Benemérita Universidad Autónoma de Puebla
Posgrado en Ingeniería del Lenguaje y del Conocimiento
Facultad de Ciencias de la Computación

Algoritmo de Membresía para Gramáticas de Reemplazo de Hiperaristas

Tesis
Que para obtener el grado de:
Dra. en Ingeniería del Lenguaje y del Conocimiento

Presenta

M. C. Yolanda Moyao Martínez

Directora:
Dra. Darnes Vilariño Ayala

Director externo:
Dr. Carlos Guillén Galván

Co-Director:
Dr. José de Jesús Lavalle Martínez

Puebla, agosto de 2021

Dedicado a
Mis papás, esposo y hermanos

Agradecimientos

Cuando quise rendirme solo
miré al inicio y ya no lo pude
ver más pero el final aunque
apenas se veía ahí seguía.

Yolanda

Agradezco

A mi papá Alejandro que aunque ya no está presente, siempre me alentó a superarme, a seguir adelante y no quedar en "el montón".

A mi querido esposo Juan Manuel quién fue un gran apoyo emocional en los momentos díficiles durante el tiempo que duró este trabajo de investigación, quién me apoyó y alentó para continuar, cuando parecía que me iba a rendir.

A mi mamá Gela y hermanos por estar siempre presentes, acompañándome y por el apoyo moral, que me brindaron a lo largo de esta etapa de mi vida.

A mi hermano el Dr. Alejandro Moyaho Martínez, quién siempre me alentó a continuar y llegar hasta el final y además colaboró en la revisión de esta tesis.

A todos mis compañeros y amigos y en especial a Andy y Bety por todo el apoyo, por las platicas, los momentos difíciles pero sobre todo por la paciencia durante esta etapa de estudio.

A mi directora de tesis la Dra. Darnes Vilariño Ayala, quién con su experiencia, conocimiento y motivación me orientó en la investigación, así como por su paciencia y apoyo incondicional para culminar el trabajo con éxito, pese al estado delicado de su salud y a las adversidades e inconvenientes que se presentaron durante este proceso.

A mis co-directores de tesis, los doctores José de Jesús Lavalle Martínez y Carlos Guillén Galván quienes siempre compartieron de su tiempo, aún sin importar que muchas veces parecía que no había resultados, a ellos que continuaron depositando su confianza en mí y que siempre estuvieron dispuestos a compartir su conocimiento.

Al Dr. David Pinto Avendaño, a quién admiro por ese apoyo incondicional y le agradezco que con su sabiduría y conocimiento me motivó para seguir adelante y desarrollarme como profesional.

A la Dra. Josefa Somodevilla por la paciencia y el apoyo brindado para concluir este trabajo de investigación con éxito, quién a lo largo de este trabajo siempre me asesoró y

apoyó en todo lo necesario.

A todos los doctores que conforman el comité tutorial por su paciencia y tiempo dedicado durante este trabajo de investigación.

A la BUAP por todo el apoyo brindado para realizar mis estudios de doctorado.

Al Programa de doctorado LKE por darme la oportunidad y por creer en mi trabajo.

Al CONACyT por el apoyo, a través del financiamiento otorgado para realizar este trabajo de investigación bajo el número de beca 484107.

Resumen

Este trabajo trata del problema de membresía en Gramáticas de Reemplazo de Hiperaristas (HRG). Dado un hipergrafo H con nodos e hiperaristas etiquetadas, dirigidas y enraizadas, el problema consiste en determinar si $H \in L(G)$, donde $G \in HRG$, es decir si H está en el lenguaje generado por G. Se conoce que el problema de membresía para HRG es, en general, intratable. Sin embargo, este problema se ha resuelto en tiempo polinomial para algún tipo restringido de HRG. El objetivo principal de esta investigación es desarrollar un algoritmo correcto con complejidad polinomial que resuelva el problema de membresía en HRG. Para lograr el objetivo fue necesario utilizar una definición alternativa de la matriz de adyacencias para hipergrafos, la cual es una generalización de la matriz de adyacencias para grafos. En este trabajo se obtuvo un algoritmo *Analizador*, cuya complejidad es del orden $O(l^5)$, donde l es el número de vértices del hipergrafo de entrada. Este algoritmo lleva a cabo el análisis directamente en la Matriz de Adyacencias del hipergrafo H. También, para el algoritmo propuesto se presenta la demostración de su corrección.

Índice general

Índice de figuras

Capítulo 1

Introducción

En este trabajo de investigación se propone un algoritmo *Analizador* para resolver el problema de membresía en Gramáticas de Reemplazo de Hiperaristas (HRG), por sus siglas en inglés. El problema de membresía consiste en reconocer el hipergrafo de entrada H a través de las reglas de producción.

Las gramáticas de hipergrafos se han desarrollado como una extensión del concepto formal de Gramáticas Libres de Contexto. En éstas, el concepto de reescritura de símbolos por cadenas se generaliza al reemplazo de hiperaristas por hipergrafos, a pesar de que en el caso de cadenas, el reemplazo de una de éstas por otra suele ser un proceso sencillo. Para el caso de hipergrafos, es necesario encontrar un hipergrafo para ser reemplazado por otro hipergrafo [1].

En ciencias de la computación un hipergrafo es una generalización de un grafo, cuyas hiperaristas pueden relacionar a cualquier cantidad de vértices, en lugar de sólo un máximo de dos como en el caso particular de grafos, también puede ser dirigido y con hiperaristas etiquetadas. La hiperarista es un elemento que relaciona a más de dos vértices, eventualmente ésta puede ser reemplazada con algún otro hipergrafo. De tal forma que la hiperarista se elimina y algún hipergrafo R se reemplaza en el hipergrafo original [2] [3].

Las HRG, son gramáticas de grafos libres de contexto presentadas por [4] y [5] como uno de los formalismos más exitosos para la descripción de lenguajes de grafos, debido a que sus propiedades teóricas del lenguaje se parecen a las que ya se tienen definidas en el ámbito de las gramáticas libres de contexto.

Las HRG, las cuales son un formalismo para la generación de lenguajes de hipergrafos, han propiciado el interés de este tipo de lenguaje debido a las aplicaciones en diferentes áreas, tales como procesamiento del lenguaje natural, en partícular en la comprensión y generación de lenguaje, en la traducción automática basada en semántica; así también en áreas como teoría de juegos, bases de datos e inteligencia artificial, entre otras [6],[7] [8], [9] y [10].

A pesar de que las propiedades teóricas del lenguaje generado por las HRG se asemejan a las de las gramáticas libres de contexto, las similitudes entre cadenas y grafos no se extienden

a uno de los problemas más importantes en el contexto de los lenguajes formales, nos referimos al problema de membresía en HRG [11].

Se conoce ampliamente que el problema de membresía para HRG es, en general, intratable; [12] y [13] sin embargo, este problema puede ser resuelto en tiempo polinomial para algunos tipos restringidos de HRG. Después de la revisión bibliográfica, se encontaron diferentes métodos que han sido propuestos para dar solución a dicho problema.

Por ejemplo, [1] presenta una versión extendida "Top-Down" para HRG del algoritmo CKY (Cocke-Younger-Kasami) de grámaticas libres de contexto para cadenas sobre grafos de grado acotado; éstos permiten ejecutar el algoritmo analizador en un tiempo polinomial respecto del tamaño del grafo de entrada.

En el contexto del procesamiento del lenguaje natural con representaciones de significado a través de grafos, [14] presenta un enfoque de análisis que usa HRG síncronas para la generación y traducción automática basada en semántica.

Por otro lado [15] recomienda el uso de HRG para describir representaciones de significado en Procesamiento del Lenguaje Natural (PLN), y en este sentido [16] propone la Representación de Significado Abstracto (AMR), por sus siglas en inglés. Chiang presenta un algoritmo para un analizador "Bottom-Up" en HRG representado a través de un sistema deductivo. Este algoritmo se basa en una buena descomposición hiperarbórea de las reglas de la gramática, aunque con ciertas restricciones, tal como, ancho y grado del hipergrafo acotados. El algoritmo también ofrece un esquema de optimización que, junto con las restricciones antes mencionadas, permite ejecutar el analizador en un tiempo polinomial.

El algoritmo que plantea [3] y [6] solamente emplean gramáticas de grafos y lo aplican para representar algunos aspectos del significado de una oración en inglés. Del mismo modo, [17] desarrolla un algoritmo para Gramáticas de Grafos Regulares [18], las cuales son una subfamilia de los Lenguajes de Reemplazo de Hiperaristas (HRL), por sus siglas en inglés.

[19] Propone un analizador eficiente bottom-up para HRG llamado analizador predictivo de reducción de cambios para una subclase de gramáticas, la cual generalizan el concepto de analizador de cadena ($SLR(1)$) a grafos.

[20] Propone un algoritmo para resolver el problema de membresía en tiempo polinomial del $O(n^2+mn)$, donde n es el tamaño del grafo de entrada y m es el tamaño de la gramática. Este algoritmo se basa en gramáticas de Grafos Acíclicos y Dirigidos (DAG) por sus siglas en inglés, las cuales son un caso especial de las HRG.

[21] Propone un algoritmo para resolver el problema del analizador uniforme, es decir que la HRG se considera como parte de la entrada, la solución que presentan es en tiempo

polinomial para un tipo de HRG que cumple dos condiciones, la preservación de reentrada y la preservación de orden, la cual se refiere al orden en que los nodos del hipergrafo de entrada pueden ser instanciados.

[22] Propone una extensión ponderada y polinomial para el algoritmo que resuelve el problema del analizador uniforme con preservación del orden de instanciación en los nodos del hipergrafo de entrada.

El algoritmo *Analizador* que aquí se propone resuelve el problema de membresía en HRG. Se consigue realizando el proceso de análisis sobre una representación matricial de H, en particular sobre la matriz de adyacencias. De tal forma que se define un nuevo tipo de Matriz de Adyacencias (MA), como una herramienta más para el análisis de hipergrafos.

El *Analizador* utiliza la definición de MA para hipergrafos. Por eso se ha definido el concepto de matriz de adyacencias, el cual es una generalización de la matriz de adyacencias de un grafo, correspondiente al hipergrafo de entrada H. En ésta las etiquetas de las hiperaristas quedan representadas a través de conjuntos, los cuales forman cada entrada de la matriz correspondiente al hipergrafo de entrada H.

El análisis de la complejidad para el *Analizador* propuesto arroja una complejidad polinomial de orden $O(l^5)$, donde l es el número de vértices del hipergrafo de entrada H. Así mismo, se ha demostrado la corrección del algoritmo *Analizador*, lo cual contrasta con las propuestas algorítmicas de otros autores.

La tesis está organizada de la siguiente manera: consta de 4 capítulos y un cuerpo bibliográfico. El capítulo 1 Introducción, en éste se abordan los antecedentes y los objetivos del trabajo de investigación. El capítulo 2 Marco teórico de la investigación, en éste se enmarcan los aspectos teóricos y conceptuales asociados a algunos conceptos relacionados con las HRG, como el mecanismo de reemplazo de hiperaristas por hipergrafos, la relación de derivación y su cerradura reflexiva y transitiva. También se describe ahí el lenguaje de hipergrafos, así como el concepto de parametrización. El capítulo 3 Proceso de Análisis, se presenta el algoritmo *Analizador* que resuelve el problema de membresía en HRG, así como el desarrollo de un ejemplo que muestra su funcionamiento; también la demostración de que el *Analizador* es correcto y el análisis de su complejidad. El capítulo 4 Conclusiones y trabajo a futuro, en este apartado se describen los principales resultados obtenidos, y las aportaciones de este trabajo de investigación. También se identifican algunas líneas de investigación apoyadas en los resultados obtenidos.

Capítulo 2

Marco Teórico

En este capítulo se presentan los antecedentes y las consideraciones teóricas bajo las que se sustenta el trabajo de investigación, de tal forma que se presenta la revisión de los avances previos logrados en esta línea de investigación y que de forma directa o indirecta abordan el tema propuesto en los objetivos y que además ayudan a justificar el trabajo de investigación desarrollado.

