

LÓGICA E INTELIGENCIA ARTIFICIAL

Por: Luis Piscoya Hermoza

INTRODUCCIÓN

Comenzaremos esta exposición señalando que compartimos los puntos de vista que consideran la expresión “inteligencia artificial” una metáfora, la misma que, a nuestro juicio, ha sido, en algunos aspectos, productiva para el desarrollo de la ciencia y para el análisis filosófico y, en otros, ha dado lugar a analogías que pueden conducir a confusiones y entusiasmos reveladores de una ausencia de reflexión fundada en la teoría de autómatas, la misma que posibilita un conocimiento de los alcances y limitaciones teóricas de los mismos.

En esta ponencia pretendemos dar fundamento a la tesis que sostiene que la metáfora, más específica que la antes mencionada, que afirma una especie de paralelismo funcional entre la mente humana y el ordenador, adolece de una limitación fundamental, cuya comprensión adecuada es decisiva para entender la capacidad específica del hombre de investigar, descubrir y crear a partir de situaciones que para un ordenador tienen carácter terminal. Dicha limitación consistiría fundamentalmente en el hecho de que, como demostraremos, el artificio de mayor potencia teórica, conocido como máquina universal de Turing, posibilita modelar o simular la función cerebral con respecto a lo que es recursivamente computable pero no en relación con aquello que, en general, puede ser calificado de inteligible o comprensible. Y si bien es verdad que todo lo computable es también inteligible, la afirmación conversas “Todo lo inteligible es computable” no es verdadera como lo muestra, por ejemplo, la existencia de conceptos matemáticos que son claramente inteligibles pero no recursivamente computables. Asimismo, para facilitar la comunicabilidad de nuestra argumentación, hemos preferido las pruebas intuitivas y, de este modo, hemos reducido el simbolismo lógico matemático a un *minimum* accesible, sin dificultades, a un lector con formación académica pero no familiarizado con lo que suele denominarse el ámbito interdisciplinario de la inteligencia artificial.

Damos por asumido el hecho histórico de que el hombre, desde la antigüedad, ha construido máquinas con el propósito presunto de transferirles, en la mayor medida posible, la ejecución de lo que genéricamente se denomina trabajo, el mismo que demanda el consumo de lo que llamaremos energía de ejecución. Sin embargo, el hombre para usar las máquinas, según el menor o mayor nivel de desarrollo de las mismas, ha tenido que gastar cantidades variables de energía de dirección. Parecería que la tendencia histórica que ha orientado la construcción de las máquinas ha sido diseñarlas de tal modo que se tienda a minimizar la energía de dirección y a maximizar la energía de ejecución. En función de este criterio se puede distinguir al menos tres tipos de máquina.

Una máquina es de tipo 1 cuando la energía de mando está integrada a la energía de ejecución, es el caso de la palanca, el arado, o la bicicleta, entre otros. En estas máquinas el hombre tiene que hacer un esfuerzo sostenido para lograr que realicen el trabajo propuesto

y la mayor o menor productividad del mismo depende de la cantidad de fuerza humana utilizada que es, al mismo tiempo, energía de dirección o de mando. Una máquina es de tipo 2 cuando la energía de mando es independiente de la energía de ejecución que puede ser, según sea el caso, la fuerza hidráulica, la eólica o la de un motor. Es el caso del molino tradicional, de un tractor, de una tejedora o de una impresora. Estas máquinas requieren de un operador que consume energía de mando a través de acciones físicas como manipular palancas o botones, que cumplen funciones análogas. Una máquina es de tipo 3 cuando su energía de mando ha sido reducida a la introducción de un conjunto finito de instrucciones, expresadas en un lenguaje L, que la máquina reconoce y acepta, realizando actos que, a su vez, son órdenes que pueden dar lugar a un trabajo que puede consumir cantidades masivas de energía de ejecución, como la cantidad de galones de carburante necesarios para hacer aterrizar un moderno avión comercial. Esta máquina se conoce como ordenador o computador y la realización de sus actos presupone, sólo al comienzo del proceso, la presencia de un programador humano o no.

El hombre ha transferido las rutinas de ejecución física a las máquinas de los tipos 1 y 2, y las tareas de procesamiento de información, tales como cálculos aritméticos, así como las rutinas de dirección, a las máquinas de tipo 3. Estas últimas tareas implican con frecuencia la toma de decisiones entre estados posibles de la máquina, determinados por las informaciones de entrada que sus sensores reciben de su “ambiente”, a través de instrucciones codificadas en un lenguaje no determinista.

