Hacia un modelo computacional unificado del lenguaje natural

Towards an unified computational model of natural language

Benjamín Ramírez González
Estudiante de doctorado UCM
benjaminramirezg@gmail.com

Resumen

¿Qué tipo de formalismo debe utilizarse para representar el lenguaje natural? Es necesario un formalismo capaz de describir adecuadamente todas las secuencias de las lenguas naturales. Pero, además, en la medida de lo posible, debe ser un formalismo sencillo, de un coste computacional reducido. Esta pregunta ha generado mucha controversia entre las principales escuelas generativas: la Gramática Transformacional y las Gramáticas de Unificación. En este artículo se defiende que, pese a las diferencias existentes, en última instancia, tales escuelas formalizan el lenguaje humano mediante un mismo tipo de formalismo bien definido: lo que Noam Chomsky llamó lenguaje independiente del contexto. Bajo el prisma de este artículo, la Lingüística actual está en condiciones de ofrecer un modelo computacional unificado del lenguaje natural.

Palabras clave

Jerarquía de Chomsky, lenguajes regulares, lenguajes dependientes del contexto, lenguajes independientes del contexto, coste computacional, complejidad computacional, lenguaje natural, formalismo, gramática generativa, Gramática Generativa Transformacional, Gramáticas de Unificación, HPSG, estructuras de rasgos, estructura compartida.

Abstract

What formalism should be used in order to formalize natural language? That formalism must be able to describe all sequences of natural languages in a right way. Moreover, as long as possible, that formalism must be simple, with a reduced computational cost. This question has triggered a great controversy among the main branches of generative Linguistics: Transformational Grammar and Unification Grammars. The claim of this paper is that, despite discrepancies, these linguistic models formalize natural language by means of the same formal language. This is a well-defined formal language: the context-sensitive language of Noam Chomsky's hierarchy. So, from the point of view of this paper, nowadays, Linguistics can offer an unified computational model of natural language.

Keywords

Chomsky's Hierarchy, regular languages, context-sensitive languages, context-free languages, computational cost, computational complexity, natural language, formalism, generative grammar, Transformational Grammar, Unification Grammars, HPSG, feature structures, feature sharing.

1 La Jerarquía de Chomsky

Toda ciencia utiliza un lenguaje formal para representar los fenómenos que estudia. En Lingüística y en Procesamiento del Lenguaje Natural, la decisión de qué tipo de lenguaje formal utilizar para modelar las secuencias de las lenguas humanas es una cuestión de suma importancia. La Lingüística es una ciencia empírica que se pregunta por la naturaleza del lenguaje humano como fenómeno natural que está representado de forma concreta (mediante una formalización determinada) en el cerebro[1]. Por su parte, en Procesamiento del Lenguaje Natural es crucial el tipo de formalización mediante el cual se codifican las secuencias de una lengua: un formalismo sencillo puede conducir a la elaboración de herramientas eficientes, mientras que un formalismo más complejo puede comprometer tal eficiencia.

Un lenguaje formal Λ es un conjunto (quizá infinito) de secuencias σ. Para formar estas secuencias se utiliza un vocabulario Υ: un conjunto finito de símbolos atómicos υ. La gramática Γ de Λ es la definición de qué concatenaciones de elementos υ son posibles en las secuencias de Λ, es decir Γ define Λ.

Es comúnmente aceptado en Lingüística y en Computación que existen cuatro clases fundamentales de lenguajes formales, con distinto nivel de complejidad. Son los cuatro tipos que estableció Noam Chomsky en la llamada Jerarquía de Chomsky[2]: lenguajes regulares, lenguajes inde-

[1] Véase (Chomsky, 2003), p. 106.

[2] (Chomsky, 1956). Véanse también (Chomsky, 1957),

pendientes del contexto, lenguajes dependientes del contexto y lenguajes recursivamente enumerables. Estos tipos de lenguajes están ordenados en orden creciente de complejidad y cada uno de ellos engloba a todos los lenguajes de complejidad menor. Chomsky estableció para cada uno de estos cuatro tipos de lenguaje un tipo de gramática correspondiente: el tipo de gramática necesaria para generar (definir) tal lenguaje. Así pues, los lenguajes regulares se pueden generar mediante gramáticas regulares, los lenguajes independientes del contexto pueden generarse mediante gramáticas independientes del contexto, etc.

El objetivo de Chomsky al definir su jerarquía era decidir qué tipo de lenguaje formal es el más adecuado para representar las secuencias de las lenguas naturales. Parece adecuado elegir el tipo más sencillo (el de menor coste computacional) de entre aquellos capaces de representar adecuadamente tales secuancias.

De forma paralela al trabajo de Chomsky, en Teoría de la Computación se definieron una serie de máquinas abstractas capaces de definir distintos tipos de lenguaje[3]. Se ha demostrado que estos tipos de lenguaje son, precisamente, los que propuso Chosmky en su jerarquía. Es decir, que estas máquinas abstractas y las gramáticas de la jerarquía de Chomsky son descripciones equivalentes de los mismos tipos de lenguaje. Los lenguajes regulares se describieron en términos de autómatas finitos, los lenguajes independientes del contexto se describieron en términos de autómatas a pila y los dependientes del contexto como autómatas acotados linealmente. Además, los lenguajes no recursivos se pueden describir mediante máquinas de Turing.

Estas máquinas abstractas son algoritmos capaces de decidir en un número finito de pasos, y con el uso de una determinada cantidad de memoria, si una secuencia forma o no parte de un lenguaje. Es por esa naturaleza algorítmica por lo que, en este artículo, se ha decidido representar los lenguajes formales por medio de este tipo de máquinas abstractas. Se espera que, con esta representación resulte fácil entender cómo el coste computacional de usar un tipo de lenguaje (un tipo de máquina) es sensiblemente mayor del de usar otro lenguaje más sencillo.

