ISSN: 0213-3563

DE LA COMPUTABILIDAD A LA HIPERCOMPUTACIÓN

From Computation to hipercomputation

Enrique ALONSO*
Dept. de Lingüística, Lenguas Modernas, Lógica y Filosofía de la Ciencia, U.A.M., enrique.alonso@uam.es

BIBLID [(0213-356)8,2006,121-146]

Fecha de aceptación definitiva: 17 de marzo de 2006

RESUMEN

Este trabajo aborda el estado actual de la cuestión en el ámbito de la computación teórica desde una perspectiva especialmente dirigida a lectores no expertos y con una clara vocación filosófica. En la primera parte, secciones 1 a 4, se ofrecen algunas de las claves para entender la importancia de la Teoría clásica de la Computación extrayendo conclusiones filosóficas de fundamental importancia para entender el nacimiento de la I.A. En su segunda parte se analizan los nuevos modelos propuestos haciendo balance de sus dificultades y su interés relativo.

Palabras clave: Informática, Máquinas de Turing, supertareas, hipercomputación.

ABSTRACT

This work analyzes the state of the art in theoretical computer science from a philosophical point of view especially directed to non expert readers. In the first part, sections 1 to 4, I offer some of the keys needed to understand the importance of classic Computation Theory obtaining at the same time some philosophical conclusions of fundamental importance to understand properly the birth of the I.A. In the second part I discuss some alternative computation models assessing the virtues and difficulties posed by them.

* Investigación parcialmente financiada por los proyectos HUM2006-12848-C02-01 y BFF2003-08998-C03.

Key words: Computer Science, Turing machines, supertasks, hypercomputation.

1. Lo que fue primero

Lo que aquí me propongo es ofrecer a personas no expertas una guía o estado de la cuestión en el terreno de lo que podríamos llamar *computación teórica*. Esta disciplina no difiere sustancialmente de aquello a lo que se suele denominar Teoría de la Computación, pero intenta acotar un poco más su alcance. Nos interesa, ante todo, aquellos aspectos de tipo fundamental más directamente relacionados con la Lógica formal y con la propia Filosofía. La computación teórica se ocupa en la actualidad del estudio de modelos abstractos destinados a imaginar métodos de ejecución de rutinas capaces de mejorar en algo los modelos teóricos vigentes. La implementación no es, sin embargo, nuestra especialidad ni nuestro fuerte.

Es muy importante entender que el estudio de la computación consiste, en última instancia, en el análisis de todo aquello que tiene que ver con la noción de tarea efectiva o *algoritmo*. Las tareas efectivas que interesan a la Teoría de la Computación son aquellas que pueden ser ejecutadas en un ordenador quizá ideal y no sometido a limitaciones físicas de espacio. Si hubiera tareas efectivas, propiamente dichas, que pudieran ser ejecutadas por seres humanos pero no por ordenadores el asunto que nos ocupa difícilmente llegaría a tener importancia para la Filosofía. Se trataría, simplemente, de una rama de la ingeniería ocupada de remedar de forma más o menos fiel las habilidades del ser humano. Pero al no existir, hasta donde sabemos, tareas efectivas merecedoras de ese calificativo que no puedan ser desarrolladas por ordenadores aunque admitiendo, eso sí, un cierto grado de idealización, el asunto adquiere una relevancia filosófica evidente. La Teoría de la Computación se convierte, al menos para el filósofo informado, en una excepcional plataforma para estudiar el modo en que nuestra mente interpreta la ejecución de tareas sometidas a reglas objetivas –algoritmos, en definitiva–. La computación teórica es el dominio en el que este tipo de consideraciones se hacen fuertes absorbiendo toda nuestra atención. La importancia de todas estas consideraciones y debates para el desarrollo de la I.A. es obvia. De hecho, se puede considerar que este trabajo cae dentro de las preocupaciones recientemente mostradas por los teóricos de la I.A. ante el franco y abierto rearme teórico de aquellos que piensan que las habilidades cognitivas humanas nunca podrán ser reproducidas por medios artificiales.

Una vez aclarados los objetivos creo que corresponde ofrecer algún tipo de información básica destinada sobre todo a disipar algunas dudas y malentendidos frecuentes, aunque he de decir que justificados. No deseo, ni ahora ni más adelante, adoptar ese tono ofendido tan frecuente en exposiciones expertas dirigidas a un público no necesariamente informado. Disipar dudas y malentendidos sólo responde al deseo de fijar un punto de partida claro y común, eso es todo.

Lo primero que interesa dejar claro en este caso es qué fue antes, los ordenadores o la teoría que analiza su comportamiento. Así formulado todo apunta a que los ordenadores que estudia la Teoría de la Computación surgieron del taller de algún visionario del mismo modo que muchos otros ingenios característicos de nuestra época. La máquina de vapor, el motor de explosión o los procesos fotográficos pueden ser ejemplos de ello. En todos estos casos el artefacto antecede a la teoría que permite explicar su comportamiento. Se sigue aquí un modelo que podríamos calificar como *artefacto primero, teoría después*. Pero lo cierto es que el nacimiento de la Teoría de la Computación no sigue, por mucho que lo parezca, esa pauta. La teoría antecede en este caso a los primeros ingenios que pueden considerarse propiamente como sus implementaciones y lo hace además en varias décadas.

Los primeros resultados que engrosan el acervo de la Teoría de la Computación se producen en un periodo extraordinariamente breve e intenso siendo pocas –o relativamente pocas– las personas que participan en ello y que están en condiciones de entender lo que está sucediendo. Durante el invierno de 1936-37 se publican de forma casi simultánea e independiente dos artículos absolutamente determinantes para toda la historia posterior. El primero lo firma Alonzo Church y se titula «An Unsolvable Problem of Elementary Number Theory»[1] mientras que el segundo, «On Computable Numbers, with an Application to the Enstscheidungsproblem»[2], se debe al genio del joven matemático inglés Alan Turing. Términos como «computable», «computabilidad» o «computación» no son aún habituales en ese momento por lo que todo lo que allí se dice se integra de forma más o menos natural dentro del dominio de la denominada *Teoría de funciones recursivas*, un área bastante abstracta y también novedosa, perteneciente al ámbito de las funciones numéricas y al de la entonces pujante Lógica matemática. Estos dos ensayos, sobre todo el segundo de ellos, contienen los resultados básicos que soportan la interpretación vigente de aquello que cabe entender por tarea efectiva o algoritmo.

La traducción de la teoría a la práctica no empieza a producirse hasta mediados de la década de 1940. No obstante, es sólo a partir de 1950 cuando se toma realmente en serio la posibilidad de construir ordenadores muy lejanos en diseño y propósito de aquello que hoy consideramos como tal.

1. «An Unsolvable Problem of Elementary Number Theory», *The American Journal of Mathematics*, vol. 58, pp. 345-363. Reimpreso en Davis, M. (ed.), *The Unsolvable*.

2. «On Computable Numbers, with an Application to the Enstscheidungsproblem», *Proceedings of the London Mathematical Society*, ser. 2, vol. 42, pp. 230-265. Reimpreso en Davis, M. (ed.), *The Unsolvable*.

2. Lo que no se puede hacer

Otro tópico que acompaña con frecuencia a todo lo que rodea a la Teoría de la Computación es el éxito. El estudio de ejecución mecánica de tareas efectivas habría nacido del intento de ir incorporando en ingenios mecánicos cantidades crecientes de lo que son nuestras habilidades cognitivas básicas. Ese proceso habría empezado por lo más simple, operaciones aritméticas elementales como suma y producto, para ir ganando terreno hasta alcanzar las sofisticadas tareas que nuestras máquinas son capaces de efectuar en la actualidad. Así vista, la Teoría de la Computación se vería reducida a los objetivos propios de la I.A. y entroncaría con una larga tradición empeñada en averiguar la sustancia de nuestra originalidad poniendo a prueba su resistencia a ser imitada. Las leyendas griegas sobre autómatas, la tradición de los muñecos animados capaces de interpretar melodías al piano, o las máquinas calculadoras de Leibniz o Pascal serían episodios ancestrales de lo que ahora es un dominio bien establecido. La máquina calculadora de Babbage supone, para este modo de entender las cosas, un hito destacado.