2.1. Hipergrafos y Gramáticas de Reemplazo de Hiperaristas

Un hipergrafo es la generalización de un grafo, donde cada **hiperarista** e [6] puede relacionar a cualquier subconjunto de vértices, los cuales de denominan vértices adjuntos, incluyendo al conjunto vacío. Un hipergrafo se forma a partir de un conjunto de vértices junto con un número variado de hiperaristas dirigidas [23], [24] y [25].

Definition 1. Un **hipergrafo con hiperaristas etiquetadas y dirigidas** es una terna $H = (V, E, lab)$, donde V es un conjunto de vértices. Cada $e \in E$ es un par (R_e, D_e), tal que $R_e \subseteq V$ es el origen de e, y $D_e \subseteq V \setminus R_e$ es el destino. C es un conjunto numerable de etiquetas, y lab es una función de E en C.

Definition 2. Un hipergrafo con hiperaristas etiquetadas y dirigidas $H = (V, E, lab)$ es **enraizado** si,

- H es acíclico y
- $V = \{v_R\} \cup N$, $\{v_R\} \cap N = \emptyset$, v_R es un vértice distinguido y llamado nodo raíz del hipergrafo H. N es el conjunto de vértices que no son raíz.

Cuando hablemos de hipergrafo, entiéndase que nos referimos a un hipergrafo enraizado con hiperaristas etiquetadas y dirigidas.

Definition 3. [26] Sea $H = (V, E, lab)$ un hipergrafo; V un conjunto de vértices, y $a, b \in V$. Entonces, a es un vértice adyacente a b si existe una hiperarista $e \in E$ tal que $a \in R_e$ y $b \in D_e$.

2.1.1. Reemplazo de Hiperaristas por Hipergrafos

En esta sección se muestran las definiciones del proceso de derivación de hipergrafos y reemplazo de una hiperarista por un hipergrafo. Estos son elementos necesarios para abordar el concepto de la HRG, el cual es fundamental en el diseño del *Analizador*.

Los vértices externos se definen como una lista ordenada de vértices distintos, llamados ext. El vértice raíz se define como un vértice designado como la raíz del hipergrafo desde donde cada hiperarista es dirigida; ambos elementos especifican cómo reemplazar una hiperarista por un hipergrafo.

Las HRG permiten manipular hipergrafos mediante la sustitución de hiperaristas. Una hiperarista $e \in H$ es reemplazada por un hipergrafo H', a través del vértice raíz y los vértices externos. Inicialmente la arista e es removida y los vértices externos de H' mapeados con los vértices adjuntos de e. Es decir, el i-ésimo y j-ésimo vértices externos de la hiperarista e son mapeados con el i-ésimo y j-ésimo vértices adjuntos en el hipergrafo H.

Definition 4. [24] Sea $\mathcal{H}_C$ la clase de todos los hipergrafos sobre el conjunto de etiquetas C; $H \in \mathcal{H}_C$ un hipergrafo, y $B \subseteq E(H)$ un conjunto de hiperaristas para ser reemplazadas. Sea $repl : B \to \mathcal{H}_C$ una función que realiza la operación de reemplazo de la manera que se describe a continuación.

El **reemplazo** de B en H hecho por $repl$ produce el hipergrafo $H[repl]$. Este se obtiene al quitar B de E_H, agregando disjuntamente los vértices e hiperaristas de $repl(e)$ por cada $e \in B$ y empatando el i-ésimo vértice externo con el i-ésimo vértice adjunto de e para cada $e \in B$. Todas las hiperaristas mantienen sus etiquetas y vértices adyacentes; los vértices externos de $H[repl]$ son los de H. Si $B = \{e_1, \ldots, e_n\}$ y $repl(e_i) = R_i$, para $i = 1, \ldots, n$, entonces se escribe $H[e_1/R_1, \ldots, e_n/R_n]$ en lugar de $H[repl]$.

La Figura 2.1 inciso b) muestra el reemplazo de la hiperarista Y por el hipergrafo H, el cual se muestra en el inciso a).

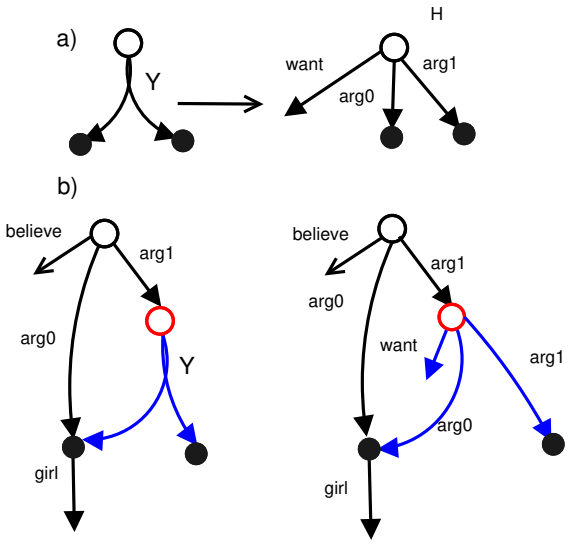

Figura 2.1: a) Hipergrafo H. b) Reemplazo de hiperarista Y por H.
(Tomada de [15])

2.1.2. Gramáticas de Reemplazo de Hiperaristas

Las HRG son elementos que pueden ser empleados para la generación y análisis de la representación semántica apoyada en hipergrafos [27]. En la Figura 2.2, inciso a), se ilustra una HRG.

Definition 5. [6] Una **Gramática de Reemplazo de Hiperaristas** (HRG) es una tupla $G = (N, T, S, P)$ donde,

- N es un conjunto finito de símbolos no terminales.
- T es un conjunto finito de símbolos terminales.
- $S \in N$ es un símbolo no terminal inicial.
- P es un conjunto finito de reglas de producción de la forma $p : A \to R$, donde $A \in N$, y R es un hipergrafo cuyas hiperaristas están etiquetadas con símbolos de $T \cup N$.

2.1.3. Derivación

Sea $e \in E$ una hiperarista etiquetada con A, y R un hipergrafo. El elemento principal de las HRG es el conjunto finito de reglas de producción. Se dice que éstas controlan el proceso de derivación pues determinan la hiperarista e, la cual debe ser reemplazada por un

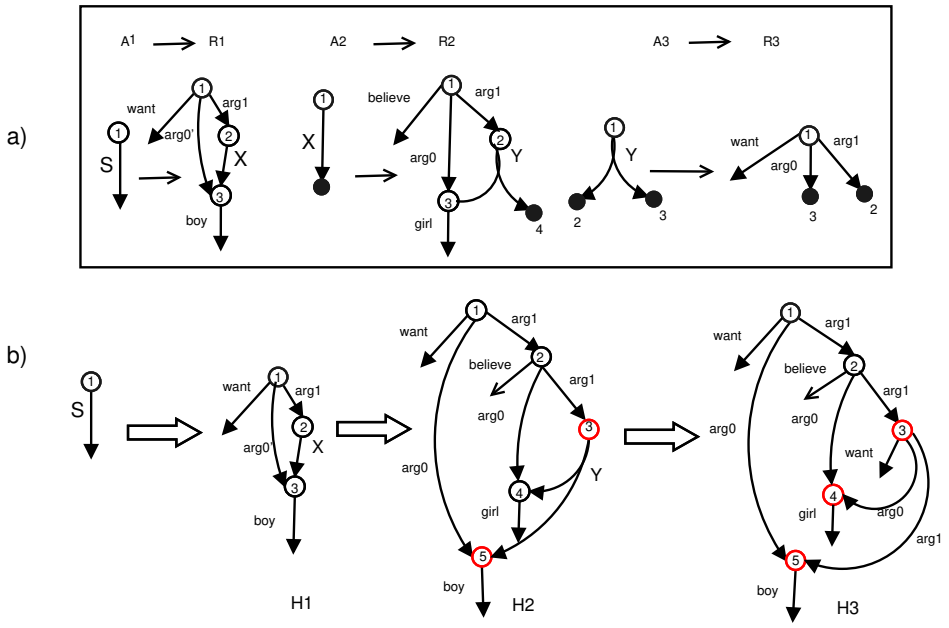

Figura 2.2: a) Reglas de producción para G. b) Derivación a partir de G para el hipergrafo H_3 que representa el significado "The boy wants the girl to believe that he wants her". (Tomada de [15])

hipergrafo. Una regla de producción de las HRG es de la forma $A \to R$ para la hiperarista e que es reemplazada. A es llamado el lado izquierdo de $A \to R$ y R el lado derecho de $A \to R$. Se tiene la siguiente definición.

Definition 6. [6] Sea N un conjunto de símbolos no terminales. Una **regla de producción** sobre N es de la forma $p = A \to R$, $A \in N$ y R es un hipergrafo cuyas hiperaristas están etiquetadas por símbolos de $N \cup T$.

La definición 6 se tomó de [6]; sin embargo, para el propósito de este trabajo, $A \in N$ corresponde a la etiqueta de la única hiperarista del hipergrafo del lado izquierdo de una producción.

El proceso de derivación inicia con el símbolo inicial S. Luego repetidamente se elige un símbolo A no terminal para ser reemplazado por el hipergrafo R, a través de alguna regla de producción $A \to R$. A su vez el hipergrafo R puede tener otras hiperaristas etiquetadas con símbolos no terminales, de tal manera que el proceso de reemplazo continúa hasta que todas las hiperaristas quedan etiquetadas con símbolos terminales [28].

Definition 7. [28] Sea G una HRG y $A \to R$ una regla de producción de G . La relación $H' \Rightarrow H''$ (H'' es derivada de H' en un solo paso) se define de la siguiente forma. H' debe

tener una hiperarista e etiquetada con A. Sean $v_1, ..., v_k$ vértices conectados de la hiperarista y $u_1, ..., u_k$ los vétices externos de R. Entonces, H'' es el hipergrafo formado después de haber removido e de H', haciendo una copia isomorfa de R e identificando v_i con las copias de u_i para cada $i = 1, ..., k$. Denotamos mediante $\overset{*}{\Rightarrow}$ a la cerradura reflexiva y transitiva de $\Rightarrow$.

Ejemplo 2.1.1. La Figura 2.2, inciso b) muestra un ejemplo del proceso de derivación a partir de G para el hipergrafo de entrada H, el cual representa "The boy wants the girl to believe that he wants her".

Inicialmente se aplica la regla de producción $A_1 \rightarrow R_1$ para reemplazar el hipergrafo cuya hiperarista está etiquetada con el símbolo inicial S, por el hipergrafo R_1. Así es obtenido el hipergrafo H_1. Posteriormente en H_1, sobre la base de la regla de producción $A_2 \rightarrow R_2$, se reemplaza la hiperarista X por el hipergrafo R_2, obteniendo así un nuevo hipergrafo H_2. Enseguida se realiza una copia isomorfa de R_2, realizando un mapeo entre los vértices 2, 3 conectados a X y los vértices externos 2, 3 de R_2.

Finalmente, en H_2, sobre la base de la regla de producción $A_3 \rightarrow R_3$, se reemplaza la hiperarista Y por el hipergrafo R_3, con lo que se obtiene el hipergrafo H_3. Después se realiza una copia isomorfa de R_3, haciendo un mapeo entre los vértices 3, 4 y 5 conectados a Y y los vértices externos 3, 4 y 5 de R_3.

Se puede decir que en el hipergrafo resultante H_3 todas las hiperaristas están etiquetadas solamente con símbolos terminales. Por lo tanto, se concluye que el hipergrafo de entrada H_3 es derivado a partir del símbolo inicial S aplicando las reglas de producción de G.

Una vez presentado el concepto de las HRG, así como el proceso de derivación, se puede abordar el concepto de lenguaje de hipergrafos. Éste se define como el conjunto de todos los hipergrafos H, tal que cada uno contiene solamente hiperaristas etiquetadas con símbolos terminales; además, estos hipergrafos pueden ser derivados a partir del símbolo S aplicando producciones de P en un número finito de pasos $S \overset{*}{\Rightarrow} H$ [29].

Definition 8. [23] Sea $H \in HRG$ el **lenguaje de hipergrafos** $L(H)$ generado por H comprende todos los hipergrafos etiquetados con símbolos terminales, los cuales se pueden derivar de S aplicando producciones de P, donde $S \in \mathcal{H}_C$ es un hipergrafo.

