1. Recapitulación

Como vimos la clase pasada, hay un tipo de producción que los lenguajes regulares no pueden describir. Veamos un ejemplo sacado de Wintner (2010).

Consideremos una lengua sobre el alfabeto Σ = {a,b} cuyos miembros son cadenas que consisten en un número n de as seguido del mismo número de bes

Por ejemplo, serán miembros de esta lengua cadenas como las de (1):

(1) {aabb, aaabbb, ab...}

Formalmente esta lengua se puede anotar como... (tildar la opción correspondiente)

- **1.** L = $\{a^n \cdot b^n | n>0\}$
- **2.** L = $\{a^n \cdot b^m | n>0\}$
- **3.** L = $\{a^* b^*\}$

Asumamos que L es un lenguaje regular y que, por lo tanto, existe un autómata de estado finito cuya lengua es L.

Ahora consideremos una lengua L_i = { $a^i | \ i > 0$ }.

Siguiendo lo que sabemos de L, podemos decir que todas las cadenas de L_i son prefijos de alguna cadena de L.

Entonces, para todo string en L_i tiene que haber un camino posible en el autómata de L que lleve del estado inicial q_0 al siguiente estado.

Si bien hay infinitas cadenas en L_i, hay finitos estados en el autómata, por definición. Por lo tanto tenemos al menos dos cadenas en L_i que nos van a llevar del mismo estado inicial al mismo estado de llegada q_1. Llamemos a estas cadenas a^j y a^k, donde $j \neq k$ (pongámosle, j=3, k=4, es decir "aaa" y "aaaa" respectivamente). El problema es que, si vemos el autómata resultante para L, partiendo de esta necesidad, podemos comprobar que ese autómata ya no es equivalente a L. ¿Por qué?

Porque este autómata reconoce las cadenas $a^j b^j$, que pertenece a L, pero también acepta $a^k b^j$, que no pertenece a L.

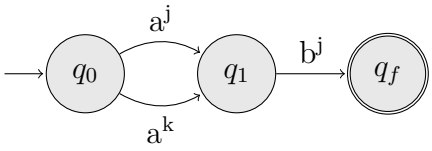


Figura 1: Autómata para L

Esta es una forma de demostrar que hay lenguajes que no son regulares y que hay una clase más compleja de lenguajes, lo que nos da pie a introducir los lenguajes independientes de contexto y las gramáticas independientes de contexto que los generan.

Vamos a revisar también los axiomas propios de estas gramáticas, que son los axiomas de dominancia y de precedencia.

Y por último, antes de verlos en acción, vamos a presentar los "parsers" que son programas que nos van a permitir aplicar las gramáticas independientes de contexto para producir el análisis sintáctico (árbol) de una oración.

2. Gramáticas independientes de contexto

2.1. Definición

Los lenguajes independientes de contexto son de tipo 2. Son lenguajes con mayor poder expresivo que los regulares, es decir, con mayores restricciones, lo que también implica mayor costo de procesamiento. Esto lo vimos la clase pasada cuando vimos la notación O y vimos que estos lenguajes se pueden procesar en tiempo polinómico.

Si bien este costo de procesamiento es menor al de los lenguajes dependientes de contexto y por eso estos lenguajes son "atractivos" para el procesamiento del lenguaje, se suele considerar a los lenguajes independientes de contexto como insuficientes para representar a las lenguas naturales.

Como ya vimos, una forma de describir los lenguajes formales es por medio de una gramática y una gramática formal está constituida por cuatro elementos.

(2) $G = <V_T, V_N , S, R>$

Para una gramática independiente de contexto que describa una lengua natural, podemos representar esa cuádrupla del siguiente modo:

(3) a. V_T = un vocabulario de símbolos terminales, que nos permitirá tener almacenadas y definidas las entradas léxicas.

b. V_N = un conjunto de símbolos no terminales, como SN, SV, etc., que nos permitirán agrupar las categorías en constituyentes y sintagmas hasta llegar a la categoría raíz.

c. S = un símbolo raíz de la categoría gramatical más compleja que se considere, normalmente oración (o S por *sentence*).

d. R = un conjunto de reglas de derivación.