Pero las cosas no fueron así. Es cierto que siempre ha existido una fuerte curiosidad por saber si nuestras habilidades podían ser imitadas por medios artificiales, pero la Teoría de la Computación no nace de ella. Sólo más tarde se produce una confluencia tras observar las posibilidades técnicas de los actuales ordenadores y entender en toda su dimensión las implicaciones de la nueva teoría. Los resultados que dan forma a la Teoría de la Computación son esencialmente negativos y se ubican dentro del campo de la Lógica matemática de forma bien directa. Los artículos que he mencionado líneas atrás hacen referencia ambos a un problema conocido como *problema de la decisión*[3]. Su enunciado plantea la cuestión de si existe o no un procedimiento de decisión –de ahí su nombre– que sea capaz de determinar en tiempo finito si una fórmula de la Lógica elemental[4] es o no un teorema. Al ser puramente formal, los resultados que afectan a la Lógica afectan también a todas las teorías que puedan expresarse con su concurso haciendo así que el problema inicialmente referido a la Lógica se refiera y afecte de igual manera a todas esas teorías derivadas. El que observa por primera vez la importancia de esta cuestión es el matemático alemán David Hilbert quien la incluye en el listado de problemas pendientes que presenta al ya famoso Congreso de Matemáticas de París de 1900.

Tanto Church como Turing observan que cualquier respuesta a este problema pasa por aclarar de manera absolutamente satisfactoria a qué se hace referencia con *procedimiento de decisión*. Es claro que un procedimiento de decisión es un tipo

3. Se trata del conocido *Entscheidungsproblem*.

4. Por Lógica elemental se entiende aquí lo que en los manuales de Lógica se describe como Lógica de Primer Orden. Este es el sistema que permite expresar buena parte de nuestro conocimiento matemático y en el cual se pueden expresar, en definitiva, todas las rutinas que implementan nuestros ordenadores. También se ha intentado emplear como lenguaje de referencia para la elaboración de un lenguaje lógicamente perfecto orientado al discurso ordinario pero sin mucho éxito, esa es la verdad.

de rutina o algoritmo que permite ofrecer la respuesta a un problema dado. Por tanto, se puede convenir que todo el asunto reposa al final en lo que se pueda entender por *procedimiento efectivo* o *algoritmo*. La respuesta que ofrecen Church y Turing al problema planteado por Hilbert, tras aclarar este último aspecto de formas realmente distintas, es negativa. No hay un algoritmo que permita determinar si dada una fórmula ésta es o no un teorema de la Lógica Elemental. Esto no quiere decir que no se pueda alcanzar esa conclusión en muchos casos, sino que no existe un método general aplicable a todos ellos. El método seguido para obtener este resultado hace que la limitación a la que se refieren tanto Church como Turing no pueda ser entendida como algo propio de la Lógica, sino más bien como una restricción imputable al modo que tenemos de entender en qué consiste una tarea efectiva. Dicho de otra forma, si por tarea efectiva ha de entenderse un procedimiento finito consistente en una serie de instrucciones cada una de las cuales da paso a la siguiente en función de ciertos factores, sin dejar en ningún momento un lugar a la elección arbitraria, entonces, el problema planteado por Hilbert no tiene solución.

La gravedad o importancia del asunto es que la solución negativa al problema de la decisión se obtiene como corolario de un resultado que, como parece deducirse de lo que acabo de decir, es mucho general, afectando en realidad, a la propia noción informal de algoritmo. El resultado general al que hago mención suele denominarse *problema de parada* y se formula del siguiente modo: no hay un procedimiento efectivo –bajo una cierta interpretación formal absolutamente genérica de esta noción[5]– capaz de determinar de manera general si otro procedimiento igualmente efectivo es capaz de finalizar arrojando un resultado cuando se le suministra un input o situación de partida. Nuestros algoritmos no parecen ser, en definitiva, tan eficaces como desearíamos ya que en general no podemos saber si van a terminar salvo que de hecho lo hagan.

Este resultado está ligado a muchos otros que expresan en distintas variantes limitaciones a aquellos problemas que podemos resolver de manera efectiva y constituye, en cierto modo, una versión abstracta o universal de todo ellos[6].

3. Una teoría única

Hablar de problemas que no se pueden resolver de manera efectiva no importa por qué medios no deja de ser, tras los esfuerzos del siglo xx por acabar con cualquier atisbo de una teoría universal en cualquier ámbito, un atrevimiento notable. Más bien deberíamos hablar de problemas que no pueden resolverse de forma efectiva dados estos o aquellos recursos o herramientas, pero no parece que

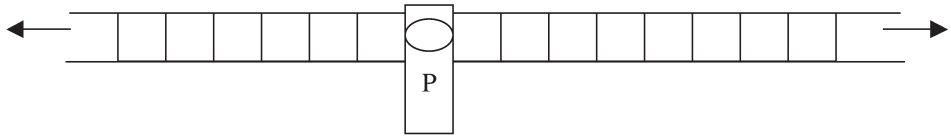
5. La traducción formal más habitual de esta noción es la que ofrecen las máquinas de Turing, pero de ellas hablaremos en breve.

6. La teoría de la complejidad es el estudio del modo en que problemas insolubles se conectan unos con otros y nace en este punto.

tenga sentido hablar de la *imposibilidad absoluta* de resolver un cierto problema. Para entender por qué la Teoría de la Computación puede hablar de ese modo sin entrar en conflicto frontal con la corriente relativista que impera hoy en día basta con tener en cuenta que el objeto de nuestro análisis es la estructura general de cualquier método que podamos denominar efectivo, sin importar su contenido. Un ordenador actual puede comportarse como una calculadora de sobremesa, como un procesador de textos, puede ejecutar las tareas típicas de un navegador o cualquier otra cosa que se nos ocurra. Todas estas tareas son efectivas exactamente en el mismo sentido del término y pueden ser expresadas en un único lenguaje universal capaz de expresar cualquier tarea efectiva que podamos concebir. Al menos hasta donde sabemos, ya que la segunda parte de este trabajo está dedicada, precisamente, a estudiar las críticas a esta interpretación universal de la efectividad de una tarea o rutina.

Para entender declaraciones tan radicales es conveniente presentar la piedra angular de este ensayo y de tantos otros dedicados a estos mismos temas. Me refiero a las denominadas *máquinas de Turing*. Las máquinas de Turing son el modelo teórico sobre el que reposa en la actualidad la Teoría de la Computación prácticamente al completo. Fueron propuestas en el artículo seminal de Turing de 1936 y constituyen una de las aportaciones más relevantes de la lógica matemática de todos los tiempos.

Pasando a las descripciones, diré que una de estas máquinas consta de tres partes bien diferenciadas. Una cinta de cálculo divida en celdas que se puede prolongar indefinidamente en ambos sentidos. Una cabeza lectora capaz de desplazarse de celda en celda en cualquier sentido de la cinta. Este dispositivo lee el contenido de la celda sobre la que está estacionada, y puede borrar el contenido de la misma, o escribir una marca en ella. Finalmente, un programa P en cuya descripción me entretendré un poco más.



Un programa no es sino una colección finita de instrucciones. Cada instrucción consta de cuatro elementos que podemos representar como nXYm. El primero y el último son números naturales. X e Y aluden al contenido de la cinta y a la acción que la cabeza lectora respectivamente. Para simplificar, sólo admitiremos la posibilidad de que una celda esté en blanco, lo que se representa como «b», o que se encuentre marcada por un símbolo previamente convenido, «*», por ejemplo. Las acciones que puede ejecutar la cabeza lectora son: imprimir una marca –si ya hay una se sobreimprime, pero no se añade–, borrar una marca, si existe, desplazarse una celda a la izquierda –que se representa como «i»– y desplazarse una celda a la derecha –representado por «d». Supongamos que una máquina ejecuta

la instrucción 1b*2. La acción que ésta realiza es la siguiente: si la celda sobre la que está estacionada la cabeza está en blanco, procede a marcar la celda y pasa a ejecutar la instrucción 2. Existe una gran cantidad de ejemplos que el lector puede consultar si así lo desea, pero por el momento basta para lo que aquí se pretende.