Las máquinas del tipo 3 han sido calificadas de inteligentes y ha gravitado en ello el hecho de que simulan la conducta humana, en sus aspectos que la diferencian de los animales, debido a que son capaces de reconocer, aceptar o rechazar, determinadas secuencias de signos de un lenguaje L. En efecto, es la capacidad de comunicarse a través de la palabra lo que, desde Aristóteles, ha sido calificado de racionalidad humana y, en este sentido, las máquinas del tipo 3 no sólo se comportan racionalmente sino que podrían ser la manifestación más lograda de lo que los filósofos de la escuela de Frankfurt han llamado razón instrumental.

Los desarrollos notables en el conocimiento de las gramáticas capaces de generar los lenguajes de máquina (software), con independencia de referencias específicas a la naturaleza o estructura física de la misma (hardware), han conducido a campos de especialización distinguibles y, en grado significativo, independizables. De este modo, los problemas de software admiten un tratamiento separado de los de hardware. Esta diferenciación ha despertado el entusiasmo de muchos psicólogos y psiquiatras, que han visto en esta analogía la posibilidad teórica de estudiar las denominadas funciones psicológicas superiores, como el pensamiento y el uso del lenguaje, con independencia de referencias neurológicas específicas. De esta manera, para la investigación del “*software humano*”, se ha buscado analogías funcionales en modelos como la máquina de Turing, que no constituye un artefacto sino un modelo matemático teórico del comportamiento aunque no de la estructura de un artefacto capaz de computar secuencias de signos conocidos como funciones recursivas.

MÁQUINA DE TURING

Puede considerarse que una máquina de Turing (en adelante T_m) es un artificio matemático definido en términos que lo capacitan para entender un lenguaje L . Puede presentársela intuitivamente, como ocurre en algunos manuales de lógica y computación, o puede presentársela de modo axiomático dentro de una teoría general de autómatas finitos, como un caso particular, en la forma exhibida por trabajos rigurosos como el de Hopcroft y Ullman, titulado *Formal Languages and their relation to Automata*. En todo caso no es fácil presentar adecuadamente una T_m , aun intuitivamente, sin entrar en detalles que pueden ser monótonos pero necesarios para no afectar la corrección del planteamiento. Por ello nos ha parecido adecuada la presentación hecha por Kleene (1968) en su libro *Introduction to Mathematical Logic* (p. 232 Y ss.), razón por la que pretenderemos exponerla, con leves modificaciones, en lo que sigue.

Una T_m es una idealización que se diferencia del ordenador o del computador humano en al menos dos aspectos sustantivos:

- 1) Una T_m es infalible en el sentido de que obedece las instrucciones sin desviación alguna; y
- 2) Una T_m tiene una memoria potencialmente infinita.

Asumiremos que operamos nuestra T_m en una sucesión de instantes I_0, I_1, I_2, \dots y que la sucesión q_1, q_2, \dots, q_k , de estados que puede tomar nuestra T_m , es finita.

Si consideramos el estado pasivo como el estado q_0 , o de máquina sin funcionamiento, entonces nuestra T_m tiene $k+1$ estados y en cada instante puede tomar cualquiera de ellos. Ella consiste de una cinta dividida en cuadraditos o celdas, la misma que por razones de simplicidad la consideramos potencialmente infinita sólo hacia la derecha. Cada celda puede estar en blanco, en cuyo caso asumiremos que se ha impreso en ella el símbolo S_0 , o puede tener impreso uno de los símbolos S_1, \dots, S_j de un alfabeto finito. Por razones de comodidad reduciremos este alfabeto al conjunto $\{B, 1\}$. La letra **B** significará "cuadradito en blanco." Así existirán sólo dos condiciones para cada cuadradito, o tiene impreso un **1**, o tiene impresa una **B** (lo segundo equivale a estar en blanco).

Asimismo, en cada instante, a partir del instante I_0 , la T_m escruta sólo un cuadradito. Para producir la transición de un instante dado al siguiente, la T_m realiza un *acto*, el mismo que consiste de una secuencia de tres operaciones: **a)** Imprime uno de los símbolos del conjunto $\{B, 1\}$ en el cuadradito escrutado o intercambia uno por otro. En este caso, eliminar **B** para imprimir **1** equivale a escribir **1** en un cuadradito en blanco; y eliminar **1** para imprimir **B**, equivale a simplemente borrar **1** y dejar la celda en blanco; **b)** La T_m se mueve solamente una celda a la derecha o una celda a la izquierda o permanece estacionaria (centrada); **c)** Cambia a otro estado o permanece en el mismo. Una *configuración* consiste en el estado de la T_m y el símbolo que está impreso en la celda escrutada en un instante dado. Cada acto que la T_m realiza en un instante dado está determinado por la *configuración* de la T_m en el

instante inmediato anterior. A la totalidad de los signos impresos en la cinta de la T_m junto con la configuración se le denomina *situación*.

