Lógica e hipercomputación

Francisco Hernández Quiroz

Departamento de Matemáticas

Facultad de Ciencias, UNAM

E-mail: fhq@fciencias.unam.mx

Página Web: www.fciencias.unam.mx/~fhq/

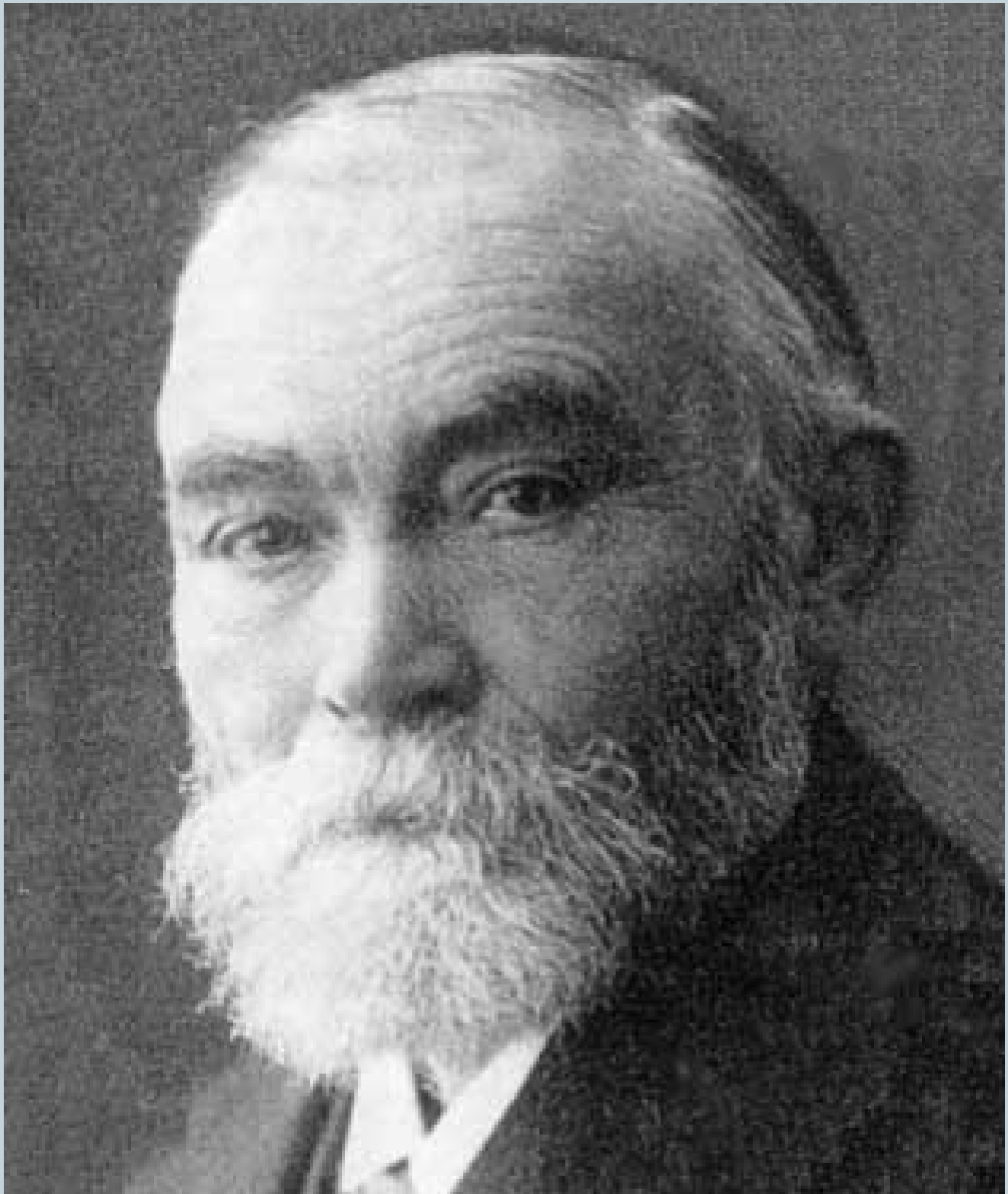
25 de marzo de 2004

“También se esperó entonces la aclaración de los misterios básicos de la humanidad: el origen de la Biblioteca y del tiempo. Es verosímil que esos graves misterios puedan explicarse en palabras: si no basta el lenguaje de los filósofos, la multiforme Biblioteca habrá producido el idioma inaudito que se requiere y los vocabularios y gramáticas de ese idioma.”