$L(H) = \{H \in \mathcal{H}_\mathcal{C} | S \Rightarrow_P^* H\}$.

2.2. Parametrización

Cuando hablamos de complejidad parametrizada de un problema, se tiene por objetivo identificar la fuente de dificultad del problema al introducir una medida auxiliar que refleja la dureza de una instancia determinada, es decir, la complejidad no solo se mide por el tamaño de la entrada, sino que también se toma en cuenta algún o algunos parámetros como medida del "punto débil"de la instancia y que además se asume que es pequeño [30] [31] [32].

La complejidad parametrizada tiene como objetivo clasificar problemas intratables dentro de otras "escalas" de tal forma que varios problemas son "díficiles" o intratables cuando su complejidad es medida solo en términos del tamaño de la entrada, pero éstos mismos se vuelven tratables con un parámetro fijo k "pequeño", el cual se considera como la medida de dureza de la instancia dada[30] [31] [32].

En la teoría de la complejidad parametrizada el objetivo es comprender en que medida contribuyen dichos parámetros al comportamiento de la complejidad global del problema, es decir en este enfoque se trata de responder la pregunta ¿Qué hace que el problema sea intratable?, de tal forma que se tiene por objetivo diseñar y analizar problemas cuya complejidad pertenece a la clase de problemas de Parámetro Fijo tratable (FPT) por sus siglas en inglés [30].

Definition 9. [30] Un problema parametrizado es un lenguaje $L \subseteq \Sigma^* \times \mathbb{N}$, donde Σ es un alfabeto finito, fijo. Para una instancia $(x, k) \in \Sigma^* \times \mathbb{N}$, k es llamado el parámetro. Haremos uso de la notación ρ para hacer énfasis cuando nos referimos a un problema parametrizado.

Ejemplo 2.2.1. Por ejemplo [31], en el área de la Verificación Automatizada, supongamos que se tiene el problema de un sistema finito de estados (estructura Kripke), como un circuito que debe cumplir con cierta propiedad, la propiedad es una fórmula ϕ en Lógica Temporal (LTL), y el problema es decidir si una estructura satisface la fórmula ϕ. Este problema es conocido como problema de verificacion del modelo LTL. Visto desde el punto de vista de complejidad clásica, este problema es intratable.

En términos de algoritmo parametrizado, este problema se puede plantear de la siguiente manera:
Entrada: Una estructura finita de Kripke $\mathcal{K} = (\mathrm{V}, \mathrm{E}, \lambda)$, y una fórmula-LTL ϕ
Parámetro: Un entero positivo k
Pregunta: ¿Decidir si una estructura satisface la fórmula ϕ?
donde k es la longitud de la fórmula de entrada ϕ y n es el tamaño de la estructura de entrada $\mathcal{K}$.

El concepto de FPT es fundamental dentro de la complejidad parametrizada, y éste permite hacer tratables a los problemas que son intratables, debido a que se puede identificar y fijar alguna restricción, es decir se puede fijar el parámetro en algún valor constante que conduzca a mejorar el comportamiento computacional de los algoritmos que resuelven dichos problemas intratables [30]. Un problema parametrizado es tratable con parámetro fijo si admite un algoritmo que trabaja en tiempo $f(k)n^{O(1)}$ para alguna función f calculable, donde k es el parámetro y n es la longitud de la entrada. Existen dos vertientes naturales para optimizar el tiempo de ejecución de un algoritmo FPT [32]:

1. Se puede optimizar la dependencia del parámetro, es decir, hacer la función f lo más baja posible.

2. Se puede optimizar el factor polinomial. es decir, hacer la constante n del exponente tan pequeña como sea posible.

Definition 10. [30] Un problema $L \subseteq \Sigma^* \times \Sigma^*$ es un parámetro fijo tratable si hay un algoritmo que de forma correcta decide, para la entrada $(x, y) \in \Sigma^* \times \Sigma^*$ ya sea $(x, y) \in L$ en tiempo $f(k)n^{\alpha}$, donde n es el tamaño de la parte principal de la entrada x, $\|x\| = n$, k es el parámetro que podemos tomar para que sea de longitud de y, $k = \|y\|$, α es una constante (independiente de k), y f es una función arbitraria.

2.2.1. Métricas de parametrización

Existen diferentes métricas de ancho de hipergrafo para resolver problemas de FPT tales como, ancho de clique, ancho de banda, ancho de rango, **ancho hiperarbóreo** y ancho de camino [33]. En partícular, el ancho hiperarbóreo de un hipergrafo es una de las métricas utilizadas con más frecuencia en algoritmos parametrizados. De manera intuitiva podemos decir, que el ancho hiperarbóreo de un hipergrafo mide que tanto se parece el hipergrafo a un árbol.

Cuando el parámetro ancho hiperarbóreo de un hipergrafo es pequeño, o de manera equivalente el hipergrafo admite una buena técnica de descomposición arbórea, entonces muchos de los problemas de hipergrafos que son intratables en general se vuelven tratables o eficientes [34].

2.2.2. Descomposición hiperarbórea

Los problemas que se basan en hipergrafos con ancho de hiperárbol acotado pueden ser tratables [35], [36] y [37], siempre y cuando cada propiedad de hipergrafo pueda ser definible en lógica monádica de segundo orden [38].

Un hiperárbol de un hipergrafo $H = (V, E)$ es una 3-tupla $\langle T, x, \lambda \rangle$, donde $T = (N, F)$ es un árbol enraizado y x y λ son funciones etiquetadoras que asocian a cada nodo $p \in N$ con dos conjuntos: $x(p) \subseteq V$ y $\lambda(p) \subseteq E$. Denotan el subárbol enraizado al nodo $p \in N$ con T_p y sea $x(T_p) = \{v \mid v \in x(w), w \in T_p\}$ [39].

Definition 11. [39] Un árbol de descomposición de un hipergrafo $H = (V, E)$ es un hiperárbol $HD = \langle T, x, \lambda \rangle$, tal que cumple las siguientes condiciones:

1. Para cada hiperarista $e \in E$, hay un nodo $p \in N$, tal que $vertices(e) \subseteq x(p)$,
2. Para cada vértice $v \in V$, el conjunto $\{p \in N \mid v \in x(p)\}$ induce un subárbol conectado de T,
3. Para cada $p \in N$, $x(p) \subseteq vertices(\lambda(p))$,
4. Para cada $p \in N$, $vertices(\lambda(p)) \cap x(T_p) \subseteq x(p)$.

El ancho de un árbol de descomposición hiperarbórea T es $width(T) = \text{máx}_{t \in T} |\lambda(t)| - 1$. Sea $\mathcal{T}_H$ la familia de todos los árboles de descomposición de H, se define el ancho del hipergrafo H como el $width(H) = \text{mín}_{T \in \mathcal{T}_H}\{width(T)\}$.

Ejemplo 2.2.2. En este ejemplo, la Figura 2.3, inciso a) muestra un hypergrafo H y el inciso b) de la misma Figura muestra una de sus descomposiciones hiperarbórea con ancho hiperarbóreo igual a 3.

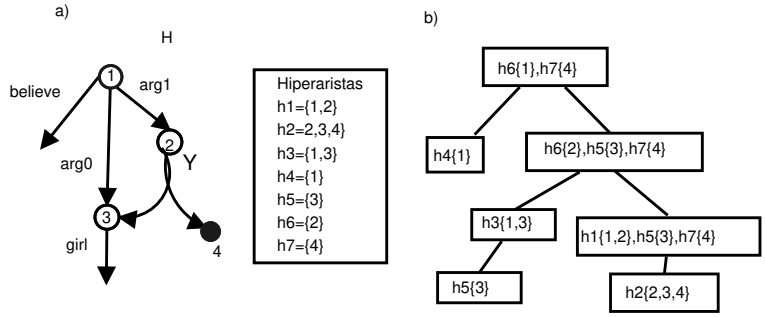

Figura 2.3: a) Hipegrafo H y b) Descomposición hiperarbórea de H.

Capítulo 3

Proceso de Análisis

3.1. Analizador

En esta sección se presenta el *Analizador* para resolver el problema de membresía en HRG. Aquél lleva a cabo el análisis en la MA del hipergrafo de entrada H. La MA para el hipergrafo de entrada H, y para cada hipergrafo R de cada una de las reglas de producción de la HRG, es creada antes de ejecutar el *Analizador*.

3.1.1. Matriz de Adyacencia

En el *Analizador* no se utiliza la definición de Matriz de Adyacencias (MA) para grafos, sino una en la cual las etiquetas de las hiperaristas de cada vértice quedan representadas en cada uno de los elemento de la matriz.

La MA de un hipergrafo es una matriz cuadrada, donde los vértices de H indexan a cada fila de la MA bajo un orden arbitrario. Sin embargo, el orden elegido debe ser consistente; es decir, indexar las filas y columnas bajo el mismo orden Esta consistencia se puede ver representada en la definición 12.

Cuando hablemos de M_H, entiéndase que nos referimos a la matriz de adyacencias del hipergrafo H. Del mismo modo, cuando hablemos de M_A, entiéndase que nos referimos a la matriz de adyacencia del hipergrafo R, donde $A \to R$.

Definition 12. Sea H un hipergrafo con hiperaristas etiquetadas y dirigidas, y sea MA una matriz cuadrada con entradas a_{ij}, definidas por

$$a_{ij} = \bigcup_{e \in E(H),\ i \in R_e,\ j \in D_e} \{lab(e)\}.$$

Para el propósito de esta investigación, conformamos la MA con base en la dirección de cada hiperarista dirigida, de tal forma que la primera fila se corresponde con el conjunto

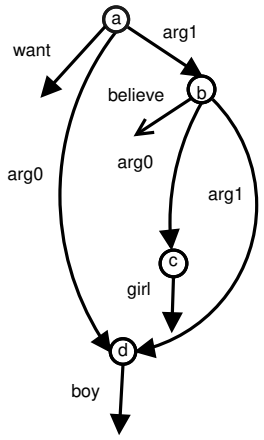

Figura 3.1: Hipergrafo H para el ejemplo 3.1.1.
(tomado de [6])

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ \infty \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & \infty \end{array} \\ \begin{pmatrix} \{\} & \{arg_1\} & \{\} & \{arg_0\} & \{want\} \\ \{\} & \{\} & \{arg_0\} & \{arg_1\} & \{believe\} \\ \{\} & \{\} & \{\} & \{\} & \{girl\} \\ \{\} & \{\} & \{\} & \{\} & \{boy\} \\ \{\} & \{\} & \{\} & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.2: M_H del hipergrafo en la Figura 3.1 para el ejemplo 3.1.1.

formado por el vértice raíz. Enseguida la segunda fila se corresponde con el conjunto formado por el vértice adyacente a las siguientes raíces ubicadas en el hipergrafo. Se procede de arriba hacia abajo y de izquierda a derecha. Se continua así hasta haber representado a todas las hiperaristas del hipergrafo en la MA.

Ejemplo 3.1.1. En las Figuras 3.1 y 3.2, se muestra el hipergrafo de entrada H con su M_H formada por cinco filas y cinco columnas. La primera fila tiene 3 conjuntos $\neq \emptyset$, los cuales representan las etiquetas de las hiperaristas adyacentes al vértice a.

Las Figuras 3.4 a 3.9 describen las M_R para el hipergrafo R de las reglas X_0, X_3, X_2 y X_1 de la HRG que se muestran en la Figura 3.3.

Ejemplo 3.1.2. Este ejemplo muestra la estructura para cada una de las MA de acuerdo a los hipergrafos que se van generando en cada uno de los 3 pasos del proceso de derivación mostrado en la Figura 2.2, inciso b).