A continuación se verá en qué consisten los tres primeros tipos de lenguaje de la jerarquía, y cuál es la naturaleza y coste computacional de las gramáticas necesarias para definirlos[4].

(Sánchez León, 2006) y (Serrano, 1975).

[3]Se sigue aquí a (Hopcroft, Motwani, y Ullman, 2001) y (Aranda et al., 2006).

[4]El cuarto tipo de la jerarquía, el más complejo, se obvia en adelante, pues queda fuera del interés de este artículo. El lector interesado en tal tipo de lenguajes puede consultar (Hopcroft, Motwani, y Ullman, 2001).

1.1 Lenguajes regulares

Los lenguajes regulares son aquellos cuyas secuencias σ son una mera concatenación lineal de elementos υ ($\sigma = \upsilon_1, \upsilon_2 ... \upsilon_n$). Podría decirse que, en los lenguajes regulares, cada σ es un objeto unidimensional, en el cual cada constituyente υ_i solo se relaciona con el resto de constituyentes de la secuencia en términos de su posición lineal relativa. En concreto, los lenguajes regulares solo admiten secuencias σ de tipo υ^n, consistentes en la concatenación de n elementos υ. Esta caracterización de las secuencias contrastará en los apartados sucesivos con la propia de las secuencias de otros lenguajes más complicados. Las gramáticas necesarias para generar lenguajes regulares tienen un coste computacional muy bajo, pues no requieren memoria alguna, salvo la necesaria para determinar el momento exacto del proceso de generación en el que se encuentran.

Un lenguaje es regular si existe un autómata de estados finitos capaz de generarlo. Un autómata de estados finitos consta de un conjunto Q de estados, uno de los cuales es el estado de inicio i. Se define también un subconjunto Φ de Q con los estados finales del autómata. Además, el autómata cuenta con un conjunto Σ de símbolos de entrada, el vocabulario a partir del cual se forman las secuencias del lenguaje que se define. Por último, el autómata cuenta con una función δ. Esta función toma un estado de Q y un símbolo de Σ y devuelve un estado de Q.

Por ejemplo, imagínese un autómata de estados finitos Γ_r que define el lenguaje Λ_r, donde Λ_r solo consta de una secuencia: la oración del español *Juan visitó Madrid*. La definición de Γ podría ser la siguiente: $Q = \{q^0, q^1, q^2, q^3\}$, $i = q^0$, $\Phi = \{q^3\}$ y $\Sigma = \{Juan, Madrid, visitó\}$. Además, δ sería la función de transición representada en la figura 1 en forma de grafo.

Figura 1: Función de transición δ en Γ_r

El funcionamiento de este autómata Γ_r en labores de procesamiento podría ser el siguiente. A Γ_r se le pasa como *input* una cadena de símbolos de Σ, por ejemplo, la cadena *Juan visitó Madrid*. La labor de Γ_r es determinar si tal cadena es una

secuencia del lenguaje Λ_r. Γ_r se encuentra en su estado de inicio q^0 y toma el primer símbolo de la cadena de entrada: *Juan*. De acuerdo con δ (figura 1), desde q^0, dada la aparición de *Juan* en la cadena de entrada, Γ_r pasa al estado q^1. A continuación, ya desde q^1, se analiza el siguiente símbolo de la cadena de entrada: *visitó*. La función δ establece que, desde q^1, Γ_r pasa a q^2 al recibir *visitó*. Del mismo modo, Γ_r pasará de q^2 a q^3 al recibir el símbolo siguiente de la cadena de entrada: *Madrid*. Se considera que el proceso termina cuando la cadena de entrada se ha consumido, o cuando δ no define transición alguna desde el estado en que se encuentra para el símbolo de la cadena de entrada que se le ha pasado. Se considera que la secuencia analizada es parte del lenguaje que define Γ_r si, cuando el proceso termina, Γ_r está en un estado final (q^3 en este caso) y la cadena de entrada se ha consumido.

Este tipo de gramática tiene un coste computacional bajo, pues solo hace una lectura secuencial de la cadena de entrada. Los pasos del proceso de análisis son tantos como símbolos haya en la cadena de entrada. Pero, ¿es este tipo de análisis adecuado para las secuencias de las lenguas naturales? No lo es. Las gramáticas regulares (los autómatas de estados finitos) no son capaces de dar cuenta del hecho de que las secuencias de las lenguas naturales obedecen a una estructura sintagmática. Las estructuras sintagmáticas son objetos bidimensionales, ordenados en una dimensión lineal y en otra de dependencia estructural. De acuerdo con ello, la oración *Juan visitó Madrid* no es solo la concatenación de palabras de la figura 1, sino que tales palabras obedecen a una estructura que pudiera representarse mediante el sistema de corchetes del ejemplo (1).

(1) $[_O\ [_{SN}\text{Juan }]\ [_{SV}\text{visitó }[_{SN}\text{Madrid }]]]$

Las gramáticas regulares (los autómatas de estados finitos) no son capaces de dar cuenta de este tipo de estructuras. Obsérvese que los corchetes del ejemplo (1) no son una mera concatenación de signos [y], sino que tales signos obedecen a ciertas reglas gramaticales. En concreto, por cada signo [de la secuencia debe haber un signo] correspondiente. Una gramática regular no tiene la memoria necesaria para generar este tipo de secuencias: en su lectura secuencial de la cadena de entrada, no tiene forma de recordar en un momento dado del proceso generativo cuántos signos [han aparecido en la secuencia para generar solo el mismo número de elementos].

Por tanto, cabe concluir que las lenguas naturales no pueden ser formalizadas adecuadamente mediante lenguajes regulares.