Jorge Luis Borges, *La biblioteca de Babel*

Calculus ratiocinator:

Un método para expresar todas las ideas y resolver todas las disputas filosóficas



Frege y lenguaje del cálculo de predicados

En el cálculo de predicados tenemos las proposiciones atómicas:

$$Q(x) \qquad P(x,y) \qquad R(x,y,z)$$

Que se pueden combinar de acuerdo con las conectivas lógicas:

$$\neg P(x,y) \qquad Q(x) \wedge R(x,y,z)$$

Además, tenemos los cuantificadores $\forall$ y $\exists$

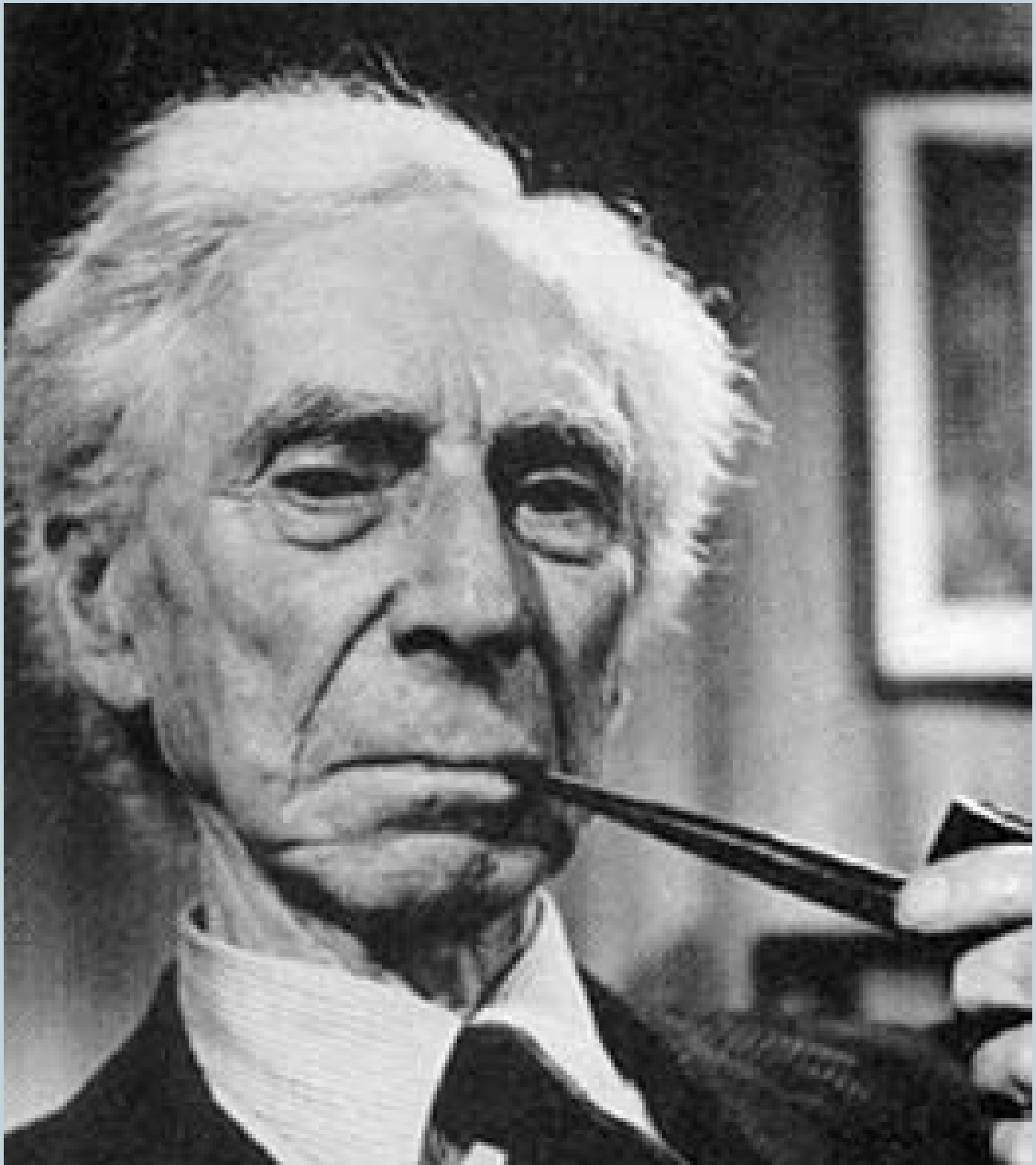
$$(\forall x \,.\, Q(x)) \qquad (\exists y \,.\, P(x,y) \Rightarrow R(x,y,z))$$

