Complejidad y aleatoriedad

Héctor Zenil

¿Cómo puede explicarse la estructura y organización en el Universo a partir de un origen primitivo y desordenado? ¿Qué nos puede decir la computación de ello? ¿Podemos distinguir entre estructura y aleatoriedad? ¿Puede una computadora producir azar?

Formas y patrones en la naturaleza

Algunas partes del Universo parecen ordenadas. La vida en la Tierra es un ejemplo de organización y estructura que contrasta con lo que se considera como el ruido de fondo, producto de la "gran explosión" (o *big bang*) que dio origen al Universo. Es también similar a lo que se puede ver en la pantalla de una televisión analógica cuando no está sintonizado ningún canal (un 1% del ruido en la televisión es, de hecho, este ruido cósmico de fondo).

Hay una teoría matemática extremadamente elegante que describe y puede explicar la aparición de patrones en la naturaleza. Se llama *teoría de la información algorítmica*, y en ella los conceptos de cálculo mecánico y universalidad, desarrollados por Alan M. Turing, son piezas medulares.

Al ver las figuras 1 y 2 se podría creer que las estructuras mostradas en ellas no son producto de un proceso natural, sino de la intervención humana. ¿Cómo pudieron formarse miles de hexágonos de piedra casi perfectos en la naturaleza? ¿Cómo se comunican los hongos unos con otros para colocarse en círculo? ¿Cómo puede la naturaleza reproducir esta estructura aparentemente sin esfuerzo y sin estar guiada por una inteligencia?

En el mundo físico existe una amplia gama de sistemas que se comportan de manera distinta, pero que comparten una característica: presentar aspectos regulares y al mismo tiempo difíciles de predecir. Un ejemplo es el clima. ¿De dónde

Figura 1. El "camino del gigante" (*giant's causeway*), en Irlanda del Norte. Consta de columnas basálticas hexagonales formadas por el relativamente rápido enfriamiento con agua de mar de lava incandescente, ocurrido hace unos 60 millones de años (*fotografía:* H. Zenil, 2009).

Figura 2. Un "anillo de hadas" (*fairy ring*) en el suelo del bosque, generado de manera natural por crecimiento diferencial de hongos en Dundee, Escocia (*fotografía:* H. Zenil, 2012).

surge esta regularidad y complejidad en la naturaleza? ¿Cómo diferenciar entre azar y estructura?

Con el uso de un programa de computación se pueden caracterizar algunas de las propiedades de los números. En el caso de los números que admiten una representación que puede escribirse como una división p/q, donde p y q son números enteros, el algoritmo de la división consiste en el programa de computadora que encuentra cocientes y residuos, y se detiene cuando el residuo es cero. Si el algoritmo no se detiene sino que entra en un ciclo infinito, entonces se dice que el número es *racional* y periódico. También hay números no racionales; es decir, que no se pueden expresar como la división de dos enteros p y q de la forma p/q. En el caso de números como π y raíz cuadrada de 2, el algoritmo produce una expansión infinita no periódica que parece tener complejidad infinita, ya que en su expansión decimal puede encontrarse cualquier número, no importa cuán largo. En estos números como π y raíz cuadrada de 2, la expansión decimal parece producir dígitos sin ningún orden aparente, a pesar de ser el resultado de operaciones muy simples.

El mundo de los programas simples

Un programa de computadora es simplemente una serie de instrucciones que se pueden seguir mecánicamente paso a paso. Un algoritmo es similar, pero un mismo algoritmo puede tener diferentes programas de cómputo que lo ejecuten y produzcan el mismo resultado. El concepto de máquina de Turing, ya analizado en otros artículos de este número de *Ciencia*, ha sido esencial para entender el concepto de algoritmo.

Una máquina de Turing es una computadora ideal que mantiene un estado interno que puede cambiar mientras lee una cinta que contiene símbolos indivisibles. La máquina lee un símbolo a la vez, y puede sustituir el símbolo por otro o dejarlo igual, dependiendo del estado interno en que se encuentre y el contenido de la cinta, que actúa como memoria ilimitada.