Cuando usemos estas gramáticas con los programas parser, las vamos a describir directamente por medio de sus reglas. Las reglas nos van a describir la estructura interna de los constituyentes de una gramática.

La notación de estas reglas consiste en una secuencia compuesta por un símbolo no terminal y el signo → seguido por una secuencia de símbolos terminales o no terminales

¿Cómo serán las reglas para describir a L?

2.2. Derivación

El símbolo → establece una relación de derivación entre dos formas pertenecientes a la gramática. Y es la relación de derivación lo que permitirá definir el lenguaje que describe una gramática.

Veamos las reglas de una gramática simple.

(4) **Derivaciones de un lenguaje L_1**

a. O → SN SV

b. SN → Det N

c. SV → V

d. Det → el | la | los | las

e. N → perro | vaca | pollitos | gaviotas

f. V → ladra | muge | pían | graznan

¿Qué oraciones serán generadas por esta gramática?

2.3. Árboles

Los árboles de derivación presentan una ayuda para visualizar derivaciones de estas gramáticas y también los veremos como *output* de los parsers.

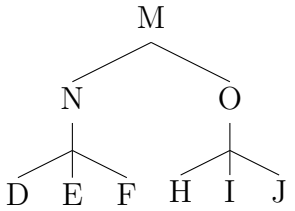
Los árboles son grafos y pueden ser descriptos de un modo formal dentro de la teoría matemática de grafos, que estudia las propiedades de estos objetos que representan relaciones binarias dentro de un conjunto. Un uso habitual es, por ejemplo, el estudio de las relaciones de individuos o cuentas en redes sociales.

Los grafos se describen por medio de dos conjuntos: el conjunto de vértices (nodos del árbol) y conjunto de aristas (ramas del árbol), que se definen como el par de nodos que cada arista conecta. Como veremos más adelante, hay un orden jerárquico en los árboles de derivación que organiza los pares que denominan las aristas, por lo cual, estos son pares ordenados.

El árbol tiene siempre un nodo raíz, único, desde el que parten las derivaciones y que para nosotros estará etiquetado con el símbolo O. Los nodos finales del árbol suelen aparecer etiquetados con los símbolos terminales, o entradas léxicas (V_T), de la gramática. En los nodos intermedios, vamos a encontrar los símbolos intermedios no terminales (V_N).

Las ramas del árbol representan las relaciones entre los nodos. Veamos un ejemplo:

(5)



En este grafo tenemos el conjunto de vértices enunciado en (6):

(6) {M, N, O, D, E, F, H, I, J}.

Y si bien no hemos usado flechas para dibujar debemos asumir que las ramas son flechas que viajan en una sola dirección: hacia abajo. Por esto el conjunto de aristas está formado por pares ordenados. El conjunto de pares ordenados para el árbol 5 se enuncia en (7):

(7) {<M,N>, <M,O>, <N,D>, <N,E>, <N,F>, <O,H>, <O,I>, <O,J>}.

Esto es así porque los árboles de derivación son grafos acíclicos dirigidos. Es decir, que las aristas tienen una dirección y, a su vez, no es posible formar un ciclo entre nodos.

¿Cuál es el nodo raíz?

¿Cuáles son los nodos intermedios?

¿Cuáles son los nodos terminales?

¿Cómo se denomina la arista que va de M a N?

2.4. Axiomas de dominancia y precedencia

Por lo visto hasta acá, podemos asumir que entre los nodos se establecen relaciones de arriba hacia abajo y, de un modo intuitivo, que también las relaciones se establecen de izquierda a derecha (ya que conocemos el orden de la secuencia de símbolos terminales). Estas relaciones se conocen como dominancia (D) y precedencia (P) sobre un conjunto N de nodos.

