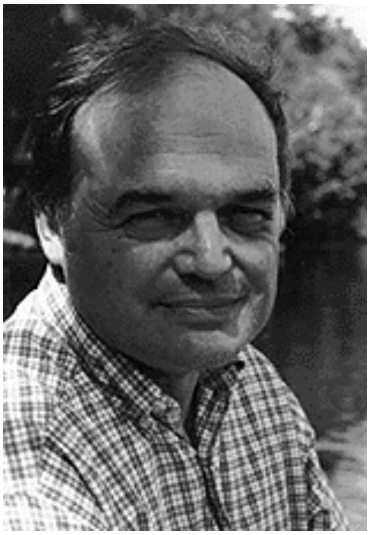
Una demostración del Teorema de Incompletitud de Gödel

por

George Boolos

George Boolos (1940–96) fue catedrático de Filosofía del Massachusetts Institute of Technology. El siguiente artículo apareció publicado con el título *New Proof of the Gödel Incompleteness Theorem* en el *Notices of the American Mathematical Society*, 36 (1989) 388-390). Reproducimos también una carta de Boolos al *Notices*: *A letter from George Boolos*, *Notices of the American Mathematical Society*, 36 (1989) 676. La Gaceta agradece a la American Mathematical Society y a Sally Sedgwick el permiso para la traducción y publicación de este material[1].



George Boolos

De casi todos los teoremas se conocen muchas demostraciones distintas. Además de la primera demostración rigurosa del Teorema Fundamental del Álgebra, Gauss dio otras tres; con posterioridad, diversos autores han encontrado muchas otras. El teorema de Pitágoras, más antiguo y más sencillo que el Teorema Fundamental del Álgebra, tiene cientos de demostraciones. ¿Hay algún gran teorema que sólo tenga una demostración? En este artículo presentamos una nueva demostración sencilla del Teorema de Incompletitud de Gödel en la forma siguiente:
No existe ningún algoritmo cuya salida contenga todos los enunciados verdaderos de la Aritmética y ningún enunciado falso.
Nuestra demostración es bastante diferente de las usuales y presupone sólo una cierta familiaridad con la lógica matemática formal. Es perfectamente completa, excepto por un detalle técnico cuya demostración sólo esbozaré.

[1] Ambos textos de George Boolos aparecen reproducidos asimismo en el libro reciente de Reuben Hersch, *What is Mathematics, really?*, Oxford University Press, Oxford, 1997

Nuestra demostración se sirve de la *paradoja de Berry*. Bertrand Russell, en varios de sus artículos, atribuía a G. C. Berry, bibliotecario de la Universidad de Oxford, la paradoja sobre *el mínimo entero que no puede ser definido con menos de catorce palabras*. La paradoja, desde luego, estriba en que se ha definido el tal entero con trece palabras. Sobre la paradoja de Berry dijo Bertrand Russell en cierta ocasión: "*Tiene el mérito de no salirse del ámbito de los números finitos*"[2]. Antes de comenzar, hemos de hablar, siquiera brevemente, sobre algoritmos y sobre "enunciados de la Aritmética", y sobre lo que "verdadero"o "falso"significan en el contexto que aquí nos concierne. Empecemos por los "enunciados de la Aritmética".

El *lenguaje de la Aritmética* contiene los signos + y × para la adición y la multiplicación, un nombre 0 para el cero, y un signo s para el sucesor (la operación de añadir 1). También contiene el signo de igualdad =, así como los símbolos lógicos ¬ (no), ∧ (y), ∨ (o), → (si ... entonces ...), ⟷ (... sí y sólo si ...), ∀ (para todo) y ∃ (para algún o existe), y los paréntesis. Las variables del lenguaje de la Aritmética son las expresiones $x, x', x'' \ldots$ construidos con los símbolos x y $'$: los valores que pueden tomar estas variables son los números naturales: $\{0, 1, 2, \ldots\}$. Nos referiremos abreviadamente a estas variables con las letras y, z, etc.

