comunicación ::
matematicalia
revista digital de divulgación matemática
Vol. 2, no. 2 (abr. 2006)

Lo que los computadores *no* pueden hacer (*)

Mike Yates
University of Manchester y University of Wales, Bangor
página web: http://www.bangor.ac.uk/~mas041

El profesor P.N. Furbank, editor general de las *Obras Completas* de Turing [F], describe a Alan Turing como "una de las principales figuras de la ciencia del siglo XX".

Hace sesenta[1] años que se publicó el artículo más famoso de Turing, en el que se introduce la idea de una *Máquina de Computación Universal*, diez años antes de que el primer ordenador digital de programa almacenado funcionase realmente.

Figura 1. Alan Turing.

Este es sólo uno de sus tantos logros. Hoy en día se sabe que el trabajo de Turing para descifrar el código alemán Enigma en Bletchley Park durante la Segunda Guerra Mundial supuso una contribución decisiva a la victoria aliada, aunque este hecho permaneció desconocido incluso para sus amigos más cercanos hasta después de su trágica muerte por ingestión de cianuro potásico en 1954.

El trabajo de Turing durante la guerra marcó significativamente la importancia posterior de las instalaciones de computación mecánicas. Aunque buena parte del trabajo repetitivo se realizaba mecánicamente, también estuvo involucrado un enorme equipo de "computadores" humanos.

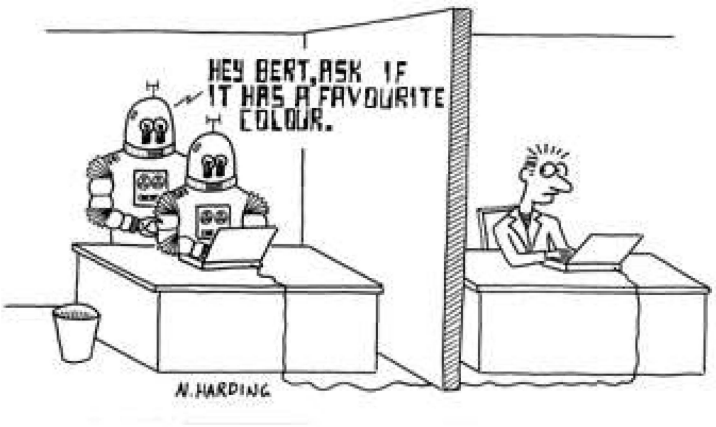
Figura 2a. La máquina Enigma.
(Fuente de la imagen: AGN, University of Hamburg, Copyright 1995, Morton Swimmer).

Figura 2b. Ampliación de los rotores de codificación.

Otra característica de su trabajo en tiempo de guerra fue el uso de la teoría de probabilidades. Parte del trabajo de Turing en este área fue altamente innovador, siendo reconocido después de la guerra a través de las publicaciones de su hasta entonces asistente (luego profesor) Jack Good, sin referencia a sus aplicaciones bélicas.

El interés de Turing en la computación continuó en la posguerra, cuando trabajó en el NPL (*National Physical Laboratory*, Laboratorio Nacional de Física) en el desarrollo de un ordenador de programa almacenado (el ACE o *Automatic Computing Engine*, Máquina de Computación Automática). En 1948 se trasladó a Manchester, donde el primer ordenador digital de programa almacenado funcionó ese mismo año.

Aunque su conexión con ese ordenador real fue, a lo sumo, tenue, Turing hizo contribuciones significativas a la teoría de la computación, en particular a la inteligencia artificial (el test de Turing), la arquitectura de computadores (el ACE) y la ingeniería del software. Debido precisamente a su contribución en este campo existe actualmente el prestigioso Premio Turing en ciencias de la computación.



En el Test de Turing de inteligencia artificial, un observador tiene que distinguir entre la máquina y un humano realizando una serie de preguntas a través de un enlace informático.

El problema de la parada

Como un ejemplo de su forma de pensar, veamos una prueba de que no existe ningún método general que permita predecir, una vez que un ordenador ha empezado un cálculo, si dicho cálculo terminará en una respuesta. Este problema es conocido como *problema de la parada para las máquinas de Turing* y fue resuelto por primera vez en la publicación [T], datada en 1937, en la que Turing introdujo las máquinas que llevan su nombre.