2.4.1. Dominancia

Informalmente, podemos entender dominancia como la relación que un nodo que se encuentra más arriba en el árbol tiene con un nodo de más abajo. En nuestro árbol 5, por ejemplo, M domina a N y a O. Y M domina también a D, E, F, H, I y J. En este sentido, la dominancia es también una relación de contención: M contiene a todos los demás nodos del árbol. En una oración, diríamos que la categoría superior, O, domina a todos los sintagmas de la oración.

Podemos, entonces, redefinir los símbolos de nuestra gramática:

- El nodo raíz es el nodo que domina a todos y que no es dominado por ningún otro (excepto sí mismo).
- Los nodos intermedios (no terminales) son los que dominan a otros (y a sí mismos).
- Los nodos terminales serán aquellos que no dominan a ningún nodo (excepto a sí mismos).

Matemáticamente, podemos formular los axiomas que describen formalmente la relación de dominancia (D).

- **A1.** D es reflexiva: $(\forall x \in N)\ [x \lhd^{\star} x]$.

Esto lo hemos visto al decir que los nodos, de todo tipo, se dominan a sí mismos y va a ser importante para excluir árboles de múltiples raíces y árboles donde la raíz sea dominada por otro nodo (Esto también implica algo que ya sabemos: los árboles derivacionales son acíclicos):

(8)

(9)

Para excluir árboles como el de 8, 9 necesitamos otro axioma:

- **A2.** Condición de raíz única:$(\exists x\ \forall y \in N)\ [x \lhd^\star y]$.

Este axioma excluye a los árboles con más de una raíz e indica que todo nodo debe tener un único nodo que lo domine. Para el nodo raíz esto es posible únicamente al ser D reflexiva.

¿Se les ocurre una frase del español que no cumpla este axioma?

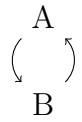
El siguiente axioma nos dice que:

- **A3.** D es transitivo:$(\forall xyz \in N)\ [((x \lhd^\star y)\ \&\ (y \lhd^\star z)) \rightarrow (x \lhd^\star z)]$.

Esto también lo hemos visto al decir que, en nuestro árbol, M no sólo dominaba a N y O, sino también a todos los demás nodos inferiores.

- **A4.** D es antisimétrico:$(\forall xy \in N)\ [((x \lhd^\star y)\ \&\ (y \lhd^\star x)) \rightarrow (x = y)]$.

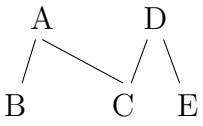
(10)



Este axioma nos indica también la propiedad dirigida y acíclica de los grafos que estamos analizando, ya que, si un nodo x domina a un nodo y, y dicho nodo y domina a su vez al nodo x, eso significa sí o sí que x e y son el mismo nodo (x = y), porque si no lo fueran, se rompería el axioma **A4.**.

- **A5.** Madre única:$(\forall xyz \in N)\ [((x \lhd^\star z)\ \&\ (y \lhd^\star z)) \rightarrow ((x \lhd^\star y) \vee (y \lhd^\star x))]$.

(11)

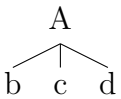


A5. nos permitirá excluir árboles como el de 8, ya que los nodos no pueden tener más de una madre o, en otros términos, un único nodo que lo *domina inmediatamente*, informalmente conocido como "madre", y del que, dicho nodo, es inmediatamente dominado (hijo). También se denominan nodos hermanos aquellos nodos inmediatamente dominados por el mismo nodo madre. La relación de dominancia inmediata es útil para referirse a relaciones locales entre nodos y es la que vemos reflejada en los pares ordenados de vértices que definen en teoría de grafos el conjunto de aristas. Simbolizaremos la relación de *dominancia inmediata* como $\lhd^+$.

Otra relación importante para nosotros es la de *dominancia exhaustiva* que es la que se da entre un conjunto de nodos terminales y un nodo que los domina a todos y que no domina a más nodos que los que forman parte de ese conjunto.

