Boletín de Matemáticas
Nueva Serie Vol.I No.2 (1994) y Vol.II No.1 (1995) (59-79)

ALGORITMOS Y GRAMATICAS: UN CAPITULO APASIONANTE DE LA TEORIA COGNITIVA

JESÚS HERNANDO PÉREZ ALCÁZAR
Departamento de Matemáticas y Estadística
Universidad Nacional de Colombia

> Durante muchos años, me han intrigado dos problemas referentes al conocimiento humano. El primero es el de explicar cómo conocemos tanto, a partir de una experiencia tan limitada. El segundo es el problema de explicar cómo conocemos tan poco considerando que disponemos de una evidencia tan amplia.
>
> Noam Chomsky

DOS GRANDES PROBLEMAS

Uno de los grandes pioneros de la teoría cognitiva es el profesor norteamericano Noam Chomsky. En este ensayo presentaremos algunos de sus muy numerosos aportes al desarrollo de esta nueva teoría, limitando la atención a las relaciones entre gramáticas y algoritmos.

Una manera muy sugestiva de abordar las relaciones entre los algoritmos matemáticos y las gramáticas de la teoría lingüística, la ofrece el propio Noam Chomsky en su importante trabajo "El conocimiento del lenguaje", publicado en español por Alianza Universidad en 1989. Nos plantea en este texto el Profesor Chomsky, dos problemas "perpendiculares" o "duales"

y cuya solución, al parecer, es la misma. De una parte, el problema de Platón y de otra, el problema de Orwell. En palabras del propio autor, estos problemas se formularían en la siguiente forma:

"Así pues, el problema de Platón consiste en explicar cómo conocemos tanto teniendo en cuenta que los datos de los que disponemos son tan escasos. El problema de Orwell consiste en explicar cómo conocemos y comprendemos tan poco, a pesar de que disponemos de unos datos tan ricos".[1]

En un extremo tendríamos la "sabiduría", y en el otro la "ignorancia" y el "dogmatismo". No debe extrañarnos que se llame al primero el "problema de Platón", y al segundo el "problema de Orwell"; Platón con su teoría de las "ideas", nos ofrece ejemplos claros de cómo "uno" -una idea- se refiere a infinitos; George Orwell por su parte, en su novela "1984", nos ha hecho una descripción alarmante de lo que es el dogmatismo y a lo que nos puede conducir - ¿nos ha conducido?-.

Siguiendo la línea de pensamiento platónico, la idea "casa", a la cual todo sujeto humano llega después de conocer tres o cuatro casas concretas, nos da la "sabiduría total" sobre las casas: ¿cómo es esto posible?. Sin embargo, y al mismo tiempo, esta idea de casa que nos hace tan sabios, nos vuelve también ignorantes y dogmáticos, pues nuestra "cultura occidental", pongamos por caso, nos impide aceptar otras modalidades de vivienda como casas: una maloca tal vez no es una casa. ¿Cómo es esto posible?

Según Chomsky, si entendemos los sujetos humanos como constructores y manejadores de algoritmos y gramáticas, encontraremos una posible respuesta a estos dos problemas que se vuelven entonces uno sólo: el hommo algoritmo-gramatical es a la vez sabio e ignorante.

Para entenderlo, debemos ver algunos asuntos previos.

Aunque la palabra "algoritmo" es de origen árabe -siglo IX- los algoritmos son tan antiguos como el pensamiento matemático mismo. Los procedimientos que utilizamos corrientemente para sumar y multiplicar, son

[1] Chomsky, Noam. El Conocimiento del Lenguaje. Alianza Universidad. Madrid, 1989.

algoritmos inventados por las culturas babilónicas alrededor del año 2000 antes de nuestra era. Sin embargo, una definición de este concepto surgió apenas a comienzos de este siglo, y en ella participaron, entre otros, autores como A.A. Markov, Emil Post y Alan Turing.

La teoría de algoritmos adquirió un impulso muy importante después de la formulación del décimo problema de Hilbert. David Hilbert en el Congreso Internacional de Matemáticas de 1900, celebrado en París, formuló una lista de 21 problemas que el ilustre matemático alemán consideraba como los más importantes en el momento. El problema número diez, planteaba encontrar un algoritmo para decidir las ecuaciones diofánticas[2]. Este problema fue resuelto en 1972, independientemente, por dos matemáticos soviéticos -Grigory Chudnovsky y Yury Matijasevich- de una manera que Hilbert nunca imaginó: no existe tal algoritmo.

Tan extraordinario resultado consolidó definitivamente la teoría de algoritmos, transformándose en una rama de las matemáticas muy productiva, la cual se ha convertido en uno de los fundamentos de la informática y la ingeniería de sistemas.

La historia del concepto de gramática es igualmente rica y larga. La gramática de Panini sobre el sánscrito, data del siglo IV antes de nuestra era, aproximadamente. Por su parte, la influyente obra de Ferdinand de Saussure fue precedida de grandes descubrimientos como los de Jacob Grimm, los de Karl Verner, y aquellos de los "jóvenes gramáticos" de la edad de oro de la filología. La escuela polaca de Roman Jacobson dio piso sólido a la moderna fonología determinando, entre otras, las reglas que distinguen los fonemas básicos de todo lengua. Este, y muchos otros episodios, abrieron el camino para que el estructuralismos norteamericano colocara a la lingüística teórica al borde de uno de sus más influyentes descubrimientos: las relaciones de equivalencia entre las gramáticas y los algoritmos.