$$(\exists x \,.\, Q(x)) \vee (\forall z \,.\, R(x,y,z))$$

El significado intuititivo de los cuantificadores es:

$\forall x \,.\, P$ Para todas los valores posibles de x, vale la fórmula P.

$\exists x \,.\, P$ Existe un valor posible de x tal que vale la fórmula P



Principia Mathematica:

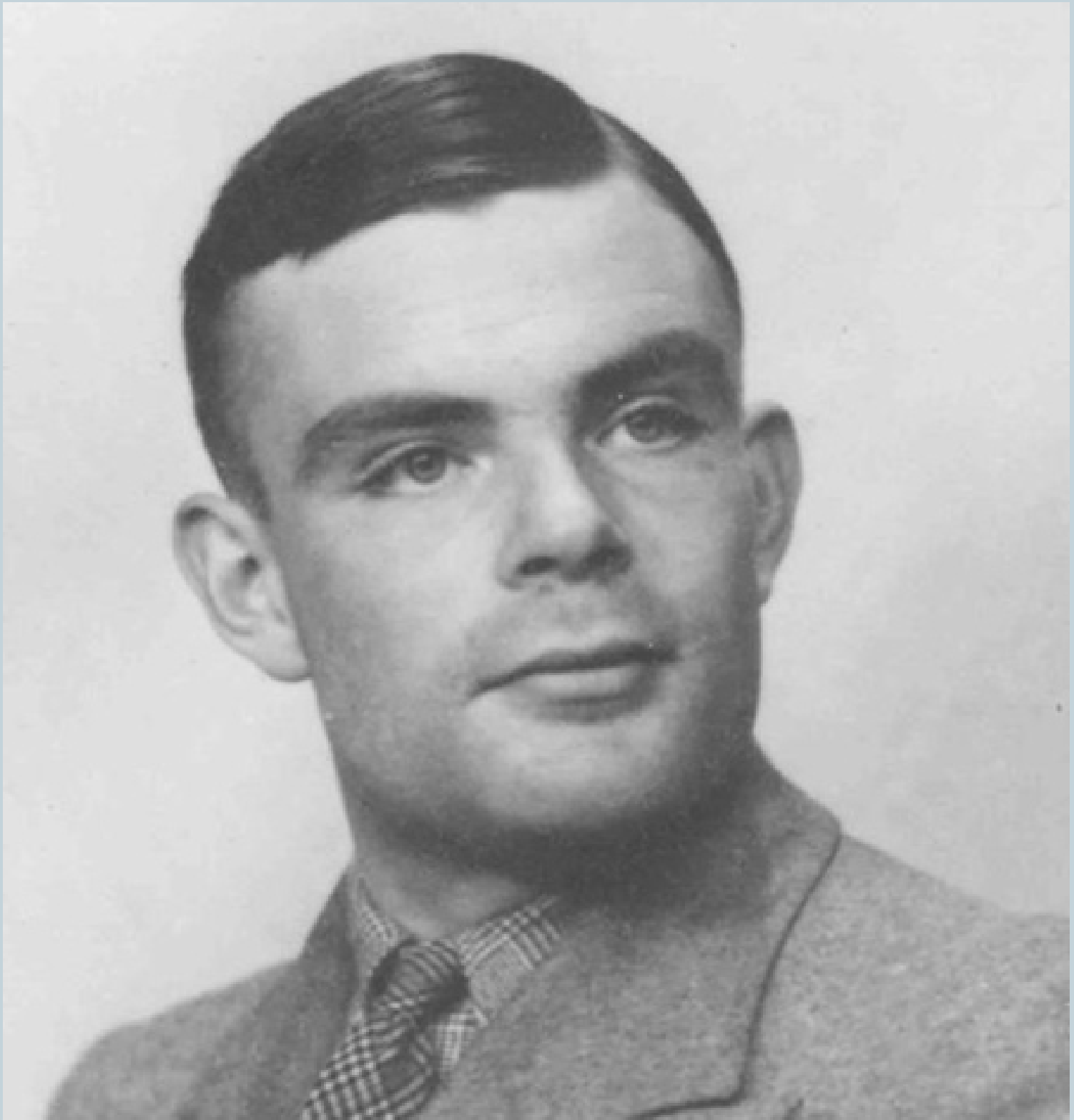
Todas las matemáticas se pueden traducir al lenguaje de la lógica