1. El proceso inicia desde el simbolo inicial S y aplica la regla de producción correspondiente a S mostrada en la Figura 2.2, inciso a) para derivar el hipergrafo H_1 mostrado

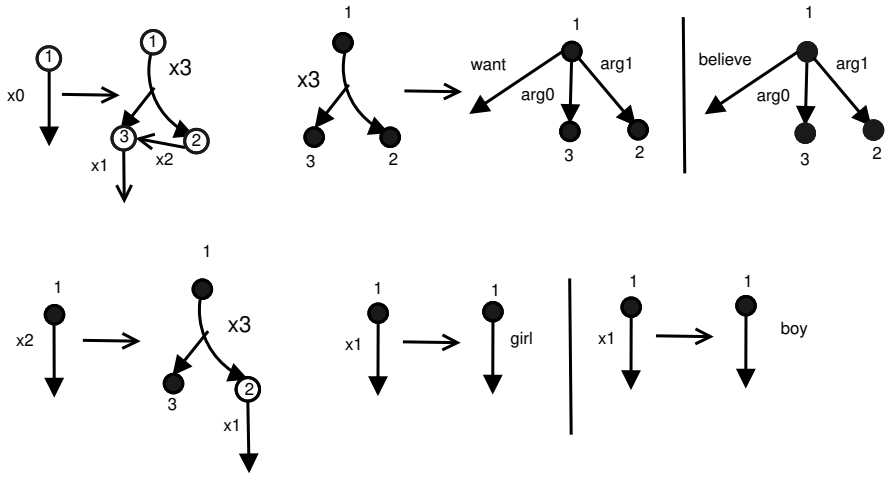

Figura 3.3: HRG para el ejemplo 3.1.1.
(tomadas de [6])

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \infty \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\ \begin{pmatrix} \{\} & \{X_3\} & \{X_3\} & \{\} \\ \{\} & \{\} & \{X_2\} & \{\} \\ \{\} & \{\} & \{\} & \{X_1\} \\ \{\} & \{\} & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.4: M_{X_0} del hipergrafo en la Figura 3.3 para el ejemplo 3.1.1.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \infty \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\ \begin{pmatrix} \{\} & \{arg_1\} & \{arg_0\} & \{want\} \\ \{\} & \{\} & \{\} & \{\} \\ \{\} & \{\} & \{\} & \{\} \\ \{\} & \{\} & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.5: M_{X_3} del hipergrafo en la Figura 3.3 para el ejemplo 3.1.1.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \infty \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\ \begin{pmatrix} \{\} & \{arg_1\} & \{arg_0\} & \{believe\} \\ \{\} & \{\} & \{\} & \{\} \\ \{\} & \{\} & \{\} & \{\} \\ \{\} & \{\} & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.6: M_{X_3} del hipergrafo en la Figura 3.3 para el ejemplo 3.1.1.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \infty \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\ \begin{pmatrix} \{\} & \{X_3\} & \{X_3\} & \{\} \\ \{\} & \{\} & \{\} & \{X_1\} \\ \{\} & \{\} & \{\} & \{\} \\ \{\} & \{\} & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.7: M_{X_2} del hipergrafo en la Figura 3.3 para el ejemplo 3.1.1.

$$\begin{array}{c} \\ 1 \\ \infty \end{array} \begin{array}{c} \begin{array}{cc} 1 & \infty \end{array} \\ \begin{pmatrix} \{\} & \{boy\} \\ \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.8: M_{X_1} del hipergrafo en la Figura 3.3 para el ejemplo 3.1.1.

$$\begin{array}{c} \\ 1 \\ \infty \end{array} \begin{array}{c} \begin{array}{cc} 1 & \infty \end{array} \\ \begin{pmatrix} \{\} & \{girl\} \\ \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.9: M_{X_1} del hipergrafo en la Figura 3.3 para el ejemplo 3.1.1.

en la Figura 3.10. En este caso la estructura de las matrices de adyacencias para H_1 y S es la misma y se muestran en las Figuras 3.11 y 3.12.

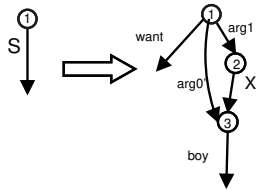

Figura 3.10: Hipergrafo H_1 para el ejemplo 3.1.2.
(tomado de [6])

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \infty \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\ \begin{pmatrix} \{\} & \{arg_1\} & \{arg_0\} & \{want\} \\ \{\} & \{\} & \{X\} & \{\} \\ \{\} & \{\} & \{\} & \{boy\} \\ \{\} & \{\} & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.11: Matriz M_{H_1} para el ejemplo 3.1.2.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \infty \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 2 & \infty \end{array} \\ \begin{pmatrix} \{\} & \{arg_1\} & \{arg_0\} & \{want\} \\ \{\} & \{\} & \{X\} & \{\} \\ \{\} & \{\} & \{\} & \{boy\} \\ \{\} & \{\} & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.12: M_S del hipergrafo en la Figura 3.10 para el ejemplo 3.1.2.

2. Ahora se tiene el hipergrafo H_1 generado en el paso 1, el cual tiene una hiperarista etiquetada con el simbolo X como se muestra en la Figura 3.10, para el siguiente paso de derivación se aplica la regla de producción correspondiente a X mostrada en la Figura 2.2, inciso a) para generar el siguiente hipergrafo H_2 mostrado en la Figura

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ \infty \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & \infty \end{array} \\ \begin{pmatrix} \{\} & \{arg_1\} & \{\} & \{\} & \{arg_0\} & \{want\} \\ \{\} & \{\} & \{arg_1\} & \{arg_0\} & \{\} & \{believe\} \\ \{\} & \{\} & \{\} & \{Y\} & \{Y\} & \{\} \\ \{\} & \{\} & \{\} & \{\} & \{\} & \{girl\} \\ \{\} & \{\} & \{\} & \{\} & \{\} & \{boy\} \\ \{\} & \{\} & \{\} & \{\} & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.13: Matriz M_{H_2} para el ejemplo 3.1.2.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ \infty \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & \infty \end{array} \\ \begin{pmatrix} \{\} & \{arg_1\} & \{arg_0\} & \{\} & \{believe\} \\ \{\} & \{\} & \{Y\} & \{Y\} & \{\} \\ \{\} & \{\} & \{\} & \{\} & \{girl\} \\ \{\} & \{\} & \{\} & \{\} & \{\} \\ \{\} & \{\} & \{\} & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.14: Matriz M_X para el ejemplo 3.1.2.

3.15. En este caso las matrices de adyacencias para H_2 y X se muestran en las Figuras 3.13 y 3.14.

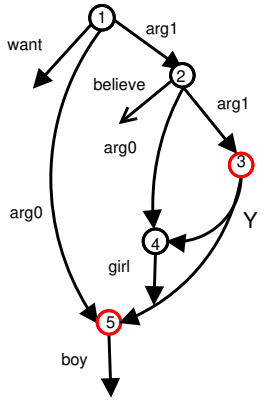

Figura 3.15: Hipergrafo H_2 para el ejemplo 3.1.2.
(tomado de [6])

3. Ahora se tiene el hipergrafo H_2 generado en el paso previo, el cual tiene una hiperarista etiquetada con el simbolo Y como se muestra en la Figura 3.16, para el siguiente paso de derivación se aplica la regla de producción correspondiente a Y mostrada en la Figura 2.2, inciso a) para generar el hipergrafo H_3 mostrado en la Figura 3.19.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ \infty \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & \infty \end{array} \\ \begin{pmatrix} \{\} & \{arg_1\} & \{\} & \{\} & \{arg_0\} & \{want\} \\ \{\} & \{\} & \{arg_1\} & \{arg_0\} & \{\} & \{believe\} \\ \{\} & \{\} & \{\} & \{Y\} & \{Y\} & \{\} \\ \{\} & \{\} & \{\} & \{\} & \{\} & \{girl\} \\ \{\} & \{\} & \{\} & \{\} & \{\} & \{boy\} \\ \{\} & \{\} & \{\} & \{\} & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.17: Matriz M_{H_3} para el ejemplo 3.1.2.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \infty \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\ \begin{pmatrix} \{\} & \{arg_1\} & \{arg_0\} & \{want\} \\ \{\} & \{\} & \{\} & \{\} \\ \{\} & \{\} & \{\} & \{\} \\ \{\} & \{\} & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.18: Matriz M_Y para el ejemplo 3.1.2.

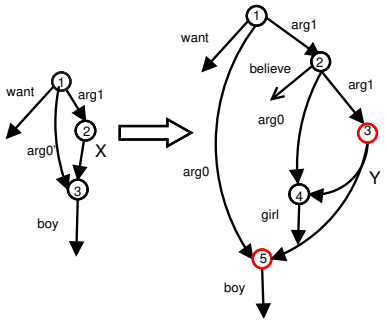

Figura 3.16: Hipergrafo H_2 para el ejemplo 3.1.2.
(tomado de [6])

En este caso las matrices de adyacencias para H_3 y Y se muestran en las Figuras 3.17 y 3.18.

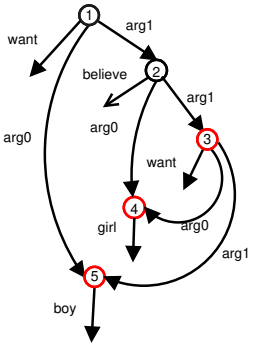

Figura 3.19: Hipergrafo H_3 para el ejemplo 3.1.2.
(tomado de [6])

3.2. Algoritmo *Analizador*

El *Analizador* recibe como entradas a la M_H de H, al conjunto de las M_A de los hipergrafos R para cada una de las reglas de producción de la HRG, y a la M_S de R para la regla inicial $S \to R$.

El *Analizador* no utiliza los vértices externos debido a que éstos se requieren solamente durante el proceso de derivación.

Cuando hablamos de empatar dos filas, entiéndase que nos referimos a que los conjuntos de ambas filas se intersectan; esto significa que las etiquetas de las hiperaristas del hipergrafo representado por éstas, son las mismas que las del hipergrafo R de la regla.

Los algoritmos *Analizador* y *Empata* van creando una copia de las filas de M_H y de la M_R del hipergrafo R conforme se va realizando el análisis. Esto ocurre así porque durante el proceso de análisis los conjuntos que se intersectaron podrían ser $\emptyset$; y en caso de que las filas no empaten entre sí, se tiene que buscar una nueva regla, por lo que es necesario recuperar los conjuntos de ambas filas.

El algoritmo *Analizador* consta de tres iteraciones anidadas.

1. La iteración más externa (líneas 2 a 24) toma cada fila f_{H_i} de la matriz de adyacencias M_H del hipergrafo de entrada H.

2. [1] La iteración intermedia (líneas 2 a 37) toma cada matriz de adyacencias M_{R_j} de R para la regla de producción, cuyas etiquetas se han almacenado en la pila.

3. [2]La iteración más interna (líneas 2 a 32) toma cada fila f_{R_k} de la matriz de adyacencias M_{R_j} para ver si empata con la fila f_{H_i}; aquí tenemos tres casos:

   a) f_{R_k} y f_{H_i} empatan (líneas 5 a 16 y 26 a 28); entonces, *Analizador* continua con el ciclo más externo.

   b) f_{R_k} y f_{H_i} no empatan (líneas 5 a 16 y 18 a 20) y f_{R_k} no contiene variables; entonces, continua con el ciclo más interno.

   c) f_{R_k} y f_{H_i} no empatan (líneas 5 a 16 y 21 a 24) y f_{R_k} contiene las variables $V_1, \ldots, V_m$; entonces, para cada $V_l, 1 \leq l \leq m$ continua con el ciclo intermedio.

[1]Esta iteración corresponde al algoritmo *Procesa*
[2]Esta iteración corresponde al algoritmo *Empata*

El algoritmo *Procesa* utiliza una pila para ir almacenando las etiquetas de todas las variables encontradas en cada fila f_R de la matriz M_R que está siendo utilizada para el análisis. Esta información se requiere ya que, cuando las filas de dicha matriz no empataron con la fila de M_H que está siendo procesada entonces, *Procesa* debe recuperar la información de la pila para regresar a la variable que le permita seguir avanzando en el análisis, para ello toma la MA de la regla, cuya etiqueta es igual que la de la variable recuperada de la pila.