Resulta sorprendente que una de estas máquinas pueda implementar un sofisticado programa de ordenador, pero de hecho es así. El formalismo en que se expresan las máquinas de Turing es el lenguaje universal al que hacía mención líneas atrás. Cualquier programa de los que existen en la actualidad puede expresarse de forma equivalente en términos de una máquina de Turing. Pero aún hay más. Durante los intensos años en que se gestaron los principios básicos de la Teoría de la Computación no sólo se propuso el modelo que acabo de ilustrar. Junto a las máquinas de Turing surgieron otras propuestas[7] destinadas igualmente a expresar tareas efectivas. Pronto se pudo comprobar que todos estos formalismos, por muy distintos que pudieran resultar, permitían formular la misma clase de procedimientos efectivos. Si una cierta tarea podía ser descrita en términos de uno de estos lenguajes podía serlo también en los restantes. El tiempo y la sencillez del método han querido que sean las máquinas de Turing el formalismo de referencia, pero muy bien podría haber sido de otro modo. Esta peculiar coincidencia llevó a diversos autores a proponer una conjetura que hoy en día conocemos como *Tesis de Church*. Su enunciado puede ser expuesto del siguiente modo: toda tarea efectiva que podamos concebir y expresar de forma precisa en un determinado formalismo puede ser expresada igualmente en el formalismo de las máquinas de Turing. Puesto que este formalismo permite trasladar la ejecución de una tarea a un ingenio mecánico, lo que la Tesis de Church sostiene, ampliando algo su alcance, es el hecho de que toda tarea que el ser humano puede realizar de manera efectiva puede ser igualmente ejecutada por un ingenio de tipo artificial.

Palabras tan gruesas reposan fuertemente en el hecho peculiar de que sólo exista una única forma de interpretar de manera precisa el significado de la noción de tarea efectiva. Es el tiempo el que ha preservado esta conjetura haciendo que pase de ser una mera hipótesis a un enunciado firmemente sostenido en nuestra experiencia. No es, y esto hay que dejarlo claro, un teorema del cual podamos ofrecer una demostración. Sólo es una tesis, y por tanto, resulta ser algo potencialmente rebatible. Su posición estratégica parece por otra parte evidente. Encontrar tareas que puedan denominarse efectivas en algún sentido plausible del término y no puedan ser encarnadas por una máquina de Turing puede representar un gran avance para todos aquellos que piensan que la mente humana es superior siempre a cualquier ingenio mecánico que aspire a imitarla. Importante, aunque no definitivo, ya que la Tesis de Church sólo compara formalizaciones distintas de una noción intuitiva.

7. Las principales fueron estas tres: *funciones recursivas generales*, *cálculo-λ* y *funciones recursivas por minimalización*.

Lo que importa ahora es entender la importancia y hasta cierto punto la excepcionalidad de esta situación. Hay pocas, poquísimas, teorías en cualquier ámbito del conocimiento que podamos concebir para las que no exista un rival capaz de diferir sustancialmente en algún punto. El dominio de las matemáticas de principios del siglo XX exhibía ejemplos notables en este punto. No hay, por ejemplo, una única forma de interpretar la noción general y abstracta de *conjunto*. No existe un único formalismo que podamos considerar como el sistema lógico por excelencia, la Lógica ideal o perfecta. No hay tampoco un sistema aritmético perfectamente fiable, ni una única geometría. ¿Por qué tendría que haber una única forma de interpretar formalmente la noción intuitiva de tarea efectiva? Sin embargo, así es.

4. Teoría de la Computación e I.A.

Quizá sea excesivo decir que la I.A. depende fuertemente de la existencia de una única forma de interpretar la idea de algoritmo, pero lo que sí es cierto es que debe su prosperidad –relativa– a algunos de los aspectos más característicos del modo en que la Teoría de la Computación ha interpretado ese concepto. El deseo de *imitar* las habilidades cognitivas del ser humano no es, como ya he dicho, algo nuevo en nuestra tradición cultural. La antigüedad de este tópico permite hablar, incluso, de obsesión. Siempre ha habido corrientes *mecanicistas* alentadas por la posibilidad de crear trasuntos artificiales del ser humano, ya sea en todo, o en parte. Pero lo que nunca ha permanecido constante es la teoría que en cada caso actuaba como marco para evaluar las posibilidades de ese proyecto. Los autómatas músicos del siglo XVIII y XIX y las máquinas calculadoras de Leibniz o Pascal se apoyan en la *mecánica racional*, los sistemas actuales en la Teoría de la Computación. ¿Qué hace que se produzca este cambio de paradigma?

Una de las características más destacadas del modelo computacional vigente es el grado de autorreferencia que incorpora sin que por ello aparezcan las tan temidas paradojas[8]. Un enunciado autorreferencial típico es el del *mentiroso* también conocido como la paradoja del *cretense*. Si afirmo mientras hablo que eso que acabo de decir es falso, incurro en la paradoja del mentiroso. El enunciado «esto que digo es falso» es autorreferencial porque tiene como objeto ese mismo enunciado y es paradójico porque no hay una forma consistente de determinar si es verdadero o falso –invito al lector a que lo intente–. ¿En qué sentido hay autorreferencia en la Teoría de la Computación? Una máquina de Turing se distingue de otra por el programa que ejecuta y un programa es, como ya hemos visto, una cadena finita de instrucciones. Asignar un número a ese programa que lo identifique plenamente es una tarea elemental que se denomina *codificación*. Un código es un entero

8. Tampoco debe entenderse que la autorreferencia sea en este caso totalmente innocua, ya que de hecho es la que se encuentra en el origen del tipo de limitaciones que he explicado en el apartado anterior.

positivo que contiene toda la información que podríamos obtener si dispusiésemos directamente del programa del cual es código. Obtener el programa codificado por un entero positivo se logra mediante la correspondiente operación de *decodificación*. Ahora bien, una máquina de Turing procesa enteros positivos y nada impide que una de estas máquinas contenga los mecanismos para decodificar enteros positivos y obtener los programas que estos representan. La autorreferencia es ahora evidente: ¿qué sucede cuando una máquina con código *i* procesa como input el número *i* y posee además los mecanismos para decodificar enteros positivos?

Turing demostró que esas máquinas dotadas de recursos para decodificar enteros obteniendo el programa que estos representan son máquinas de Turing igualmente. No forman una categoría aparte ni son precisos recursos extra de ningún tipo. Son lisa y llanamente una máquina de Turing más. El resultado por el que se llega a esta conclusión se conoce como *teorema de existencia de máquinas universales* y fue demostrado por Turing en el mismo texto en que ofrece la solución negativa al problema de la decisión. Una *máquina universal* U(x,y) es una máquina de Turing cuyo programa actúa al menos sobre dos inputs –enteros positivos–. El primero de ellos, *x*, es tomado como el código de una cierta máquina de Turing y el segundo, *y*, como el input cuyo procesamiento se ordena a la máquina con código *x*. El resultado que la máquina *x* arroja cuando procesa el input *y* es tomado como el propio resultado de ejecutar U(x,y). Si M^x representa la máquina con código *x*, lo que acabo de decir puede expresarse perfectamente en la siguiente igualdad: $U(x,y)=M^x(y)$.

¿Qué importancia tiene la existencia de máquinas universales para el tema de la I.A.? Hasta la aparición de la Teoría de la Computación la imitación de tareas típicamente humanas o bien era tratada como una mera hipótesis –las más de las veces en el dominio de la ciencia ficción– o se limitaba a algún aspecto concreto y aislado. El detractor de esta posibilidad, al que de forma genérica bautizaremos como *mentalista*, siempre podía apelar a la ausencia de un método para incorporar en un ingenio mecánico todas las facultades cognitivas del ser humano de forma simultánea. La cosa cambia a partir de la demostración del teorema de existencia de máquinas universales. Porque, ¿cuántas tareas concretas puede ejecutar una máquina universal? La respuesta es obvia: todas aquellas que pueden ser ejecutadas mediante máquinas de Turing. Basta con identificar su código.

A partir de este momento, la posibilidad de concebir la imitación de las habilidades cognitivas del ser humano toma tierra dejando de ser una mera ficción o sueño de filósofo. Pero aún ha de pasar algún tiempo hasta que los investigadores tomen conciencia de la aparición de una nueva disciplina a bautizar bajo el rótulo de I.A. Uno de los primeros autores en plantear las preguntas pertinentes es Turing, de nuevo, al proponer la cuestión de *si puede pensar una máquina*, donde máquina ha de interpretarse ya bajo el paradigma de la Teoría de la Computación. El artículo correspondiente ve la luz en 1950[9]. Pero el hito más destacado, al menos

9. «Computing Machinery and Intelligence», *Mind* 49, 1950, pp. 433-460.

para la comunidad formada en torno a la I.A., es la Conferencia de Dartmouth celebrada en 1956. Es allí donde se configura el programa que ha estado vigente durante las últimas décadas.