Las computadoras digitales de hoy pueden considerarse en cierta forma implementaciones de máquinas de Turing y, gracias a una propiedad fundamental de éstas (la *universalidad*), el lector puede sustituir el concepto de máquina de Turing por el concepto intuitivo de computadora digital común, con la que seguramente está más familiarizado. La universalidad es la propiedad que tiene una máquina de Turing de ser programada para comportarse como cualquier otra máquina de Turing. Una máquina *universal* de Turing es, en principio, capaz de producir cualquier grado de "complejidad". Por ejemplo, el tipo de complejidad (o aleatoriedad) que uno puede ver en la expansión decimal de π.

La complejidad no necesariamente proviene de un algoritmo complejo o complicado. Si la constante ma-

temática π = 3.141592654… puede portar tal aparente complejidad en su expansión decimal (ya que no presenta repeticiones, pues no es un número racional), una pregunta legítima sería: ¿qué tan comunes son los algoritmo o programas de computadora que, al ser ejecutados, produzcan la misma complejidad de, por ejemplo, los dígitos de π? Hay programas de computadora muy simples (cortos) capaces de producir cualquier parte de la expansión decimal de π, por lo que a π se le conoce como un número *calculable* (o *computable*).

¿Cuál es el programa de cómputo más simple que produce un comportamiento aparentemente complejo o aleatorio como π? Stephen Wolfram encontró un programa extremadamente sencillo (llamado autómata celular "regla 30"; veáse la Figura 3), cuya evolución no parece presentar ninguna estructura regular (incluso ejecutando millones de pasos), tal como lo hace π.

Un *autómata celular* es un modelo de computación que comienza a partir de una condición inicial, por ejemplo una celda negra a la que se le aplican las reglas del autómata celular para producir una nueva configuración. En el caso de autómatas celulares unidimensionales, como la regla 30 de Wolfram (véase Figura 3), cada nueva configuración es un renglón obtenido al aplicar la regla descrita en la parte superior a cada una de las celdas del renglón, para producir el siguiente. Los autómatas celulares elementales de Wolfram consideran únicamente los vecinos más cercanos para determinar el color de la celda en el renglón siguiente. Para blanco y negro (o cero y uno) hay solamente 256 posibles reglas que Wolfram estudia a detalle.

¿Qué hay de la forma en que crecen los seres vivos? ¿Son sus mecanismos muy distintos de los que dan origen a fenómenos como los hexágonos de Irlanda o los anillos de hadas? Si en los programas de computadora es tan sencillo producir esta complejidad y estructura, es posible que la naturaleza explote este tipo de recursos computacionales para generar sus patrones (véase Figura 4). En otras palabras, algunos de los patrones que presenta la naturaleza podrían ser resultado de programas que son a la vez simples en descripción, pero muy ricos en comportamiento.