1.2 Lenguajes independientes del contexto

El segundo tipo de lenguajes formales que estableció Chomsky en su jerarquía es el de los llamados lenguajes independientes del contexto. Son lenguajes cuyas secuencias se pueden entender como objetos bidimensionales. Si las secuencias de los lenguajes regulares eran de tipo υ^n, las de los lenguajes independientes del contexto son de tipo ${\upsilon_a}^n{\upsilon_b}^n$. Es decir, en las secuencias de los lenguajes independientes del contexto los elementos υ no solo están concatenados en una dimensión lineal, sino que pueden estar agrupados en subsecuencias entre las que se pueden establecer determinadas relaciones. Por ejemplo, en una secuencia de tipo ${\upsilon_a}^n{\upsilon_b}^n$, puede establecerse que el número n de elementos υ de tipo a debe ser idéntico al número de elementos b que aparece a continuación. En definitiva, estas secuencias son concatenaciones lineales de elementos υ (primera dimensión), pero además, entre estos elementos υ se pueden establecer relaciones estructurales locales (varios elementos υ adyacentes en la dimensión lineal pueden entenderse como constituyentes de una subsecuencia ϕ de σ).

Este tipo de lenguajes sí pueden caracterizar adecuadamente la naturaleza sintagmática de las secuencias de las lenguas naturales. Pero las gramáticas necesarias para generar lenguajes independientes del contexto son más complejas que las gramáticas regulares. Una gramática independiente del contexto podría formalizarse como un autómata de estados finitos aumentado con una pila.

Un autómata a pila consta del mismo conjunto de estados Q que un autómata de estados finitos, con su estado de inicio i, con un subconjunto Φ de estados finales, el vocabulario Σ que determina las palabras posibles de las secuencias del lenguaje que se define, y la conocida función de transición δ. Pero además, el autómata a pila cuenta con un vocabulario de pila Υ. La función δ de un autómata a pila toma tres argumentos: un estado de Q, un símbolo de Σ y un símbolo de pila Υ. El resultado de la función es un par formado por un estado de Q y un nuevo símbolo de pila de Υ. En definitiva, una pila es una memoria que se va alterando según el autómata hace transiciones. Es un modo sencillo de recordar en un momento dado del análisis cuántos elementos de un determinado tipo se han visto ya en la cadena de entrada.

Por ejemplo, imagínese un autómata a pila Γ_{ic} que define un lenguaje Λ_{ic}, donde Λ_{ic} es un con-

junto de secuencias de corchetes [ ]. En una secuencia σ de Λ_{ic}, por cada $[_i$ que se abre, debe aparecer después un $]_i$ que lo cierre. Tal autómata Γ_{ic} podría obedecer a lo siguiente: $Q = \{q^0, q^1\}$, $i = q^0$, $\Phi = \{q^1\}$, $\Sigma = \{[,]\}$ y $\Upsilon = \{p\}$. Además, δ sería la función de transición representada en la figura 2 en forma de grafo.

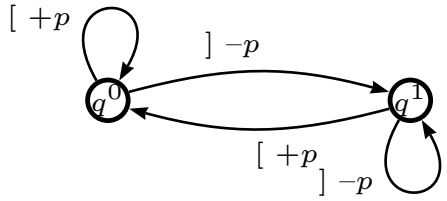

Figura 2: Función de transición δ en Γ_{ic}

Esta función δ establece que, cada vez que se encuentra un símbolo [en la cadena de entrada, se añade un símbolo p a la pila (véase $+p$), y cada vez que se encuentra], se elimina un símbolo p de la pila. Las pilas son estructuras de almacenamiento de datos en las cuales el último elemento que entra es el primero en salir (de ahí su nombre). En este caso, la pila es una memoria que recuerda en cada momento cuántos elementos [están abiertos, a fin de saber cuántos elementos] correspondientes se deben añadir. De este modo Γ_{ic} establece que las secuencias deben tener el mismo número de símbolos de apertura que de cierre. Imagínese que los símbolos [,] y p tienen un índice, de modo que cada símbolo $[_i$ debe cerrarse por el correspondiente $]_i$. Además, por cada $[_i$ se añade a la pila un p_i, y la lectura de p_i en la pila es la única que admite la aparición del correspondiente corchete de cierre $]_i$. Esto permitiría establecer que los pares de corchetes de las secuencias deben anidarse y no cruzarse. Un autómata de este tipo permite analizar las estructuras sintagmáticas de las secuencias de las lenguas naturales, como la representada en el ejemplo (1), que se repite aquí en (2).

(2) $[_O\ [_{SN}$Juan] $[_{SV}$visitó $[_{SN}$Madrid]]]

Obviamente, la máquina utilizada para hacer un análisis adecuado de la estructura sintagmática de (2) sería mucho más compleja que Γ_{ic}. Pero la cuestión crucial es que tal autómata no puede ser un mero autómata de estados finitos, sino que deberá contar, al menos, con una memoria equivalente a la pila de Γ_{ic}. Como parece obvio, un autómata a pila tiene, en principio, un coste computacional añadido sobre el propio de un autómata de estados finitos (un coste de memoria: la pila). No obstante, es importante señalar que los autómatas a pila, como ocurriera con los de estados finitos, solo hacen una lectura secuencial de la cadena de entrada. Es cierto que cada paso que efectúan es más costoso que el correspondiente paso en un autómata de estados finitos, pero el número de pasos no es mayor: un autómata a pila ejecuta tantos pasos de análisis como elementos tenga la cadena de entrada.

1.3 Lenguajes dependientes del contexto

Se ha visto que un autómata a pila sí es capaz de modelar estructuras sintagmáticas. Pero hay ciertos fenómenos sintácticos de las lenguas naturales que este tipo de lenguajes no pueden modelar: el movimiento y el ligamiento. Véanse los siguientes ejemplos del español. En (3) el pronombre *lo* está ligado por el antecedente *Juan*: *lo* y *Juan* se refieren, por ello, al mismo individuo. De un modo similar, en (4), el pronombre interrogativo *Qué* y la huella *h* están vinculados: gracias a ello, *Qué* se linealiza en la margen izquierda de la oración, pero conserva su valor de objeto del verbo *descubrirán*.