La obra pionera de Chomsky, de los años cincuenta, bien puede ser

[2] Una ecuación diofántica es de la forma $P = 0$, donde P es un polinomio con coeficientes enteros y las soluciones que se buscan son números enteros.

objetada por los lingüistas; pero, de cualquier manera, su descubrimiento de la jerarquía gramático- algorítmica forma parte sustancial del pensamiento matemático contemporáneo, y constituye uno de los capítulos centrales de la teoría cognitiva.

Retomando el motivo planteado al comienzo, debemos partir del asombroso hecho siguiente: la cantidad de oraciones -bien formadas- del español (o de cualquier otra lengua) es infinita. En efecto, supuesto lo contrario, existiría una oración, llamémosla X, cuya longitud -en cantidad de fonemas- es la mayor posible; X sería entonces una de las oraciones "más largas"[3]. Esto es contradictorio pues la oración "la oración X es una de las oraciones más largas", nos daría otra estrictamente más larga. Ahora bien, como lo han mostrado Jacobson y sus colaboradores, la cantidad de fonemas de una lengua es finita -33 para el caso del español- lo que nos da una doble situación a explicar: de una parte, no toda secuencia finita de fonemas -expresión- es una oración y de otra, aunque la cantidad de fonemas y reglas gramaticales es finita, la de oraciones es infinita.

Tenemos así, en cada lengua, un caso típico del problema de Platón: una colección finita -los fonemas y las reglas gramaticales- nos permite manejar una colección infinita- las oraciones-. Este, también, es un ejemplo para el problema de Orwell: el manejo de una lenguaje se vuelve un hábito y no parece necesario el uso de otra diferente; pongamos por caso, nos acostumbramos al español y resulta entonces difícil acceder al inglés, al francés, al alemán, etc. De hecho, en el español hay suficiente espacio -infinito- como para permanecer en él indefinidamente.

La clave de todo ésto, según Chomsky, radica en "la gramaticalidad": de una parte, la gramática separa las expresiones en "buenas" y "malas" y de otra, las expresiones buenas o gramaticales o bien formadas, construidas a partir de un alfabeto finito y con ayuda de una cantida finita de reglas, conforman una colección infinita. En total, cada gramática permite el manejo desde la finitud, de una infinitud, lo cual nos daría una primera respuesta al problema de Platón; pero, al mismo tiempo, una gramática

[3] Podrían existir otras oraciones con longitud igual a la de X.

introduce cierta idea de "corrección" o de que las cosas andan "bien", lo cual produce un determinado apego y una gran limitación y, tendríamos así una explicación inicial para el problema de Orwell.

Veamos ahora algunas de las ideas básicas e iniciales de Chomsky.

LA JERARQUIA CHOMSKIANA

El diagrama que sigue es la representación plana de un "algoritmo regular" $\mathcal{B}_1$:

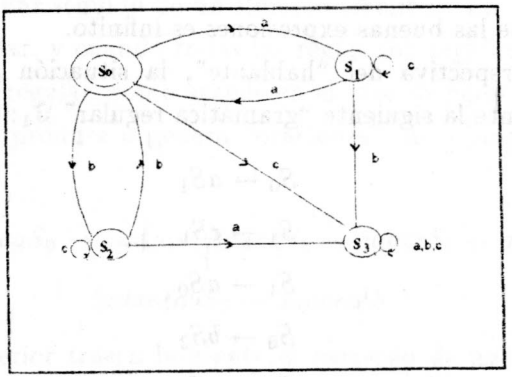


Figura 1

La anterior estructura matemática está constituida por: un alfabeto terminal $\mathcal{B} = \{a, b, c\}$, un alfabeto auxiliar $\mathcal{C} = \{S_0, S_1, S_2, S_3\}$, y unas reglas de transformación indicadas por las flechas. Los elementos de $\mathcal{B}$ se denominan "estímulos" o según el caso "fonemas", "palabras" o "símbolos terminales"; los de $\mathcal{C}$ reciben el nombre de "estados" o símbolos auxiliares uno de los cuales - S_0 - se denomina estado inicial y algunos otros, estados finales - en el ejemplo hay sólo un estado inicial y coincide con S_0-, situación que se señala con un doble círculo.

Las reglas se traducen en cambios de un estado a otro; así la flecha vertical marcada con b en la figura 1, y que va desde S_1 a S_3, lo cual podríamos representar también sí: $(b, S_1) \rightarrow S_3$, o simplemente BS_1S_3, se lee así: Si se recibe el estimulo b en estado S_1, se pasa al estado S_3.

Este algoritmo distingue secuencias finitas. Pongamos por caso, la ex-

presión "*aaaccabb*" es una "oración" o "expresión bien formada", pues partiendo del estado inicial S_0 y recorriendo la secuencia de estímulos de izquierda a derecha como un "oyente", se sigue el camino $S_0 \to S_1 \to S_0 \to S_1 \to S_1 \to S_1 \to S_0 \to S_2 \to S_0$ que culmina en el estado final S_0. Por el contrario, la secuencia "accabacc" no es "correcta", dado que deja el algoritmo en el estado S_2 que no es terminal. Este primer ejemplo muestra claramente las dos situaciones previamente comentadas: el conjunto E de todas las expresiones se particiona en dos $E = B \cup M$, donde el de las "buenas" expresiones es B y el de aquellas que son "malas" es M; además, el conjunto B de las buenas expresiones es infinito.

