La máquina de Turing

Article · January 2000

CITATIONS
0

READS
1,266

1 author:

Manuel Alfonseca
Universidad Autónoma de Madrid
183 PUBLICATIONS **1,100** CITATIONS

LA MÁQUINA DE TURING

Manuel Alfonseca

En 1931, Kurt Gödel (1906-1978) publicó su famoso artículo *Sobre las Proposiciones Formalmente Indecidibles en Principia Mathematica y Sistemas Relacionados*, que quizá sea la realización matemática más importante del siglo XX. En síntesis, el teorema de Gödel demuestra que toda formulación axiomática consistente de la teoría de números contiene proposiciones indecidibles: siempre habrá en ella afirmaciones verdaderas que no se pueden demostrar.

En 1937, el matemático inglés Alan Mathison Turing (1912-1953) publicó otro artículo famoso (sobre los *Números Calculables*), que desarrolló el teorema de Gödel y que puede considerarse el origen oficial de la informática teórica. En este artículo introdujo la *máquina de Turing*, una entidad matemática abstracta que formalizó el concepto de *algoritmo* y resultó ser la precursora de las computadoras digitales. Con ayuda de su máquina, Turing pudo demostrar que existen problemas irresolubles, tales que ninguna máquina de Turing (y, por ende, ningún ordenador) será capaz de obtener su solución, por lo que a Alan Turing se le considera el padre de la *teoría de la computabilidad*.

Una de las fotos de Turing hechas tras su elección como fellow *de la* Royal Society *(*National Portrait Gallery, *Londres).*

33

Una máquina de Turing puede considerarse como una cinta infinita dividida en casillas, cada una de las cuales contiene un símbolo. Sobre dicha cinta actúa un dispositivo que puede adoptar diversos estados y que, en cada instante, lee un símbolo de la casilla sobre la que está situado. En función del símbolo que ha leído y del estado en que se encuentra, realiza las tres acciones siguientes: pasa a un nuevo estado, imprime un símbolo en lugar del que acaba de leer, y se desplaza una posición hacia la izquierda, o hacia la derecha, o bien la máquina se para.

El funcionamiento de una máquina de Turing puede representarse mediante una tabla de doble entrada. Las filas están encabezadas por los estados, las columnas por los símbolos escritos en la cinta. En cada posición de la tabla hay tres elementos: el estado siguiente, el símbolo que se escribe en la cinta y el movimiento de la cabeza (I,D,P).

	1	0	b
p	q0D	p0I	rbD
q	q1D	q0D	p0I
r		r1D	sbP
s			

Tabla de una máquina de Turing

También puede haber posiciones en blanco. Por ejemplo, la tabla de una máquina de Turing podría ser como se muestra en la página anterior.

En las casillas de la cinta de esta máquina de Turing puede haber tres símbolos: 0, 1, o la casilla puede estar en blanco (lo que se representa en la tabla con la letra b). Observando la tabla, es fácil ver que los estados de esta máquina corresponden a las acciones siguientes:

Estado p: mientras encuentra el símbolo 0, lo ignora y avanza hacia la izquierda. En cuanto encuentra el símbolo 1, lo sustituye por 0, pasa al estado q y avanza hacia la derecha. Si encuentra una casilla en blanco, pasa al estado r y avanza hacia la derecha.

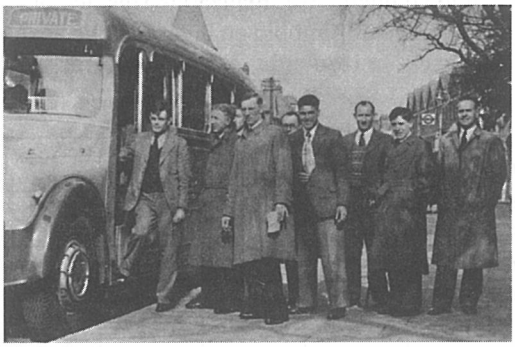
Estado q: mientras encuentra los símbolos 0 y 1, los ignora y avanza hacia la derecha. En cuanto encuentra un blanco, escribe un cero, pasa al estado p y avanza hacia la izquierda. Su función, por tanto, es añadir un 0 al final de la cadena de ceros y unos, sobre la primera casilla en blanco situada a la derecha de la misma.

Estado r: mientras encuentra 0, lo sustituye por 1 y avanza hacia la derecha. En cuanto encuentra una casilla en blanco, pasa al estado final y se detiene la máquina.

Al principio, la máquina está en el estado p, n casillas consecutivas de la cinta de entrada contienen un uno, las restantes están en blanco, y la cabeza lectora apunta al último uno. Pensando un poco se verá que, cuando la máquina se pare, la cinta contendrá 2n unos. En cierto modo, esta máquina de Turing multiplica por dos.

33

El concepto de máquina de Turing es tan general y potente, que es posible construir una máquina que sea capaz de simular el comportamiento de otra máquina de Turing cualquiera. Esto es lo que se llama *máquina de Turing universal*. Gracias a su existencia, podemos disponer de ordenadores electrónicos, que no son más que máquinas generalizadas capaces de realizar cualquier cálculo computable.



Alan Turing (primero por la izquierda) subiendo al autobús en 1946 junto con otros miembros del Walton Athletic Club.

Estudiando la máquina que había inventado, Turing demostró la existencia de un problema irresoluble: ¿es posible construir un algoritmo que, dada una máquina de Turing cualquiera M1, nos diga si esa máquina acabará por pararse al leer cierta cinta, o bien si seguirá funcionando indefinidamente, moviéndose siempre hacia la derecha, siempre hacia la izquierda, o realizando ciclos más o menos complejos?