Nuestra T_m opera de izquierda a derecha. La cinta es finita por la izquierda y potencialmente infinita por la derecha. Solamente en el caso de que la celda escrutada sea la "más izquierda", una configuración que ordene un desplazamiento a la izquierda no determinará un *acto*.

Para computar el valor y para la función $f(x)$ es necesario asumir el instante I_0 como el instante en que la T_m y su cinta están listas: la celda del extremo izquierdo de la cinta está en blanco y a su derecha el valor de x está representado por $x + 1$ apariciones de $1 \in \{B, 1\}$ en $x + 1$ celdas. La T_m está escrutando la celda impresa "más a la derecha" y diremos que la T_m computa un valor y para el argumento x si partiendo de la situación que corresponde al instante I_0 la T_m después de un número finito de instantes posteriores toma el estado pasivo q_0 con una B en la cinta, la misma que a su izquierda tiene las $x + 1$ apariciones de $1 \in \{B, 1\}$ que representan el valor de x , y a su derecha tiene las $y + 1$ apariciones de $1 \in \{B, 1\}$ que representan el valor de y ; todas las otras celdas están en blanco y la T_m nuevamente está en la posición de escrutar la celda impresa que está "más a la derecha" (reiteramos que a la celda con una B no la consideramos impresa sino en blanco o borrada). Asimismo, para indicar la celda que está escrutada por la T_m escribimos un número sobre ella que, además, indica el número del estado en el que se encuentra la T_m . En el caso de que para cada valor de x , la T_m compute un valor y , entonces diremos que $f(x)$ es una función T_m computable. En particular nos interesa $f(x) = x + 1$, esto es, la función sucesor.

Por ejemplo, si nuestra T_m es aplicada al número natural 1 como argumento, la situación de T_m lista, en el instante I_0 puede representarse como sigue:

1					
B	1	1	B	B	B

El 1 de la parte superior muestra que la celda escrutada es la tercera, y que la T_m está en su primer estado activo q_1 . Asimismo, en el estado q_1 la celda escrutada es aquella que está impresa y se encuentra más a la derecha. Las siguientes están en blanco. Las dos apariciones de $1 \in \{B, 1\}$ representan, por convención, el valor de x , que en este caso es el número natural 1 . En un instante posterior I_r , la T_m habrá computado el valor $y = 2$ (imprimiendo $y + 1$ apariciones de $1 \in \{B, 1\}$) para el argumento 1 , puesto que $f(x)$ es la función sucesor. La situación del instante I_r puede graficarse como sigue:

							0
B	1	1	B	1	1	1	

A fin de mejorar nuestra comprensión de la naturaleza monótona y exhaustiva cómo procede una máquina de Turing, en lo que sigue describiremos detalladamente una T_m^* , específica, capaz de computar la función sucesor $f(x) = x + 1$, cuando el valor de x es igual a 1 . Esta descripción consistirá en precisar para cada uno de los k estados activos de la T_m^* y para cada una de las dos condiciones posibles de la celda escrutada, **B**, **1**, cuál es el acto que debe ser realizado por la T_m^* en un instante I_i dado, lo que equivale a hacer explícito un conjunto de instrucciones condicionales a través de una tabla (**T1**) de doble entrada para sus q_1, q_2, \dots, q_{11} estados posibles y para sus dos condiciones posibles, **B**, **1**, para cada una de las celdas escrutadas. Ello equivale al producto cartesiano de $\{q_1, \dots, q_{11}\} \times \{B, 1\}$. Cada instrucción es una n -tupla de la lista de símbolos **P**, **E**, **L**, **C**, **R** que termina en un número que señala el estado que debe asumir la T_m^* en el siguiente instante. **P** es una abreviación de "imprimir", **E** de "borrar", **L** de "mover una celda a la izquierda", **C** de "no mover" o "centrar" y **R** de "mover una celda a la derecha."

Tabla T1

Estados de T_m^*	Condición de la celda escrutada	
	B	1
q_1	CO	R2
q_2	R3	R9
q_3	PL4	R3
q_4	L5	L4
q_5	L5	L6
q_6	R2	R7
q_7	R8	ER7
q_8	R8	R3
q_9	PR9	L10
q_{10}	CO	ER11
q_{11}	PCO	R11

Los estados de la T_m^* pueden entenderse como un conjunto de 11 instrucciones condicionales, cada una con dos componentes de la forma "Si **B**, entonces...", o de la forma "Si **1**, entonces, ...". Estas instrucciones se cumplen en 25 actos correspondientes a 25 instantes, siendo el primero de ellos I_0 , tal como se especifica en la tabla **T2**.