Algorithm 1 *Analizador*(M_H, C_R, A)

```
1: Salida: {true, false}
2: q ← 1
3: M_R ← A
4: push(P_Var[tope], A)
5: Mientras (q − 1) < |M_H|
6:     f_H ← next(f_H, M_H)
7:     f_AH ← f_H
8:     empatafH ← false
9:     res ← Empata(f_AH, M_R)
10:    R → Procesa(f_H, C_R, P_Var, res, M_R)
11:    si R = V_f
12:      empatafH ← true
13:      proc[q] ← f_H
14:      q ← q + 1
15:      ir a 5
16:    si R = V_m
17:       ir a 7
18:    si R = V_n
19:      empatafH ← false
20:      rompeCiclo
21:    en otro caso
22:      ir a 9
23:   continua
24: regresa empatafH
```

Ejemplo 3.2.1. El funcionamiento del Algoritmo 1 se muestra para la matriz M_H de la Figura 3.2, para las M_{R_j} de las reglas de producción de la HRG, las cuales se muestran en las Figuras 3.4 a 3.9.

En la iteración más externa el *Analizador* **toma la fila** a de la matriz M_H y enseguida la matriz de adyacencias, la correspondiente a la regla del símbolo inicial M_{X_0} del hipergrafo R de la regla X_0. A continuación en la línea 9, llama al algoritmo $Empata(a, M_{X_0})$ con el

Algorithm 2 *Procesa*$(f_{AH}, C_R, P_{Var}, res, M_R)$

```
1: Salida: {V_m, V_f, V_n}
2: fin ← true resp ← V_R
3: Mientras ¬vacia(P_Var)
4: switch
5:     case res = V_e:
6:         pop(P_Var[tope])
7:         Para cada conjunto x de f_AH
8:           si x ≠ ∅ fin ← false
9:             rompeCiclo
10:        si fin and P_Var[tope] = res_-d
11:          Mientras P_Var[tope] ≠ res_-i
12:            pop(P_Var[tope])
13:            resp ← V_m ir a 37
14:        continua
15:        resp ← V_f
16:        M_R ← BuscaMA(C_R, P_Var[tope])
17:        pop(P_Var[tope])
18:        push(P_Var, P_Var[tope]_-i)
19:    case res = V_-d and res ≠ P_Var[tope]
20:        push(P_Var, res)
21:    case res = V_p
22:        M_R ← BuscaMA(C_R, P_Var[tope])
23:        si P_Var[tope] ≠ P_Var[tope]_-i
24:          pop(P_Var[tope])
25:          push(P_Var[tope]_-i)
26:        ir a 37
27:    case res = V_n
28:        Mientras(eti(M_R) ≠ P_Var[tope] or C_R ≠ ∅)
29:          M_R ← next(M_R, C_R)
30:          si (eti(M_R) = P_Var[tope])
31:            M_R ← BuscaMA(C_R, P_Var[tope])
32:            ir a 37
33:          en otro caso
34:           resp ← V_n
35:            ir a 37
36:          continua
37: regresa resp
```

Algorithm 3 $Empata(f_{AH}, M_R)$

1: Salida: $\{V_e, V_n, V_i, V_d, V_p\}$
2: Mientras($M_R \neq \emptyset$)
3: $f_R \leftarrow next(f_R, M_R)$ $f_{AR} \leftarrow f_R$
4: $empataf = V_n$
5: Para cada conjunto $x \neq \emptyset$ de f_{AH}
6: $i \leftarrow false$
7: Para cada conjunto $y \neq \emptyset$ de f_{AR}
8: si $(x \cap y) \neq \emptyset$
9: $x \leftarrow x - (x \cap y)$
10: $y \leftarrow y - (x \cap y)$
11: $i \leftarrow true$
12: rompeCiclo
13: continua
14: si $\neg i$
15: rompeCiclo
16: continua
17: Para cada $y \neq \emptyset$ de f_{AR}
18: Sea $V \in y$
19: si $V \in T$
20: ir a 3
21: en otro caso
22: $empataf \leftarrow V_{-d}$
23: $y \leftarrow y - \{V\}$
24: rompeCiclo ir a 32
25: continua
26: si i
27: $empataf \leftarrow Ve$
28: rompeCiclo
29: en otro caso
30: $empataf \leftarrow V_p$
31: continua
32: regresa $empataf$

Algorithm 4 $BuscaMA(CR, PVar[tope])$

1: Salida: $\{\emptyset, M_A\}$
2: $M_{AR} \leftarrow 0$
3: Para cada M'_R de C_R
4: si $etiq(M'_R) = PVar[tope]$
5: $M_{AR} \leftarrow M'_R$
6: regresa (M_{AR})

$$
\begin{array}{r}
\\
\boldsymbol{f_{A_H}} = a \\
\\
\boldsymbol{f_{A_R}} = 1 \\
\\
\boldsymbol{f_{A_H}} = a \\
\\
\boldsymbol{f_{A_R}} = 1
\end{array}
\begin{array}{c}
\begin{array}{ccccc} a & b & c & d & \infty \end{array} \\
\begin{pmatrix} \{\} & \{arg_1\} & \{\} & \{arg_0\} & \{want\} \end{pmatrix} \\
\begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\
\begin{pmatrix} \{\} & \{arg_1\} & \{arg_0\} & \{want\} \end{pmatrix} \\
\begin{array}{ccccc} a & b & c & d & \infty \end{array} \\
\begin{pmatrix} \{\} & \{\} & \{\} & \{\} & \{\} \end{pmatrix} \\
\begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\
\begin{pmatrix} \{\} & \{\} & \{\} & \{\} \end{pmatrix}
\end{array}
$$

Figura 3.20: Empatando fila a de la matriz M_H con fila 1 de M_{X_3}.

propósito de que en la iteración más interna verifique si la fila 1 de la M_{X_0} empata con la fila a de M_H.

Así que el algoritmo $Empata(a, M_{X_0})$, en las líneas 21 a 26, regresa la variable X_3 contenida en la fila 1. En la iteración intermedia el algoritmo $Procesa(a, C_R, P_{Var}, X_3, M_{X_0})$, en las líneas 19 a 20 empila la variable X_3 en la pila P_{Var}. Enseguida, en la iteración más interna en las líneas 4 y 32 el algoritmo $Empata(a, M_{X_0})$ regresa V_p; ya que todos los conjuntos de la fila 1 de M_{X_0} están vacíos; es decir, que ya no hay más variables para empilar. Es entonces que el algoritmo $Procesa(a, C_R, P_{Var}, V_p, M_{X_3})$ en la iteración intermedia en las líneas 21 a 26, toma la matriz M_{X_3}.

Posteriormente el algoritmo $Empata(a, M_{X_3})$ en la iteración más interna toma la fila 1 de la matriz de adyacencias M_{X_3}. Después en la iteración que inicia en la fila 5 a 16 toma los conjuntos $\{arg_1\}$, $\{arg_0\}$ y $\{want\}$ de la fila a y valida que se intersecten con $\{arg_1\}$, $\{arg_0\}$ y $\{want\}$ de la fila 1. Simultáneamente a cada conjunto de ambas filas les va restando la intersección de las dos, como se muestra en la Figura 3.20.

Luego, en la iteración intermedia el algoritmo $Procesa(a, C_R, P_{Var}, V_e, M_{X_3})$ en las líneas 5 a 6 y 16 a 18 desempila a X_3 para tomar el siguiente elemento en la pila; es decir, X_0. Enseguida, el algoritmo $Analizador(M_H, C_R, S)$, en las líneas 13 a 17 marca la fila a como procesada.

Nuevamente, en la iteración más externa el *Analizador* **toma la fila** b de la matriz M_H y enseguida la matriz de adyacencias, la correspondiente a la regla del símbolo inicial M_{X_0} del hipergrafo R de la regla X_0. A continuación en la línea 9, llama al algoritmo $Empata(b, M_{X_0})$ con el propósito de que en la iteración más interna verifique si la fila 2 de la M_{X_0} empata con la fila b de M_H.

Así que el algoritmo $Empata(b, M_{X_0})$, en las líneas 21 a 26, regresa la variable X_2 contenida en la fila 2. En la iteración intermedia el algoritmo $Procesa(b, C_R, P_{Var}, X_2, M_{X_0})$, en las líneas 19 a 20 empila la variable X_2 en la pila P_{Var}. Enseguida, en la iteración más interna en las líneas 4 y 32 el algoritmo $Empata(b, M_{X_0})$ regresa V_p; ya que todos los conjuntos de la fila 2 de M_{X_0} están vacíos; es decir, que ya no hay más variables para empilar. Es entonces que el algoritmo $Procesa(b, C_R, P_{Var}, V_p, M_{X_2})$ en la iteración intermedia en las líneas 21 a 26, toma la matriz M_{X_2}.

A continuación en la línea 9, llama al algoritmo $Empata(b, M_{X_0})$ con el propósito de que en la iteración más interna verifique si la fila 1 de la M_{X_2} empata con la fila b de M_H.

Así que el algoritmo $Empata(b, M_{X_2})$, en las líneas 21 a 26, regresa la variable X_3 contenida en la fila 1. En la iteración intermedia el algoritmo $Procesa(b, C_R, P_{Var}, X_3, M_{X_3})$, en las líneas 19 a 20 empila la variable X_3 en la pila P_{Var}. Enseguida, en la iteración más interna en las líneas 4 y 32 el algoritmo $Empata(b, M_{X3})$ regresa V_p; ya que todos los conjuntos de la fila 1 de M_{X2} están vacíos; es decir, que ya no hay más variables para empilar. Es entonces que el algoritmo $Procesa(b, C_R, P_{Var}, V_p, M_{X_2})$ en la iteración intermedia en las líneas 21 a 26, toma la matriz M_{X2}.

Posteriormente el algoritmo $Empata(b, M_{X2})$ en la iteración más interna toma la fila 1 de la matriz de adyacencias M_{X2}. Posteriormente el algoritmo $Empata(b, M_{X_3})$ en la iteración más interna toma la fila 1 de la matriz de adyacencias M_{X3}. Después en la iteración que inicia en la fila 5 a 16 toma los conjuntos $\{arg_1\}$, $\{arg_0\}$ y $\{believe\}$ de la fila b y valida que no se intersecten con $\{arg_1\}$, $\{arg_0\}$ y $\{want\}$ de la fila 1. Simultáneamente a cada conjunto de ambas filas les va restando la intersección de las dos, como se muestra en la Figura 3.21.

$$
\begin{array}{rc}
 & \begin{array}{cccc} a & b & c & d & \infty \end{array} \\
\boldsymbol{f_{A_H}} = b & \begin{pmatrix} \{\} & \{\} & \{arg_0\} & \{arg_1\} & \{believe\} \end{pmatrix} \\
 & \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\
\boldsymbol{f_{A_R}} = 1 & \begin{pmatrix} \{\} & \{arg_1\} & \{arg_0\} & \{want\} \end{pmatrix} \\
 & \begin{array}{ccccc} a & b & c & d & \infty \end{array} \\
\boldsymbol{f_{A_H}} = b & \begin{pmatrix} \{\} & \{\} & \{\} & \{\} & \{believe\} \end{pmatrix} \\
 & \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\
\boldsymbol{f_{A_R}} = 1 & \begin{pmatrix} \{\} & \{\} & \{\} & \{want\} \end{pmatrix}
\end{array}
$$

Figura 3.21: No empata fila b de la matriz M_H con fila 1 de M_{X_3}.

Después en el ciclo más interno *Empata* en las líneas 4 y 32 regresa V_n para que en la

$$
\begin{array}{rc}
 & \begin{array}{ccccc} a & b & c & d & \infty \end{array} \\
\boldsymbol{f_{A_H}} = b & \begin{pmatrix} \{\} & \{\} & \{arg_0\} & \{arg_1\} & \{believe\} \end{pmatrix} \\
 & \begin{array}{ccccc} 1 & 2 & 3 & \infty & \end{array} \\
\boldsymbol{f_{A_R}} = 1 & \begin{pmatrix} \{\} & \{arg_1\} & \{arg_0\} & \{believe\} & \end{pmatrix} \\
 & \begin{array}{ccccc} a & b & c & d & \infty \end{array} \\
\boldsymbol{f_{A_H}} = b & \begin{pmatrix} \{\} & \{\} & \{\} & \{\} & \{\} \end{pmatrix} \\
 & \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\
\boldsymbol{f_{A_R}} = 1 & \begin{pmatrix} \{\} & \{\} & \{\} & \{\} \end{pmatrix}
\end{array}
$$

Figura 3.22: Empatando fila b de la matriz M_H con fila 1 de M_{X_3}.

iteración intermedia, *Procesa* en las líneas 27 a 32 toma otra opción para M_{X_3}, así que en la iteración que inicia en la fila 5 a 16 toma los conjuntos $\{arg_1\}$, $\{arg_0\}$ y $\{believe\}$ de la fila b y valida que se intersecten con $\{arg_1\}$, $\{arg_0\}$ y $\{believe\}$ de la fila 1. Simultáneamente a cada conjunto de ambas filas les va restando la intersección de las dos, como se muestra en la Figura 3.22.