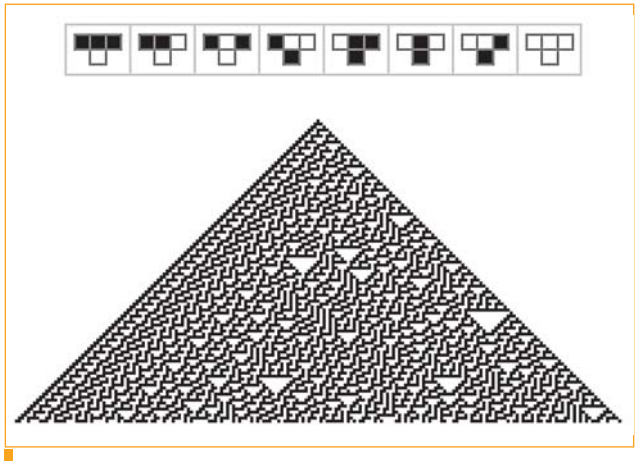
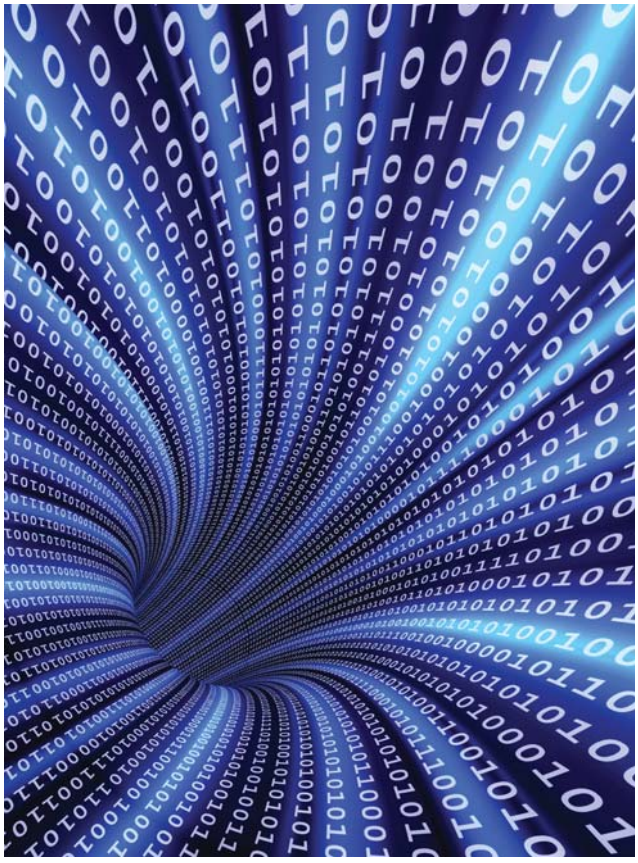


Figura 3. Evolución determinista del autómata celular elemental regla 30 (en esta imagen se muestran 100 pasos o renglones). La regla 30 genera un tipo de comportamiento difícil de predecir si no se hace correr la regla paso a paso, incluso si se comienza de la condición inicial más simple (una celda negra). El ícono en la parte superior de la evolución del programa muestra la regla que se le aplica a cada renglón para producir el siguiente. La regla 30 es, bajo casi cualquier estándar, el programa de computadora más simple que produce aparente aleatoriedad.

Probabilidad algorítmica

A Leonid Levin no le interesaba la pregunta común de si un programa de computadora termina o

Figura 4. Concha de *Conus marmoreus,* un caracol altamente venenoso. Se puede encontrar en el Océano Índico y en el Pacífico. Las conchas de caracol crecen por capas, y la pigmentación sigue un tipo de regla que produce formas (triángulos) en la superficie. Las similitudes con los programas de computadora de Wolfram son evidentes (*fotografía y composición:* H. Zenil, 2012).



```
long k=4e3,p,a[337],q,t=1e3;
main(j){for(;a[j=q=0]+=2,k;)
for(p=1+2*k;j<337;q=a[j]*k+q%p*t,a[j++]=q/p)
k!=j>2?:printf("%.3d",a[j2]%t+q/p/t);}
```

Figura 5. Un programa escrito en el lenguaje C de 141 caracteres, que al ejecutarse imprime mil dígitos decimales de π (Gjerrit Meinsma, www.boo.net/~jasonp/pipage.html). Es mucho más probable generar este programa (o uno equivalente) tecleando al azar que acertar los mil primeros dígitos de π. Este programa, comprimido con GZIP, ocupa solamente 713 bytes.

no, sino la frecuencia con que aparece un resultado cada vez que se ejecuta tal programa. Lo que un programa de cómputo produce solamente puede considerarse un resultado cuando el programa se detiene, pues mientras no se detenga, el programa puede reescribir lo que ya había escrito. Imaginemos que escribimos un programa de cómputo al azar para ejecutarse en una computadora (por ejemplo, una máquina universal de Turing que garantiza que el programa, sea cual sea, pueda ejecutarse). La pregunta es: ¿hay algún resultado que pueda esperarse con mayor o menor frecuencia, o el resultado que imprima un programa al azar va a ser tan arbitrario como el programa que se ejecuta?

La respuesta que Levin propone es sorprendente y constituye uno de los resultados más importantes en las ciencias de la computación. Levin concluye que el resultado de un programa de computadora cuyas instrucciones son elegidas al azar no es un resultado al azar, sino uno que favorece dramáticamente la producción de resultados simples; todo lo contrario al azar. En otras palabras, ¡alimentar una computadora con programas aleatorios produce patrones y estructuras! ¿Cómo es esto posible?