Para poder analizar la demostración de este resultado, es necesario comentar algunas cosas sobre recuentos y listas o *sucesiones*. Decimos que los elementos de un conjunto son *contables* si se pueden enumerar en una única sucesión.

El conjunto de los números naturales se puede enumerar sin problema: 0, 1, 2, 3,... y así hasta el infinito. Para enumerar los enteros, positivos y negativos, en una única sucesión, podemos escribir 0, 1, -1, 2, -2, 3, -3,... y así sucesivamente, con lo que de nuevo no tenemos ningún problema.

Las fracciones o *racionales* llevan un poco más de trabajo. Es usual hacerlo en dos dimensiones, usando una tabla o *matriz*. Veamos qué ocurre con los racionales positivos; a los negativos se extiende como hicimos con los enteros.

Existen muchas repeticiones –para empezar, todos los elementos de la diagonal son iguales–, por lo que este algoritmo es poco eficiente. Pero funciona. Continuando *indefinidamente*, cualquier fracción estará en algún lugar de esta matriz. Para escribir la matriz como una sucesión, tomamos las diagonales de SO (suroeste) a NE (noreste), con lo que obtenemos:

1, 1/2, 2, 1/3, 2/2, 3, 1/4, 2/3, 3/2, 4,...

El siguiente paso es probar un teorema muy famoso, el Teorema de Cantor, que dice que los números *reales* no son contables de esta manera. El conjunto de los números reales incluye números como Pi (3'14159...) que no pueden ser escritos como un cociente de dos enteros.

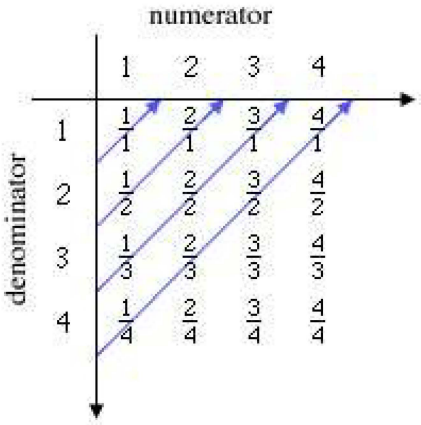


Figura 3. Tabla de fracciones. Las fracciones pueden ser enumeradas tabulándolas y luego contándolas a lo largo de las diagonales, mostradas en azul.

Demostración del Teorema de Cantor

Mostraremos que no podemos contar todas las sucesiones binarias o, en otras palabras, las sucesiones infinitas de ceros y unos.

Supongamos que pudiéramos y veamos que obtenemos una contradicción. Etiquetamos cada sucesión binaria B_1, B_2, B_3,... *ad infinitum*. Como anteriormente, hagamos una lista con los elementos de cada sucesión en una tabla o matriz.

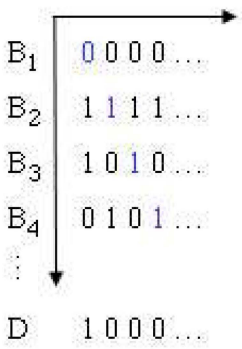


Figura 4. Tabla de sucesiones binarias. Una posible lista de sucesiones binarias, la sucesión D, está construida invirtiendo los elementos de la diagonal, mostrada en azul.

Definimos ahora una sucesión binaria D, eligiendo un 0 como primer término de la sucesión D si 1 es el primer término de B_1 y un 1 si es 0. Entonces, elegimos un 0 como *segundo* término de D si 1 es el segundo término de B_2 y un 1 si es 0, y así sucesivamente. La sucesión binaria resultante, D, no puede estar en la lista, porque si lo estuviera entonces tendría que coincidir con alguna sucesión B_n para algún n. Pero nos hemos asegurado deliberadamente que la n-ésima columna de D difiere de la de B_n, lo que supone una contradicción.

Sin importar cómo hagamos la lista de las sucesiones binarias, siempre encontraremos una nueva sucesión, D, que no está en la lista.

Este procedimiento se denomina *diagonalización*. Como se puede ver, hemos obtenido una regla simple para ello, de tal manera que dada una regla para contar una lista de números binarios siempre tendríamos otra regla para calcular un número binario diagonal que no está en esa lista.