Luego, en la iteración intermedia el algoritmo $Procesab, C_R, P_{Var}, V_e, M_{X3})$ en las líneas 5 a 6 y 16 a 18 desempila a X_3 para tomar el siguiente elemento en la pila; es decir, X_2. Enseguida, el algoritmo $Analizador(M_H, C_R, S)$, en las líneas 13 a 17 marca la fila b como procesada.

Nuevamente, en la iteración más externa el *Analizador* **toma la fila** c de la matriz M_H y a continuación en la línea 9, llama al algoritmo $Empata(c, M_{X_2})$ con el propósito de que en la iteración más interna verifique si la fila 2 de la M_{X_2} empata con la fila c de M_H.

Así que el algoritmo $Empata(c, M_{X_2})$, en las líneas 21 a 26, regresa la variable X_1 contenida en la fila 2. En la iteración intermedia el algoritmo $Procesa(c, C_R, P_{Var}, X_1, M_{X_2})$, en las líneas 19 a 20 empila la variable X_1 en la pila P_{Var}. Enseguida, en la iteración más interna en las líneas 4 y 32 el algoritmo $Empata(c, M_{X_2})$ regresa V_p; ya que todos los conjuntos de la fila 2 de M_{X_2} están vacíos; es decir, que ya no hay más variables para empilar. Es entonces que el algoritmo $Procesa(c, C_R, P_{Var}, V_p, M_{X_2})$ en la iteración intermedia en las líneas 21 a 26, toma la matriz M_{X_1}.

Posteriormente el algoritmo $Empata(c, M_{X_1})$ en la iteración más interna toma la fila 1 de la matriz de adyacencias M_{X_1}. Posteriormente el algoritmo $Empata(c, M_{X_1})$ en la iteración más interna toma la fila 1 de la matriz de adyacencias M_{X_1}. Después en la iteración que inicia en la fila 5 a 16 toma los conjuntos $\{\}$, $\{\}$ y $\{girl\}$ de la fila c y valida que no se

$$
\begin{array}{rc}
 & \begin{array}{ccccc} a & b & c & d & \infty \end{array} \\
\boldsymbol{f_{A_H}} = c & \begin{pmatrix} \{\} & \{\} & \{\} & \{\} & \{girl\} \end{pmatrix} \\
 & \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\
\boldsymbol{f_{A_R}} = 1 & \begin{pmatrix} \{\} & \{\} & \{\} & \{boy\} \end{pmatrix} \\
 & \begin{array}{ccccc} a & b & c & d & \infty \end{array} \\
\boldsymbol{f_{A_H}} = c & \begin{pmatrix} \{\} & \{\} & \{\} & \{\} & \{girl\} \end{pmatrix} \\
 & \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\
\boldsymbol{f_{A_R}} = 1 & \begin{pmatrix} \{\} & \{\} & \{\} & \{boy\} \end{pmatrix}
\end{array}
$$

Figura 3.23: No empata fila c de la matriz M_H con fila 1 de M_{X_1}.

$$
\begin{array}{rc}
 & \begin{array}{ccccc} a & b & c & d & \infty \end{array} \\
\boldsymbol{f_{A_H}} = c & \begin{pmatrix} \{\} & \{\} & \{\} & \{\} & \{boy\} \end{pmatrix} \\
 & \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\
\boldsymbol{f_{A_R}} = 1 & \begin{pmatrix} \{\} & \{\} & \{\} & \{boy\} \end{pmatrix} \\
 & \begin{array}{ccccc} a & b & c & d & \infty \end{array} \\
\boldsymbol{f_{A_H}} = c & \begin{pmatrix} \{\} & \{\} & \{\} & \{\} & \{\} \end{pmatrix} \\
 & \begin{array}{cccc} 1 & 2 & 3 & \infty \end{array} \\
\boldsymbol{f_{A_R}} = 1 & \begin{pmatrix} \{\} & \{\} & \{\} & \{\} \end{pmatrix}
\end{array}
$$

Figura 3.24: Empatando la fila c de la matriz M_H con la fila 1 de M_{X_1}.

intersecten con $\{\}$, $\{\}$ y $\{boy\}$ de la fila 1. Simultáneamente a cada conjunto de ambas filas les va restando la intersección de las dos, como se muestra en la Figura 3.23.

Después en el ciclo más interno *Empata* en las líneas 4 y 32 regresa V_n para que en la iteración intermedia, *Procesa* en las líneas 27 a 32 toma otra opción para M_{X_1}, así que en la iteración que inicia en la fila 5 a 16 toma los conjuntos $\{\}$, $\{\}$ y $\{girl\}$ de la fila c y valida que se intersecten con $\{\}$, $\{\}$ y $\{girl\}$ de la fila 1. Simultáneamente a cada conjunto de ambas filas les va restando la intersección de las dos, como se muestra en la Figura 3.24.

Luego, en la iteración intermedia el algoritmo $Procesa(c, C_R, P_{Var}, V_e, M_{X_1})$ en las líneas 5 a 6 y 16 a 18 desempila a X_1 para tomar el siguiente elemento en la pila; es decir, X_0. Enseguida, el algoritmo $Analizador(M_H, C_R, S)$, en las líneas 13 a 17 marca la fila c como procesada.

A continuación en la línea 9, llama al algoritmo $Empata(d, M_{X_0})$ con el propósito de que en la iteración más interna verifique si la fila 3 de la M_{X_0} **con la fila** d de M_H.

Así que el algoritmo $Empata(d, M_{X_0})$, en las líneas 21 a 26, regresa la variable X_1 contenida

$$
\begin{array}{rc}
 & \begin{array}{ccccc} a & b & c & d & \infty \end{array} \\
\boldsymbol{f_{A_H}} = d & \begin{pmatrix} \{\} & \{\} & \{\} & \{\} & \{boy\} \end{pmatrix} \\
 & \begin{array}{ccccc} 1 & 2 & 3 & \infty \end{array} \\
\boldsymbol{f_{A_R}} = 1 & \begin{pmatrix} \{\} & \{\} & \{\} & \{boy\} \end{pmatrix} \\
 & \begin{array}{ccccc} a & b & c & d & \infty \end{array} \\
\boldsymbol{f_{A_H}} = d & \begin{pmatrix} \{\} & \{\} & \{\} & \{\} & \{boy\} \end{pmatrix} \\
 & \begin{array}{ccccc} 1 & 2 & 3 & \infty \end{array} \\
\boldsymbol{f_{A_R}} = 1 & \begin{pmatrix} \{\} & \{\} & \{\} & \{boy\} \end{pmatrix}
\end{array}
$$

Figura 3.25: No empata fila d de la matriz M_H con fila 1 de M_{X_1}.

en la fila 3. En la iteración intermedia el algoritmo $Procesa(d, C_R, P_{Var}, X_1, M_{X_0})$, en las líneas 19 a 20 empilar la variable X_1 en la pila P_{Var}. Enseguida, en la iteración más interna en las líneas 4 y 32 el algoritmo $Empata(d, M_{X_0})$ regresa V_p; ya que todos los conjuntos de la fila 3 de M_{X_0} están vacíos; es decir, que ya no hay más variables para empilar. Es entonces que el algoritmo $Procesa(d, C_R, P_{Var}, V_p, M_{X_0})$ en la iteración intermedia en las líneas 21 a 26, toma la matriz M_{X_1}.

Posteriormente el algoritmo $Empata(d, M_{X_1})$ en la iteración más interna toma la fila 1 de la matriz de adyacencias M_{X_1}. Posteriormente el algoritmo $Empata(d, M_{X_1})$ en la iteración más interna toma la fila 1 de la matriz de adyacencias M_{X_1}. Después en la iteración que inicia en la fila 5 a 16 toma los conjuntos {}, {} y $\{boy\}$ de la fila d y valida que se intersecten con {}, {} y $\{boy\}$ de la fila 1. Simultáneamente a cada conjunto de ambas filas les va restando la intersección de las dos, como se muestra en la Figura 3.25.

Enseguida, el algoritmo $Analizador(M_H, C_R, S)$, en las líneas 13 a 17 marca la fila d como procesada y como ya nos hay más filas que procesar en M_H, entonces devuelve true para denotar que H sí es generado por la HRG.

3.2.1. Demostración de la corrección del *Analizador*

Demostraremos que el algoritmo *Analizador* es correcto, pero primero se demuestra una proposición más general.

Proposición 3.2.1. *Sean $H_I \in \mathcal{H}_C$ y HRG una gramática de reemplazo de hiperaristas; entonces $H_I \in L_A(HRG)$ si y sólo si Analizador(M_{H_I}, C_R, A).*

$H \in L_A(HRG)$: Se realiza una demostración por inducción sobre el número k de pasos para

hacer una derivación.

$k = 1 :\ A \Rightarrow H_I$. En este caso se tiene el hipergrafo H_I de entrada y se reconoce en un solo paso, en cuyo caso el algoritmo $Analizador(M_{H_I}, C_R, A)$ realizaría lo siguiente: tomaría las matrices de adyacencias M_{H_I} del hipergrafo de entrada H_I y M_A del hipergrafo R de la regla de producción $A \to R$, las cuales tienen la siguiente forma.

$$\begin{array}{c} \\ n_1 \\ n_2 \\ \vdots \\ n_{m-1} \\ \infty \end{array} \begin{array}{c} \begin{array}{ccccc} n_1 & n_2 & \cdots & n_{m-1} & \infty \end{array} \\ \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1m-1} & h_{1m} \\ h_{21} & h_{22} & \cdots & h_{2m-1} & h_{2m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ h_{m-11} & h_{m-12} & \cdots & h_{m-1m-1} & h_{m-1m} \\ \{\} & \{\} & \cdots & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.26: Matriz M_{H_I} para la demostración.

$$\begin{array}{c} \\ n'_1 \\ n'_2 \\ \vdots \\ n'_{m-1} \\ \infty \end{array} \begin{array}{c} \begin{array}{ccccc} n'_1 & n'_2 & \cdots & n'_{m-1} & \infty \end{array} \\ \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1t-1} & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m-1} & a_{2m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m-11} & a_{m-12} & \cdots & a_{m-1m-1} & a_{m-1m} \\ \{\} & \{\} & \cdots & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.27: Matriz M_A para la demostración.

Al comparar estas dos matrices, se podrían dar los siguientes dos casos:

caso a) Ambas matrices se empatan en todas sus filas, pues de acuerdo a la iteración más externa en las líneas 2 a 24, el *Analizador* tomaría cada una de las filas de M_{H_I} para que en la iteración más interna *Empata* tome cada fila de M_A y en las líneas 5 a 16 y 26 a 28: verificaría que ambas filas se empatan pues los conjuntos $h_{ij} = a_{ij}$, donde $i = 1, ..., m-1$ y $j = 1, ..., m$. A continuación, en la iteración intermedia en las líneas 5, 6, 15 y 37: *Procesa* regresa V_f para que el *Analizador* en las líneas 11 a 15 elimine cada fila empatada de M_{H_I} y regrese el valor *true*.

El algoritmo *Procesa* utiliza una pila porque las HRG en las que se basa el análisis pueden ser resursivas, de tal forma que se requiere gurdar las etiquetas de las varibales correspondientes, las cuales determinan el orden en que estas serán utilizadas durante el análisis.