En el ejemplo 12, A *domina exhaustivamente* al conjunto de nodos {b, c, d} y no hay ningún otro nodo, llamémoslo x, que esté dominado por A y no es parte del conjunto.

(12)



Entonces, basandonos en los axiomas de dominancia y la relación de dominancia exhaustiva podemos definir qué es un constituyente:

Un constituyente es un conjunto de nodos *dominados exhaustivamente* por un único nodo.

2.4.2. Precedencia

Informalmente, la precedencia es lo que "se lee antes". En la oración anterior, "Informalmente" precede a "la", "la" a "precedencia"

y así sucesivamente. Formalmente, la precedencia tiene implicancias importantes en la concepción de las gramáticas y la vamos a anotar con el símbolo $\prec$.

Digamos primero que en la precedencia también se refleja la relación de "árbol familiar" entre nodos. En este caso, entre nodos hermanos:

- **Nodos hermanos**: Un nodo precede a su nodo hermano si los dos nodos están dominados inmediatamente por la misma madre y el nodo que precede emerge de una rama colocada a la izquierda de su nodo hermano.

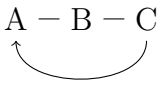
También vamos a definir una relación de precedencia inmediata entre nodos:

- **Precedencia inmediata**: Un nodo precede inmediatamente a otro si el primero esta inmediatamente a la izquierda del nodo al que precede.

Con estas definiciones en mente, podemos definir las propiedades de la precedencia (P) de forma axiomática como lo hicimos con la dominancia (D):

- **A6.** P es transitiva: $(\forall xyz \in N)\ [((x \prec y)\ \&\ (y \prec z)) \rightarrow (x \prec z)]$.

Esto excluye árboles como:

(13) 

Si lo pensamos en la cadena sonora, sería como pronunciar A antes de B, B antes de C y C antes de A... Tampoco podríamos pronunciar dos cosas al mismo tiempo, por eso:

- **A7.** P es asimétrica:$(\forall xy \in N)[((x \prec y) \rightarrow \neg(y \prec x)]$

(14) 

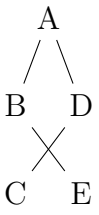
De este axioma se sigue que la precedencia no es reflexiva:

- **T1.** P es irrreflexiva:$(\forall x \in N)\ [\neg(x \prec x)]$

A su vez, entre P y D hay una condición de exclusividad, es decir que si un nodo domina al otro, no puede precederlo y viceversa. Con esta condición, P y D organizan el total de relaciones posibles en nuestro conjunto de nodos y aristas.

- **A8.** Condición de exclusividad: $(\forall xyz \in N)\ [((x \prec y) \vee (y \prec x)) \leftrightarrow \neg((x \lhd^{\star} y) \vee (y \lhd^{\star} x))]$.

Por último, la precedencia cumple con otra condición muy importante para nosotros, que es la condición de "no enredamiento", es decir, que no se puede dar algo como el siguiente árbol:

(15) 

- **A9.** Non-tangling condition: $(\forall wxyz \in N)\ [((w \prec s\ x)\ \&\ (w \lhd^{\star} y)\ \&\ (x \lhd^{\star} z)) \rightarrow (y \prec z)]$.

Este axioma refleja la asunción de Chomsky de que no existen constituyentes discontinuos, o, en nuestro árbol 15, que las ramas no pueden cruzarse.

¿Se les ocurre una frase del español que no cumpla este axioma?

2.5. Chomsky Normal Form

Hasta aquí asumimos que las reglas de derivación podían tomar la forma de un símbolo no terminal a la izquierda del símbolo de derivación $\rightarrow$, y de cualquier secuencia de símbolos no terminales o terminales a la derecha.

Esta forma se puede normalizar sin afectar el poder generativo de la gramática y a nosotros nos va a ser útil esta normalización para alguna de las aplicaciones de parser.