Con todo esto ya podemos entender suficientemente bien lo que verdadero y falso significan en el lenguaje de la Aritmética; por ejemplo, $\forall y \exists x, x = sy$ es un enunciado *falso*, por que no es cierto que todo número natural x sea el sucesor de algún número natural y. (El cero es un contraejemplo: no es el sucesor de ningún número *natural*.) Por otro lado,

$$\forall x \exists y, (x = (y + y) \vee x = s(y + y))$$

es un enunciado verdadero: porque para todo número natural x hay un número natural y tal que ora $x = 2y$ bien $x = 2y + 1$. Vemos también que hay muchas nociones que se pueden expresar con el lenguaje de la Aritmética, verbigracia, "ser menor que": $x < y$ puede definirse mediante:

$$\exists z(sz + x = y)$$

(para algún número natural z, el sucesor de z más x da y). Y ahora vemos que

$$\forall x \forall y \, [ss0 \times (x \times x) = (y \times y) \rightarrow x = 0]$$

es —bueno, póngase a prueba, ¿es cierto o falso? (Una buena sugerencia: $\sqrt{2}$ es irracional). Para el propósito que aquí nos ocupan, no hace falta formalizar la sintaxis y la semántica del lenguaje de la Aritmética más allá de lo ya expuesto. Por *algoritmo* entendemos un procedimiento o rutina computacional (sea automática, efectiva o mecánica) de las habituales; por ejemplo, un programa en un lenguaje de computación como *C*, *Basic*, *Lisp*, ... , una máquina

[2]Bertrand Russell, *On Insolubilia and Their Solution by Symbolic Logic* en *Essays in Anlysis*, editores Douglas Lackey y George Brazillier, Nueva York, 1973, p. 210.

de Turing, una caja registradora, un algoritmo de Markov, ... , o lo que sea. Suponemos que el algoritmo tiene una *salida*, las cosas que "imprime" durante o al finalizar la computación. (Desde luego, un algoritmo podría producir una salida *vacía*.) Si el algoritmo es un sistema formal, entonces su salida no es otra cosa que el conjunto de enunciados que se pueden probar dentro del sistema. Aunque el lenguaje de la Aritmética contiene sólo los símbolos s, $+$, y $\times$ de ciertas operaciones, muchos enunciados de Matemáticas se pueden formular como enunciados en el lenguaje de la Aritmética, incluyendo proposiciones tan famosas como el Último Teorema de Fermat, la Conjetura de Goldbach, la Hipótesis de Riemann y la hipótesis comúnmente aceptada de que $P \neq NP$. De manera que si hubiera un algoritmo que imprimiera todos los enunciados verdaderos de la Aritmética sin incluir ninguno falso —lo que según el teorema de Gödel que aquí nos ocupa no es posible— tendríamos una forma de averiguar si estas proposiciones todavía sin dilucidar[3] son ciertas o no, y tendríamos además un procedimiento para decidir si un enunciado cualquiera S que se puede expresar en el lenguaje de Aritmética es cierto: basta con iniciar el algoritmo, y simplemente esperar hasta ver si el algoritmo imprime S o su negación $\neg S$. (A la postre imprimirá exactamente uno de los dos: S o $\neg S$, pues el algoritmo imprime sólo verdades y ninguna falsedad, y, ciertamente, ora S bien $\neg S$ es verdadero.) Pero, por desgracia, no hay que preocuparse por si el algoritmo va a invertir demasiado tiempo en responder a una pregunta concreta cuya respuesta nos interese, porque, como vamos a ver a continuación, no hay ningún algoritmo capaz de hacerlo, ni siquiera uno que fuera absurdamente lento. Para demostrar que no existe ningún algoritmo cuya salida incluya todos los enunciados verdaderos de la Aritmética y sin contener ninguno que sea falso, supondremos que M es un algoritmo cuya salida no contiene ningún enunciado falso de la Aritmética y demostraremos, entonces, como hallar un enunciado verdadero de la Aritmética que no se encuentra en la salida de M, lo que demostrará el teorema. Para cualquier número natural n, denotaremos por $[n]$ la expresión que consta de 0 precedido por n símbolos s de sucesor. Por ejemplo, [3] es $sss0$. Obsérvese que la expresión $[n]$ representa al número n.

Necesitamos una definición más: Diremos que una fórmula $F(x)$ *nombra* al número natural n si el siguiente enunciado es parte de la salida del algoritmo M:

$$\forall x(F(x) \longleftrightarrow x = [n])\,.$$

Obsérvese como la definición de *nombra* hace referencia al algoritmo M. Así, por ejemplo, si

$$\forall x(x + x = ssss0 \longleftrightarrow x = ss0)$$

se encuentra en la salida de M, entonces la fórmula $x + x = ssss0$ nombra al número 2. Ninguna fórmula puede nombrar a dos números diferentes, porque si

$$\forall x((F(x) \longleftrightarrow x = [n])$$

[3]Este artículo es anterior a la demostración de Wiles del Último Teorema de Fermat.