La noción detrás de la medida de Levin es relativamente sencilla, pero muy poderosa. Si uno deseara producir al azar los dígitos de la constante matemática π, tendría que intentarlo una y otra vez hasta conseguir los primeros números consecutivos correspondientes a un segmento inicial de la expansión decimal de π (por ejemplo, los primeros diez dígitos: 141592654 después de la parte entera de π, que empieza con 3). En este caso, la probabilidad de producir al azar estos diez dígitos decimales consecutivos es muy pequeña: de exactamente 1 entre los diez posibles dígitos (del 1 al 10); es decir, 1/10 para cada dígito multiplicado por el número deseado de dígitos consecutivos, en este caso $(1/10)^{10}$ para los primeros 10 dígitos.

Pero si en lugar de lanzar dígitos al azar para producir un segmento de la expansión decimal de π, se lanzaran programas de computadora al azar para ejecutarlos en una computadora, la situación cambiaría dramáticamente debido a que existen programas de computadora muy cortos que producen segmentos muy largos de la expansión de π. Por ejemplo, si la expansión de π se escribiera en binario, la probabilidad de producir

π con un programa de computadora puede escribirse como $1/2^{\text{longitud}(p)}$, con *p* siendo el programa que produce π.

Hay que recordar que la probabilidad de producir al azar un dígito decimal de π es 1/10, y la probabilidad de producir al azar dos dígitos consecutivos de π es 1/10 multiplicado por 1/10; es decir, $(1/10)^2$. En general, la probabilidad es $(1/10)^n$ para *n* dígitos. Sin embargo, lo que nos dice la probabilidad algorítmica es que objetos que tienen estructura, como π, que es el resultado de dividir la circunferencia de cualquier círculo entre su diámetro, y para los cuales hay un programa corto de computadora que los produce, la probabilidad de encontrar el programa que produce π es mucho mayor comparada con la probabilidad de producir el objeto original dígito por dígito.

Este concepto de probabilidad algorítmica, desarrollado por Levin, fue también sugerido por Ray Solomonoff, en el contexto de la inteligencia artificial, como solución al *problema de predicción* (o *inducción*), ya que permitiría elegir, de entre una sucesión de eventos, el más probable de ocurrir, basado en el tamaño del programa de computadora más corto que produjese el evento. Esto permite que un sistema de inteligencia artificial dotado de un método de predicción basado en probabilidad algorítmica favorezca los eventos más simples con mayor estructura, detectando regularidades (objetos con baja complejidad de Kolmogorov).

Complejidad de Kolmogorov

Imaginemos que se nos proporcionan dos secuencias de ceros y unos (bits) y que se nos pregunta cuál de ellas parece resultado de un proceso que genera cada bit de la secuencia al azar. Digamos que las secuencias son:

(a)

0101010101010101010101010101010101010101010101010101010101010101

(b)

1111001001011110001100011111001101101001110111011011111001010110100001

La cadena (a) permite una descripción de 36 caracteres: "treinta y cinco repeticiones de «01» (si el lector