El paso del tiempo y la ausencia de resultados definitivos ha venido provocando una creciente sensación de desconfianza que ha llevado a muchos investigadores a empezar a mirar a su alrededor. Quizá la teoría de fondo posee insuficiencias que de algún modo pueden estar bloqueando el progreso de la I.A. Los fundamentos de la interpretación estándar de la computabilidad, a la que he dedicado las secciones anteriores, tienen que ser revisados y criticados a la búsqueda de posibilidades que puedan haber quedado ocultas. Esta nueva vía de investigación empieza a ser reconocida bajo el rótulo de *hipercomputación*. No se trata de un ámbito dotado de un objetivo y metodología demasiado definidos, sino de un conjunto de estrategias que intentan proponer modelos teóricos alternativos a aquellos de que partieran Turing y Church en la década de 1930. Este tipo de trabajo ha adquirido alguna relevancia en nuestros días al mostrar conexiones interesantes con otras teorías hasta ahora muy alejadas, pienso en la física teórica, en la mecánica cuántica, o en el propio evolucionismo. El resto de este trabajo ofrece una guía de campo para aquellos que quieran entrar en contacto con lo que se está haciendo en este terreno y atreverse, quizá, a proponer sus propias opiniones.

5. De la Computación a la Hipercomputación

El término *hipercomputación* se emplea para referirse a la siguiente idea:

> …el cómputo de funciones o números que no pueden ser calculados en el sentido de Turing […] i.e., no pueden ser calculados a través de un número finito de pasos realizados por un agente humano dotado de lápiz y papel que opera de manera efectiva [Copeland 2004, p. 251].

Su uso empieza a extenderse durante la década de 1990 convirtiéndose en una referencia habitual a partir del *Hypercomputation Wokshop* celebrado en el *University College* en 2000. Existen, no obstante, otras expresiones que pueden encontrarse en la literatura para referirse a esa misma idea. «Supercomputación», «computación fuera de los límites de Turing», y «computación no clásica» son quizá las más frecuentes.

Lo interesante de la idea misma de *hipercomputación* es que durante los últimos años ha dejado de ser tratada como una curiosidad propia de lógicos o filósofos para ser considerada desde el propio frente de la I.A. Creo que nos corresponde intentar entender por qué es así.

10. Copeland por ejemplo.

Son varios los autores que ven el origen de la hipercomputación en el concepto de *oráculo*[10]. La primera vez que se introduce este misterioso concepto es en la obra de Turing, «Systems of Logic based on Ordinals», publicada en 1939 y en la que se resumen los resultados de su Tesis Doctoral. Turing presenta la idea de una forma directa y sin muchas concesiones, como por otra parte es habitual en él:

> Supongamos que se nos suministran ciertos recursos sin especificar aptos para resolver cuestiones numéricas; algo similar a un oráculo. No entraremos en la naturaleza de este oráculo salvo para dejar claro que no puede tratarse de una máquina. Con la ayuda de este oráculo formamos un nuevo tipo de máquina (sean éstas las O-máquinas) que tiene como uno de sus procesos básicos el de resolver un determinado problema numérico [Davis 1965, p. 167].

Aunque no es nada fácil determinar exactamente el papel que este concepto desempeña en un texto tan complejo como aquel[11], me inclino por seguir la interpretación que hará Rogers[12] años más tarde cuando opta por ver en este concepto tan sólo un recurso técnico útil para introducir la idea de *computabilidad relativa* y con ella los denominados *grados de reducibilidad*. No se trataría, en definitiva, de una noción destinada al consumo filosófico sino tan solo a la dura tarea de analizar y comparar la insolubilidad de ciertos problemas numéricos.

Lo cierto es que no parece nada plausible que con el concepto de oráculo, tal y como se presenta en la cita anterior, Turing estuviera intentado extender los límites de su concepción original de la computabilidad. Lo que sí juega un papel importante en esta dirección es su idea de generar una serie de sistemas –lógicas– con respecto a la cual sea posible definir un límite, es decir, lo que el propio Turing denomina una *ordinal logic*. Este sistema límite se podría emplear entonces para probar más cosas de las que se pueden probar en cada uno de los sistemas de esa serie[13]. Hablaré de ello más tarde.

Los intentos por desarrollar modelos alternativos de la computabilidad tienen, eso me temo, un origen mucho menos espectacular. Si tomamos en serio los datos que aporta el propio curso de los acontecimientos, encontramos que el primer desafío al modelo computacional clásico tiene lugar con la introducción del *modelo conexionista* y la definición del concepto de *red neuronal*. Una de estas redes está formada por una cantidad determinada de neuronas interconectadas para las cuales se han especificado una serie de *potenciales de activación*. Por debajo de ese potencial la neurona no se activa. Cuando se alcanza o supera esa cifra la neurona produce un impulso característico que es transmitido por la red.

A una máquina de Turing se opondría una red neuronal, existiendo alguna posibilidad de que estas últimas fueran capaces de hacer algo que queda más allá

11. Turing no vuelve a mencionar el asunto fuera del apartado 4 de la obra mencionada.
12. Rogers, 1967, p. 128.
13. [Turing, 1939, §8].

del alcance de las primeras. Para evitarnos suspenses innecesarios advertiré de la existencia de resultados posteriores por los que se establece que una red neuronal estándar[14] no es más potente que una máquina de Turing. Interesa eso sí destacar cuál es el fondo del asunto. La capacidad de una máquina de Turing para imitar en principio nuestras habilidades cognitivas en lo tocante a la ejecución de tareas efectivas no reside, desde luego, en la similitud estructural con nuestro cerebro. El uso del modelo computacional clásico para desarrollar las posibilidades de la I.A. tiene que apoyarse necesariamente en alguna teoría que aclare ese punto. Esto es de lo que se ocupa, precisamente, lo que se ha venido a llamar paradigma funcionalista. Lo que nos importa del funcionalismo se puede resumir ahora en una sucinta afirmación acerca de los criterios por los cuales dos entidades deben ser consideradas como entidades del mismo tipo. Para el funcionalismo esto sucede cuando realizan la misma función, extremo que se determina a su vez por la conducta observable. No quiere decir que en tal caso sólo estemos ante una y la misma entidad, sino que a parte de su materia bruta y su localización espaciotemporal esas entidades serán en todo lo relevante, entidades del mismo tipo. Sólo así se puede salvar la evidente distancia que existe entre la organización estructural de nuestro cerebro, fielmente reproducida por una red neuronal, y la estructura formal de una máquina de Turing.

La hipótesis planteada en su momento por el modelo conexionista reposa sobre principios muy distintos. La similitud estructural cuenta, y mucho, a la hora de obtener conductas y comportamientos similares. Si el cerebro humano se estructura en redes de neuronas sólo se podrán imitar, comprender y reproducir sus habilidades mediante modelos que reflejen fielmente la estructura de la entidad que produce esa conducta, el cerebro en este caso. Este enfrentamiento continúa hoy en día bajo formas algo evolucionadas pero que recuerdan aún los términos básicos del problema.

Resulta curioso que estas alternativas vieran la luz de forma prácticamente simultánea en la Dartmouth Conference de 1956 impidiendo así que el vasto proyecto de la I.A. quedase claramente identificado con una sola de ellas. Pienso que es el desarrollo posterior de la ingeniería informática lo que favorece la tendencia a asimilar I.A. con el modelo computacional clásico, y no otra cosa.

La segunda línea de investigación que debe ser mencionada a la hora de entender los orígenes y atractivo de la *hipercomputación* tiene un origen completamente diferente. Procede, de hecho, de las discusiones filosóficas surgidas en torno a la posibilidad lógica de concebir lo que se denomina una *supertarea*.

Este concepto se remonta a su vez a las paradojas de Zenón y muy en especial a la de *Aquiles y la Tortuga*. Resulta llamativo que el estudio de este tópico, es decir, la posibilidad de concebir sin contradicción una tarea compuesta de infinitos pasos que finaliza, no obstante, en tiempo finito se iniciara algún tiempo antes que la propia Teoría de la Computación. Los autores que protagonizan esta

14. Veremos más adelante que eso mismo no se puede afirmar con rotundidad en ciertos tipos de redes consideradas en la actualidad.

polémica no tienen en mente un concepto de tarea que pueda traducirse sin más al que nace años más tarde con Turing. Creo, no obstante, que podemos centrarnos sin mayor problema en aquel caso en que cada uno de los pasos de una de estas tareas se ejecuta de forma efectiva y se conecta de ese mismo modo con los restantes.