(3) [Juan$_i$ [cree [que [lo$_i$ descubrirán]]]]

(4) [Qué$_i$ [crees [que [descubrirán [h$_i$]]]]]

Estas relaciones i de (3) y (4) —ligamiento y movimiento— van más allá de la complejidad propia de una estructura sintagmática: son relaciones que se establecen, dentro de una estructura sintagmática, entre elementos alejados tanto en la dimensión lineal como en la jerárquica[5]. Se suele decir de estos fenómenos que son no locales.

Por tanto, las secuencias de (3) y (4), con sus identidades i, no son meras estructuras sintagmáticas, no son meros objetos bidimensionales. Si se quisiera representar en términos espaciales que el índice i de *Juan* y el de *lo* es el mismo objeto, sería necesaria una tercera dimensión[6].

[5] La dimensión que forma la jerarquía de constituyentes sintagmáticos.

[6] Imagínese una secuencia unidimensional σ formada por la concatenación de n elementos de tipo υ ($\sigma = \upsilon_1, ..., \upsilon_n$). En tal secuencia unidimensional, el elemento υ_i solo puede ocupar una posición: la posición i. Pero imagínese que σ se concibe como una de las dos coordenadas (dimensiones) que permiten definir el plano ϕ. Junto a $\sigma = \upsilon_1, ..., \upsilon_n$ existe entonces $\sigma' = \upsilon'_1, ..., \upsilon'_n$, y cualquier punto del plano ϕ se define como $\phi(\upsilon_i, \upsilon'_j)$. Entonces, en este nuevo objeto bidimensional ϕ, al elemento υ_i de σ (que en σ, objeto unidimensional, solo ocupaba la posición i) le corresponden n posiciones distintas: las posiciones $\phi(\upsilon_i, \upsilon'_1), ..., \phi(\upsilon_i, \upsilon'_n)$. Del mismo modo, el punto $\phi(\upsilon_i, \upsilon'_j)$ solo cuenta con una posición en el plano ϕ. Pero si ϕ forma parte de la definición del cubo Υ (un objeto tridimensional), de nuevo, en Υ, a $\phi(\upsilon_i, \upsilon'_j)$ le corresponden n posiciones: $\Upsilon(\upsilon_i, \upsilon'_j, \upsilon''_1), ..., \phi(\upsilon_i, \upsilon'_j, \upsilon''_n)$. Recuérdese que las estructuras sintagmáticas de (3) y (4) son objetos bidimensionales. En tales planos bidimensionales no es posible que un punto —el índice i— ocupe dos posiciones distintas. Por tanto, si el índice i ha de concebirse como un

Precisamente, si los lenguajes regulares definían secuencias unidimensionales y los independientes del contexto secuencias bidimensionales, los lenguajes independientes del contexto permiten formalizar las secuencias como objetos tridimensionales (multidimensionales, en realidad). Es decir, estos lenguajes pueden estar formados por secuencias del tipo ${v_a}^n {v_b}^n {v_c}^n$, donde es necesario coordinar tres dimensiones: el número de elementos de tipo v_a, de tipo v_b y de tipo v_c debe ser el mismo, debe ser n.

Para generar estos lenguajes, la Teoría de la Computación define un nuevo tipo de máquina abstracta: el autómata acotado linealmente. Estas máquinas cuentan, como las anteriores, con un conjunto Q de estados (con su estado de inicio i y sus estados finales Φ), un vocabulario Σ y una función de transición δ. No tienen una pila, pues, como se verá, su memoria supera con creces a la de la pila. Lo característico de los autómatas acotados linealmente es el modo en el que leen la cadena de entrada. Los autómatas anteriores hacían una mera lectura secuencial de esta. Los autómatas acotados linealmente, en cambio, pueden también retroceder en la lectura, volver a avanzar, etc. Incluso son capaces de modificar los símbolos de la cadena de entrada. Dicho en otros términos, la función δ de estos autómatas toma un estado de Q y un símbolo de Σ y devuelve tres elementos: un estado de Q, un símbolo de Σ y una instrucción $\leftarrow$ o $\rightarrow$. El estado de Q es el estado al que se desplaza el autómata como consecuencia de la transición. El símbolo de Σ es el símbolo con el que la transición rescribe la posición de la cadena de entrada que se acaba de leer. Y la instrucción $\leftarrow$ o $\rightarrow$ determina si el siguiente elemento de la cadena de entrada que se va a leer es el que se encuentra a la derecha del actual o el que se encuentra a su izquierda.

Sería muy complicado crear y explicar un autómata acotado linealmente capaz de generar las estructuras sintagmáticas de (3) y (4). Baste, entonces, con explicar su lógica. En los análisis de (3) y (4), *lo* y *h* deben compartir índice con *Juan* y *Qué*, respectivamente. De un modo u otro, la aparición de *lo* y *h* está legitimada por la existencia de un antecedente. Esta relación no puede regularse haciendo una mera lectura secuencial de la cadena de entrada. Es necesario un análisis que, cuando observa la aparición de *lo* o *h*, compruebe que estos tienen un antecedente legítimo. Para ello es necesaria una memoria más compleja que pila de los autómatas a pila. La memoria de los autómatas acotados linealmente es, precisamente, su capacidad de recordar la cadena de entrada y de volver sobre ella para hacer las comprobaciones oportunas. Por tanto, un hipotético autómata acotado linealmente Γ_{dc} que definiese el lenguaje Λ_{dc} formado por las dos estructuras de (3) y (4) podría leer secuencialmente la cadena de entrada y, al encontrar *lo* o *h*, volver sobre sus pasos para comprobar si existen los antecedentes *Juan* o *Qué*.

