Computar y compactar

*Horacio Faas**

El golpe asestado por Gödel en 1931 al programa de Hilbert asombró en su momento y aún sigue asombrando. Como se sabe, señala una de las limitaciones más importantes a los formalismos y se han propuesto desde entonces varias vías para superarlas. Pero también se ha naturalizado el asombro, para mostrar que en los sistemas puede uno esperar ciertos resultados: por ejemplo, si el sistema es compacto (expresión de lo infinito mediante lo finito) se pierden ciertas intuiciones; y si se enfoca a un sistema formal como un programa de computadora que produce teoremas se advierte que tal limitación es natural. En este sentido, el trabajo se refiere principalmente a los programas elaborados por Chaitin para computar con una máquina universal de Turing y a cómo puede mostrarse que el número omega de Chaitin (elaborado sobre la base de la probabilidad de la detención de un programa) es absolutamente incomputable. Se trataría de una entidad matemática definible pero no compactable (en el sentido de compresible), y que se puede comparar con un enunciado *à la Gödel*, verdadero pero indemostrable.

La compacidad de un sistema puede definirse como su característica de que la relación de consecuencia lógica establecida en cualquier caso entre infinitas premisas y conclusión puede también establecerse entre algún grupo finito de esas premisas y la conclusión. En general, es considerada como una virtud, ya que hace más manejable la noción de consecuencia lógica. ¿Cuándo agotaríamos las infinitas premisas? Pero también plantea limitaciones. Algunas inferencias que se catalogarían intuitivamente como válidas se pierden por la compacidad. Quizás el ejemplo más famoso es la regla omega. Supongamos que un predicado A es verdadero de todo número natural, 0, 1, 2, ... ; es decir, son verdaderos los enunciados $A(0)$, $A(1)$, $A(2)$, ... Entonces uno afirmaría sobre esa base "Para todo n, $A(n)$ es verdadero." Pero en lógica clásica esta no es una inferencia válida ya que este último enunciado no se deduce de ningún subconjunto finito de las infinitas premisas iniciales.

Esa tensión entre lo finito y lo infinito ha constituido un problema y un incentivo permanentes a lo largo de la historia de la matemática. Como la finitud es lo único manejable, la propuesta de finitismo que en su momento realizó Hilbert fue ampliamente aceptada por la comunidad matemática. Aquí se parte de una cantidad finita de axiomas, y los signos y las reglas son también finitos, pero los resultados pueden ser infinitos, al menos potencialmente. Por desgracia, en 1931 Gödel demostró que todo sistema finitista es incompleto. Para ello construyó (con medios finitistas) un enunciado que dice de sí mismo que es indemostrable, y que es expresable como un enunciado de la teoría de números. A él mismo le corresponde un número, que luego fue llamado un número de Gödel.

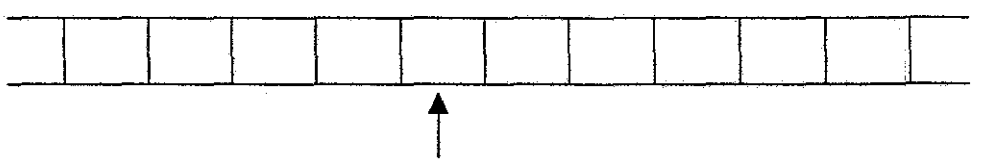
* Universidad Nacional de Córdoba.

El sorprendente resultado de Gödel fue aceptado recelosamente por muchos matemáticos y lógicos (incluyendo a Hilbert), pero otros, como von Neumann, lo acogieron plenamente. Turing, preocupado por problemas similares, logró incorporarlo como derivado de uno de sus teoremas más inesperados: Es imposible suministrar un método general que indique para todo programa de computación si se detendrá o no. Lo que antecede es una versión traducida a lenguaje actual de algo que Turing había afirmado para sus máquinas, llamadas luego máquinas de Turing. El problema al que se refería es el problema de la detención (*halting problem*).

Pero lo destacable aquí, como también lo correspondiente a los trabajos coetáneos de Church y de Post, era que aquel resultado comenzaba a aparecer repetidamente en enfoques diferentes (que luego se mostraron como equivalentes entre sí), surgía cada vez más "naturalmente".

Más recientemente, Gregory Chaitin ha mostrado que el teorema de incompletud de Gödel era algo de esperar en un sistema apoyándose en su teoría algorítmica de la información. Según él mismo declara, ha seguido los pasos de Gödel y de Turing, pero se ha valido ahora de las herramientas concretas que son las computadoras digitales para arribar a esos resultados.