y

$$\forall x(F(x) \longleftrightarrow x = [p])$$

son ambas ciertas, entonces también son ciertas

$$\forall x(x = [n] \longleftrightarrow x = [p])$$

y

$$[n] = [p]\,,$$

y el número n debe ser igual al número p. Además, para cada número i, hay tan sólo una cantidad finita de fórmulas que contienen i símbolos. (Puesto que hay sólo 16 símbolos primitivos en el lenguaje de la Aritmética, hay a lo sumo 16^i fórmulas con i símbolos.) Por tanto, para cada i, hay tan sólo un número finito de números que se pueden nombrar mediante fórmulas que contengan i símbolos. Por consiguiente, para cada m hay un número finito (en realidad, $\leq 16^{m-1} + 16^{m-2} + \ldots + 16^1 + 16^0$) números que se pueden nombrar mediante fórmulas de menos de m símbolos; hay algún número que no se puede nombrar mediante ninguna fórmula que tenga menos de m símbolos; y por consiguiente hay un número más pequeño entre aquellos que no se pueden nombrar mediante fórmulas de menos de m símbolos. Sea $C(x, z)$ una fórmula del lenguaje de la Aritmética que dice que una fórmula que contiene z símbolos nombra a x. El detalle técnico que mencionamos anteriormente y que necesitamos ahora es que cualquiera que sea el algoritmo M, hay una tal fórmula $C(x, z)$. Esbozaremos la construcción de $C(x, z)$ en el comentario (3) de la página 526. Consideremos ahora $B(x, y)$, la fórmula

$$\exists z(z < y \wedge C(x, z))$$

Esta fórmula $B(x, y)$ dice que se puede nombrar a x mediante alguna fórmula que contiene menos de y símbolos.

Se ahora $A(x, y)$ la fórmula

$$(\neg B(x, y) \wedge \forall a(a < x \rightarrow B(a, y)))\,.$$

$A(x, y)$ dice que x es el menor número que no puede ser nombrado mediante una fórmula que contenga menos de y símbolos. Sea k el número de símbolos en $A(x, y)$. $k > 3$. Finalmente, sea $F(x)$ la fórmula

$$\exists y(y = ([10] \times [k] \wedge A(x, y)))\,.$$

$F(x)$ dice que x es el menor número que no se puede nombrar mediante ninguna fórmula que contenga menos de $10k$ símbolos. ¿Cuántos símbolos contiene F? Bueno, [10] contiene 11 símbolos, $[k]$ contiene $k + 1$, $A(x, y)$ contiene k, y hay otros 12 símbolos (puesto que $y = x'$): así que en total $2k + 24$. Como $k > 3$, $2k + 24 < 10k$, y $F(x)$ contiene menos de $10k$ símbolos. Ya observamos anteriormente que para cada m hay un número que es el más pequeño entre los que no pueden ser nombrados mediante fórmulas que contienen menos de

m símbolos. Para $m = 10k$, sea n el menor de tales números. Entonces $F(x)$ no nombra a n; en otras palabras,

$$\forall x(F(x) \longleftrightarrow x = [n])$$

no se encuentra en la salida de M. ¡Pero

$$\forall x(F(x) \longleftrightarrow x = [n])$$

es un enunciado verdadero, puesto que n *es* el menor número no nombrado por ninguna fórmula que contenga menos de $10k$ símbolos! Así que hemos encontrado un enunciado verdadero que no está en la salida de M, a saber,

$$\forall x(F(x) \longleftrightarrow x = [n])$$

Q.E.D.

Algunos comentarios sobre la demostración:

1. En nuestra demostración, los símbolos son las "sílabas", y de la misma manera que "veinticuatro" contiene $4 < 24$ sílabas, $([10] \times k)$ contiene $(k + 15) < 10k$ símbolos.

2. En una nota biográfica[4] sobre Kurt Gödel, Georg Kreis cuenta como Gödel atribuía su éxito no tanto a la inventiva matemática como a prestar atención a las sutilezas filosóficas. Gregory Chaitin comentó en alguna ocasión que una de sus demostraciones de la incompletitud se asemejaba a la paradoja de Berry más que a la paradoja del mentiroso de Epiménides ("Lo que estoy diciendo ahora no es cierto")[5]. La demostración de Chaitin usa la noción de *complejidad* de un número natural, *i.e.*, el número mínimo de instrucciones en el programa que en una máquina de Turing imprime ese número, y de varias otras nociones de la Teoría de Computación. Nuestra demostración no usa ninguna de estas nociones, aunque las observaciones que hemos reseñado de Kreisel y Chaitin, y que el autor leyó más o menos al mismo tiempo, supusieron un acicate para nuestro trabajo.