Este tipo de lenguaje formal, el lenguaje dependiente del contexto, sí es capaz de dar cuenta de esta clase de fenómenos sintácticos. No obstante, la complejidad de estos lenguajes es sensiblemente mayor a la propia de los lenguajes regulares y también a la de los lenguajes independientes del contexto. La observación crucial que permite hacer tal afirmación es el hecho de que las máquinas utilizadas para reconocer estos lenguajes (los autómatas acotados linealmente) permiten lecturas no secuenciales de la cadena de entrada. Con las máquinas anteriores, el tiempo de análisis de una secuencia de entrada de longitud n estaba siempre definido por una función lineal: la máquina empleaba siempre n pasos. En cambio, con un análisis no secuencial de las cadenas de entrada, el tiempo puede ser polinómico. Una máquina puede leer los n elementos de la secuencia de entrada m veces. Por tanto, los pasos del análisis son n^m.

2 La Gramática Transformacional

Se han visto en 1 ciertos fenómenos (estructura sintagmática, ligamiento y movimiento) que suponen un reto para la elaboración de formalizaciones adecuadas de las secuencias de las lenguas naturales. Se ha sugerido que el lenguaje formal necesario para modelar adecuadamente esos fenómenos debe ser lo que Chomsky llamó un lenguaje independiente del contexto. Este tipo de lenguajes formales tiene una cierta complejidad. Por tanto, su aplicación al lenguaje natural quizá comprometa la eficiencia computacional de las formalizaciones. Esta ha sido una preocupación central de ciertas escuelas lingüísticas generativas (Gramáticas de Unificación, entre otras) que han intentado desarrollar modelos más sencillos, más adecuados en términos de coste de procesamiento.

No obstante, la Gramática Generativa Transformacional que fundara Noam Chomsky en los años 50 del siglo XX[7] sí ha utilizado de forma sistemática formalizaciones de las lenguas naturales

único objeto, este debe definirse en una tercera dimensión. En ese objeto tridimensional resultante, i sí puede actuar como índice de dos elementos diferentes.

[7] (Chomsky, 1957)

que, de un modo u otro, consisten en lenguajes independientes del contexto. Para la Gramática Transformacional, las secuencias de las lenguas naturales son estructuras de constituyentes como las vistas en (3) y (4) —se repiten aquí en (5) y (6)—.

(5) [Juan$_i$ [cree [que [lo$_i$ descubrirán]]]]

(6) [Qué$_i$ [crees [que [descubrirán [h$_i$]]]]]

Es decir, en este modelo gramatical, tales estructuras no son solo la representación de la historia derivativa de las secuencias. Para el modelo transformacional, las estructuras de (5) y (6) son objetos sobre los que las reglas de la gramática operan. A las reglas gramaticales que trabajan sobre este tipo de estructura se les ha llamado transformaciones o reglas transformacionales. Por ejemplo, una transformación podría tomar como *input* una estructura como la de (7) y mover *Qué* desde la posición de base a la periferia izquierda, para dar lugar a (6).

(7) [Crees [que [descubrirán [qué]]]]

Como se ha visto en 1.3, este tipo de operación solo es posible para gramáticas equivalentes a autómatas acotados linealmente. Por tanto, el modelo transformacional entiende las lenguas naturales como lenguajes dependientes del contexto[8].

3 Gramáticas de Unificación: HPSG

Las Gramáticas de Unificación nacieron en los años 80, como solución a los problemas de procesamiento en tiempo real de los modelos gramaticales previos. Las más importantes son GPSG, LFG y HPSG. GPSG (*Generalized Phrase Structure Grammar*) fue desarrollado por Gazdar, Pullum y Sag —(Gazdar et al., 1985)—. Hoy se encuentra en desuso. En cambio, LFG (*Lexical Functional Grammar*) se encuentra en pleno desarrollo —(Kaplan, Ronald, y Bresnan, 1982), (Dalrymple, Lamping, y Saraswat, 1995) y (Bresnan, 2001)—. HPSG (*Head-Driven Phrase Structure Grammar*) es la evolución de GPSG, y será el modelo que se use como referencia a continuación —(Pollard y Sag, 1987), (Pollard y Sag, 1994) y (Sag, Wasow, y Bender, 2002)—.

A diferencia de lo que ocurre en la Gramática Transformacional, en HPSG, las estructuras de constituyentes (como las de (5) o (6)) no se contemplan propiamente como objetos gramaticales. Los sintagmas de tales estructuras se crean a partir de sus constituyentes inmediatos, pero la relación entre el sintagma creado y sus constituyentes no se recuerda: el sintagma no guarda la información de cuáles son los constituyentes que lo formaron. En este sentido, las estructuras del tipo de (5) o (6), en HPSG, no existen. No es posible, por tanto, para una gramática de tipo HPSG, tomar una estructura como la de (7) para efectuar directamente sobre ella el movimiento necesario para generar (6). No es posible, sencillamente, porque, en HPSG, la estructura de constituyentes de (7) no existe como objeto gramatical.

Esta limitación hace de HPSG un modelo gramatical más sencillo que la Gramática Transformacional, en términos computacionales. En principio, cabría pensar que este modelo es equivalente a una gramática independiente del contexto, pues solo permite analizar estructuras sintagmáticas. Entonces, ¿cómo se consige en HPSG modelar aquellos fenómenos cuya complejidad va más allá, como el ligamiento y el movimiento? Se hace mediante el uso generalizado de estructuras de rasgos.