El argumento de Turing

Finalmente, daremos una idea que cómo el argumento de Turing (relacionado con el aún más famoso argumento de Kurt Gödel en 1931) lleva este razonamiento un paso adelante.

La prueba bosquejada aquí no es la original de Turing, pero está bastante relacionada. Gran parte del artículo clásico de Turing está dedicado a describir su noción de máquina de computación y la universalidad de ésta. Cualquier cosa que pueda ser calculada a través de una lista finita de reglas, puede ser computada por una de estas máquinas.

Brevemente, podemos pensar en una máquina de Turing como en una caja negra que realiza un cálculo de algún tipo sobre un número de *entrada*. Si el cálculo llega a una conclusión, o se *para*, entonces se devuelve un número de salida. En caso contrario, la máquina continúa indefinidamente. Existe un número infinito de máquinas de Turing, puesto que hay un número infinito de cálculos que se pueden realizar con una lista finita de reglas.

Una de las consecuencias de la teoría de Turing es que existe una *Máquina de Turing Universal*: en otras palabras, una máquina que puede simular todas las posibles máquinas de Turing. Esto significa que podemos pensar en las máquinas de Turing como contables y enumeradas T_1, T_2, T_3,... por una Máquina Universal a través de algún tipo de listado alfabético. Turing usó este hecho para dar su propia versión del Teorema de Gödel, el *problema de la parada*: no existe un procedimiento mecánico que permita decidir si una máquina de Turing se parará o no, dada una cierta entrada.

La irresolubilidad del problema de parada

Representemos el resultado de usar la n-ésima máquina de Turing, T_n, en la entrada i como $T_n(i)$.

Supongamos que exista un procedimiento para decidir si $T_n(i)$ se detiene o no para todos los valores de n y de i. Por un proceso de diagonalización similar al anterior, podemos definir una nueva máquina de Turing, digamos D, que se parará para todas las entradas y devolverá la siguiente salida para la entrada i:

$$0, \text{ si } T_i(i) \text{ no se para,}$$
$$T_i(i)+1, \text{ si } T_i(i) \text{ se para.}$$

Pero esta máquina D debe ser una de esas máquinas. En otras palabras, D debe ser T_d para algún d. Sin embargo, definimos la máquina D para que diera una respuesta diferente que T_d con entrada d: una contradicción.

La sofisticación extra que encontramos aquí con respecto al argumento de diagonalización original se apoya en que (1) todo el listado realizado es computable, y (2) una máquina T_n puede o no pararse al llevar a cabo sus cálculos.

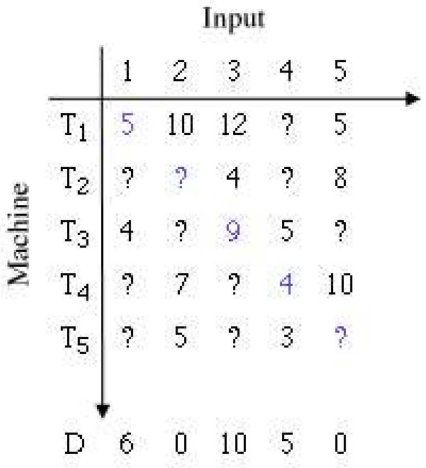
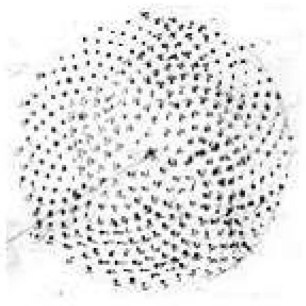


Figura 5. Se podría utilizar una regla de parada para realizar una tabla de la salida $T_n(i)$, usando un signo de interrogación para representar cálculos que nunca se paran. Esta tabla es sólo ilustrativa; sus contenidos no han sido elegidos con ningún orden particular de máquinas de Turing en mente.

Nada de esto forma parte del argumento de diagonalización original de Cantor. Este tipo de diagonalización computable fue utilizado por primera vez en el trabajo pionero de Gödel, Turing y otros en la década anterior a la Segunda Guerra Mundial, y ha permanecido como una técnica importante a lo largo del tiempo. El trabajo duro consiste realmente en la formulación de las diversas definiciones de computabilidad, ¡pero eso es otra historia!



¿Qué es la vida?

En los últimos años antes de su muerte, Turing estaba trabajando en algo totalmente diferente, algo que había estado cercano a su corazón desde sus días escolares: la *morfogénesis*, el origen de la forma biológica.