El problema de la decisión

Una *interpretación* es la asignación de significado a los elementos del cálculo. Por ejemplo, podemos pensar que la fórmula $P(x, y)$ significa "x es mayor o igual que y". La fórmula será verdadera dependiendo de qué valores asignemos a x y a y.

Una fórmula es *válida* si y sólo si es verdadera en toda interpretación. La fórmula $P(x, y) \vee \neg P(x, y)$ es verdadera en toda interpretación y, por tanto, es válida.

El problema de la validez consiste en contestar la pregunta "¿α es válida?" para toda fórmula del cálculo de predicados.



Máquinas de Turing

Un alfabeto es un conjunto (finito de símbolos). Sean Σ y Γ dos alfabetos con un símbolo especial: $\sqcup$ (espacio). Además, se tienen otros dos símbolos: $\vdash$ (inicio de la cinta) y $\dashv$ (final de la cinta) que están en Γ pero no en Σ.

Una máquina de Turing es una estructura $M = (Q, \Sigma, \Gamma, \delta, s, t, r)$ donde:

Q es un conjunto de estados;

Σ es el alfabeto de entrada;

Γ es el alfabeto de trabajo;

$\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \times \Gamma \rightarrow Q \times \Gamma \times \{1, 0, 1\}^2$ es la función de transición;

s, t y $r \in Q$ son los estados inicial, de aceptación y de rechazo, respectivamente.

Una *configuración* es un cuarteta ordenada (p, α, i, j), donde $p \in Q$, α es una cadena de símbolos del alfabeto de trabajo, i es la posición en la cinta de entrada y j es la posición en la cinta de salida. La *configuración inicial* es $(s, \vdash \sqcup^{\omega}, 0, 0)$.

Sea β una cadena de símbolos del alfabeto de entrada. Una *transición* en una máquina de Turing *M* se define así:

$(p, \alpha, i, j) \overrightarrow{M,\beta} (q, \alpha_{[j/b]}, i + d, j + e)$ sii $\delta(p, \beta_i, \alpha_j) = (q, b, d, e)$.

Sean M una máquina de Turing y β una cadena del alfabeto de entrada. Decimos que *M acepta* β sii

$$(s, \vdash \sqcup^{\omega}, 0, 0) \overrightarrow{{}_{M,\beta}}^{*} (t, \alpha, i, j);$$

y *M rechaza* β sii

$$(s, \vdash \sqcup^{\omega}, 0, 0) \overrightarrow{{}_{M,\beta}}^{*} (r, \alpha, i, j);$$

El lenguaje aceptado por una máquina de Turing M se denotará por $L(M)$.

El tiempo de las máquinas de Turing

Sea n la longitud de la cadena de entrada β y sea $f(n)$ el número *máximo* de pasos necesarios para que la máquina de Turing M acepte o rechace β.

$f(n)$ es, entonces, *la complejidad temporal* para el reconocimiento del lenguaje $L(M)$.

Por ejemplo, decidir si un número representado por una cadena binaria de n dígitos es par toma n pasos en una máquina de Turing muy simple.

Si se codifica de manera adecuada una fórmula β del cálculo de proposiciones con n proposiciones atómicas, el saber si β es satisfactible toma 2^n pasos, aproximadamente.

Otros modelos computacionales

- Cálculo λ de Church.

- Funciones recursivas.

Tesis de Church-Turing

"El conjunto de funciones computables por medio de una máquina de Turing es idéntico al conjunto de funciones que se pueden calcular por medio de un procedimiento mecánico descrito por medio de una serie de instrucciones llevadas a cabo por una persona o una máquina."