Turing supuso que se puede resolver el problema. Entonces será posible construir una máquina, M2, que lo resuelva. En la cinta de M2 colocaremos una descripción de M1 (su tabla de transiciones) y una copia de la cinta que debe leer. Además, podemos programar M2 de manera que, si M1 se para, M2 siga funcionando indefinidamente; por el contrario, si M1 no se para, M2 debe pararse. Puesto que M2 acepta la descripción de cualquier máquina de Turing, le proporcionamos su propia descripción. Nos encontramos entonces con una contradicción: por construcción, M2 debe pararse si M2 no se para, y viceversa. Por tanto, tenemos que renunciar a la hipótesis de que sea posible construir la máquina M2. Dicho de otro modo: el problema de la parada de la máquina de Turing es irresoluble.

Se dice que la máquina de Turing es *computacionalmente completa*, con lo que queremos decir que puede resolver cualquier problema *recursivamente enumerable*, que equivale a un problema resoluble por un ordenador digital. Por esta razón, la máquina de Turing resulta ser un modelo adecuado de la actuación de los ordenadores digitales, aunque no es el único. Existen otros mecanismos computacionalmente completos, como las redes neuronales, los sistemas clasificadores de Holland, etc.

La máquina de Turing ha encontrado aplicación en el campo de la complejidad de algoritmos, que compara la dificultad de distintos métodos para la resolución de un problema. Se ha detectado un tipo de problemas (*NP-completos*), que se conjetura son imposibles de resolver en un tiempo razonable cuando el número de elementos es grande. Este grupo incluye problemas como el del viajante de comercio o el de la búsqueda del camino mínimo en un grafo. Los dos campos de problemas P (más sencillos) y NP (potencialmente más complejos) denuncian en sus siglas su relación con la máquina de Turing, pues los primeros se definen como los que se pueden resolver en tiempo polinómico (P) en una máquina de Turing determinista, y los segundos los que se pueden resolver en tiempo polinómico en una máquina de Turing no determinista (de aquí la N).

Existen diversas modificaciones de la máquina de Turing, que han dado lugar a campos muy interesantes de la Informática Teórica. Algunas son equivalentes a la máquina original en su poder computacional, como las máquinas de Turing no deterministas, que en cada posición de la tabla pueden incluir distintas posibilidades; las máquinas con varias cintas; o aquéllas que en cada momento sólo pueden realizar alguna de sus tres acciones posibles. Otras restricciones conducen a máquinas menos generales, como el autómata lineal acotado, una máquina de Turing con cinta finita; el autómata a pila, que sólo puede leer unidireccionalmente de su cinta de entrada, aunque puede escribir y leer en una segunda cinta, que funciona como una pila; y el autómata finito determinista, que puede considerarse como un autómata a pila al que le hemos quitado la pila.

En la década de 1950, el norteamericano Avram Noam Chomsky (1928-) revolucionó la Lingüística con su *Teoría de las Gramáticas Transformacionales*, que proporcionó una herramienta que podía aplicarse a los lenguajes naturales y facilitaba el estudio y la formalización de los lenguajes de ordenador. Esta

teoría resultó tener una relación sorprendente con la de máquinas abstractas, hasta el punto de que ambas son *isomorfas*.

Chomsky clasificó los lenguajes formales de acuerdo con una jerarquía de cuatro grados, cada uno de los cuales contiene a todos los siguientes. El más general se llama *Tipo 0* e incluye todos los lenguajes posibles. Los de *Tipo 1*, también llamados *Lenguajes Sensibles al Contexto*, tienen algunas limitaciones, aunque se permite que la sintaxis de las palabras dependa de su contexto. Algunos lenguajes naturales (como el alemán-suizo y el bambara) tienen construcciones gramaticales de esta clase. Los de *Tipo 2* se llaman también *Lenguajes Independientes del Contexto* y restringen la libertad de formación de reglas gramaticales, pues la sintaxis de una palabra debe ser independiente de su contexto. La mayor parte de los lenguajes naturales y todos los de ordenador pertenecen a este grupo. Por último, los lenguajes del *Tipo 3*, los más sencillos, se llaman también *Lenguajes Regulares*.

Esta jerarquía de lenguajes es paralela a la de máquinas abstractas, en el sentido de que los lenguajes de cada tipo pueden representarse mediante máquinas equivalentes, que pueden ayudar a resolver cierto tipo de problemas o algoritmos. La tabla adjunta indica la relación entre las tres jerarquías.

		Problemas No Resolubles
Lenguajes Tipo 0 de Chomsky	Máquinas de Turing	Problemas Recursivamente Enumerables
Lenguajes Tipo 1 de Chomsky	Autómatas Lineales Acotados	Problemas Sensibles al Contexto
Lenguajes Tipo 2 de Chomsky	Autómatas a Pila	Problemas Independientes del Contexto
Lenguajes Tipo 3 de Chomsky	Autómatas Finitos Deterministas	Expresiones Regulares

Tabla que indica la relación entre las tres jerarquías

Bibliografía

Alfonseca, M.; Sancho, J.; Martínez Orga, M. A.: *Teoría de Lenguajes, Gramáticas y Autómatas*. Promosoft, Publicaciones R.A.E.C., Madrid, 1997. ISBN: 84-605-6092-9.

Brookshear, J. G.: *Teoría de la Computación*. Addison-Wesley, Madrid, 1989.

Chomsky, N.; Schutzenberger, M.P.: *The algebraic theory of context-freé languages*". Computer Programming and Formal Systems, pp. 118-161, North Holland, Amsterdam, 1963.

Fernández, G.; Sáez Vacas, F.: *Fundamentos de Informática, Lógica, Autómatas, Algoritmos y Lenguajes*. Anaya Multimedia, Madrid, 1995.