El hecho de que el número de actos no coincida con el número de estados de la T_m^* se explica porque hay instrucciones que ordenan volver al mismo estado (esto es, hacer un *loop*) al pasar de un acto a otro, o, también hay otras que ordenan un *feedback* (es el caso de

q_6 que dice "Si **B**, entonces **R2**" y de q_8 que dice "Si **1**, entonces **R3**"). La situación del instante I_{23} es la de vuelta al estado inactivo o de parada (*halting*). El acto de I_{24} es sólo una confirmación del estado de parada. Los *loops* y *feed backs* puedan apreciarse en el diagrama de flujo D_1 , que equivalente a la Tabla T_1 .

En términos generales, la T_m^* es un artificio matemático que opera examinando exhaustivamente en la cinta el estado en que se encuentra, en un determinado instante, y la condición correspondiente de la celda escrutada, la misma que puede ser **B** o **1**. El recorrido lo hace de izquierda a derecha y de derecha a izquierda, escrutando en cada acto sólo una celda, sin omitir ninguna, hasta alcanzar el estado q_0 de parada. Durante este proceso imprime 'unos' y los borra, según lo ordenen las instrucciones, hasta llegar a la situación que se muestra en la línea correspondiente al instante I_{23} .

Las instrucciones contenidas en la Tabla T_1 constituyen un algoritmo porque permiten computar $x + 1$ para cualquier valor que tome x dentro del conjunto N , esto quiere decir que para cada valor de x , el proceso establecido por la tabla T_1 genera una T_m que llega a la situación de parada. Si consideramos a la situación del instante I_{23} , $B11B111^0$, como una palabra w en un lenguaje formal, podemos decir que el proceso establecido por la tabla T_1 es recursivo porque es un algoritmo que para toda palabra w decide si ha sido o no generada por la situación correspondiente a I_0 .

Lo anterior implica que una T_m puede computar funciones más complejas como las correspondientes a la suma, multiplicación y potenciación de la aritmética, por citar algunos ejemplos. El requisito es que estas funciones sean definidas recursivamente de tal manera que las operaciones más complejas sean reducidas a las más simples. Esto se logra a través de las tres definiciones siguientes:

$$x + 0 = x$$

$$x \cdot 0 = x$$

$$x^0 = 1$$

$$X + Y' = (X + Y)'$$

$$x \cdot y' = x \cdot y + x$$

$$x^{y'} = x^y \cdot x$$

Tabla T2

Tabla de actos de Tm^* para computar $f(x)$, para $x = 1$

Instantes	Situación de Tm^* (Tm^* vs cinta)						
l_0	B	1	1^1	B	B	B	B
l_1	B	1	1	B^2	B	B	B
l_2	B	1	1	B	B^3	B	B
l_3	B	1	1	B^4	1	B	B
l_4	B	1	1^5	B	1	B	B
l_5	B	1^6	1	B	1	B	B
l_6	B	1	1^7	B	1	B	B
l_7	B	1	B	B^7	1	B	B
l_8	B	1	B	B	1^8	B	B
l_9	B	1	B	B	1	B^3	B
l_{10}	B	1	B	B	1^4	1	B
l_{11}	B	1	B	B^4	1	1	B
l_{12}	B	1	B^5	B	1	1	B
l_{13}	B	1^5	B	B	1	1	B
l_{14}	B^6	1	B	B	1	1	B
l_{15}	B	1^2	B	B	1	1	B
l_{16}	B	1	B^9	B	1	1	B
l_{17}	B	1	1	B^9	1	1	B
l_{18}	B	1	1	1	1^9	1	B
l_{19}	B	1	1	1^{10}	1	1	B
l_{20}	B	1	1	B	1^{11}	1	B
l_{21}	B	1	1	B	1	1^{11}	B
l_{22}	B	1	1	B	1	1	B^{11}
l_{23}	B	1	1	B	1	1	1^0
l_{24}	B	1	1	B	1	1	1^0