En la iteración intermedia, en las líneas 19 a 20, *Procesa* empila la etiqueta de la variable encontrada en la fila de M_{H_I}, posteriormente esta etiqueta puede ser desempilada porque se cumple el caso de las líneas 5 a 18, es decir que la fila de M_{H_I} fue empatada con alguna de las filas de la MA para la regla que está siendo utilizada. Así que *Procesa* desempila la etiqueta correspondiente a la variable de la MA de la regla que será utilizada para continuar con el análisis o también empieza a desempilar en las líneas 21 a 26 para el caso en que ya fueron almacenadas todas las etiquetas correspondientes a las variables encontradas en la fila de M_{H_I} que está siendo procesada.

caso b) Cuando ambas matrices no se corresponden, porque al menos hay alguna fila en la que no se empatan las dos matrices. De acuerdo a la iteración más interna en las líneas 4, 5 a 16 y 32: *Empata* devolvería la variable V_n para que en la iteración intermedia en las líneas 27 a 32: *Procesa* tomará una nueva matriz de adyacencias M_A para el hipergrafo R de la regla de producción $A \to R$ y verificaría nuevamente, si se da el caso a) o b).

La hipótesis de inducción es $A \Rightarrow^i H'$ si y sólo si $Analizador(M'_H, C_R, A)$.

$k = i + 1$: Si el número de pasos es $i + 1$ entonces $A \Rightarrow^i H'$; por lo tanto, debe existir una hiperarista X y una producción de la forma $X \to R$, tal que $A \Rightarrow^i H'[X/R] \Rightarrow H$. Por la hipótesis de inducción, $Analizador(M_{H'}, C_R, A)$ regresaría *true*; pero como H' contiene una hiperarista etiquetada X y HRG contiene una regla de producción de la forma $X \to R$, entonces el algoritmo $Analizador(M_{H_I}, C_R, A)$ haría lo siguiente: tomaría la matriz de adyacencias M_{H_I} del hipergrafo de entrada H, la cual tiene la siguiente forma.

$$\begin{array}{c} \\ n_1 \\ n_2 \\ \vdots \\ n_{q-1} \\ \infty \end{array} \begin{array}{c} \begin{array}{ccccc} n_1 & n_2 & \cdots & n_{q-1} & \infty \end{array} \\ \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1q-1} & h_{1q} \\ h_{21} & h_{22} & \cdots & h_{2q-1} & h_{2q} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ h_{q-11} & h_{q-12} & \cdots & h_{q-1q-1} & h_{q-1q} \\ \{\} & \{\} & \cdots & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.28: Matriz M_{H_I} para la demostración.

Por hipótesis de inducción y en base a alguno de los casos a) o b) presentados anteriormente se han empatado las m filas de M_{H_I}, las cuales además han sido eliminadas de M_{H_I}. De tal modo que M_{H_I} tiene la siguiente forma.

$$\begin{array}{c} \\ n_{m+1} \\ \vdots \\ n_{q-1} \\ \infty \end{array} \begin{array}{c} \begin{array}{ccccccc} n_1 & n_2 & \cdots & n_{m+1} & \cdots & n_{q-1} & \infty \end{array} \\ \begin{pmatrix} h_{m+11} & h_{m+12} & \cdots & h_{m+1m+1} & \cdots & h_{m+1q-1} & h_{m+1q} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ h_{q-11} & h_{q-12} & \cdots & h_{q-1m+1} & \cdots & h_{q-1q-1} & h_{q-1q} \\ \{\} & \{\} & \cdots & \{\} & \cdots & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.29: Matriz M_{H_I} para la demostración.

Ahora, *Analizador* tomaría la matriz de adyacencias $M_{H'}$ del hipergrafo de entrada H', la cual tiene la siguiente forma.

$$\begin{array}{c} \\ n_1 \\ n_2 \\ \vdots \\ n_m \\ n_{t-2} \\ n_{t-1} \\ \infty \end{array} \begin{array}{c} \begin{array}{ccccccc} n_1 & n_2 & \cdots & n_{t-2} & n_{t-1} & \infty \end{array} \\ \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1t-2} & h_{1t-1} & h_{1t} \\ h_{21} & h_{22} & \cdots & h_{2t-2} & h_{2t-1} & h_{2t} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ h_{m1} & h_{m2} & \cdots & h_{mt-2} & h_{mt-1} & h_{mt} \\ h'_{t-21} & h'_{t-22} & \cdots & \{X\} & h'_{t-2t-1} & h'_{t-2t} \\ h'_{t-11} & h'_{t-12} & \cdots & h'_{t-1t-2} & h'_{t-1t-1} & h'_{t-1t} \\ \{\} & \{\} & \cdots & \{\} & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.30: Matriz $M_{H'}$ para la demostración.

En la iteración intermedia en las líneas 21 a 26: *Procesa* desempilará a X y tomará la matriz de adyacencias M_X para el hipergrafo R de la regla $X \to R$, la cual tiene la siguiente forma.

$$\begin{array}{c} \\ n'_1 \\ n'_2 \\ \vdots \\ n'_{p-1} \\ \infty \end{array} \begin{array}{c} \begin{array}{ccccc} n'_1 & n'_2 & \cdots & n'_{p-1} & \infty \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p-1} & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p-1} & x_{2p} \\ \cdots & \cdots & \cdots & \cdots\cdots & \cdots \\ x_{p-11} & x_{p-12} & \cdots & x_{p-1p-1} & x_{p-1p} \\ \{\} & \{\} & \cdots & \{\} & \{\} \end{pmatrix} \end{array}$$

Figura 3.31: Matriz M_X para la demostración.

En este caso, las $p-1$ filas de la matriz M_X empatarían con $t-1$ filas de $M_{H'}$, pues la fila n_{t-2} de $M_{H'}$ no empata con ninguna fila de M_X. Esto se observa en la iteración más interna en las lineas 5 a 16 y 21 a 24: *Empata* comprobará que ambas matrices no se corresponden debido a que $M_{H'}$ contiene a la variable X.

De tal modo, que la matriz de adyacencias M_{H_I} para el hipergrafo de entrada H_I tiene la siguiente forma.

$$\begin{array}{c} \\ n'_1 \\ n'_2 \\ \vdots \\ n'_m \\ n'_{m+1} \\ \cdots \\ n'_{q-1} \\ \infty \end{array} \begin{array}{c} \begin{array}{ccccc} n'_1 & n'_2 & \cdots & n'_{q-1} & \infty \end{array} \\ \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1q-1} & h_{1q} \\ h_{21} & h_{22} & \cdots & h_{2q-1} & h_{2q} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ h_{m1} & h_{m2} & \cdots & h_{mq-1} & h_{mq} \\ \cdots x_{11} & & \cdots & & x_{1p} \cdots \\ & & & & \\ \cdots x_{p1} & & \cdots & & x_{pp} \cdots \\ \cdots \{\} & & \cdots & & \{\} \cdots \end{pmatrix} \end{array}$$

Figura 3.32: Matriz M_H que subsume a M_X para la demostración.

En esta matriz M_H las primeras m filas corresponden a aquellas que ya han sido empatadas por la hipótesis de inducción. Las filas restantes $q-m$ aún por empatar, corresponden a M_X subsumida en M_H. Al comparar las $q-m$ filas de la matriz M_H con las $p-1$ filas de M_X, se puede dar cualquiera de los casos a) o b), explicados con anterioridad. De tal modo, que la matriz de adyacencias M_H para el hipergrafo de entrada H tiene la siguiente forma.

$$\begin{array}{c} \\ \infty \end{array} \begin{array}{c} \begin{array}{ccccc} n'_1 & n'_2 & \cdots & n'_{q-1} & \infty \end{array} \\ \begin{pmatrix} \cdots \{\} & & \cdots & & \{\} \cdots \end{pmatrix} \end{array}$$

Figura 3.33: Matriz M_H de H generado por la HRG para la demostración.

Por lo tanto, el *Analizador* devuelve $true$.

$H_I \notin L_A(HRG)$**:** En este caso el algoritmo $Analizador(M_{H_I}, C_R, A)$ haría lo siguiente, tomaría la matriz de adyacencias M_{H_I} del hipergrafo de entrada H_I y cada una de las matrices de adyacencias M_A del hipergrafo R de la regla de producción $A \to R$ y en la iteración más interna en las líneas 5 a 16: *Empata* comprobaría que ambas matrices no se corresponden. Entonces, en la iteración intermedia en las líneas 21 a 26: *Procesa* desempilaría cada una de las M_A restantes y en la iteración más externa en las líneas 5 a 24: *Analizador* comprobaría que M_{H_I} no se corresponde con ninguna de las M_A. De este modo, el proceso de análisis ya no puede continuar y el algoritmo *Analizador* regresa el valor $false$.

Ahora se presenta la demostración de la corrección del algoritmo $Analizador(M_H, C_R, S)$

Corolario 3.2.1. *Sean $H \in \mathcal{H}_T$ y HRG una gramática de reemplazo de hiperaritas; entonces $H \in L_S(HRG)$ si y sólo si $Analizador(M_H, C_R, S)$.*

Demostración. La prueba se sigue directamente sustituyendo $H \in \mathcal{H}_T$ por H y el símbolo inicial S por A en el desarrollo de la demostración de la proposición 3.2.1.

3.2.2. Complejidad del Analizador

La complejidad en tiempo del *Analizador* es de orden polinomial. Esta complejidad está determinada por el número de vértices del hipergrafo de entrada H (denotado como l) y por una constante igual al número de reglas de la HRG (denotado por p). Primero analizaremos la **complejidad para el algoritmo *Empata***, luego para el algoritmo *Procesa* y finalmente para el algoritmo *Analizador*.

Líneas 6 a 17. El número de entradas de f_{A_H} está representado por $|f_{A_H}|$ y el número de entradas de f_{A_R} por $|f_{A_R}|$. Cada entrada es un conjunto representado por x y y y $min(|x|, |y|)$ es el tiempo que lleva ejecutar el bloque de sentencias de la línea 7 a 16. El tiempo necesario para ejecutar la iteración que inicia en la línea 6 y finaliza en la línea 17, se representa en la siguiente sumatoria.

$$t_1 = \sum_{i=1}^{|f_{A_H}|} \sum_{j=1}^{|f_{A_R}|} min(|x|, |y|).$$

Líneas 18 a 28. El número de entradas de f_{A_R} está representado por $|f_{A_R}|$, donde cada entrada es un conjunto y C_1 es el costo necesario para ejecutar el bloque de sentencias, el que inicia en la línea 19 a 27. El tiempo necesario para ejecutar la iteración que inicia en la línea 18 y finaliza en la línea 28, se representan en la siguiente sumatoria.

$$t_2 = \sum_{j=1}^{|f_{A_R}|} C_1.$$

Líneas 3 a 35. El número de filas de cada regla M_R está representado por $|M_R|$. El tiempo necesario para ejecutar las iteración que inicia en la línea 3 y termina en la línea 35, se representa en la siguiente sumatoria.

$$t_3 = \sum_{k=1}^{|M_R|} (t_1 + t_2) = |M_R||f_{A_H}||f_{A_R}|min(|x|, |y|)$$

$$+|f_{A_R}|C_1.$$

Sabemos que el número de filas en M_R para cualquier regla, no puede ser mayor que el número de vértices de H. Lo mismo sucede para el número de conjuntos de f_{A_H}, el número de conjuntos de f_{A_R} y también para $min(|x|, |y|)$. Por lo tanto,

$$t_3 = l(l^3 + lC_1) = l^4 + l^2C_1.$$

Entonces, la complejidad de *Empata* es de $O(l^4)$.

La **complejidad para el algoritmo *Procesa*** se calcula de la siguiente forma:

En las líneas 6 a 14. El número de entradas de f_{A_H} está representado por $|f_{A_H}|$, donde cada entrada es un conjunto y C_2 es el costo necesario para ejecutar el bloque de sentencias que inicia en la línea 7 a 13. Así el tiempo que toma ejecutar la iteración que inicia en la línea 6 y finaliza en la línea 14, se representa en la siguiente sumatoria.

$$t_4 = \sum_{i=1}^{|f_{A_H}|} C_2.$$

Líneas 28 a 36. El número de reglas de C_R es igual a la constante p y C_3 es el costo necesario para ejecutar el bloque de sentencias que inicia en la línea 29 a 35. La siguiente sumatoria representa el tiempo requerido para ejecutar las iteración que inicia en la línea 28 y termina en la línea 36.

$$t_5 = \sum_{i=1}^{|C_R|} C_3 = |C_R|C_3 = pC_3.$$

Líneas 3 a 37. El número de variables por cada fila de cada regla M_R está representado por $|P_{Var}|$. De este modo. la siguiente sumatoria representa el tiempo requerido para ejecutar la iteración que inicia en la línea 3 y termina en la línea 37.

$$t_6 = \sum_{k=1}^{|P_{Var}|} (t_4 + t_5) = |P_{Var}||f_{A_H}|C_2 + pC_3.$$

Asumimos que el número de variables en P_{Var} para cualquier fila no puede ser mayor que el número de vértices de H. Lo mismo ocurre con el número de conjuntos de f_{A_H}. Por lo tanto,

$$t_6 = l(lC_2 + pC3) = l^2C_2 + lpC_3.$$

Entonces, la complejidad de *Procesa* es de $O(l^2)$.