Yo he preparado un par de máquinas de Turing muy simples, las más simples de todas, para ayudar a presentar algunas de las ideas de Chaitin. Recuérdese que una máquina de Turing consta de una cinta infinita en ambas direcciones en la que aparecen los datos y los resultados de las operaciones, y que estas operaciones son solamente, para nuestro caso, moverse un lugar a la derecha, moverse un lugar a la izquierda, borrar un signo o poner un signo. Los lugares de la cinta, por comodidad, se representan por cuadraditos, y la posición de la máquina por una flechita. La máquina opera solamente sobre un cuadro por vez.



Las máquinas se representarán aquí por sus instrucciones escritas: grupos de cuatro caracteres. Las he elegido bastante tontas, pero ilustrativas de lo que quiero: una de ellas lee un cuadro en blanco, le escribe un palote y allí se queda; la otra sólo se corre de cuadro en cada instrucción. Una de ellas se detiene, la otra, no, y ello es fácilmente previsible en los casos elegidos.

Recordemos que cuando se definen máquinas de Turing por códigos lingüísticos, cada renglón empieza y termina con indicaciones de estados (llamémosles primero y cuarto lugar), y en los que llamaremos lugares segundo y tercero hay: en el segundo, lo que la máquina percibe en el lugar de la cinta en que se encuentra (lo único que puede percibir, por otra parte), y en el tercero, qué debe hacer ante esa situación dada por el estado y lo que percibió. Supongamos, por ejemplo, un par de máquinas de Turing extremadamente simples, una de las cuales se detiene y la otra, no. Presentaré a ambas por sus códigos lingüísti-

cos, con las siguientes convenciones habituales dadas por la Tabla 1 (los únicos signos que aparecen en la cinta de la máquina son el blanco – i.e. el cuadro vacío – y el palote – representado por una A):

Tabla 1

Símbolo	Explicación
S_n	Estado de la máquina. En el lugar de la letra n va un número entero positivo que indica cuál es el estado
D	Instrucción de que se corra un lugar hacia la derecha sobre la cinta.
I	Ídem hacia la izquierda.
B	Si aparece en el segundo lugar del renglón, indicación de que allí no hay nada; si en el tercero, indicación de que hay que borrar lo que haya.
A	Si aparece en el segundo lugar del renglón, indicación de que allí hay un palote (' |'); si en el tercero, indicación de que tiene que haber uno.

Convengamos en que el estado inicial de las máquinas es siempre S_0.

Máquina 1:

S_0BAS_0

Máquina 2:

S_0BDS_0

La máquina 1 inicia su actividad frente a un espacio en blanco, pone allí un palote y se detiene. La 2, frente a la misma situación, se corre un lugar a la derecha, pero como allí encuentra nuevamente esa situación, hace lo mismo y ya no se detiene nunca.

Si se escribe la descripción de la máquina 1 en código ASCII, letra a letra y número a número concatenados, resulta un numeral binario de cierta longitud.

Las máquinas más complicadas tienen por supuesto muchos renglones, pero se puede concatenar los renglones en su orden y resulta también un numeral binario.

Desde Turing se ha llamado computable a un número para el cual existe un procedimiento mecánico (una de sus máquinas en ese momento, un programa de computadora hoy) que lo calcula. π, por ejemplo, es computable y, aunque nunca se agotará la serie de dígitos que van apareciendo en su expansión decimal, cada uno de ellos está calculado con precisión.

Como se sabe, las computadoras digitales actuales utilizan como lenguaje de máquina un código binario y se llama *bit* al elemento mínimo de información en ese código (un 0 o un 1, por ejemplo). Si se representan los programas de computadora mediante tal código, sus longitudes resultan comparables. Pueden entonces introducirse los siguientes conceptos:

Complejidad de un programa = Longitud en bits de un programa

Complejidad de un número = Longitud en bits del programa más corto que lo genera

Compresibilidad de una expresión = Posibilidad de hallar un programa más corto que ella misma y que la genere

Una *máquina universal de Turing* (MUT) es una máquina de Turing capaz de ejecutar cualquier instrucción dada para una máquina con objetivos específicos. Sería el homólogo de lo que actualmente es una computadora digital. Una MUT, por supuesto, puede también describirse como un número binario

El siguiente es el enunciado de Chaitin, comparable al de Gödel: "Encuentre una serie de dígitos binarios cuya complejidad se pueda demostrar como mayor que la cantidad de bits que hay en este programa." Este enunciado se origina en la paradoja de Berry, así como el de Gödel lo hacía en la de Richard, emparentada con la del mentiroso, basadas ambas en la autorreferencia. La paradoja de Berry, que como se sabe fue planteada por Russell y ha inmortalizado a un librero de Oxford, G.G. Berry, entre los lógicos, se expresa así:

> "el menor entero positivo que no puede especificarse en menos de mil millones de palabras."