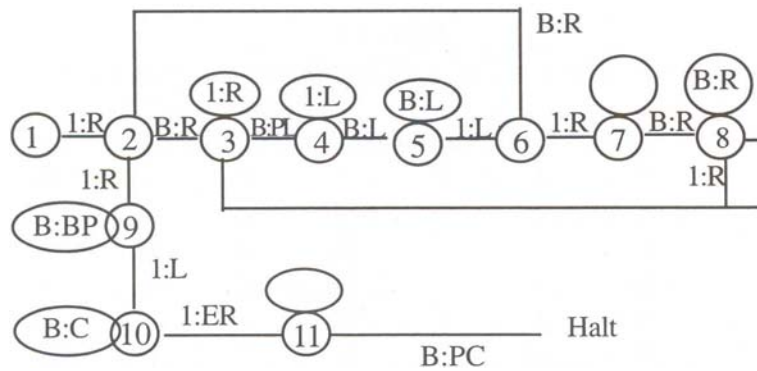


Diagrama D1

De este modo la potenciación se reduce al producto y el producto a la suma, la misma que se reduce, exceptuando el caso de la identidad, al cómputo de $x + 1$, vale decir al cómputo de la función sucesor, lo que realiza el algoritmo de la tabla **T1**. Por consiguiente, la tabla **T1** permite que una T_m compute cualquier función reducible a la función sucesor, lo cual la hace especialmente potente para los usos normales de la matemática aplicada.

El investigador Alonso Church propuso, en 1936, una tesis conocida como la conjetura de Church, la misma que afirma que toda función, que es intuitivamente computable, es también efectivamente computable a través de funciones recursivas generales y, un año más tarde, Turing demostró que para toda función recursiva general existe una T_m capaz de computarla. De este modo, la conjetura de que toda función intuitivamente computable es recursivamente computable se ha convertido en la conjetura ChurchTuring, la misma que no tiene contraejemplo conocido, pues no se sabe de una función computable que no lo sea también por alguna T_m o por funciones recursivas generales, llamadas funciones *lambda*. Posteriormente, y desde otra perspectiva centrada en el análisis de sucesiones de símbolos de un alfabeto dado, Marcov (1951) formuló una teoría de algoritmos con resultados equivalentes a los de Church y Turing. Por añadidura, se ha demostrado que cualquier computación que pueda ser ejecutada por una computadora digital moderna puede ser descrita por medio de una T_m (Vid. Hopcroft & Ullman, 1969, p.80).

Lo anterior explica el mecanismo a través del cual el hombre ha podido transferir a la máquina tareas que se estiman correspondientes a funciones superiores, tal es el caso de los cálculos numéricos. Esto ha posibilitado que también se deriven al computador tareas de dirección y ha dado lugar a que se formule la metáfora del paralelismo funcional entre el cerebro humano y una T_m , que es el artificio que más directamente ofrece analogía con el cálculo humano, materializado en una tarea que puede ser descompuesta en sus elementos más simples, de tal manera que se puede generar una lista enumerable y exhaustiva de un proceso, como el descrito por la tabla **T2**.

LIMITACIONES DE LA MÁQUINA DE TURING

La T_m está sujeta al menos a dos limitaciones, muy relacionadas entre sí pero distinguibles. La primera se refiere directamente a la posibilidad de construir al menos una función numérica $\Psi(x)$ no computable por una T_m -universal (T_m -U) que sí es capaz de computar cualquier función $\phi(x)$, para la que existe un número r finito de instantes, lo que equivale a decir que en el instante I_r la T_m -U hace alto o tiene el problema del halting resuelto (una T_m universal es una máquina capaz de simular la conducta de cualquier T_m). La otra limitación es conocida como el problema del "castor laborioso" (*busy beaver*) y se refiere a la productividad de la T_m medida en términos de la mayor o menor longitud de los segmentos de 'unos' que puede imprimir sobre una cinta que en el instante lo está en blanco. El problema del castor laborioso de clase ($n,2$) puede formularse, para la T_m que hemos descrito, en los términos siguientes: para T_m s (máquinas de Turing) de n estados, 2 símbolos, **B**, **I**, y 2 movimientos, **R**, **L**, el número N de T_m s posibles se define $N = (4(n+1))2^n$. La pregunta es ¿cuál de estas N máquinas de Turing posibles hará alto con más 'unos' escritos sobre la cinta?

La respuesta a estos problemas requiere un esclarecimiento previo de la naturaleza de los mismos. En la medida que ambas limitaciones se reducen aun interrogante acerca de la computabilidad de una función pueden ser expresadas a través de la pregunta:

¿existe una T_m -U capaz de computar la función $G(x)$? Responder **Sí** significa que existe un algoritmo para el cómputo de $G(x)$. Responder **No** admite dos interpretaciones. La primera es que se puede demostrar que no existe algoritmo capaz de computar $G(x)$, lo que implica la insolubilidad de la computabilidad de $G(x)$ por una T_m -U pero no la insolubilidad del problema planteado, pues éste ha quedado cerrado en tanto se ha demostrado la corrección de la respuesta negativa. La segunda posibilidad es que la respuesta **No** haga referencia a que no se conoce si existe o no un algoritmo para la computabilidad de $G(x)$. En este caso el problema queda abierto y la investigación científica tiene un reto más que afrontar.