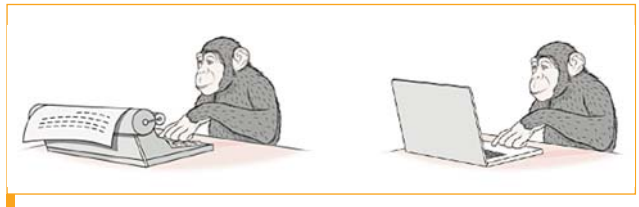


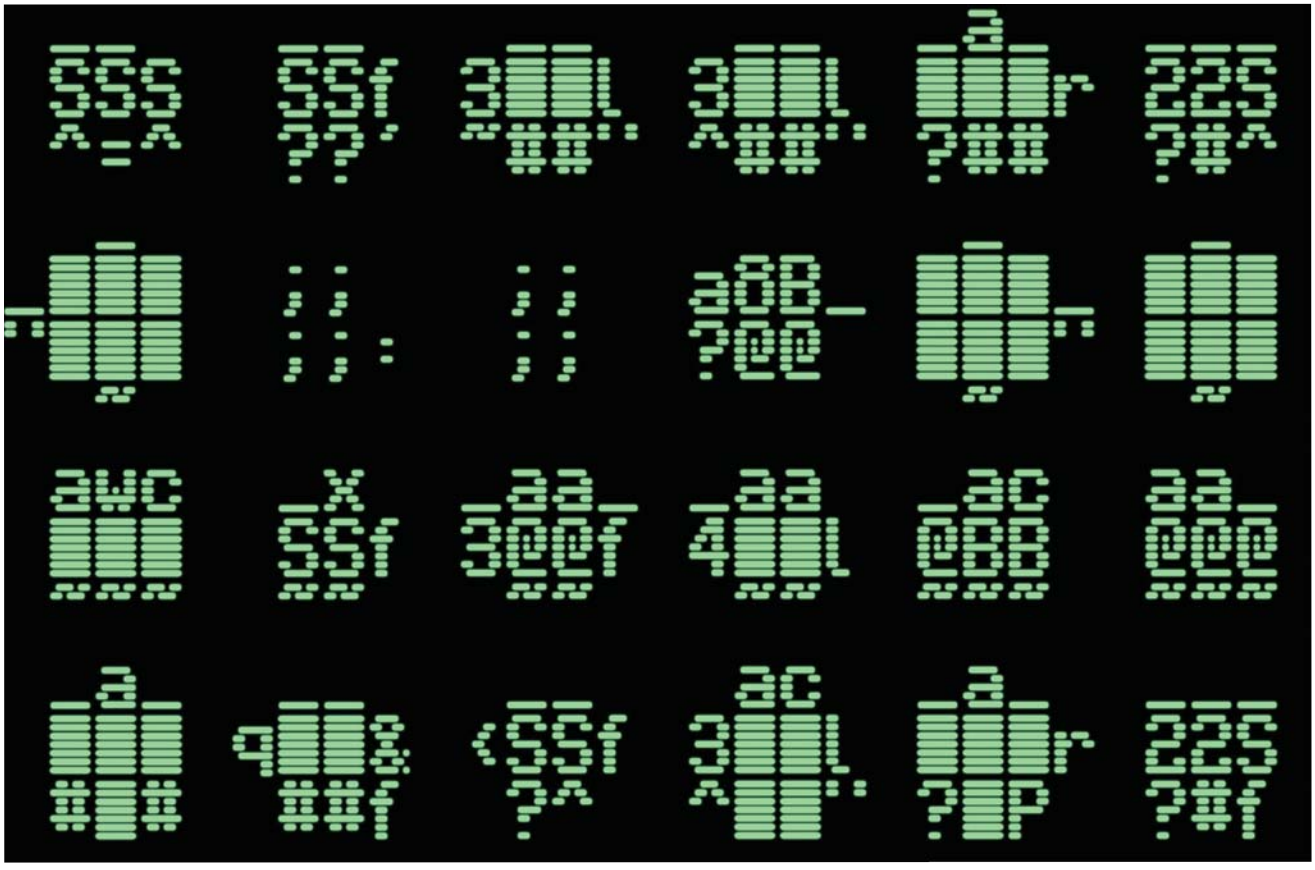
Figura 6. Un mono tecleando en una máquina de escribir contra un mono tecleando en una computadora. El segundo tiene mejores probabilidades de producir un objeto con estructura, ya que los objetos con estructura tienen descripciones cortas. En cambio, para el mono en la máquina de escribir, los objetos con estructura y los objetos aleatorios tienen la misma probabilidad de ser producidos (imagen: H. Zenil, 2012).

lo desea, puede evitar escribir «01» con sólo usar unos pocos caracteres más). Pero no parece haber forma de describir de modo tan compacto la cadena (b), que ha sido producida aleatoriamente (como resultado de arrojar una moneda y asignarle 0 a la cara y 1 al reverso). De este modo, (b) es más compleja que (a). Los objetos más complejos son aquellos para los que no existe ninguna descripción (compresión) más pequeña que tales objetos.