Desde la perspectiva del "hablante", la situación anterior se puede describir mediante la siguiente "gramática regular" $\mathfrak{G}_1$:

$$S_0 \to aS_1$$
$$S_1 \to cS_1$$
$$S_1 \to aS_0$$
$$S_0 \to bS_2$$
$$S_2 \to cS_2$$
$$S_2 \to bS_0$$
$$S_0 \to cS_3$$
$$S_1 \to bS_3$$
$$S_2 \to aS_3$$
$$S_3 \to aS_3$$
$$S_3 \to bS_3$$
$$S_3 \to cS_3$$
$$S_1 \to a$$
$$S_2 \to b$$

Una gramática regular es también una estructura matemática. En el ejemplo anterior, se tiene un alfabeto terminal $\mathcal{B} = \{a, b, c\}$, un alfabeto auxiliar $\mathcal{C} = \{S_0, S_1, S_2, S_3\}$, y unas "reglas gramaticales regulares". Como en el

caso de los algoritmos regulares, los elementos de $\mathcal{B}$ se denominan también "fonemas", "símbolos terminales", etc.

El símbolo auxiliar S_0 recibe el apelativo de "axioma", pues al igual que en las teorías axiomáticas, se comienza con este símbolo el proceso de producción o generación de una oración. En general, una regla gramatical tiene la forma $\alpha \rightarrow \beta$ donde α y β son expresiones arbitrarias del alfabeto $\mathcal{B} \cup \mathcal{C}$, y se lee " β reemplaza α". Cuando α y β tienen alguna forma especial, la regla recibe, a su vez, un apelativo especial. En el ejemplo en consideración, α es un símbolo auxiliar, y β es, o bien un símbolo terminal o un símbolo auxiliar seguido de un símbolo terminal. En tal caso la regla se denomina regular, y cuando todas las reglas son regulares la gramática se llama entonces regular, como sucede en el caso de nuestro ejemplo.

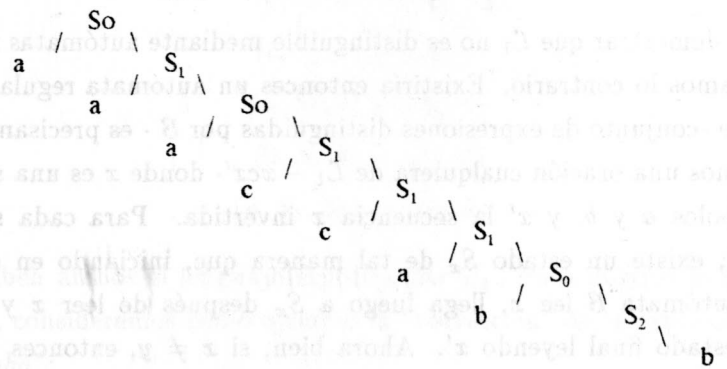
Una gramática produce o genera "oraciones"; un ejemplo para el caso $\Im_1$ es el siguiente:

$$S_0 \rightarrow aS_1 \rightarrow aaS_0 \rightarrow aaaS_1 \rightarrow aaacS_1 \rightarrow aaaccS_1 \rightarrow aaaccaS_0 \rightarrow$$

$$aaaccabS_2 \rightarrow aaaccabb$$

El proceso anterior trae a la mente el recuerdo de una "demostración matemática": se inicia con el axioma S_0 y cada vez se aplica una "regla de deducción", que en el caso que estamos tratando es una "regla gramatical".

Esto mismo se puede ver en forma de árbol, así:



La gramática $\Im_1$, produce las mismas oraciones que distingue el algoritmo $\mathcal{B}_1$, y son por lo tanto equivalentes. Este fue uno de los descubri-

mientos cruciales de Noam Chomsky: traducir cada tipo de algoritmo en una clase especial de gramática y recíprocamente, lo que origina una fuerte relación entre lingüística y matemática.

Chomsky tuvo un motivo adicional para el desarrollo de sus investigaciones: el debate con el conductismo. No es casual el hecho de haber utilizado la palabra "estímulo" en la descripción de los algoritmos regurales, llamados también autómatas regulares. Charles Hockett, conocido lingüista seguidor del conductismo, había lanzado la hipótesis según la cual la conducta lingüística sería explicable mediante un algoritmo regular, análogo a como sería el comportamiento de un ratón frente a la consecución de alimento. Una argumentación sencilla, pero elegante, permite concluir la imposibilidad de capturar la riqueza de los lenguajes comunes mediante los algoritmos regulares. En efecto, las oraciones del tipo "Dábale arroz a la zorra el abad", "Anita lava la tina", o "Las Nemocón no comen sal" cuya estructura "especular" es de gran complejidad, son muy frecuentes en los lenguajes comunes. Las estructuras de este tipo de oraciones se describen, respectivamente, mediante los lenguajes formales $\mathcal{L}_1 = \{aca, abcba, abacaba, \ldots\}$ y $\mathcal{L}_2 = \{aa, abba, abaaba, \ldots\}$. Los autómatas regulares no pueden reconocer ninguno de estos lenguajes. En el primer caso, se tendrían las estructuras especulares con espejo c, mientras en el segundo, se estarían describiendo las estructuras espectaculares sin espejo.