Russell 1914, Weyl 1927 y Blake 1926 consideraron de forma independiente los distintos aspectos del concepto de *supertarea* inaugurando –o retomando– una tradición que se reaviva a mediados del siglo xx con los trabajos de Thomson 1954 –y su *lámpara prodigiosa*–, de Benacerraf 1962 y de Chihara 1965.

El responsable de que esta polémica entre en contacto con el ámbito de la computación teórica parece ser, a falta de otra evidencia, G. Boolos cuando en su *Computability and Logic,* Boolos & Jeffrey 1974 introduce el concepto de *Zeus Machine*. En el momento actual podemos considerar a J. Copeland como el principal defensor de esta opción sostenida a través de lo que tanto él como sus seguidores denominan *acelerating Turing Machines* –ATM's de aquí en adelante–. De lo que se trata es, en definitiva, de analizar si es posible concebir, al menos en principio, tareas efectivas que adopten la forma de una supertarea. Si tal cosa es posible, parece que estaríamos ante tareas que ninguna máquina de Turing podría ejecutar sin que por ello se pueda considerar que no se trata, en algún sentido del término, de una tarea efectiva.

Muy cerca de esta segunda línea de investigación en hipercomputación podemos encontrar otra que se centra en el intento de superar ciertos resultados de limitación relativos al alcance y potencia de la Lógica a través del uso de sistemas basados en ordinales. Es en este punto en el que el trabajo de Turing mencionado líneas atrás adquiere auténtica importancia. Feferman recuperaría esta idea en 1962 con su *Transfinite recursive progressions of axiomatic theories* siendo E. Gold quien en 1965 sitúa el problema en el campo de la computación no clásica al introducir el concepto de *Limiting Recusion*. Esta línea de investigación comparte con la anterior el intento de sacar algún partido del concepto de *infinito* para obtener consecuencias que vayan algo más allá de los requisitos estrictamente finitistas aceptados por el modelo computacional clásico.

La última línea de investigación que parece tener consecuencias por lo que hace al concepto de hipercomputación tiene que ver con el desarrollo de la denominada *computación cuántica*. Este asunto, dada su orientación, queda fuera del alcance de este estudio y me limito sólo a mencionarlo.

El creciente interés por la hipercomputación no se explica, no obstante, si no se tiene en cuenta el creciente sentimiento de desánimo que reina entre los impulsores de la I.A. Sentimiento que en ocasiones puede ser difícil de apreciar para aquellos que nos encontramos fuera de la comunidad en que se produce. Tengo la impresión de que toda la polémica abierta entorno al decaimiento de la fe en la I.A. tiene que ver y mucho con el propio agotamiento de la primera generación de investigadores dedicados profesionalmente a su desarrollo y de sus esquemas y paradigmas teóricos. Las altas expectativas suscitadas en su día, difíciles de cumplir en cualquier caso, también están pasando su factura.

6. Algunos de los principales modelos en hipercomputación

No creo que tenga mucho sentido analizar las numerosas propuestas, con todas sus variantes, que han sido defendidas durante los últimos años en el ámbito de la hipercomputación. Lo más seguro es que acabáramos por formar genealogías más parecidas a las primitivas taxonomías, o bestiarios, con que los naturalistas dieron sus primeros pasos en su complejo mundo. Llegaríamos, como ellos mismos hicieron, a considerar varias veces la misma especie unas veces bajo un cierto aspecto y otras a partir de otro muy distinto. No ahorraremos el esfuerzo.

Tras estudiar un número suficiente de las muchas propuestas disponibles en el mercado se observa que es posible apreciar la existencia de dos formas características de afrontar el problema.

La primera de ellas entiende que la descripción de cualesquiera modos alternativos de interpretar la noción intuitiva de algoritmo es una tarea que pertenece propiamente al dominio de la lógica o las matemáticas. Se respetaría así el mismo origen que en su día tuvo lo que hoy denominamos modelo clásico o estándar. Esta estrategia puede ser calificada, si de dar un nombre se trata, como *estrategia formal*. La segunda vía entiende que esta forma de entender las cosas sería más bien parte del problema que de la solución. Para hallar realmente modelos alternativos a la interpretación vigente de la computabilidad se hace preciso identificar fenómenos en la naturaleza que no puedan ser representados en forma apropiada dentro del marco formal de que hoy disponemos. Sería el propio hecho físico el que puede ayudar a superar los límites teóricos del modelo clásico. Parece razonable calificar esta alternativa como *estrategia material*.

Más adelante tendremos la oportunidad de ver de qué manera se relacionan estas dos estrategias. La tensión existente entre estas dos estrategias tiene que ver claramente con un problema que va mucho más allá de los objetivos que nos hemos marcado aquí. Se trata, en definitiva, de otro episodio más de la pugna establecida entre software y hardware a la hora de interpretar su función y peso en la interpretación del la computabilidad. Éste es un asunto que nunca ha parecido estar zanjado por completo.

Estrategia formal

Los modelos que se obtienen al adoptar esta estrategia suelen estar basados en la alteración de alguna de las características de la arquitectura básica de una máquina de Turing[15]. El primer cambio que podemos considerar tiene que ver, cómo no, con el número de pasos que es posible ejecutar en un dispositivo antes de alcanzar un

15. La obtención de la jerarquía de modelos de menor potencia que las propias máquinas de Turing que conocemos por el nombre de Jerarquía de Chomsky responde, en definitiva, a esta misma idea.

estado final o de parada. Gold en Gold 1965 introduce la distinción entre *limit states* y *successor states*, terminología típica en el manejo de series inductivas.

Esta idea ha sido retomada en los últimos años por Hamkin y Lewis en su definición de las *Infinite Time Turing Machines* –ITTM–. El primer estado límite de una de estas ITTM's sería ω. Este sería el estado que resulta de considerar una máquina de Turing estándar que no finaliza su rutina en un número finito de pasos. Es importante aclarar que en este caso el problema no reside tanto en el modo en que una de estas máquinas puede alcanzar uno de estos estados límite concebidos claramente como una mera ficción formal[16]. Lo interesante de esta propuesta, como de otras del mismo tipo, se encuentra en las consecuencias de tipo lógico que pueda haber de cara a definir con precisión funciones numéricas no Turing-computables. Estas funciones podrían ser, no obstante, calculables de forma efectiva en algún nuevo sentido del término.

Las *Zeus Machines* definidas por Boolos y Jeffrey o las *Acelerating TM's* de Copeland añaden a la intuición básica de Gold o Hamkin y Lewis idealizaciones dotadas de un cierto contenido físico o material. Se aportan datos del *modus operandi* de aquellas máquinas que en cada caso se suponen capaces de ejecutar supertareas. El matiz, como vamos a ver a continuación, puede resultar importante. Las máquinas de Zeus o las máquinas aceleradas de Copeland trabajan ambas de tal modo que cada paso que ejecutan dentro de una rutina se realiza en la mitad del tiempo que el anterior. Es decir, su forma de actuar sigue, por lo que al tiempo de ejecución se refiere[17], la serie $1/n^2$. Como es sabido el sumatorio de esta serie converge de tal modo que nunca supera la unidad. El interés de estas propuestas reside seguramente en la gran cantidad de información que arrojan con respecto al modo de ejecutar una supertarea. Esto no significa, quede claro, que estemos ante propuestas realistas orientadas a la fabricación de experimentos de laboratorio. No hay datos que permitan orientar, al menos en el estado actual de la cuestión, nuestros pasos en esa dirección.

Lo que sí sucede cuando *encarnamos* propuestas tan abstractas como las actuales es que automáticamente surgen ante nosotros una gran cantidad de hipótesis basadas en los distintos aspectos y detalles del modo de operar de estas máquinas. Detalles que, en muchas ocasiones, son extremadamente sugerentes. Alguno de ellos podría sugerir, por qué no, posibles implementaciones en un futuro que, eso es cierto, tampoco parece cercano.

Las críticas más recientes al modelo clásico proceden, sin embargo, de otro concepto de larga tradición entre nosotros, el de interacción. Las máquinas de

16. En una conversación mantendida con Hamkin recientemente, el autor se mostró especialmente interesado en dejarme claro que su desarrollo nunca había pretendido ser nada distinto de un modelo matemático carente de cualquier tipo de consecuencia práctica real.