Las limitaciones antes referidas tienen respuesta negativa en el sentido de la primera interpretación. Vale decir, pueden considerarse problemas cerrados, lo que no impide que el análisis de los mecanismos implicados sea productivo desde el punto de vista epistemológico.

Para explicar los lineamientos intuitivos de la primera limitación procederemos a construir el índice de la T_m^* de la tabla **T1**. Para el efecto, formularemos primero su *código* escribiendo horizontalmente y de manera sucesiva las instrucciones que constituyen la T_m , anteponiendo a cada una de ellas la condición de la celda escrutada, esto es, uno de los símbolos B,1.

c) BCO, 1R2; BR3, 1R9;.....; BPCO, 1R11

Para construir este código de la T_m^* . hemos usado 16 símbolos incluyendo los puntos y comas, y las comas que hemos introducido para separar las instrucciones y sus componentes. Estos 16 símbolos son **C L R , ; B O 1 2 3 4 5 6 7 8 9**.

Si interpretamos el código **c** como un número escrito en un sistema en base 16, obtendremos un entero positivo, bastante grande, que es el índice de muestra T_m . De la misma manera para toda T_m se puede obtener un índice **i** que la describe y establece su patrón de conducta.

Si llamamos **a** al índice de una T_m -U, asumimos que existe una función $\varphi_i(x)$ computable por una T_m -U con índice **i** y definimos la función $\Psi(a)$ como sigue:

$$\Psi(a) = \begin{cases} \varphi_a(a) + 1 & \text{si } \varphi_a(a) \text{ es computable} \\ 0 & \text{en otro caso} \end{cases}$$

se puede demostrar por reducción al absurdo que $\Psi(a)$ no es computable por T_m -U. En efecto si $\Psi(a)$ fuera computable, la T_m -U simularía una máquina con índice **p** que denotamos T_{m_p} , la misma que computaría $\varphi(a)$ para $i=p$ y para cualquier valor de **a**. Por

tanto, tendríamos $\Psi(\mathbf{a}) = \varphi_p(\mathbf{a})$ y $\Psi(\mathbf{p}) = \varphi_p(\mathbf{p})$, pero como hemos asumido, $\varphi_i(\mathbf{x})$ es computable, entonces $\varphi_p(\mathbf{p})$, también lo es, lo que aplicando la definición dada establece $\Psi(\mathbf{p}) = \varphi(\mathbf{p}) + 1$, resultado que contradice la igualdad anterior. Esto significa, intuitivamente, que no existe una Tm que sea capaz de computar una función, $\varphi_i(\mathbf{x})$ que toma el valor $\mathbf{x} = \mathbf{i}$, esto es, ninguna Tm puede computar su propio índice o, en otras palabras, la computabilidad del índice de cualquier Tm por ella misma es insoluble.

En relación con el problema del castor laborioso, el problema se reduce a demostrar que no existe una Tm capaz de computar la función \mathbf{p} , definida para una Tm de \mathbf{n} estados así:

$\mathbf{p}(\mathbf{n})$ = la productividad de las Tms de \mathbf{n} estados más productivos.

Este problema fue resuelto por Tibor Rado (1962) quien demostró, por reducción al absurdo, que si existiera tal Tm, a la cual podemos llamar **BB**, tal máquina computaría la desigualdad $\mathbf{p}(\mathbf{n}+2\mathbf{k}) > \mathbf{p}(\mathbf{p}(\mathbf{n}))$, probado que existen como resultados previos $\mathbf{p}(\mathbf{n}+1) > \mathbf{p}(\mathbf{n})$ y $\mathbf{p}(\mathbf{n}+11) > 2\mathbf{n}$. Esto conduce, omitiendo detalles, a la contradicción $0 \geq 1$, lo que significa que la **BB** no existe.

Un problema adicional, muy ligado al anterior, es el del *halting* que puede ser formulado en términos de una proposición que afirma que no existe algoritmo alguno que nos permita reconocer cuáles son las Tms que se mueven indefinidamente y que, consecuentemente, tienen una productividad igual a cero. Si se asume como primera premisa el resultado de una demostración condicional que establece: Si el problema del *halting* fuera soluble entonces la función \mathbf{p} sería soluble por medios intuitivos. Y si añadimos como segunda premisa la conjetura de Church, la misma que para éste caso concreto asume la forma: Si \mathbf{p} es computable por medios intuitivos, entonces existe una Tm que compute \mathbf{p} . En consecuencia, utilizando el resultado de Tibor Rado que prueba que no existe Tm alguna que compute \mathbf{p} , podemos deducir de manera lógicamente inobjetable que el problema del *halting* no es soluble.