¿Cómo pueden simples células saber cómo crecer en formas estructuradas relativamente enormes? La idea crucial de que la información genética podría ser almacenada a nivel molecular había sido descrita en la charla de Schrödinger de 1943 *What is life?*, y Crick y Watson estaban ocupados en esos momentos en el descubrimiento de ese secreto, a través de la estructura del ADN. Dada la

Figura 6. Girasol meticulosamente dibujado a mano por Turing.

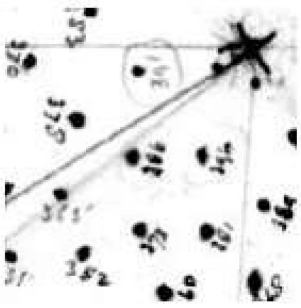


Figura 7. Una sección ampliada.

producción de moléculas por los genes, Turing estaba buscando una explicación de cómo una sopa química podía dar lugar a un patrón biológico.

El primer objetivo de su teoría fue un intento de resolver el problema clásico de la filotaxia, la disposición de las hojas en una planta. Una de las características de este tema, conocida desde los tiempos de Kepler, era la aparición natural de la sucesión de Fibonacci 1, 2, 3, 5, 8, 13, 21,... Por tanto, estaba claro que las matemáticas jugaban un papel importante. [Para más información sobre la sucesión de Fibonacci, ver *The life and numbers of Fibonacci* en el número 3 de *Plus Magazine*].

Turing también propuso que el patrón de las marcas de los animales seguía reglas matemáticas debido a señales químicas. Esta idea tuvo suertes dispares, aunque recientemente el interés de los biólogos se ha visto revitalizado. Usando su teoría, investigadores japoneses han observado cambios predichos por Turing en los patrones del pez cebra.

Lecturas adicionales

Véase la página web indicada para más detalles sobre las *Obras Completas* de Turing:

The Alan Turing bibliography, por Andrew Hodges.

El trabajo definitivo sobre su vida (de lectura obligada) es:

Andrew Hodges: *Alan Turing: The Enigma*. Hardback version: Burnett Books, 1983; paperback version: Vintage Books, 1992.

Un nuevo ángulo sobre Turing puede ser encontrado en:

Andrew Hodges: *Turing*. En la serie *The great philosophers*, Phoenix, 1997.

Una guía sobre sus días escolares es:

D'Arcy Wentworth Thompson: *On growth and form*. Cambridge University Press, 1917 (segunda edición, 1942).

Referencias

[F] P.N. Furbank (ed.): *Collected Works of A.M. Turing*. North−Holland, Elsevier Science.

[T] A. Turing: On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society (Series 2)*, vol. 42 (1936-7), pp. 230−265.

---

[1] En la actualidad, casi setenta. Obsérvese que el artículo que aquí se traduce fue publicado en 1998 (N. del T.).

Sobre el autor

Mike Yates es profesor emérito de la Universidad de Manchester, y profesor honorario de la Universidad de Gales en Bangor. Jubilado anticipadamente en la Universidad de Manchester, su principal motivación ha consistido en hacer las matemáticas más interesantes y accesibles, y su imagen pública menos intimidatoria. Entre sus méritos figura el haber sido conferenciante invitado en el International Congress of Mathematicians celebrado en Vancouver (Canadá) en 1974. Es el editor del cuarto y último volumen de las obras completas de Alan Turing, publicado por Elsevier en diciembre de 2001.

Sobre el traductor

David Iglesias Ponte es licenciado (1999) y doctor (2003) en Matemáticas por la Universidad de La Laguna. Tras disfrutar de una beca Fulbright en Penn State University (USA) durante los años 2004 y 2005, actualmente es contratado Juan de la Cierva en el Instituto de Matemáticas y Física Fundamental del Consejo Superior de Investigaciones Científicas. Desarrolla su investigación en el ámbito de la geometría diferencial.

---

matematicalia
revista digital de divulgación matemática

(*) Este artículo apareció en el número 5 (mayo 1998) de *Plus Magazine*. *Matematicalia* agradece a los responsables del Millennium Mathematics Project de la Universidad de Cambridge la autorización para publicar su traducción al castellano. (Traductor: David Iglesias Ponte).