La **complejidad para el *Analizador*** se calcula de la siguiente forma:

Línea 5 a 24. El tiempo requerido para ejecutar la iteración que inicia en la línea 5 y termina en la línea 24, se representa en la siguiente sumatoria.

$$t_7 = \sum_{i=1}^{|M_H|}(t_3 + t_6) = l^4 + l^2C_1 + l^2C_2 + lpC_3.$$

Como el número de vértices de H es una cota superior para M_H; es decir, $l > |M_H|$, entonces el tiempo total T para *Analizador* se calcula de la siguiente forma:

$$T = l(l^4 + l^2C_1 + l^2C_2 + lpC_3) = l^5 + l^3C_1 + l^3C_2 + l^2pC_3.$$

Por lo tanto, la complejidad de *Analizador* es de $O(l^5)$.

3.2.3. Discusión de la Complejidad

Muchos de los problemas intratables en grafos e hipergrafos se pueden resolver de manera eficiente sobre clases de instancias con ancho hiperarbóreo acotado [26],[37].

Hay un resultado que dice que todos los hipergrafos cuando tienen ancho hiperarbóreo acotado pertenecen a la clase de problemas FPT. Los hipergrafos enraizados, dirigidos y acíclicos tienen ancho hiperarbóreo acotado [40], [34] [41], [38] y [35].

De lo anterior se deduce que el problema de membresía en las HRG, que se ha resuelto, pertenece a la clase FPT, ya que las HRG se han restringido a hipergrafos con ancho hiperarbóreo acotado [41], [38] y [35]. No obstante, este tipo de HRG restringidas son suficientes para modelar los hipergrafos que se obtienen en la AMR, las cuales han sido la motivación de este trabajo de tesis.

Capítulo 4

Conclusiones

Se revisaron los trabajos relacionados con el problema de membresía en HRG, con el propósito de identificar los métodos propuestos para resolver dicho problema. Para este algoritmo que se propone se tuvo que utilizar una definición alternativa de matriz de adyacencias, la cual es una generalización para la matriz de adyacencias de un grafo. En contraste con otras propuestas para el problema de membresía, este concepto alternativo de matriz de adyacencias permite abordar el problema a través de un enfoque algebraico conjuntista.

Posteriormente, se propuso el algoritmo *Analizador* para resolver el problema de membresía en HRG, el cual lleva acabo el análisis directamente en la Matriz de Adyacencias correspondiente al hipergrafo de entrada H a diferencia de otros autores, los cuales proponen llevar a cabo el análisis directamente en el hipergrafo H.

Finalmente, se presentó de forma explícita el análisis de la complejidad para el algoritmo propuesto, la cual es polinomial del orden $O(l^5)$, donde l es el número de vértices del hipergrafo de entrada. Además, se pudo concluir que el algoritmo propuesto es eficiente y correcto, para lo cual se desarrolló la demostración de su corrección.

4.1. Contribuciones

Las contribuciones de este trabajo de investigación son:

1. Se ha presentado el algoritmo *Analizador* para resolver el problema de membresía en HRG.

2. Se ha presentado el análisis de la complejidad para el algoritmo propuesto, la cual es polinomial del orden $O(l^5)$, donde l es el número de vértices del hipergrafo de entrada. Además este análisis se ha presentado de manera explícita a diferencia de las presentadas por otros autores.

3. Se ha definido el concepto de matriz de adyacencias, el cual es una generalización de la matriz de adyacencias de un grafo, correspondiente al hipergrafo de entrada H.

4. Es importante destacar que se ha demostrado la corrección del algoritmo *Analizador*, lo cual contrasta con las propuestas algorítmicas de otros autores.

4.2. Trabajo a Futuro

1. Se realizará la implementación del algoritmo *Analizador* propuesto.

2. Se analizará la alternativa de aplicar el algoritmo propuesto a otras áreas y no solo a PLN para resolver problemas que pueden ser modelados a través de la Representación de Significado Abstracto. Áreas como Teoría de Juegos, Bases de Datos e Inteligencia Artificial, entre otras.

Bibliografía

[1] Lautemann, C.: The complexity of graph languages generated by hyperedge replacement. Acta Informatica **27**(5) (Apr 1990) 399–421

[2] Ehrig, H., Engels, G., Kreowski, H.J., Rozenberg, G., eds.: Handbook of Graph Grammars and Computing by Graph Transformation: Vol. 2: Applications, Languages, and Tools. World Scientific Publishing Co., Inc., River Edge, NJ, USA (1999)

[3] Drewes, F., Kreowski, H.J., Habel, A.: Handbook of graph grammars and computing by graph transformation. World Scientific Publishing Co., Inc., River Edge, NJ, USA (1997) 95–162

[4] Bauderon, M., Courcellel, B.: Graph expressions and graph rewritings. Mathematical systems theory **20**(1) (1987) 83–127

[5] Habel, A., Kreowski, H.J.: May we introduce to you: Hyperedge replacement. In Ehrig, H., Nagl, M., Rozenberg, G., Rosenfeld, A., eds.: Graph-Grammars and Their Application to Computer Science, Berlin, Heidelberg, Springer Berlin Heidelberg (1987) 15–26

[6] Peng, X., Song, L., Gildea, D.: A synchronous hyperedge replacement grammar based approach for amr parsing. In: Proceedings of the Nineteenth Conference on Computational Natural Language Learning, Association for Computational Linguistics (2015) 32–41

[7] Peuser, C.: From hyperedge replacement grammars to decidable hyperedge replacement games. In Mazzara, M., Ober, I., Salaün, G., eds.: Software Technologies: Applications and Foundations - STAF 2018 Collocated Workshops, Toulouse, France, June 25-29, 2018, Revised Selected Papers. Volume 11176 of Lecture Notes in Computer Science., Springer (2018) 463–478

[8] Rozenberg, G.: Handbook of graph grammars and computing by graph transformation. (01 1997)

[9] Bauer, D., Rambow, O.: Hyperedge replacement and nonprojective dependency structures. In Chiang, D., Koller, A., eds.: Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+12), June 29 - July 1, 2016, Heinrich Heine University, Düsseldorf, Germany, The Association for Computer Linguistics (2016) 103–111

[10] Gallo, G., Scutellà, M.: Directed hypergraphs as a modelling paradigm. Decisions in Economics and Finance **21** (02 1998) 97–123

[11] Björklund, H., Drewes, F., Ericson, P., Starke, F.: Uniform parsing for hyperedge replacement grammars. Journal of Computer and System Sciences **118** (2021) 1–27

[12] Aalbersberg, I., Rozenberg, G., Ehrenfeucht, A.: On the membership problem for regular dnlc grammars. Discrete Applied Mathematics **13**(1) (1986) 79–85

[13] Lange, K.J., Welzl, E.: String grammars with disconnecting or a basic root of the difficulty in graph grammar parsing. Discrete Applied Mathematics **16**(1) (1987) 17–30

[14] Jones, B., Andreas, J., Bauer, D., Hermann, K.M., Knight, K.: Semantics-based machine translation with hyperedge replacement grammars. In: Proceedings of COLING 2012, The COLING 2012 Organizing Committee (2012) 1359–1376

[15] Chiang, D., Andreas, J., Bauer, D., Hermann, K.M., Jones, B., Knight, K.: Parsing graphs with hyperedge replacement grammars. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics (2013) 924–932

[16] Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., Schneider, N.: Abstract meaning representation for sembanking. In: Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, Association for Computational Linguistics (2013) 178–186

[17] Gilroy, S., Lopez, A., Maneth, S.: Parsing graphs with regular graph grammars. In: Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017), Vancouver, Canada, Association for Computational Linguistics (August 2017) 199–208

[18] Courcelle, B.: The monadic second-order logic of graphs V: on closing the gap between definability and recognizability. Theor. Comput. Sci. **80**(2) (1991) 153–202

[19] Drewes, F.: Dag automata for meaning representation. In: Proceedings of the 15th Meeting on the Mathematics of Language, Association for Computational Linguistics (2017) 88–99

[20] Björklund, H., Drewes, F., Ericson, P.: Between a rock and a hard place uniform parsing for hyperedge replacement dag grammars. (02 2016) 521–532

[21] Björklund, H., Drewes, F., Ericson, P., Starke, F.: Uniform parsing for hyperedge replacement grammars. Journal of Computer and System Sciences **118** (11 2020)

[22] Björklund, H., Drewes, F., Ericson, P.: Parsing weighted order-preserving hyperedge replacement grammars. In: Proceedings of the 16th Meeting on the Mathematics of Language, Toronto, Canada, Association for Computational Linguistics (July 2019) 1–11

[23] Habel: Hyperedge Replacement: Grammars and Languages. 1st edn. University of Bremen, United States of America (1991)

[24] Rozenberg, G., Welzl, E.: Boundary nlc graph grammars basic definitions, normal forms, and complexity. Information and Control **69** (04 1986) 136–167

[25] Engelfriet, J.: Handbook of formal languages, vol. 3. Springer-Verlag New York, Inc., New York, NY, USA (1997) 125–213

[26] Gottlob, G., Grohe, M., Musliu, N., Samer, M., Scarcello, F.: Hypertree decompositions: Structure, algorithms, and applications. In Kratsch, D., ed.: Graph-Theoretic Concepts in Computer Science, Berlin, Heidelberg, Springer Berlin Heidelberg (2005) 1–15

[27] Groschwitz, J., Koller, A., Teichmann, C.: Graph parsing with s-graph grammars. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, The Association for Computer Linguistics (2015) 1481–1490

[28] Aguiñaga, S., Chiang, D., Weninger, T.: Learning hyperedge replacement grammars for graph generation. CoRR **abs/1802.08068** (2018)

[29] Minas, M.: Hypergraphs as a uniform diagram representation model. In: Selected Papers from the 6th International Workshop on Theory and Application of Graph Transformations. TAGT'98, London, UK, UK, Springer-Verlag (2000) 281–295

[30] Downey, R.G., Fellows: Parameterized computational feasibility, in feasible mathematics. **13** (1995) 219–244

[31] Flum, J., Grohe, M.: Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series). Springer-Verlag, Berlin, Heidelberg (2006)

[32] Pilipczuk, M.: Tournaments and Optimality: New Results in Parameterized Complexity. PhD thesis, Univessity of Bergen Norway (August 2013)

[33] Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and Its Applications. OUP Oxford (2006)

[34] Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. 1st edn. Springer Publishing Company, Incorporated (2015)

[35] Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theor. Comput. Sci. **209**(1-2) (December 1998) 1–45

[36] Gildea, D.: Grammar factorization by tree decomposition. Comput. Linguist. **37**(1) (March 2011) 231–248

[37] Bodlaender, H.L.: Treewidth: Algorithmic techniques and results. In Prívara, I., Ružička, P., eds.: Mathematical Foundations of Computer Science 1997, Berlin, Heidelberg, Springer Berlin Heidelberg (1997) 19–36

[38] Courcelle, P.B., Engelfriet, D.J.: Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach. 1st edn. Cambridge University Press, New York, NY, USA (2012)

[39] Gottlob, G., Miklós, Z., Schwentick, T.: Generalized hypertree decompositions: Np-hardness and tractable variants. J. ACM **56**(6) (September 2009)

[40] Adler, I., Gottlob, G., Grohe, M.: Hypertree width and related hypergraph invariants. Eur. J. Comb. **28**(8) (2007) 2167–2181

[41] Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness i: Basic results. SIAM J. Comput. **24**(4) (August 1995) 873–921