Para resolver si alguna de las dos secuencias es, sin lugar a dudas, más sencilla que la otra, o si la aparente repetición de la primera secuencia puede realmente aprovecharse a pesar de repetirse sólo tres veces, es necesario fijar un lenguaje que no permita las ambigüedades del lenguaje natural. Este lenguaje es el que utilizan las computadoras, y la complejidad algorítmica propuesta por el ruso Andrei Kolmogorov y el argentino-estadounidense Gregorio Chaitin es la longitud del programa más corto, medido en número de bits, que produce la secuencia cuando se ejecuta en una computadora (máquina universal de Turing).

A este tipo de complejidad algorítmica se le llama *complejidad de Kolmogorov* o *de Kolmogorov-Chaitin*, y es considerada una medida universal de complejidad porque utiliza el modelo universal de Turing, y es lo suficientemente general para capturar un gran número de propiedades intuitivas que solemos relacionar con la aleatoriedad (por ejemplo, la impredecibilidad y la ausencia de regularidades).

Sin embargo, esta generalidad trae como costo que no exista un algoritmo que, dado un objeto, prediga el tamaño del programa más corto que genere el objeto; es decir, su complejidad algorítmica. Esto ocurre por la misma razón por la que existen números no calculables: porque las máquinas de Turing no pueden resolver el problema de la detención (es decir, no puede saberse con certeza si un programa va a detenerse o no), lo que significa que no se puede medir con absoluta certeza la complejidad algorítmica de un objeto, pero se le puede aproximar encontrando programas cortos que generen el objeto en cuestión.



A manera de conclusión

Los métodos para aproximar la complejidad de Kolmogorov han producido muchas aplicaciones, algunas sorprendentes, que van desde la clasificación de idiomas por familias lingüísticas y de especies animales, hasta la detección de correo no deseado (*spam*), la determinación de autoría o plagio de un texto, y la clasificación de imágenes, por mencionar algunos ejemplos. Esto es prueba del poder de los conceptos de complejidad algorítmica discutidos aquí, y en los que el trabajo de Alan Turing ha sido un pilar fundamental.

Héctor Zenil es investigador en el Laboratorio de Comportamiento y Teoría de la Evolución del Departamento de Ciencias de la Computación de la Universidad de Sheffield, Inglaterra, y responsable del grupo Algorithmic Nature, en Francia. Es doctor en Ciencias por la Universidad de Lille 1 y maestro en Filosofía de la Ciencia por la Sorbona de París. Cursó la licenciatura en Matemáticas en la Facultad de Ciencias de la Universidad Nacional Autónoma de México (UNAM). Es miembro del Comité Consultivo del Centenario de Alan Turing, en el Reino Unido, *senior research associate* para Wolfram Research, y miembro del Sistema Nacional de Investigadores en el extranjero (SNI, Conacyt). Es autor de más de 30 artículos y libros.

hectorz@labores.eu

Lecturas recomendadas

Chaitin, Gregory (2011), *Matemáticas, complejidad y filosofía/Mathematics, complexity and philosophy*, edición español/inglés, Valparaíso, Midas.

Gleick, James (2012), *The information: a history, a theory, a flood*, Nueva York, Fourth Estate.

Joosten, Joost J., Fernando Soler-Toscano y Héctor Zenil (2011), "Complejidad descriptiva y computacional en máquinas de Turing pequeñas", *Lógica universal e unidade da ciência*, Centro de Filosofia das Ciências da Universidade de Lisboa. Disponible en <arxiv.org/ftp/arxiv/papers/1010/1010.1328.pdf>.

Juárez Martínez, Genaro, Héctor Zenil y Christopher Rhodes Stevens (2011), *Sistemas complejos como modelos de computación*, Londres, Luniver Press. Disponible en <http://uncomp.uwe.ac.uk/WCSCM2011/Principal.html>.

Wolfram, Stephen (2002), *A new kind of science*, Illinois, Wolfram Media.