17. Esto tiene, como parece obvio, consecuencias con respecto al resto de los parámetros: número de celdas, longitud de la cinta de cálculo, recuperación de la información del sistema y otras más complejas relativas a los aspecto formales de los espacios en que podrían operar estas máquinas.

Turing son, al menos así se afirma, entidades completamente transparentes que, curiosamente, se comportan como auténticas cajas negras, agujeros negros quizá, mientras dura la rutina que ejecutan. Nunca emiten un solo bit de información ni aceptan ningún dato externo mientras este proceso tiene lugar. El *pi-calculus* desarrollado por Milner puede ser entendido como un intento, aunque bastante sofisticado, de desarrollar esta idea. Las *coupled Turing Machines* de Copeland pueden ser consideradas también dentro de este apartado[18].

No puede resultar extraño que esta estrategia formal concentre buena parte de su atención en el intento de encontrar una forma de calcular la función asociada al *problema de parada*. Resulta curioso que pese al intenso estudio de este problema sean pocos los autores que se han parado a considerar la estructura lógica del problema de parada y de la función asociada a su cálculo. Esta estructura se basa por entero en la denominada técnica de diagonalización. Sólo T. Ord y T.D. Kieu –en T. Ord y T.D. Kieu 2005– parecen reparar en este aspecto del problema. En un trabajo monográfico dedicado a este punto M. Manzano y el propio autor de estas líneas han mostrado que la correcta interpretación de esta técnica y de su uso puede ser de gran ayuda cuando se intentan entender ciertas limitaciones del concepto vigente de función o tarea computable. Y eso mismo, obviamente, vale para la interpretación de una supertarea.

Como ya dije líneas atrás, ninguna de las dos estrategias descritas, la formal y la material, son por completo independientes. El problema para todos aquellos que han propuesto soluciones dentro de la línea formal es, como por otra parte parece evidente, la posibilidad de implementarlas. Esto lleva, en general, a la búsqueda e identificación de algún fenómeno de tipo natural capaz de representar o encarnar de alguna forma las propiedades formales de cada una de estas propuestas. Un ejemplo citado con profusión es el de los espacios de Malament-Hoghart los cuales serían capaces, al menos en principio, de albergar ingenios del tipo de las ATM's o las ITTM's. Para defenderse del cargo de ingenuidad, esta estrategia suele alegar que sus propuestas no contienen muchas más idealizaciones que aquellas que se pueden observar en una máquina de Turing. Afirmación que, a decir verdad, resulta bastante cuestionable pese a su posible eficacia como línea de defensa de sus posiciones.

Estrategia material

En este caso seré breve. La idea es, tal y como ya avancé líneas atrás, la búsqueda de modelos dotados de capacidades causales capaces de superar las

18. En cierta ocasión y para analizar esa misma falta de interacción con el medio introduje las máquinas-λ en las cuales se aceptaban constantemente inputs nuevos incorporándolos en el proceso de cómputo. Este planteamiento servía para proponer ciertos desafíos al modelo estándar que áun pueden tener vigencia.

limitaciones formales que sí poseen de los mecanismos basados en la Lógica. Esta estrategia afronta el peligro de producir propuestas que de un modo u otro colapsen sobre el modelo clásico. La objeción hecha por los escépticos es clara y certera. Dime con qué recursos –lenguaje– expresas una tarea ejecutable en uno de estos presuntos mecanismos no estándar y yo me comprometo a encontrar una traducción al modelo suministrado por la computación estándar. Al fin y al cabo todo se reduce a la comparación de los lenguajes en que se expresan las pretendidas rutinas que en cada caso se ejecutan. Y buscar posibles traducciones es, claramente, el vicio, o virtud, del lógico. Esta situación es la que encontramos, por ejemplo, cuando se analizan las posibilidades y alcance de la *computación cuántica*. Para comprender si algo puede ser visto o no como un avance real en el cálculo de funciones no-Turing computables tenemos que acudir, en definitiva, al análisis formal de las funciones afectadas. Y al proceder así no es raro encontrase ante problemas de implementación no muy distintos de los que se afronta desde la estrategia formal. Un ejemplo destacado de este conflicto potencial es qué se encuentra al estudiar las redes neuronales. Es bien sabido que las redes neuronales sólo divergen del modelo clásico cuando los valores de activación en la red son números reales o cuando los propios inputs son números reales. Pero, ¿cómo se especifica completamente y de forma finita un número real? ¿Cómo sabemos de qué número se trata?

7. La Hipercomputación entendida como un análisis crítico del modelo clásico

Lo cierto es que en la actualidad no se puede ser muy optimista en relación a los progresos reales de la hipercomputación, aunque tampoco se puede desestimar sin más este empeño. Por decirlo de una forma gráfica: aún queda mucho partido por jugar. Parece obvio que aún estamos muy lejos de encontrar posibles implementaciones físicas de los modelos teóricos que se han barajado hasta la fecha. Pero tampoco es evidente que estos modelos muestren conductas que, evaluadas en su conjunto, resulten preferibles a aquellas que son características del modelo clásico. Es posible, al menos sobre el papel, que una ATM's pueda calcular de manera efectiva funciones que las propias TM's no pueden ejecutar en modo alguno. Entre ellas, y de forma destacada, el propio problema de parada para la clase de las TM's. Sin embargo, algún precio habrá que pagar por ello. Es muy posible, por ejemplo, que nos veamos obligados a aceptar una interpretación nueva y extraña de lo que significa ejecutar una tarea de manera efectiva, o que no podamos identificar la rutina en cuestión del mismo modo que hacemos en el caso clásico. Se trataría, en definitiva, de una situación con pros y contras. Los lógicos sabemos bien que toda adquisición de poder expresivo en un sistema suele venir acompañada de la pérdida de propiedades relevantes que a veces nos son muy caras.

Lo que sí es cierto es que la consideración sistemática de posibles alternativas al modelo clásico está permitiendo elaborar una profunda y poderosa revisión de

algunos de los aspectos más problemáticos del modelo computacional propuesto por Turing. En lo que sigue voy a centrarme, precisamente, en aquellas críticas que me parecen más interesantes.

Como ya dije líneas atrás, los esfuerzos realizados por los proponentes de las ATM's para equiparar las idealizaciones de su modelo a aquellas que acompañan a las propias TM's han hecho que prestemos algo más de atención a un hecho sobre el cual se suele pasar sin mayores problemas. Shagrir y Copeland han insistido mucho en que la demanda de una cinta de cálculo expandible sin límite hecha a la hora de exponer el modo de operar de una TM no es inocente en absoluto. Según ellos esta idealización no sería de mucha menor cuantía que la que permite que una ATM ejecute un número infinito de pasos en tiempo finito. En el caso de las TM's la idealización afecta al espacio –la dimensión de la cinta de cálculo– mientras que en las ATM's se referiría al tiempo. Con independencia de la oportunidad o habilidad de estos autores a la hora de proponer este argumento, lo cierto es que el modelo clásico no compromete una concepción actual del concepto de infinito, mientras que la propuesta de Copeland sí lo hace. Una ATM exige una cantidad infinita de material capaz de contener una cantidad infinita de información y esto, como cualquiera puede entender, no es una idealización menor, sino una necesidad real del modelo. Esto no significa que las ATM's resulten ser una propuesta *lógicamente imposible*, lo que no es el caso, sino más bien que su implementación física resulta conceptualmente imposible. No obstante para todo hay soluciones. E.B. Davis en Davis 2001 propone que la cinta de cálculo y las celdas de que se compone sigan en el caso de las ATM's una pauta de disminución adaptada también a una serie convergente. Si cada celda posee la mitad de tamaño que la anterior, la cinta alcanzará un límite, aunque al precio nada despreciable de operar en un espacio denso y continuo.