Como la cantidad de palabras es finita, y los números son infinitos, la inmensa mayoría de números requiere al menos mil millones de palabras para ser designados. Tomemos al primero de este grupo, y ése será nuestro número que, según lo definimos, no puede especificarse en menos de mil millones de palabras, pero en nuestra expresión inicial lo hemos especificado en mucho menos.

Así como Gödel transformó su enunciado en un enunciado aritmético, Chaitin elabora un enunciado análogo al de la paradoja de Berry al que transforma en un enunciado que se refiere a la complejidad de programas de computación:

> "el primer entero positivo del cual se puede demostrar$_{SAF}$ (en un SAF) que tiene la propiedad de que no puede especificarse$_{MUT}$ (mediante MUT) por un programa de computadora con menos de mil millones (N) de bits."

Aquí SAF indica sistema axiomático formal.

El hecho interesante es que hay un programa de computadora de longitud

$$\log_2 N + c_{SAF}$$

en bits para calcular este número que supuestamente no podía ser calculado por ningún programa de menos de N bits, y la longitud indicada es menor que N.

Para demostrarlo, se hace así:

Se hacen correr todas las demostraciones posibles en el sistema axiomático formal, por orden de tamaño. Se le aplica el algoritmo chequeador de demostraciones a cada una de ellas y se eliminan las inválidas. Se busca entonces la primera que requiere N bits para un entero positivo particular. El algoritmo usado, aunque muy lento, es teóricamente eficaz, es

el chequeador de demostraciones, y su longitud en bits es c_{SAF} , más el número N, cuya longitud en bits es $\log_2 N$.

Y Chaitin construyó el concepto preciso de un número que es absolutamente incompresible, el número omega mayúscula (Ω), la probabilidad de detención de un programa:[1]

$$0 < \Omega = \Sigma_{\text{p se detiene}} 2^{-|p|} < 1$$

que se muestra como absolutamente incompresible, y por consiguiente como un número que es absolutamente azaroso, lo cual mostraría según él la presencia del azar en la matemática pura.

¿Qué hay de común entre todas estas cosas que pueda ayudarnos a comportarnos más confiadamente frente a ellas? Pues bien, hay que recordar el gran papel que juegan las analogías como propulsoras de avances en el conocimiento. el concepto de *compacidad* en los sistemas formales debe éste, su nombre, a la geometría, e indica algo así como homogeneidad en toda la estructura del objeto. En cuanto a definiciones precisas, cada una de ellas ha de darse con respecto al sistema en que se lo define; así, van Fraassen [1987], por ejemplo, define *compacidad-I* y *compacidad-U*, como distintos entre sí y a la vez distintos de lo que llama *vinculación semántica finitaria*, pero aclara que en lógica elemental – por la negación de exclusión – los tres conceptos son equivalentes.[2] Nuestro concepto se corresponde con el último de van Fraassen, y es también el que toma Read [1995] para referirse a los problemas con la regla omega.[3]

Continuando la analogía geométrica, podría decirse que un sistema capaz de generar algoritmos para calcular números encuentra, en cada algoritmo, una "compresión" del número, especialmente cuando el número es irracional y se lo toma como representado por su notación decimal. Como antes dijimos, π, representado decimalmente como 3,141592... , se puede comprimir en el programa que lo calcula, y es, por lo tanto compresible.

Pero el número omega mayúscula es absolutamente incompresible, y haberlo encontrado, aunque ha sido sin duda muy trabajoso y ha requerido una gran dosis de ingenio, tal como ocurrió con el enunciado G de Gödel, aparece ahora mucho más natural que este último en su momento.

Frente a esta situación, y tal como conjetura Chaitin, podría proponerse a los matemáticos dos cuestiones: 1) deberían estar dispuestos, como los físicos, a aceptar la inclusión de axiomas nuevos; 2) también deberían aceptar "inductivamente" que un resultado varias veces repetido podría considerarse verdadero, al menos provisoriamente. ¿Quizás entonces podría justificarse la regla omega de esta manera, sobre una cantidad grande pero finita de premisas?

Notas

[1] Chaitin [1998], passim.

[2] Van Fraassen [1987], pp. 53, 54, 55.

[3] Read [1995], pp. 42 y ss.

Bibliografía

Chaitin [1998]: Chaitin, Gregory, *The Limits of Mathematics*. Singapore. Springer-Verlag.

Read [1995]: Read, Stephen, *Thinking about Logic*. Oxford University Press.

Van Fraassen [1987]: van Fraassen, Bas C., *Semántica Formal y Lógica*. México: Universidad Nacional Autónoma de México.