La forma normalizada que vamos a ver se conoce como Chomsky Normal Form y restringe la definición de las reglas a dos formas: del lado derecho de la derivación debe haber un solo símbo terminal o exactamente dos símbolos no terminales:

Si volvemos a mirar algunos ejemplos que fuimos usando hasta ahora, ¿cuál respeta la CNF y cuál no?

- **1.** SN $\rightarrow$ Det N
- **2.** S $\rightarrow$ aSb
- **3.** V $\rightarrow$ ladra | muge | pían | graznan

2.6. Algunas limitaciones de las CFG

2.6.1. Constituyentes discontinuos y dependencias cruzadas

El lenguaje que consiste de una secuencia de símbolos y su imagen espejada se conoce con el nombre de **lenguaje espejado o palindrómico**:

(16) {aa, bb, abba, aabbaa, bbbb, baab, aabbbbaa, ababbaba...}

Los lenguajes espejados son lenguajes independientes de contexto.

En el lenguaje natural, se puede encontrar una muestra de lenguajes espejados en el fenómeno del *central-embedding*.

(17) *Hans Peter Marie schwimmen lassen sah*
Hans Peter Marie nadar hacer vio

Un tipo de lenguaje muy similar a los lenguajes espejados es el llamado **lenguaje copia** (*copy language*). Partee *et al* (2012: 534) lo define del siguiente modo:

(18) $\{xx|\ x \in \{a,b\}^{+}\}$

Es decir, el lenguaje copia es un conjunto de cadenas formadas por la concatenación de dos subcadenas idénticas.

(19) {aabaaaba, aaaa, abab, abaaba, aaabaaab, bbabba...}

Sin embargo, a pesar de su semejanza con los lenguajes espejados, el lenguaje copia se distingue porque implica dependencias cruzadas en lugar de espejadas.

(20) aabaaaba

Para dar cuenta de esto, una gramática independiente de contexto es insuficiente y se precisa, en su lugar, una gramática sensible al contexto.

Ahora bien, ¿existen fenómenos lingüísticos que respondan a este tipo de esquema? Sí. Un ejemplo son las dependencias cruzadas que se dan por causa del llamado *salto del afijo*:

(21) John will have -∅ be -en eat -ing pie

(22) Juan no va a pod- -er dej- ar de mir- -ar la serie.

Un ejemplo más radical de dependencias cruzadas son las llamadas dependencias cruzadas del holandés:

(23) *omdat ik Cecilia de nijlpaarden zag helpen voeren*
porque yo Cecilia el hipopótamo mirar ayudar alimentar

(Adaptado de Stabler 2004: 701)

¿Existe un fenómeno similar a este en español? La clase pasada vimos uno que según algunos autores se ajusta a esta descripción: las oraciones con *respectivamente*.

(24) a. Catalina y Martín tomaron respectivamente un Campari y un Fernet.

b. Catalina, Martín y Federico tomaron respectivamente un Campari, un Fernet y una cerveza.

c. Catalina, Martín, Federico y Victoria tomaron respectivamente un Campari, un Fernet, una cerveza y un aperitivo.

2.6.2. Proliferación de reglas

Por último, en la ejercitación van a poder apreciar cómo el manejo de un fenómeno como la concordancia multiplica de manera explosiva las reglas de estas gramáticas. Al menos en la versión simplificada que vimos hasta ahora.

Veámoslo en un ejemplo breve:

- O → SN SV
- SN → Det N
- Det → el | la
- N → casa | cohete
- SV → V
- V → vuela

Esta gramática produce secuencias gramaticales como “el cohete vuela” y “la casa vuela”. Pero también produce secuencias agramaticales como “la cohete vuela” y “el casa vuela”.