Para demostrar que $\mathcal{L}_1$ no es distinguible mediante autómatas regulares supongamos lo contrario. Existiría entonces un autómata regular $\mathcal{B}$ cuyo lenguaje -conjunto de expresiones distinguidas por $\mathcal{B}$ - es precisamente $\mathcal{L}_1$. Escojamos una oración cualquiera de $\mathcal{L}_1$ – xcx'- donde x es una secuencia de símbolos a y b, y x' la secuencia x invertida. Para cada secuencia de xcx', existe un estado S_x de tal manera que, iniciando en el estado S_0 el autómata $\mathcal{B}$ lee x, llega luego a S_x después de leer x y culmina en un estado final leyendo x'. Ahora bien, si $x \neq y$, entonces $S_x \neq S_y$ pues de otra manera el autómata $\mathcal{B}$ leería la secuencia xcy' que no es espectacular. Esto significa que el autómata $\mathcal{B}$ debería tener un conjunto infinito de estados, lo que contradice el carácter finito de estas estructuras

matemáticas. En conclusión, $\mathcal{L}_1$ no es un lenguaje regular, mucho menos los lenguajes comunes cuya riqueza en construcciones espectaculares es muy amplia. Por lo tanto, los lenguajes comunes no son regulares y así, la conducta lingüísticas no sería similar a la conducta ratonil.

La erudición matemática de Chomsky, quien conocía los trabajos de Turing y de Post, condujo a que relacionara otro tipo de algoritmos con las gramáticas "ahormacionales" (o estructuras "parsing" en inglés), descubriendo así un nuevo nivel en la jerarquía algoritmico-gramatical.

La "gramática libre de contexto" $\mathfrak{G}_2$:

$$S \to aSa$$
$$S \to bSb$$
$$S \to c;$$

y el "algoritmo con memoria" $\mathcal{B}_2$ (Z_0 representa la memoria);

$$V_M = \{Z_0, a, b, c\}, \quad V_T = \{a, b, c\}, \quad C = \{S_0, S_1, S_2\}, \quad F = \{S_3\},$$

$$(a, S_0, e) \to (S_0, a)$$
$$(b, S_0, e) \to (S_0, b)$$
$$(c, S_0, e) \to (S_1, e)$$
$$(a, S_1, a) \to (S_1, \delta)$$
$$(a, S_1, b) \to (S_1, \delta)$$
$$(a, S_1, Z_0) \to (S_2, \delta)$$

describen ambos el lenguaje espectacular $\mathcal{L}_1$. Para ilustrar lo que esto significa, consideremos como ejemplo la "corrección" de la expresión "*ababcbaba*":

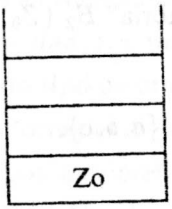
a) Mediante la demostración:

$$a \to aSa \to abSba \to abaSaba \to ababSbāba \to ababcbaba.$$

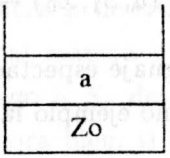
b) En forma de árbol:

c) Mediante el algoritmo con memoria:

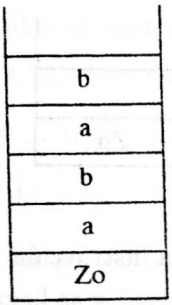
i) Se inicia en el estado S_0 y se activa (Z_0) la memoria



ii) Se recibe el estímulo a y se aplica la instrucción $(a, S_0, e) \rightarrow (S_0, a)$, que se lee: "si se recibe el estímulo a en estado S_0, no importa qué se tenga en la memoria (de esta manera se lee el símbolo e), se pasa al estado S_0 y se coloca a en memoria":

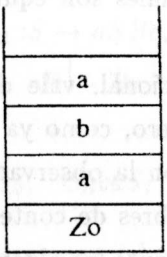


iii) Al recibir los estímulos b, a, b se aplican las dos primeras instrucciones, se permanece en el estado S_0 y la memoria queda como sigue:

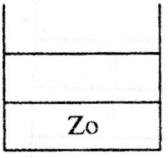


iv) Con la llegada del estímulo c se aplica la instrucción $(c, S_0, e) \rightarrow (S_1, e)$, con lo cual se cambia de estado y no se modifica la memoria. La instrución anterior se lee como sigue: "si en estado S_0 se recibe el estímulo c, sin importar lo que se tiene en la memoria (e), se pasa al estado S_1 y no se modifica lo registrado en la memoria (e)".

v) Al aparecer b se aplica la instrucción $(b, S_1, b) \rightarrow (S_1, \sigma)$ que se lee: "al recibir el estímulo b en estado S_1 y con b en la memoria, se pasa al estado S_1 y se borra (σ es el símbolo para borrar) el símbolo que aparece en la parte superior de la memoria". Con esto la memoria queda en la siguiente forma:



vi) Con la aparición subsiguiente de los estímulo a, b, a, se permanece en el estado S_1 y se van borrando estos símbolos en la memoria, la cual queda después de tres pasos, como sigue:



vii) Finalmente, se aplica la instrucción $(e, S_1, Z_0) \rightarrow (S_2, \sigma)$, que deja la memoria "inactiva" y se llega al estado final S_2. Esta Ultima instrucción se lee de la siguiente manera: estando en estado S_1 con Z_0 en la memoria; sin recibir ningún estímulo (e), se pasa al estado S_2 y se borra Z_0 en la memoria.