La forma de nuestro razonamiento es:

$$((H \rightarrow C) \& (C \rightarrow T) \& (\sim T)) \rightarrow \sim H$$

donde 'H' es la proposición "El problema del halting es soluble" y el componente ' $C \rightarrow T$ ' es la conjetura de Church.

IMPLICACIONES EPISTEMOLÓGICAS

En la medida que una Tm es un artificio matemático, sus limitaciones son teóricas o de principio y, en consecuencia, ellas no pueden ser superadas en sus aplicaciones prácticas, pues tienen una naturaleza más restrictiva que la imposibilidad física de construir una máquina de movimiento perpetuo. El origen de estas limitaciones radica en las características de los lenguajes que reconoce y acepta. Así el problema del *halting* puede ser reformulado en los siguientes términos: dada una Tm arbitraria y una situación inicial con un segmento arbitrario, eventualmente vacío, de unos impresos en la cinta, no existe

algoritmo alguno que permita decidir si la Tm hará alto o no. Desde otro ángulo, este mismo problema se reduce a probar que la suposición de que existe una Tm- U que permite decidir si cualquier Tm arbitraria hace alto o no, conduce a una contradicción, la misma que consiste en afirmar y negar que dicha Tm acepta un determinado lenguaje **L**.

El hecho de que un lenguaje que acepta una Tm deba ser recursivo, obliga a que ella sólo pueda ser usada como modelo de procesos enumerables y finitos, sin embargo la teoría general de la Tm, que permite describirla y demostrar sus alcances y limitaciones, presupone la existencia y utilización de conjuntos no enumerables de cardinalidad 2^{\aleph_0} , esto es, la Tm es recursiva pero la teoría de la Tm, que es uno de los productos notables del pensamiento humano, no lo es.

Así, por ejemplo, la teoría de la Tm explica por qué existen funciones no computables a partir del hecho de que el conjunto de las funciones numéricas recursivas es infinito y enumerable, en cambio, el conjunto de todas las funciones numéricas posibles es infinito pero no es enumerable, porque no lo es el conjunto potencia del conjunto de los enteros positivos, como lo demostró Cantor, probando que intentar confeccionar una lista exhaustiva de los elementos de un conjunto de cardinalidad 2^{\aleph_0} conduce a contradicción.

Por tanto, la tesis del paralelismo funcional entre el cerebro y un ordenador de propósito general, representado por una Tm, en el mejor de los casos, podría explicar el modo recursivo de pensar pero deja sin explicación el pensamiento que se expresa a través de conceptos no recursivos, como el de número transfinito, y conceptos elementales y de implicancias generalizadas, como el de no enumerabilidad. Dicha tesis no sólo implica atribuir a la Tm una generalidad que no tiene, sino también conlleva una comprensión inadecuada del pensamiento expresado en todas las teorías conocidas como indecibles, tal es el caso de la lógica de predicados de primer orden o de la teoría formal de números, para las cuales no existe algoritmo alguno que permita decidir cuándo una fórmula arbitraria es un teorema y cuándo no lo es, como lo demostró A. Church (1936) en términos de la definibilidad de funciones *lambda*.

Estos resultados son equivalentes a afirmar que la ausencia de un procedimiento de decisión para el conjunto de los teoremas de los sistemas mencionados consiste en la inexistencia de Tm alguna capaz de computar dicho conjunto.

Lo anterior significa que cualquier modelación o simulación de la función cerebral debe distinguir claramente entre lo inteligible o comprensible y lo recursivamente computable. Todo lo recursivamente computable es inteligible, pero no todo lo inteligible es recursivamente computable. La sucesión de los números cantorianos $\aleph_0, \aleph_1, \aleph_2, \dots$ es inteligible, pero ninguno de ellos es obtenible por métodos recursivos. Inclusive, si nos atenemos a los métodos recursivos o a las Tms, en muchos casos tenemos que conformamos con lo inteligible y no con lo efectivamente computable en términos del valor preciso de una función. Si consideramos el conjunto de las Tms de 100 estados, existe $163, 216^{100}$ de ellas. Examinando este conjunto se sabe que una de ellas hará alto con $((((7!)!)!)!)$ 'unos' en la cinta, que es una cifra fácil de entender pero prácticamente indeterminable, aun

imprimiendo mil millones de 'unos' por segundo durante diez mil millones de años (Korfhage, 1970). Consecuentemente, la T_m capaz de computar $((((7!)!)!))!$ es inteligible pero no, al menos por ahora, construible. Empero, también es inteligible que $((((7!)!)!))!$ es muy grande pero finito, y que en relación con \aleph_0 es pequeño.