Lo que sí me resulta interesante destacar son las idealizaciones sobre las que reposa el modelo clásico es su capacidad para recordarnos que Turing toma como punto de partida una situación experimental construida ad hoc en la cual se consideran ciertos elementos y parámetros dejando fuera otros. Esto no tiene nada de malo, pero es bueno recordarlo. Ese punto de partida, como indica Cleland –Cleland 2003, p. 212– podría haber sido otro. La imagen que Turing tiene en mente –Turing 1936, §9– y que tan adecuadamente representa la forma de operar de manera efectiva de un ser humano, cuando aborda *cierto tipo de problemas*, podría omitir componentes que quizá puedan ser reasumidos por un modelo matemático alternativo. Se sugiere, de hecho, que el punto de vista inicialmente adoptado por Turing no es del todo inocente ya que se orienta claramente al intento de establecer una solución negativa al *Entscheidungsproblem*. Sólo más tarde y quizá apoyándose en los éxitos de sus mecanizaciones en la decodificación del código alemán *enigma*, se atreve Turing a extender las consecuencias de su experimento mental al ámbito de cualquier tarea efectiva que el ser humano sea capaz de ejecutar. Determinar este punto es objetivo, quizá, de un análisis pormenorizado de los hechos que queda fuera del alcance de este trabajo y cuyas conclusiones, así

me lo parece, nunca van a poder ser muy definitivas. La idea general, resulta, sin embargo, bastante atractiva. Quizá porque permite entender las dificultades que el modelo clásico tiene con algunos aspectos absolutamente básicos de nuestra actividad cognitiva. Pienso ahora en nuestra capacidad para abordar rutinas orientadas a fines previamente fijados y con respecto a los cuales hay que rendir cuentas. También pienso en la naturalidad con la que procedemos a efectuar cambios sustanciales de nuestras rutinas sin que por ello reconsideremos el fin al que se dirige. No digo que todas estas maniobras no puedan ser reproducidas de algún modo por métodos computacionales basados en el modelo clásico, digo, más bien, que la forma en que es posible hacerse cargo de aspectos como los anteriores y otros similares no es siempre convincente. Y, desde luego, resulta forzada y altamente problemática, cuando es lo más normal en nuestra forma de afrontar tareas. Todas estas situaciones se producen, seguramente, porque el modelo ideal que Turing tenía en mente a la hora de proponer su modelo no intenta responder a una conducta excesivamente realista. El operario que Turing toma como punto de partida no actúa, justo es reconocerlo, de un modo que podamos considerar *típicamente* humano.

¿Existen otros modelos que puedan servir como punto de partida para establecer traducciones más potentes de la forma en que los seres humanos actuamos cuando procedemos de manera efectiva?

Como ya dije líneas atrás, las TM's son bastante refractarias a la interacción con el entorno y también con otras máquinas del mismo tipo. Ya sé que estamos acostumbrados a hablar de programas que ejecutan subrutinas para otros dados, o de intercambios cliente-servidor. Pero éste no es el punto. Todos estos procesos están controlados siempre desde una unidad central que se mantiene férreamente aislada del medio mientras controla toda esa posible serie de interacciones. Quizá el mejor modo de verlo sea trabajar con una máquina de Turing un momento para ver de qué forma se suspende el tiempo mientras ésta ejecuta sus distintas instrucciones, y de qué modo tan artificial se imita la interacción con el medio o los procesos de cambio que tan habituales nos resultan en la vida cotidiana.

Este comentario se resume perfectamente en la siguiente declaración: el modelo computacional clásico nunca ve más de una única máquina asociada a la ejecución de una determinada tarea. No existen recursos teóricos para expresar otra posibilidad. No es difícil entender entonces por qué resulta complicado hablar de la *evolución* o *cambio* de una determinada máquina, de la influencia del medio o del impacto de que los inputs recibidos tienen sobre su conducta. No es que sea imposible hacerlo, sino que resulta claramente forzado. Las TM's son mecanismos extremadamente discretos con respecto a su forma de actuar. Nunca aportan información intermedia destinada a emitir informes del proceso que media entre la recepción de un input y la emisión de un output. ¿Es posible diseñar modelos en los que la rutina que ejecuta una máquina sea controlada públicamente por otras que interactúan y quizá incluso compiten a la hora de ofrecer la mejor respuesta?

En relación con este problema encontramos el de aportar una *autonomía real* a los productos de nuestra ingeniería informática. Hasta ahora se ha venido

combinando *inteligencia* y *autonomía* –libre albedrío– de tal modo que es siempre el primero de los dos términos el que obtiene la prioridad. Se supone, por ejemplo, que una de las consecuencias de una inteligencia artificial genuina habrá de ser la producción de una conducta original e independiente. Bringsjord, Bello y Ferrucci –Bringsjord, Bello y Ferrucci, 2000– sugieren que se tome en consideración una alternativa al conocido Test de Turing en la que el componente fundamental sea la originalidad y no la inteligencia o la habilidad para contestar preguntas. Estos autores denominan a su test como Test de Lovelace en honor de Lady Lovelace conocida pionera de las posibilidades de la I.A. No obstante, aún se puede ir más lejos si a todo esto se añade el problema de la *especiación* como asunto propio de la I.A. Es decir, no sólo se trataría de evaluar la capacidad de una entidad artificial para producir conducta original, sino también de su habilidad para escapar al control del ser humano y persistir a lo largo del tiempo transmitiendo a otras entidades del mismo tipo sus pautas básicas. Se trata de algo que por ahora pertenece más al dominio de la ciencia ficción que a las preocupaciones propias del ingeniero de software o el teórico de la computación. Hay datos en el horizonte que indican que las cosas pueden estar cambiando en este punto. Diré algo más a este respecto más adelante. En cualquier caso, es justo reconocer la exigencia de movimientos dentro del ámbito de la computación teórica proclives a tomar en consideración conceptos que hasta no hace mucho hubieran sido vistos como una simple extravagancia. La importancia del medio, la presión ejercida sobre una rutina para conseguir una mejor eficiencia, su adaptación a las circunstancias, la plasticidad, etc. son ejemplos de aquello a que me refiero.

8. Análisis inverso

Aunque hay muchos problemas insolubles el primero de todos ellos y al cual se pueden reducir los demás a través de distintas medidas de complejidad es el *problema de parada*. Este problema permite definir un conjunto numérico K consistente en:

> K={x/ $f_x(x)$ tales que la x-ésima función numérica arroja un resultado cuando se le suministra el valor x como input}

Determinar si un entero positivo dado *i* pertenece o no a K constituye la forma canónica y habitual de referirse al problema de la decisión.

Los modelos en computación no-clásica ensayan sus habilidades sobre este problema considerándose que una solución satisfactoria del mismo constituye la mejor tarjeta de presentación posible. No se trata de una obsesión, sino de un hecho directamente inducido por la posición estratégica de K. K es *completo* con respecto a la insolubilidad en términos de funciones Turing-computables. Representa, en cierto sentido, el comportamiento de la insolubilidad de cualquier problema, por extraño que esto nos pueda parecer. Carece pues de un contenido

propio considerándose por tanto como una expresión puramente formal y abstracta de la insolubilidad.

Es bastante sorprendente que pese a la importancia de K a la hora de definir los límites de la computabilidad clásica, sean bastantes pocos los autores que investigan modelos alternativos que han dedicado algún tiempo a analizar cómo se llega a la conclusión de que K no es computable[19]. Como ya he dicho más atrás, la técnica empleada para mostrar ese extremo se conoce como técnica de diagonalización. Mi propuesta consiste pues en llevar a cabo un análisis inverso del estudio de la insolubilidad consistente en lo siguiente:

> Dada la estructura formal de la demostración por la que se establece la insolubilidad del problema de parada, ¿qué condiciones debería poseer cualquier otra clase de procedimientos para que fuera capaz de contener entre sus miembros uno que pudiera resolver el problema de parada para las máquinas de Turing[20]?

Esta estrategia, que recuerda en algo a la que emplea Gödel cuando analiza lo que él considera el desafío mecanicista[21], permite abordar de forma novedosa ciertos problemas clásicos en Computación teórica. Por ejemplo,

i. ¿Basta el argumento diagonal clásico para rechazar la existencia de una *solución efectiva* del problema de parada aunque basada en procedimientos distintos a las TM's?
ii. ¿Puede concebirse sin contradicción la existencia de un algoritmo particular para resolver el problema de parada de cada función Turing-computable sin que exista un algoritmo general para todas ellas?
iii. ¿Pueden resolver las ATM's el problema de parada para las TM's?

La respuesta a la primera de estas preguntas parece claramente negativa. Si se analiza la estructura lógica de la demostración diagonal por la que se alcanza la solución negativa al problema de parada, se observa que esta conclusión sólo se alcanza haciendo intervenir explícitamente a la Tesis de Church. Es decir, se da por supuesto que si existe un algoritmo tal, entonces éste debe corresponder a alguna función Turing-computable, sea ésta la i-ésima… Sin la intervención de esta hipótesis lo único que se demuestra es que la función del problema de parada no es Turing-computable. Todas estas consideraciones son, en cualquier caso, bastante obvias y no muestran la potencia del método de análisis inverso que vengo comentando. Si se insiste un poco más en este tipo de metodología las conclusiones dejan ya de ser tan obvias. En el caso anterior, por ejemplo, lleva al estudio de las rela-

19. Lo que se establece para ser exactos es que no hay un procedimiento efectivo expresable un términos de una función computable capaz de determinar en tiempo finito y para todo entero positivo si ese número pertenece o no al conjunto K.