Como se llegó a un estado final, y la memoria quedó inactiva, el algoritmo reconoce la expresión dada.

Todo lo anterior ilustra varios puntos importantes:

1) Las gramáticas libres de contexto utilizan reglas en las cuales la expresión α es un símbolo auxiliar y la expresión β es arbitraria. Este tipo de reglas se llaman libres de contexto, y aquellas gramáticas en las cuales toda regla es de esta forma, se llaman "libres de contexto" o "ahormacionales". Las reglas regulares son casos particulares de reglas ahormacionales; por lo tanto, toda gramática regular es una gramática ahormacional.
2) Las gramáticas ahormacionales son equivalentes a los algoritmos con memoria.
3) El lenguaje $\mathcal{L}_1$ es ahormacional, vale decir distinguible por una gramática ahormacional; pero, como ya se comentó, $\mathcal{L}_1$ no es regular. Combinando esto con la observación 1) es posible concluir ahora que las gramáticas libres de contexto son más potentes que las regulares. Equivalentemente los algoritmos con memoria tienen mayor capacidad que los regulares.

Se podría conjeturar ahora que el español, o cualquier otro lenguaje común es ahormacional. Sin embargo, las gramáticas de los lenguajes comunes utilizan reglas "sensibles al contexto" del tipo $*\alpha\ \Psi \rightarrow *\beta\ \Psi$, en las cuales β reemplaza a α siempre y cuando α se encuentre entre $*$ y Ψ. Una gramática donde toda regla sea sensible al contexto, se denomina "sensible al contexto". Tal es el caso de la siguiente gramática:

$$S \rightarrow AB, \quad A \rightarrow AA, \quad B \rightarrow BB, \quad AB \rightarrow aB, \quad AB \rightarrow cB, \quad Aa \rightarrow aa,$$
$$aB \rightarrow af, \quad cB \rightarrow Cg, \quad Ac \rightarrow cc, \quad fB \rightarrow ff, \quad gB \rightarrow gg.$$

Como en el caso regular, las gramáticas sensibles al contexto son equivalentes a otro tipo de algoritmos llamados linealmente acotados, y son más potentes que las gramáticas libres de contexto.

El Cuadro No. 1 que se muestra a continuación, completa la jerarquia chomskiana construida en los años 50. Hoy en día esta jerarquía ha sido ampliada y notablemente enriquecida.

GRAMATICAS	ALGORITMOS
Regular	Regulares
Libres de Contexto	Con memoria
Sensibles al contexto	Linealmente acotados
Transformacionales	De Turing

Cuadro No. 1

Los siguientes son ejemplos, respectivamente, de una gramática transformacional y de un algoritmo de Turing.

$$\Re_3 : S \rightarrow aSBc, \quad S \rightarrow abc$$
$$cB \rightarrow Bc, \quad bB \rightarrow bb$$

$$\Re_3 : S_0bDS_0, \quad S_0aDS_0, \quad S_0caS_1, \quad S_1aIS_2, \quad S_2baS_2, \quad S_2caS_2$$

La forma de leer las instrucciones del algoritmo de Turing se ilustran con los siguientes casos:

S_0bDS_0 : Si en estado S_0 se recibe el estímulo de b, se queda en estado S_0 y se desplaza un lugar a la derecha (D).

S_1aIS_2 : Si en estado S_1 se recibe el estímulo de a, se cambia el estado S_2 y se desplaza un lugar a la izquierda (I).

S_0caS_1 : Si en estado S_0 se recibe el estímulo de c, se cambia al estado S_1 y se cambia c por a.

Una producción de $\mathfrak{R}_3$ sería como sigue : $S \rightarrow aSBc \rightarrow aaSBBcc \rightarrow aaaSBBBccc \rightarrow aaaabcBBBccc \rightarrow aaaaBcBBcc \rightarrow aaaabBBcBccc \rightarrow aaaabBBBcccc \rightarrow aaaabbBBcccc \rightarrow aaaabbbBcccc \rightarrow aaaabbbbcccc.$

Una producción de $\mathfrak{I}_3$ es por ejemplo : $S_0bac \rightarrow bS_0ac \rightarrow baS_0c \rightarrow baS_1a \rightarrow bS_2aa.$

Aquí, la expresión bac se convirtió en baa.

El cómputo que se muestra a continuación ilustra el papel especial del símbolo b (blanco) en los algoritmos de Turing y una característica particular de estos algoritmos.

$$cS_0baa \rightarrow cbS_0aa \rightarrow cbaS_0a \rightarrow cbaaS_0b \rightarrow cbaabS_0b \ldots$$

Como se ve, el cómputo no termina, lo cual ilustra una diferencia notable de los algoritmos de Turing respecto de los otros tipos de algoritmos. El símbolo b funciona como "blanco", es decir, no hay estímulo. (Esta característica es importante porque muestra que el cómputo mediante un algoritmo puede ser interminable).