Esta tesis no es demostrable, pero si refutable.

El problema de la detención

No existe un método general para determinar si una máquina de Turing *M* algún día nos dará una respuesta positiva o negativa dada una cadena de entrada β. Este resultado se conoce como el problema de la detención (*halting problem*).

. . . y de nuevo el problema de la decisión

Para todo fórmula *alpha* existe una máquina de Turing *M* tal que

$\models \alpha$ si y sólo si es posible saber si *M* se detiene con la entrada β

Otros métodos más rápidos

- Máquinas de Turing no deterministas (en las cuales existe más de una transición posible en cada paso).

- Máquinas de Turing con oráculo (un dispositivo que contesta sí o no a una pregunta convenientemente planteada).

- Las futuras computadoras cuánticas (si es que se pueden construir).

Más allá de las máquinas de Turing

- Familias arbitrarias de circuitos booleanos [6].
- Máquinas de Turing “aconsejadas” [3].
- Sistemas dinámicos [4].
- Redes neuronales recurrentes [6].

Redes neuronales

Función de actualización de una red neuronal con *n* procesadores y *m* entradas:

$$x_i(t+1) = \sigma\left(\sum_{j=1}^{n} a_{ik} x_j(t) + \sum_{j=1}^{m} b_{ij} u_j(t) + c_i\right)$$

x_i es uno de los procesadores que componen la red, aij y b_{ij} son los pesos de las conexiones, u_j es un procesador de entrada, c_i es una constante interna de cada procesador y t es la transición anterior.

Poder computacional de las redes

- Si los pesos pueden ser sólo números enteros, las redes puede calcular las mismas funciones que un autómata regular (un modelo más débil que una máquina de Turing).

- Si los pesos pueden ser números racionales, las redes son equivalentes a las máquinas de Turing.

- Si los pesos pueden ser números reales, las redes pueden calcular las mismas funciones que un autómata “aconsejado”. En particular, si se acepta un número de pasos exponencial entonces todos los lenguajes son reconocibles.

¿Se pueden construir estos sistemas?

Las redes analógicas recurrentes son equivalentes a ciertos tipos de sistemas dinámicos. Ejemplo: una partícula que rebota entre espejos parabólicos [4].

Pero estos sistemas son muy sensibles al “ruido” por su naturaleza caótica (y, por tanto, propensos a errores).

Un resultado de [6] plantea la posibilidad teórica de corregir este problema: “durante los primeros q transiciones de una red

sólo importan los primeros $O(q)$ dígitos de los pesos y de los valores de activación de los componentes de la red".

En otras palabras, es posible trabajar con una red de precisión finita si nos interesa un resultado de precisión finita.

Sin embargo, la posibilidad de construir una red de este tipo es un debate abierto.

Conclusiones

- Es necesario replantear el modelo computacional de la mente en términos de otros modelos computacionales.

- En particular, hay una serie de posibles ventajas teóricas que ofrece el nuevo modelo: sobrepasar las limitaciones de los sistemas formales, reducir el tiempo de solución de muchos problemas (acercándonos a la velocidad con que actúa un ser inteligente, etc.)

- Es necesario analizar las objeciones al modelo computacional en este nuevo contexto.

Bibliografía

[1] G.S. Boolos& R.C. Jeffrey, *Computability and Logic*, Cambridge University Press, Open University Set Book, 1989.

[2] D. Deutsch, "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer", *Proceedings of the Royal Society of London*, A400, 96–117, 1985.

[3] R.M. Karp & R.J. Lipton, "Some Connections between Uniform and Nonuniform Complexity Classes", en *Proceedings of the 12th ACM Symposium on Theory of Computing*, 302–309, 1980.

[4] C. Moore, "Generalized Shifts: Unpredictability and Undecidibility in Dynamical Systems", *Nonlinearity*, 4, 199–230, 1991.

[5] R. Penrose, *The Emperor's New Mind*, Oxford University Press, 1989.

[6] H.T. Siegelmann, *Neural Networks and Analog Computation: Beyond the Turing Limit*, Birkhäuser, Progress in Theoretical Computer Science, 1999.

[7] A.M. Turing, "Computing Machine and Intelligence", *Mind* 59, 433–460, 1950.