En HPSG, toda realidad gramatical (las unidades léxicas, las reglas gramaticales, etc.) se formaliza como tipo τ consistente en una estructura de rasgos[9]. Una característica crucial de las estructuras de rasgos que usa HPSG es el hecho de que dos o más rasgos de una estructura pueden compartir valor. Es decir, el valor de tales rasgos es el mismo objeto. Si dos rasgos comparten valor, sus valores deben ser idénticos, y si el valor de uno se modifica, el del otro también ha de hacerlo de forma consecuente. Véase en la figura 3 un ejemplo de estructura de rasgos caracterizadora

[8] En realidad, (Peters y Ritchie, 1973) demostraron que una gramática transformacional, tal como se definía en sus primeras versiones —(Chomsky, 1957), (Chomsky, 1965)— generan lenguajes aún más complejos: lenguajes recursivamente enumerables, según la jerarquía de Chomsky. La demostración de estos autores estaba fundamentada en el hecho de que las transformaciones de esas primeras versiones del modelo eran capaces de eliminar elementos de una secuencia. Las formulaciones posteriores de lo que es una regla transformacional —(Chomsky, 1986)— eliminan esta posibilidad. Se puede pensar que la Gramática Transformacional, con esta nueva formulación, solo genera lenguajes dependientes del contexto. (Chomsky, 1965) da por sentado que la complejidad computacional del modelo transformacional es la propia de un lenguaje dependiente del contexto.

[9] Un tipo τ consiste en una estructura de rasgos: un conjunto de 0 o más rasgos ϕ. Por ejemplo, el tipo τ_i se puede definir como una estructura de rasgos con los rasgos ϕ_a y ϕ_b. Además, cada rasgo ϕ se concibe como un par atributo valor, donde el atributo es un identificador del rasgo y el valor debe ser uno de los tipos τ definidos en la gramática. Se entiende que los tipos τ_a y τ_b en τ_0, en tanto que tipos, son estructuras de rasgos que, a su vez, pueden tener también sus rasgos y tipos anidados. Véase (Shieber, 1986).

de un verbo transitivo. Como puede verse, la estructura recoge la categoría gramatical verb, los rasgos de persona, género y número y los dos argumentos propios de una verbo transitivo. Como se ve, el primer argumento del verbo y el verbo mismo comparten el valor de los rasgos de persona y número. Con ello se formaliza el hecho de que sujeto y verbo concuerdan en persona y número.

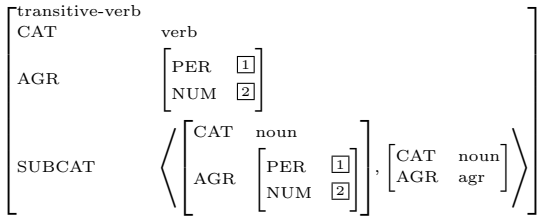

Figura 3: Estructura de rasgos en HPSG

Gracias a la estructura compartida, cada vez que una regla de la gramática crea un nuevo sintagma a partir de sus constituyentes inmediatos, en tal sintagma se puede recoger toda la información de dichos constituyentes que pudiera ser útil. Imagínese que el sintagma γ tiene la información i codificada en su estructura de rasgos. Tal sintagma γ se toma como constituyente de un nuevo sintagma β. Gracias a la estructura compartida, i puede pasar a la estructura de rasgos de β. De nuevo, si β se toma como constituyente inmediato de α, i puede pasar de β a α. A este proceso sucesivo de ascenso de información a lo largo de la estructura sintáctica se le llama habitualmente en la bibliografía de HPSG percolación de rasgos. Es esta percolación de rasgos la que permite establecer relaciones (como las propias del ligamiento y el movimiento) entre elementos alejados lineal y jeráquicamente en las secuencias de las lenguas naturales.

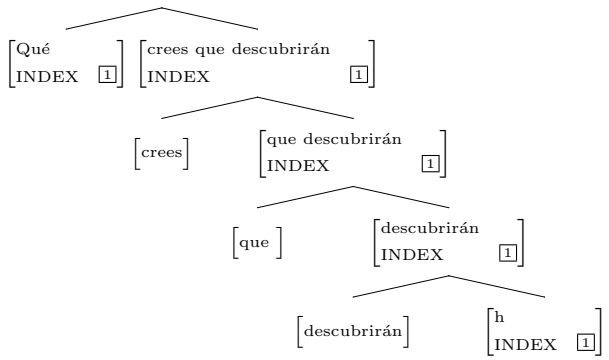

Figura 4: Percolación de rasgos en HPSG

Por ejemplo, en la figura 4 se representa el proceso derivativo que lleva a generar la oración de (6). La relación entre la huella *h* y *Qué* se establece mediante la percolación del índice 1. La unión del verbo *descubrirán* a una huella desencadena el ascenso del índice de la huella al sintagma resultante. Cada vez que se crea un sintagma a partir de un constituyente con este índice, el índice se recoge en dicho sintagma resultante. De este modo, *Qué* puede identificar su índice con el de su huella de forma local, pues este índice ha ascendido paso a paso hasta el sintagma al que *Qué* se une.

En resumen, la Gramática Transformacional contempla las estructuras de constituyentes como objetos gramaticales reales, sobre los que una regla puede operar: una regla puede mover elementos destro de la estructura, puede identificar elementos alejados lineal y jerárquicamente, etc. Obsérvese la semejanza entre este tipo de procesos y los propios de los autómatas acotados linealmente vistos en 1.3. La cadena de entrada de estos autómatas se recordaba, íntegra, durante toda la derivación, y toda ella era accesible en todo momento. El entender las estructuras sintagmáticas completas como *input* de las reglas gramaticales tiene una complejidad equivalente. Se concluye, entonces, como ya se ha dicho anteriormente, que la Gramática Transformacional formaliza las lenguas humanas como lenguajes dependientes del contexto.