La información que suministra el Cuadro No. 1 es doble: establece la equivalencia entre tipos de gramáticas y tipos de algoritmos, y cada uno de estos aparece en la jerarquía según su capacidad. Se pueden hacer entonces afirmaciones del siguiente tenor: todo lenguaje generado por una gramática transformacional es reconocido por algún algoritmo de Turing y recíprocamente; todo lenguaje reconocido por un algoritmo linealmente acotado, es reconocido por algún algoritmo de Turing; existen lenguajes generados por gramáticas sensibles al contexto, que no son generados por gramáticas libres de contexto, etc.

Precisando un poco, podemos establecer ahora una definición de "gramática transformacional". Obviamente, según la información del Cuadro No.1, es una estructura matemática equivalente a los algoritmos de Turing.

Dado el alfabeto terminal $\mathcal{B}$ (conjunto finito de símbolos), una gramática transformacional $\mathfrak{R}$ es una estructura $(\mathcal{C}, \{S\}, \wp)$ en la cual $\mathcal{C}$ es un conjunto finito cuyos elementos se llaman símbolos auxiliares, S es un símbolo auxiliar especial llamado "el axioma" de la gramática y, $\wp$ un conjunto finito de reglas de la forma $\alpha \rightarrow \beta$ (donde α, β son expresiones del alfabeto $\mathcal{B} \cup \mathcal{C}$). Entre las reglas de $\wp$, unas de ellas tiene la forma $S \rightarrow \beta$ (una producción o demostración se inicia siempre con el axioma).

Una "demostración" o "generación" de la gramática $\mathfrak{R}$, es una secuencia finita $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ de expresiones, de manera tal que $\alpha_1 = S$ y, cada α_k se deduce mediante la aplicación de una de las reglas de $\wp$ de alguna de las expresiones anteriores $\alpha_1, \ldots, \alpha_{k-1}$. Una "generación" o "producción" ó "construcción", es una demostración en la cual la última expresión α_n no contiene símbolos auxiliares.

El lenguaje $\mathcal{L}$ generado ó construido por la gramática $\mathfrak{R}$, es el conjunto de todas las expresiones del alfabeto $\mathcal{B}$ que aparecen como último elemento de una construcción- demostración.

Así pues, las expresiones del alfabeto $\mathcal{B}$ se particionan en dos conjuntos: $\mathcal{L}$ (lenguaje construido por $\mathfrak{R}$) y M (expresiones agramaticales o mal construidas). El conjunto $\mathcal{L}$ es evidentemente infinito, lo que deja "mucho espacio" para las construcciones con la gramática $\mathfrak{R}$.

Desde esta perspectiva, tal vez no la del propio Chomsky, es posible ahora retomar más claramente una de las afirmaciones del comienzo de este escrito: el ser humano crea gramáticas transformacionales (lo que le permite el manejo de infinidades mediante instrumentales finitos - el alfabeto y el conjunto de reglas-) pero, cada gramática así creada permite construcciones infinitas (lo cual "impide" o al menos dificulta el paso a otras gramáticas).

Quedan, sin embargo, muchos interrogantes. Uno de ellos particularmente importante es el siguiente: ¿existen gramáticas más poderosas que las gramáticas transformacionales ? Discutiremos esto en la siguiente sec-

ción.

LA HIPOTESIS DE PUTNAM

No hace muchos años el filósofo y matemático norteamericano Hilary Putnam, planteó la conjetura sobre la generalidad y universalidad de las gramáticas transformacionales la cual todavía hoy no tiene una respuesta definitiva. Según Putnam, todo lenguaje común es "recursivamente enumerable"; vale decir, el conjunto de las oraciones (bien formadas) de una lengua, es generado por alguna gramática transformacional. Escogiendo, mediante algún criterio de simplicidad, la mejor gramática transformacional que genera las oraciones de una lengua, p.e. el español, ésta sería entonces la gramática de dicha lengua.

No es difícil hacer un listado de las objeciones esgrimidas contra la hipótesis de Putnam y contra la propia teoría Chomskiana. Por ejemplo, ¿cómo establecer de antemano las oraciones de una lengua?. Este tipo de crítica es tan severa como la de quienes señalan que el concepto de gramática planteado hasta el momento es meramente combinatorio (sintáctico) y, no tiene en cuenta la semántica ni los usos de una lengua (pragmática).

La validez relativa de todas estas objeciones, no disminuye el interés en la hipótesis de Putnam. Hay todavía mucho trabajo por hacer.

En este ensayo hemos elegido mostrar los aspectos positivos del planteamiento y así, haremos más bien un pequeño repaso de aquellos hechos en los cuales se fundamenta la propuesta de Putnam. (Con esto hacemos una declaración de dogmatismo: preferimos jugar al juego de Chomsky).

Ante todo, retomemos la historia de la teoría de algoritmos. Un episodio interesante en este proceso, lo constituye todo el trabajo de investigación relacionado con otra hipótesis importante reconocida con el nombre de "Tesis de Church" (por el lógico Alonzo Church). Hay varias formas de plantear esta conjetura; una muy interesante nos permitirá relacionarla con la de Putnam: Todo lo que sea intuitivamente computable, lo es mediante algún algoritmo de Turing.

