ es válido ssi H es planar, $CONEXO(H)$ es válido ssi H es conexo, $COLORES_3(H)$ es válido ssi H es 3-coloreado.
Entonces el predicado $EXIGEN(H) = (PLANAR \wedge CONEXO \wedge \neg COLORES_3)(H)$ predicados desde $\mathcal{G}$. es compatible, pues $PLANAR$, $CONEXO$ y $COLORES_3$ son $\mathcal{HEG}$-compatible (Vea Habel [Ha 92]) y la compatibilidad es cerrada respecto a las Operaciones Booleanas, entonces el predicado $EXIGEN$ es también compatible, de manera que la pregunta: "Tienen todos los grafos generados la propiedad $EXIGEN$, es decir son ellos planar y conexos pero no 3-coloreados" puede ser aplicada como exigencia adicional u objeción al algoritmo de inferencia. Un caso interesante de tratar son los ejemplos negativos, los cuales son similares a los modelos empleados, pero con la diferencia que ellos no aparecen en el lenguaje generado.

6 DISCUSION

Este trabajo es una continuación del algoritmo de inferencia presentado en [Je 95]. En este nuevo contexto pueden ser aplicadas algunas propiedades de los grafos como exigencias sobre las gramáticas de reemplazo, las cuales se traducen en un concepto de compatibilidad. Esta idea sirve para restringir el número de gramáticas aceptadas, pero no sobre el número general de gramáticas construídas. No obstante resulta de singular interes el poder constatar que ciertas propiedades *Compatibles*, tales como grafo planar, conexo, k-coloreado, funcionan eficientemente para el proceso de derivación de nuestro algoritmo de inferencia. Por otra parte es interesante visualizar propiedades de las gramáticas de grafos, las cuales pueden adaptarse en difentes campos de aplicación, como son las Redes Neuronales [HM 96].
A pesar de todos estos resultados, existen todavía una serie de aspectos a considerar que deberían ser considerados en próximas publicaciones, algunos de ellos son:

1. Elaborar nuevas Bases de Inferencia, es decir construirse nuevas inicializaciones y transformaciones para el proceso de inferencia.

2. El proceso de inferencia es altamente no-determinístico, de manera que tomar medidas sintácticas para tal efecto, serían de gran valor. Cuales serán las más apropiadas ?.

3. La implementación del algoritmo de inferencia es con seguridad un duro desafío.

4. Gramáticas Collage (vea Habel, Kreowski [HK 91]) y Redes Neuronales (vea Heider, Moraga [HM 96]) pueden ser una interesante aplicación de nuestro algoritmo de inferencia.

References

[CER 79] V. Claus, H. Ehrig, G. Rozenberg (eds.): Graph-Grammars and Their Application to Computer Science and Biology, Lect. Not. Comp. Sci. 73, 1979.

[EKR 91] H. Ehrig, H.-J. Kreowski, G. Rozenberg (eds.): Graph-Grammars and Their Application to Computer Science, Lect. Not. Comp. Sci. 532, 1992.

[ENR 83] H. Ehrig, M. Nagl , G. Rozenberg (eds.): Graph-Grammars and Their Application to Computer Science, Lect. Not. Comp. Sci. 153, 1983.

[ENRR 87] H. Ehrig, M. Nagl, G. Rozenberg, A. Rosenfeld (eds.): Graph-Grammars and Their Application to Computer Science, Lect. Not. Comp. Sci. 291, 1987.

[Fu 82] K.S. Fu: Syntactic Pattern Recognition and Applications, Prentice-Hall, Englewood-Cliffs, N.J., 1982.

[Ha 92] A. Habel: Hyperedge Replacement: Grammars and Languages, Lect. Not. Comp. Sci. 643, 1992.

[HK 91] A. Habel, H.-J. Kreowski: Collage Grammars, Lect. Not. Comp. Sci. 532, 411-429, 1991.

[HM 96] R. Heider, C. Moraga: Evolutionärer Entwurf neuronaler Netze unter Verwendung von Graph-Grammatiken. Forschungsbericht 594, Universidad Dortmund, 1996.

[Je 95] E. Jeltsch: Un algoritmo para inferir gramáticas de grafos. Actas III Encuentro Chileno de Computación, 1995.

[Lu 93] S. M. Lucas: New Directions in Grammatical Inference: Colloquium.IEE, Savoy Place, London WC2R OBL, UK, 1993.