En cambio, en HPSG y el resto de Gramáticas de Unificación, las estructuras de constituyentes no se contemplan como objetos gramaticales. Las reglas de estas gramáticas no pueden tomar una estructura de constituyentes y establecer directamente una relación no local entre distintos elementos de esta. Una gramática de este tipo es equivalente a un autómata a pila: la lectura de la cadena de entrada es secuencial, pues, en cada momento de la derivación, la gramática no tiene acceso a la estructura de constituyentes previamente derivada. Solo se necesita una memoria equivalente a una pila que permita regular la correcta apertura y cierre de constituyentes sintagmáticos.

Para establecer relaciones no locales, en HPSG la información relevante debe ser recogida sucesivamente en cada sintagma tal como se muestra en la figura 4. Este proceso de percolación de rasgos se ha formalizado mediante estructuras de rasgos donde (esto es crucial) dos o más rasgos pueden tomar por valor el mismo objeto (se dice que tales rasgos comparten estructura). En principio, se entiende que este modelo gramatical, dada esta restricción fundamental, es más económico en términos computacionales que la

Gramática Transformacional.

4 Hacia un modelo unificado

En este apartado se defiende que los modelos gramaticales anteriormente expuestos no son propuestas irreconciliables. Se asume que HPSG es un modelo más sencillo desde el punto de vista de coste computacional. Pero se defiende también que tal diferencia no es en realidad una diferencia fundamental, sino, más bien, una diferencia de grado.

Esta reflexión está fundamentada en dos observaciones. La primera consiste en que las estructuras de rasgos con estructura compartida son, en sí, objetos tridimensionales. Obsérvese que una estructura de constituyentes con relaciones no locales —(5) y (6) con su coindización i— es perfectamente equivalente a una estructura de rasgos donde unos rasgos se anidan dentro de otros y donde dos rasgos, quizá en niveles de anidación distintos, comparten valor. Por tanto, tanto las estructuras sintagmáticas con relaciones no locales como las estructuras de rasgos que usa HPSG son objetos tridimensionales, o secuencias propias de un lenguaje dependiente del contexto. Para manejar tales estructuras complejas serán necesarias gramáticas dependientes del contexto, equivalentes a autómatas acotados linealmente: gramáticas capaces de recordar y manipular la estructura completa en todo momento de la derivación.

De hecho, en las gramáticas computacionales basadas en HPSG, las relaciones de dependencia sintagmática se representan en forma de estructuras de rasgos[10], tal como se muestra en la figura 5, donde ARGS es el rasgo cuyo valor es la lista de constituyentes de un signo.

$$\left[\begin{array}{l}\textsc{s}\\ \textsc{args}\left\langle\left[\begin{array}{l}\textsc{np}\\ \textsc{args}\left\langle[\ldots],[\ldots]\right\rangle\end{array}\right],\left[\begin{array}{l}\textsc{vp}\\ \textsc{args}\left\langle[\ldots],[\ldots]\right\rangle\end{array}\right]\right\rangle\end{array}\right]$$

Figura 5: Dependencia sintagmática con estructuras de rasgos

Sería perfectamente posible representar las estructuras sintagmáticas de (5) y (6) del modo esbozado en esta figura 5, y marcar la coindización i de tales estructuras mediante una relación de estructura compartida. Asúmase, entonces, que es posible formalizar tanto una Gramática Transformacional como una Gramática de Unificación (HPSG) mediante este tipo de estructuras de rasgos con estructura compartida. Es fácil observar, entonces, que la única diferencia real entre estas gramáticas consiste en las restricciones que presentan sus reglas en cuanto a qué rasgos anidados tienen acceso. Si una Gramática Transformacional se caracteriza por recordar en cada momento toda la estructura previamente derivada, sus reglas tendrán acceso a todos los rasgos ARGS anidados en la estructura de rasgos. En cambio, las reglas de una gramática de tipo HPSG tienen vedado el acceso a la información de tales rasgos[11].

Obsérvese que, desde este punto de vista, la complejidad computacional de los dos modelos gramaticales no es esencialmente distinta. Ambos, de un modo u otro, tienen que gestionar objetos complejos, tridimensionales, para lo cual tendrán que utilizar gramáticas equivalentes, en ambos casos, a autómatas acotados linealmente, a gramáticas dependientes del contexto. Esta naturaleza dependiente del contexto estará en el corazón mismo del algoritmo de análisis —llámese *parser*— de las Gramáticas Transformacionales. Las gramáticas de tipo HPSG, por su parte, utilizarán *parsers* dotados de algún sistema que compruebe si las estructuras de rasgos de dos constituyentes son compatibles, y que calcule cuál ha de ser la estructura del sintagma resultante dadas las estructuras de sus constituyentes. Estos cálculos, dada la complejidad de las estructuras de rasgos con estructura compartida, los debe afrontar una gramática dependiente del contexto. En definitiva, en ambos casos, la complejidad del proceso alcanza el nivel propio de un lenguaje dependiente del contexto. Esto no implica que la diferencia de coste computacional de ambos modelos no sea significativa, pero sí se deduce de aquí que tal diferencia no es fundamental.

La segunda observación que sugiere la posibilidad de alcanzar un modelo computacional unificado para el lenguaje natural es la siguiente. Se observa en la Gramática Transformacional una preocupación por acotar el dominio de acción de las reglas de la gramática. Por ejemplo, en el modelo de fases de (Chomsky, 2008), la estructura sintagmática se divide en tramos a los que

[10] Véase (Bender, Flickinger, y Oepen, 2002). Cabe preguntarse por qué se representan las relaciones de dependencia sintagmática en gramáticas de tipo HPSG si los constructos formados por sintagma y constituyentes no se contemplan como objetos gramaticales. Se hace por motivos técnicos, como la representación de la ordenación lineal de los constituyentes de un sintagma. Esto no compromete el carácter independiente del contexto de la gramática, pues las reglas siguen sin tener acceso a la información de tales constituyentes anidados.