Un planteamiento como el anterior, surge de manera muy natural cuando se trata de precisar (definir) un concepto tan importante como el de algoritmo. ¿Qué es un algoritmo?. Mencionamos en la sección anterior cuatro variantes distintas: los algoritmos regulares, los algoritmos con memoria, los linealmente acotados y, los de Turing. Sin embargo, todos ellos son clases particulares de algoritmos de Turing. Otros investigadores como A.A. Markov y Emil Post, propusieron definiciones diferentes (algoritmos de Markov y algoritmos de Post), que resultaron asimilables a la definición de Turing.

Otra familia interesante de algoritmos son los llamados "algoritmos celulares". El de John Conway, conocido como "juego de la vida", es de mucho interés y ha suscitado numerosas investigaciones en los últimos años. Aquí, una vez más, estas definiciones resultan también identificables como algoritmos de Turing.[4]

La manera como esta identificación se ha venido logrando, ha consistido en traducir el concepto de algoritmo al de "función computable" o "recursiva".

Veamos un ejemplo.

El siguiente algoritmo de Turing es una sumadora:

$$S_0|bS_1, \quad S_1bIS_2, \quad S_2|IS_2, \quad S_2bIS3, \quad S3|bS3.$$

Un cómputo sencillo deja entender el por qué:

$$S_0\,|\,|\,|\,b\,|\,|\,|\rightarrow S_1\,|\,|\,b\,|\,|\,|\rightarrow bS_1\,|\,|b\,|\,|\,|\,|\rightarrow b\,|\,b\,|\,|\,|\,|\rightarrow$$
$$b\,|\,|S_2b\,|\,|\,|\,|\rightarrow b\,|\,|\,bS_3\,|\,|\,|\,|\,b\,|\,bs_3b\,|\,|\,|\,.$$

En este caso, el alfabeto terminal $\{b,|\}$ permite codificar los números y las parejas de números en la forma siguiente :

$$0 = |\,, \qquad 1 = |\,|\,, \qquad 2 = |\,|\,|\,,\ldots$$
$$(0,0) = |\,b\,|\,, \qquad (2,1) = |\,|\,|\,b\,|\,|\,, (2,3) = |\,|\,|\,b\,|\,|\,|\,,\ldots$$

[4] En el hermoso libro de Gardner, reseñado al final, se encuentra una explicación de lo que es el "juego de la vida".

En el cómputo efectuado, la pareja $(2,3)$ se ha convertido en una expresión en la cual aparecen cinco "palitos", lo que significa entonces que $2+3$ es igual a cinco[5]; hecho que todos ya sabíamos, sólo que lo anterior muestra que también lo "saben" los algoritmos de Turing.

Así, la función

$$\mathbb{N} \times \mathbb{N} \to \mathbb{N}$$
$$(n, m) \to n + m$$

es Turing-computable; es decir, existe un algoritmo de Turing que computa esta función -el algoritmo del ejemplo anterior.

Esto define el conjunto $\mathbf{C}$ de todas la funciones Turing-computables; o sea, las funciones f de dominio $\mathbb{N}^n$ para algún $n \geq 0$ y codominio $\mathbb{N}$, y para las cuales existe un algoritmo de Turing que computa f. Lo que se ha descubierto es que este conjunto $\mathbb{C}$ puede definirse independientemente del concepto de algoritmo de Turing y por ello, se le denomina conjunto de funciones recursivas.

Lo que se postula entonces -y esta es otra forma de plantear la tesis de Church- es que el conjunto de las funciones recursivas cuya definición no depende del concepto de algoritmo, es idéntico al conjunto de las funciones computables, en el sentido de Turing, o de Markov o de Post. Este resultado es de una gran riqueza y sugiere claramente que la definición de Turing es una "buena" definición: capturaría el concepto intuitivo de calculabilidad ó computabilidad. Es posible formular lo anterior de otra manera: todo cálculo realizable en un computador, puede efectuarse mediante algún algoritmo de Turing.

Ahora bien, los algoritmos de Turing poseen otra característica crucial: existen algoritmos de Turing universales. Esto último significa lo siguiente: para cada $n \geq 0$ existe un algoritmo de Turing $\Im_n$ de tal manera que, si $f : \mathbb{N}^n \to \mathbb{N}$ es una función recursiva. $\Im_n$ computa f y ese procedimiento

[5] En la respuesta o resultado, el convenio para identificar números es diferente: el número de palitos es el número que resulta.

de existencia es algorítmico.

Estas propiedades de los algoritmos de Turing, son "curiosamente" similares a otras muy características de los lenguajes comunes: pues estos últimos son universales; vale decir, son instrumentos muy eficientes para la comunicación -un poco en contravía con Wittgenstein, todo lo decible se puede decir- y además, con ellos se puede hablar acerca de ellos mismos.

Reformulemos ahora la hipótesis de Putnam.