Para que esto no suceda, deberíamos agregar símbolos no terminales que indiquen la diferencia de género:

- O → SNfem SV
- O → SNmasc SV
- SNfem → Detfem Nfem
- SNmasc → Detmasc Nmasc
- Detfem → la
- Detmas → el
- Nfem → casa
- Nmasc → cohete
- SV → V
- V → vuela

Esto implica no solamente que las reglas se multiplican con cada aspecto de estos fenómenos, sino también que entendemos estos fenómenos como axiomas dados y no podemos dar cuenta de ninguna otra explicación sobre ellos.

2.7. Ejercitación

1. Dibuje todos los árboles posibles para las siguientes oraciones:

- Fernando miró al perro con rabia.
- Macarena saludaba a la vecina con la escoba.
- Pablo estornudó fuertemente.

2. Escriba las reglas derivativas de una gramática que genere las siguientes oraciones (y no otras):

- El león está cansado.
- La chita está cansada.
- Los leones están cansados.
- Las chitas están cansadas.

(25) **Pequeña ayuda para 2:**
O → SNplfem SVplfem
O → SNsgfem SVsgfem
... → ...

2.8. Parsers

Un parser es un programa que toma un "*input*", generalmente una cadena de caracteres (*string*), y devuelve una estructura de datos. La acción que el parser realiza la vamos a llamar, entre nosotres, "parsear" (*parsing*). Parsear es el proceso de analizar una cadena de símbolos, que pueden ser símbolos pertenecientes al lenguaje natural o a un lenguaje formal, según las reglas de una gramática formal, para llegar a una estructura de datos. Un parser es lo que, por ejemplo, tomará el código de python que veremos en clase y devolverá un "parse tree" que permite que el código "humano" de python se convierta en código ejecutable por una máquina. De este modo podemos ver que en computación, el uso de programas de parseo excede su uso en lenguas naturales, pero el funcionamiento es el mismo en cierta medida.

2.9. Parsers sintáctico

Dada una gramática y una oración, el proceso de parseo es el proceso de asignar a la oración la estructura sintáctica correspondiente, de acuerdo a la gramática, y así determinar si una oración es gramatical o no. Es decir, en nuestro contexto, parsear es analizar sintácticamente una oración partiendo de una gramática y devolver el árbol correspondiente si esta oración pertenece efectivamente a la gramática (y si no, también... en muchos casos que vamos a ir viendo).

Un "parser" es, por lo tanto, un analizador sintáctico automatizado que recibe una gramática y una oración como "*input*" y aplica en una serie de pasos las reglas de la gramática sobre la oración hasta conseguir como "*output*" el análisis sintáctico subyacente a la cadena.

Si bien "*input*" y "*output*" suelen ser los mismos, no todos los parsers son iguales. Según cómo apliquen las reglas de la gramática, los hay de distinto tipo:

- **Top-Down**: El parser parte del símbolo terminal de la gramática y desarrolla cada una de las reglas posibles de derivación hasta encontrar la que coincide con el texto de la oración.

- **Bottom-Up**: El parser parte de la cadena de palabras de la oración y les asigna un las posibles categorías no terminales que irá combinando según las reglas hasta llegar a la categoría raíz.

- **Dynammic Programming**: El parser guarda resultados intermedios de la aplicación de reglas que podrá ir a buscar para completar la tarea de análisis de un modo efectivo.

- **Probabilistic Parsers**: Se basan en gramáticas que asignan peso (o probabilidad) a las estructuras para producir el análisis más probable.

A su vez, para estas variedades, encontraremos diferentes programas que los implementan de modos levemente distintos.

Referencias

Partee, B., Meulen, A., y Wall, R. (2012). *Mathematical methods in linguistics*. Kluwer Academics, Dordrecht.

Stabler, E. P. (2004). Varieties of crossing dependencies: structure dependence and mild context sensitivity. *Cognitive Science*, 28(5):699–720.

Wintner, S. (2010). Formal language theory. En Clark, A., Fox, C., y Lappin, S., editores, *The Handbook of Computational Linguistics and Natural Language Processing*, pp. 11–42. Willey Blakwell, Massachusetts.