Consecuentemente, los psicólogos cognitivistas, que investigan en el sentido de construir teorías que expliquen el pensamiento humano y que invocan como modelo de simulación la T_m , tendrían que tomar en consideración el hecho científicamente establecido que el ámbito de lo inteligible es inmensamente más amplio y rico que el de lo computable. Podría ocurrir que nuestro nivel de desarrollo teórico y tecnológico no nos permita por ahora mejor simulador de la función pensante que una T_m , pero en todo caso, se deberá ser explícitamente consciente de que se está simulando sólo una parte del conjunto de lo inteligible, el mismo que tiene entre sus elementos las teorías más elaboradas que ha producido la actividad superior conocida como pensamiento, que excede largamente, como hemos demostrado, el ámbito de lo efectivamente computable.

BIBLIOGRAFÍA

Andur Pedersen, Stig: *Mathematical Models in Cognitive Science*. Moscú. Abstracts del CLFMC-87, Tomo 2, pp. 404-405.

Boole, George y Jeffrey, Richard: *Computability and Logic*. Cambridge University Press, 1974.

Bunge, Mario: *The place of Psychology in the System of Knowledge*. Moscú. Abstracts del CLFMC-87, Tomo 2, pp. 369-71.

Burgin, M.S.: *The notion of Algorithm and the Turing-Church Thesis*. Moscú. Abstracts del CLFMC-87, Tomo 5, Parte 1, pp. 138-140.

Cohen, Jonathan: *A note in the Evolutionary Theory of Software Development*. Moscú. Abstracts del CLFMC-87, Tomo 2, pp. 489-490.

Dadong, Liang: *The Relationship between Human Thought and Artificial Intelligence*. Moscú. Abstracts del CLFMC-87, pp. 149-52.

Delclaux, Isidoro y Seoane, Julio: *Psicología cognitiva y procesamiento de información*. Madrid. Ediciones Pirámide. 1982.

Gurevich, Yuri: *Logic and the Challenge of Computer Science*. Moscú. Abstracts del CLFMC-87, Tomo 5, Parte 1, pp. 144-146.

- Hopcroft, John y Ullman, Jeffrey: *Formal Languages and their relation to Automata*. Addison Wesley Publishing Company, Series in Computer Science and Information Processing, 1969.
- Humphries, Jill: *Artificial Intelligence and Human Mental States*. Moscú. Abstracts del CLFMC-87, Tomo 2, pp.378-380.
- Johnson, Mark: *Grammar as Logic, Parsing and Deduction*.1987. Moscú. Abstracts del VIII Congreso Internacional de Lógica, Filosofía y Metodología de la Ciencia (CLFMC-87), Tomo 1, pp. 468-470.
- Kleene, Stepehn: *Mathematical Logic*. John Wiley and Sons, Inc.1967.
- Korfhage, Robert: *Lógica y Algoritmos*. México. Editorial Limusa Wiley.1970.
- Krishnamurthy, E.V.: *Non-Archimedian Valuation-Its Philosophy and practical utility for rational recursive computacion*. Moscú. Abstracts del CLFMC-87, Tomo 1, pp.143-145.
- Kushner, B.A.: *A counterexample in the Theory of Constructive Functions*. Moscú. Abstracts del CLFMC-87 pp. 147-148.
- Lucchesi, Claudio, Simon, I. et al.: *Aspectos teóricos da computação*. Río de Janeiro. Instituto de Matemática Pura y Aplicada, 1979.
- Marras, Ausonio: *Mental Images and the Frame Problem in Artificial Intelligence*. Moscú. Abstracts del CLFMC-87, Tomo 2, pp.400-403.
- Seiffert, Helmut: *Einführung in die Wissenschaftstheorie*. München, Verlag C.H. Beck, n. Auflage, 1991.
- Seiffert, Helmut y Radnitzky, Gerard: *Handlexikon zur Wissenschaftstheorie*. München, Deutscher Taschenbuch Verlag GmbH & Co. KG, 1992.
- Striebing, Lothar: *Computerisierung-Eine Herausforderung für die Gesellschaftsphilosophie*. Moscú. Abstracts del CLFMC87. Tomo 2, pp. 544-546.
- Swift, Clare: *Logical Inconsistencies in Mathematical Models of Neural Systems*. Moscú. Abstracts del CLFMC-87, Tomo 2, pp.417-420.
- Wall, Robert: *Introduction to Mathematical Linguistics*. PrenticeHall, Inc. 1972.
- Zlatoustova, Egorov, Raskin: *Some restrictions of Machine Modelling of Human Communication, Language and Thought*. Moscú. Abstracts del CLFMC-87. Tomo 1, pp. 547550.