[4]Georg Kreisel, "Kurt Gödel, 28 April 1906–14 January 1978". *Biographical memoirs of Fellows of the Royal Society* 26 (1980), p. 150.

[5]Como exposición de las demostraciones de incompletitud de Chaitin, recomendamos Martin Davis, "What is computation?" en *Mathematics Today*, editor Lynn Arthur Steeen, Vintage Books, Nueva York, 1980, páginas 241-267. La observación de Chaitin se encuentra en Gregory Chaitin, "Computational complexity and Gödel's incompleteness theorem", (Abstract) *Notices of the American Mathematical Society* 17 (1970), página 672.

3. Vamos ahora a esbozar la construcción de una formula $C(x, z)$ que dice que x es un número nombrado por una fórmula que contiene z símbolos. Las ideas clave son las siguientes:

   (a) considerar que podemos interpretar que algoritmos como M actúan sobre "expresiones", o lo que es lo mismo, sobre sucesiones finitas de símbolos;

   (b) la posibilidad de asignar códigos numéricos, de forma análoga a como se hace con los códigos ASCII, a los símbolos (los lógicos suelen llamar números de Gödel a estos códigos);

   (c) el uso de ciertos trucos de Teoría de Números para codificar expresiones como números y operaciones sobre expresiones como operaciones sobre los números que las codifican;

   (d) y, finalmente, que todas estas operaciones numéricas se pueden definir en términos de la suma, la multiplicación y las nociones de la Lógica.

   De esta forma, las argumentaciones sobre símbolos y expresiones (y de sucesiones finitas de expresiones, etc.) pueden, por consiguiente, codificarse como argumentaciones sobre los números naturales que los codifican. Para construir una fórmula que dice que alguna fórmula que contiene i símbolos nombra a n se escribe una fórmula que dice que hay una sucesión de operaciones del algoritmo M (que operan sobre expresiones) que generan la expresión que consta de $\forall$, $\exists$, los i símbolos de alguna fórmula $F(x)$ del lenguaje de la Aritmética, $\longleftrightarrow$, x, $=$, n símbolos consecutivos de sucesor s, 0 y $\wedge$. La numeración de Gödel y ciertos trucos de Teoría de Números permiten entonces que toda esa discusión de símbolos, sucesiones y operaciones de M pueda ser codificada como fórmulas de la Aritmética.

4. Ambas demostraciones, la nuestra y la estándar, hacen uso de la numeración de Gödel. Además, las verdades indemostrables en nuestra demostración y en la estándar pueden, en ambas casos, obtenerse mediante la substitución de un nombre por un número en una cierta fórmula crucial. Hay, sin embargo, una importante diferencia entre ellas. En la prueba tradicional, el número cuya nombre se substituye es el código de la fórmula en la que se substituye; en la nuestra es el verdadero número del que la fórmula es *verdad*. En vista de esta distinción parece justificado decir que en nuestra demostración, al contrario de lo que sucede en la demostración usual, la *diagonalización* no desempeña papel alguno.

Carta de George Boolos

Algunos lectores de mi demostración del Teorema de Gödel han alabado su brevedad, creyendo, en apariencia, que el uso de la paradoja de Berry es responsable de su concisión. Parece, por tanto, oportuno hacer notar que, siempre y cuando ya se disponga de una sintaxis aritmetizada, se puede dar una demostración aún más corta, en esencia la del propio Gödel en la introducción de su famoso "Sobre proposiciones formalmente indecidibles ... ". A saber:

Digamos que m se aplica sobre n, si $F(n)$ es parte de la salida de M, donde $F(n)$ es la fórmula con número de Gödel m. Sea $A(x, y)$ la expresión "se aplica en"y sea n el número de Gödel de $\neg A(x, x)$. Si n se aplica sobre n, el enunciado falso $\neg A([n], [n])$ es parte de la salida de M, lo que es imposible; por consiguiente, n no se aplica sobre n y $\neg A([n], [n])$ es una verdad que no es parte de la salida de M.

Este argumento oculta la gran cantidad de trabajo que se precisa para construir una fórmula $A(x, y)$ adecuada; la demostración de la existencia de esa fórmula clave $C(x, y)$ en mi demostración requiere al menos el mismo esfuerzo. Lo que al autor considera de particular interés en su demostración basada en la paradoja de Berry no es su brevedad sino que exhibe una razón distinta del porqué de la incompletitud de los algoritmos.

Traducción: Carmelo Alonso Torres