[11] Valga esta formulación personal de lo que definieron los fundadores de HPSG en (Pollard y Sag, 1987) como *Locality Principle*.

se llama fases. Mientras se está derivando una fase, las reglas de la gramática sí tienen acceso a todos los constituyentes anidados en esta. Pero una vez la fase se ha completado, las reglas no tienen acceso a sus constituyentes. Utilizando la formalización de la figura 5, en el modelo de fases, las reglas solo tienen acceso a la información de los rasgos ARGS anidados en la misma fase en la que la derivación opera en ese momento; pero no tendrán acceso a los rasgos ARGS más anidados, los pertenecientes a las fases anteriores. Esta limitación debe ir acompañada de un proceso de ascenso de información que recuerda a la percolación de rasgos de HPSG vista en 3.

En resumen, las estructuras de rasgos usadas en HPSG son objetos tridimensionales, secuencias propias de un lenguaje dependiente del contexto. Dada esta realidad, tanto la Gramática Transformacional como HPSG deben hacer frente a una complejidad computacional que, en esencia, es del mismo tipo. Llegada la discusión a este punto, surge la siguiente pregunta: dentro de la estructura sintagmática, ¿cuál debe ser el ámbito de acción de las reglas de la gramática? Las distintas escuelas dan distintas soluciones a esta pregunta. Pero, en favor de la sencillez computacional, todas pretender acotar tal ámbito.

5 Conclusión

Se ha discutido en los apartados previos cuál es el formalismo adecuado para dar cuenta de las lenguas naturales: cuál es el formalismo de menor coste computacional de entre los capaces de describir adecuadamente tales lenguas. Se han presentado los distintos tipos de lenguajes formales posibles y se ha discutido qué formalización han utilizado la Gramática Transformacional y las Gramaticas de Unificación (en concreto, HPSG). Se ha llegado a la conclusión de que las diferencias entre las formalizaciones de estas escuelas no son fundamentales. El panorama resultante es el siguiente. Tanto la Gramática Transformacional como las Gramáticas de Unificación formalizan las secuencias de las lenguas naturales mediante lenguajes dependientes del contexto. En este sentido, la Lingüística ha alcanzado un cierto consenso. Por razones de economía computacional, parece conveniente, no obstante, formalizar tales secuencias como objetos en los cuales las relaciones no locales se reduzcan, en la medida de lo posible, a procesos locales cíclicos. Es decir, parece deseable que, en cada paso local de creación de estructura, el sintagma resultante recoja, de un modo u otro, la información de sus constituyentes que, presumiblemente, pudiera ser necesaria en un momento posterior de la derivación. La alternativa sería que la derivación recordara en todo momento toda la estructura creada y pudiese acceder a ella, lo cual resulta de un coste computacional inasumible. HPSG opta por una interpretación radical de esta idea: todos los procesos gramaticales son locales, pues ninguna regla tiene acceso a los constituyentes del sintagma sobre el que opera. El modelo de fases de la Gramática Transformacional opta por una interpretación menos exigente: solo dentro de ciertos tramos estructurales (las fases), las reglas tienen acceso a los constituyentes anidados.

Bibliografía

Aranda, Joaquín, Natividad Duro, José Luis Fernández, José Jiménez, y Fernando Morilla. 2006. *Fundamentos de lógica matemática y computación*. Sanz y Torres.

Bender, Emily M., Dan Flickinger, y Stephan Oepen. 2002. The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precission grammars. CSLI, Stanford University.

Bresnan, J. 2001. *Lexical-Functional Syntax*. Oxford, Basil Blackwell.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions PGIT*, 2:113–124.

Chomsky, Noam. 1957. *Syntactic Structures*. Mouton and co., N.Y. publishers.

Chomsky, Noam. 1965. *Aspects of the Theory of Sintax*. The MIT Press, Cambridge, Massachusetts.

Chomsky, Noam. 1986. *Knowledge of Language: Its Nature, Origins and Use*. Praeger Publishers, N.Y., USA.

Chomsky, Noam. 2003. *Sobre la naturaleza y el lenguaje*. Cambridge University Press.

Chomsky, Noam. 2008. On phases. En *Foundational Issues in Linguistic Theory*. The MIT Press, Cambridge, Massachusetts, páginas 133–166.

Dalrymple, M., J. Lamping, y V. Saraswat. 1995. *Formal Issues in Lexical Functional Grammar*. CSLI Publications.

Gazdar, G., E. Klein, G. Pullum, y I. Sag. 1985. *Generalized Phrase Structure Grammar*. Harvard University Press.

Hopcroft, John E., Rajeev Motwani, y Jeffrey D. Ullman. 2001. *Introduction to Automata Theory, Languages and Computation*. Pearson Education.

Kaplan, Ronald, y Joan Bresnan. 1982. Lexical-functional grammar: a formal system for grammatical representation. En Joan Bresnan, editor, *The mental representation of grammatical relations*. Cambridge: The MIT Press, páginas 173–281.

Peters, P. Stanley y R. W. Ritchie. 1973. On the generative power of transformational grammar. *Information Science*, 6:49–83.

Pollard, Carl y Ivan A. Sag. 1987. *Information-Based Syntax and Semantics*. The University of Chicago Press.

Pollard, Carl y Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press.

Sag, Ivan A., Thomas Wasow, y Emily Bender. 2002. *Syntactic Theory: a Formal Introduction*. CSLI Publications.

Serrano, Sebastián. 1975. *Elementos de lingüística matemática*. Editarial Anagrama.

Shieber, Stuart M. 1986. *An Introduction to Unification Based Approaches to Grammar*. CSLI Publications.

Sánchez León, Fernando. 2006. Gramáticas y lenguajes formales. Departamento de Lingüística Computacional de la Real Academia Española.