Un subconjunto $\mathcal{C} \subset \mathbb{N}$ de números naturales es recursivamente enumerable, si existe una función $f : \mathbb{N} \to \mathbb{N}$ recursiva (Turing-computable), de tal manera que $\mathcal{C} = \{f(n) \mid n \in \mathbb{N}\} = f(\mathbb{N})$. Ahora bien, codificando los fonemas del español (o de cualquier otra lengua) es posible asignar a cada secuencia de fonemas un número, y en esta forma cada expresión española (secuencia finita de fonemas en español) se convierte en un número. Así las cosas, el conjunto $\mathbf{E}$ de las oraciones del español (este sería propiamente el español) es un subconjunto de $\mathbb{N}$ y entonces, lo que estaría afirmando Putnam es que $\mathbf{E}$ es un conjunto recursivamente enumerable. O sea, el conjunto $\mathbf{E}$ (las oraciones del español), se podría colocar algorítmicamente en una lista infinita. Una buena versión sobre lo que hacen las gramáticas, y además una buena hipótesis para tratar de meter un lenguaje común en un disquete.

Lo anterior, claro está, supone que un lenguaje común permanece estático o como lo decía Saussure, en sincronía. De todas maneras, el cambio de lenguajes con el tiempo, no afecta el alcance de la hipótesis de Putnam. La hipótesis es discutible desde ángulos como los mencionados en la sección anterior; sin embargo, el ánimo de quienes persiguen encajonar la mente en el "silicio" la ha escogido como la mejor de las guías para el trabajo que desarrollan.

UN COMENTARIO FINAL

Con la definición de algoritmo en el "bolsillo" (algoritmo = algoritmo de Turing = gramática generativa), resulta posible retomar cuestiones como la planteada por Hilbert sobre las ecuaciones diofánticas. Fue este el método

que aplicaron Matijasevich y Chudnovsky para probar que no existe ningún algoritmo para decidir si una ecuación diofántica tiene o no solución. La teoría es entonces "maravillosamente cerrada": los algoritmos no pueden con todo, y es por lo tanto una teoría "sabia", "no dogmática". De hecho, reconoce "lo otro", admite la existencia de un "afuera". Por eso, no debe resultar extraño que las gramáticas generativas no puedan dar cuenta de toda la riqueza de los lenguajes comunes.

CHOMSKY, UNO DE LOS CREADORES DE LA TEORIA COGNITIVA

No en vano Noam Chomsky aparece como uno de los creadores de la teoría cognitiva: tratando de encontrar respuesta a sus dos grandes problemas (típicos de la teoría cognitiva) nos ha conducido a grandes descubrimientos. Mencionemos algunos de ellos:

1. No es posible ser sabio sin al mismo tiempo ser dogmático. En realidad podríamos decir lo mismo positivamente: para evitar el dogmatismo es indispensable manejar al menos dos gramáticas (o algoritmos) diferentes. En el lenguaje de Basil Bernstein, esto podría formularse asi: muévete dentro de varios códigos, sin que te vuelvas un oportunista.
2. Tendríamos dos grandes modalidades de creatividad: fuerte y débil. Un sujeto fuertemente creativo construye gramáticas (o algoritmos). Un sujeto débilmente creativo maneja gramáticas (algoritmos) construidas por otros. Habría entonces, dos grandes modalidades de investigación: una al interior de un "paradigma" dado y la otra construyendo nuevos "paradigmas".
3. Cierto tipo de estructuras cognitivas requieren "memoria". Tal sería el caso de las estructuras especulares.
4. El más importante de todos: las destrezas algoritmico-gramaticales son instrumentos cognitivos de gran capacidad y utilidad, pero deben existir otras habilidades, pues las primeras son limitadas.

5. La lingüística y la matemática no son teorías extrañas; vale decir, las destrezas lingüísticas y lógicas no son ajenas las unas a las otras. Construir una oración- estructuralmente - es lo mismo que demostrar un teorema. (Resulta entonces bastante difícil de aceptar que la lógica matemática aparezca tan complicada cuando es tan fácil aprender a hablar).
6. Desde el punto de vista didáctico, encontramos en esta historia y con esta teoría, una herramienta poderosa: las dificultades en el aprendizaje de nuevas "teorías" ó "algoritmos-gramáticas" obedecen, simplemente, a que se conoce, comprende y utiliza otra u otras "teorías" ó "algoritmos-gramáticas".

Hay aquí una coincidencia muy profunda entre Piaget (estructuras cognitivas), Chomsky (algoritmos-gramáticas), Kuhn (paradigmas), Bachelard (concepciones), Foucault (epistemes), Vigotsky (zonas de desarrollo próximo), y Von Foester (No se puede ver que no se ve lo que no se ve) todos ellos coinciden en plantear el aprendizaje como un proceso que exige "desequilibrar" para despues volver a "equilibrar".

LECTURAS RECOMENDADAS

Chomsky, Noam. **El conocimiento del lenguaje**. Alianza Universidad. Madrid. 1989.

Davis, Martin and Elaine Weyuker. **Computability, Complexity, and Languages.** Fundamentals of theoretical computer science. Academic Press. New York - London. 1983

Gardner, Howard. **La nueva ciencia de la mente. Historia de la revolución cognitiva.** Ediciones Paidos. Barcelona - Buenos Aires - México. 1988.

Gardner, Martin. **Ruedas, vida y otras diversiones matemáticas.** Editorial Labor S.A. Barcelona, 1985.

Gross M., Lentin A. **Introduction to formal grammars.** Springer - Verlag. Berlin - Heidelberg - New York. 1970.