20. Esto no supone que esa nueva clase no pudiera enfrentarse a un problema de parada de contenido similar al que afecta a la clase de Funciones Turing computables. De hecho, es lo más probable.

21. Esto se puede ver en sus conversaciones con H. Wang al respecto.

ciones que han de existir entre una clase dada de procedimientos y aquella otra a la que podría pertenecer el método capaz de resolver el problema de parada de la primera de estas clases. La conclusión, que ofrezco aquí sin entrar en detalles, podría exponerse como sigue:

> El método empleado por un determinado procedimiento H para identificar y hacer referencia a los elementos de otra clase P no puede ser definible en P.

Ese determinado procedimiento H podría abordar el problema de parada en P sin contradicción, aunque el precio a pagar podría ser alto. La clave en este caso, y otros similares, estriba en los métodos empleados para interpretar la enumerabilidad efectiva de una clase de procedimientos. En el caso de las TM's esa enumeración es definible en términos de una máquina de Turing. Conclusiones como las anteriores pueden ser vistas como consecuencias más o menos directas de un conocido teorema en Computación teórica: el Teorema de Forma Normal de Kleene.

Del segundo de los problemas mencionados más arriba no diré nada por ahora. Las ambigüedades que lo rodean impiden tratarlo de forma concisa y comprensible como corresponde a un trabajo de este tipo. Queda ahí por tanto. Diré, no obstante, que algo de ello se puede encontrar en el procedimiento que Penrose emplea en el argumento mentalista expuesto en su famoso *Shadows of the Mind*. Este argumento pretende mostrar la definitiva superioridad de la mente sobre cualquier ingenio mecánico que, pretendidamente pueda aspirar a igualar su potencia. Y lo hace explotando teoremas como el que conduce a la insolubilidad del problema de parada o los conocidos teoremas de indecidibilidad de Gödel. Buena parte de su éxito reposa, al menos esa es mi opinión, en la existencia de una respuesta afirmativa a la pregunta formulada en ii, cosa que Penrose no hace y ni siquiera plantea.

El tercero de los problemas resulta interesante por diversas razones. Los oráculos introducidos por Turing en su obra de 1939 son, desde cualquier punto de vista posible, auténticas cajas negras. No sabemos nada acerca del modo en que operan ni de su estructura interna. Las ATM's pueden ser consideradas oráculos, pero por razones muy distintas. Resuelven cuestiones numéricas que quedan fuera del alcance de las TM's, pero en este caso disponemos de mucha información acerca del modo en que trabajan. De hecho pueden ser consideradas TM's que actúan de un modo muy especial. Estas consideraciones permiten quizá hablar de un nuevo reino de objetos formado por oráculos que no se comportan estrictamente como cajas negras ya que sabemos algo de cómo se comportan y podemos emplear ese conocimiento para determinar su conducta y rasgos formales. Un buen término para referirnos a ellos podría ser el de *oráculos translúcidos* los cuales se comportan no como cajas negras sino como *cajas grises*.

Un problema aún latente –y pienso que por mucho tiempo– en el dominio de las ATM's y otras máquinas que de algún modo pretenden trabajar con infinitos actuales es definir adecuadamente las condiciones ideales bajo las que operan estos mecanismos.

Consideremos el caso de las ATM's. Como ya he dicho, una ATM es, ante todo, una TM. Supongamos que una de estas máquinas no alcanza un estado de parada estándar, es decir, no se detiene arrojando un input, sino que continúa acelerando de algún modo. Parece evidente que el estado final de una de estas máquinas es el *límite* de algún proceso infinito que no nos resulta accesible. Resulta imprescindible, por tanto, introducir algún tipo de interfaz que recoja el resultado obtenido por la máquina que opera de forma acelerada, si existe, y refleje que no existe en otro caso. El modo más simple de apreciar este comportamiento es suponer que una ATM consta de un mecanismo que ejecuta en un *régimen acelerado* un programa previamente dado para escribir el resultado de esa rutina, si existe, deteniéndose en una celda en blanco de otro modo. Podemos suponer, sin pérdida de generalidad, que este proceso siempre termina en *k* segundos. De este modo una ATM alcanza el mismo resultado que sus correlatos en el dominio de las TM's cuando este existe alcanzando de otro modo un resultado previamente convenido a los k segundos[22]. De hecho, se puede decir que una ATM computa la versión total –totalmente definida– de las funciones que computan las máquinas de estándar Turing. La clase de las funciones totalmente definidas está sometida al alcance de la denominada técnica de diagonalización. De ella ya he hablado en este ensayo y no volveré a hacerlo ahora. Su descripción detallada queda, además, fuera del alcance de este trabajo. Basta decir que mediante este procedimiento se puede establecer la imposibilidad de enumerar –a veces de forma efectiva, otras mediante cualquier procedimiento– ciertas clases de objetos. Enumerar una clase de objetos sólo significa que es posible en principio asignar un nombre –quizá más de uno– a cada uno de esos objetos. Es otra forma de decir que la colección en cuestión está formada por objetos bien definidos los cuales pueden distinguirse y separarse de los demás mediante un nombre. Resulta pues que la clase de las funciones que pueden ser calculadas mediante máquinas de Turing y tales además que están definidas para todos sus posibles argumentos –arrojan siempre un valor– no es efectivamente enumerable. Esto no quiere decir que la clase de funciones cuyos valores se pueden calcular mediante el uso de máquinas de Turing[23] no sea enumerable a su vez. De hecho, sabemos que es enumerable de forma efectiva. Lo que sucede es que no hay modo de separar las funciones computables que están definidas para todos sus valores de aquellas que pueden quedar indefinidas para algunos de ellos.

22. Es decir, la máquina acelerante posee algún dispositivo extra que captura el evento correspondiente a la ejecución de la rutina de la máquina estándar sobre la que opera. Si ese evento consiste en un determinado resultado, la máquina acelerante lo ofrece a su vez como output. Si el evento consiste en que la máquina de Turing no finaliza arrojando un output, entonces la máquina acelerante informa arrojando un output previamente asociado a esa circunstancia.

23. Las máquinas de Turing son efectivamente enumerables gracias a la existencia de un mecanismo efectivo para asignar códigos a sus programas. De ahí que las funciones computables por medio de máquinas de Turing resulten también efectivamente enumerables ya que basta identificar cada función con una de las muchas máquinas que computa sus valores.

Y, por tanto, no es posible saber cuándo un determinado input no arrojará un output. Si juntamos todos estos datos, nos damos cuenta que las ATM son máquinas de Turing que funcionan de una forma peculiar y en consecuencia, y puesto que las máquinas de Turing forman una clase enumerable[24], las ATM's también lo son a su vez. Cada ATM calcula una función numérica y sólo una y por tanto no debería haber más funciones ATM calculables que máquinas acelerantes. Cada ATM tiene, como lo tiene la máquina de Turing asociada, un código o número de serie que la identifica y que puede ser transmitido a la función que cada una de ellas computa. Pero de ser así, la clase de funciones ATM-computables es enumerable y además lo es de forma efectiva. Líneas más atrás dijimos, sin embargo, lo contrario. Esta contradicción ha llevado a algunos autores más informados de las técnicas formales de análisis a concluir que el concepto de supertarea que las ATM's pretenden ilustrar es autocontradictorio y en consecuencia inviable. A mi juicio esto sólo prueba las dificultades conceptuales que rodean conceptos que juegan con la idea de infinito de forma tan explícita. Por otra parte, no es claro que la contradicción aludida exista realmente. La técnica que sirve para establecer que la clase de las funciones Turing-computables definidas para todos sus posibles inputs no puede ser reconocida por medios efectivos quizá no pueda aplicarse del mismo modo sobre funciones calculables por medio de ATM's. La razón no es fácil de explicar y se trata, además, de una conjetura que se apoya en un cierto modo de interpretar el modus operandi de una de estas máquinas. El problema básico sí se puede exponer de todos modos. Las máquinas de Turing no están sometidas a ningún comportamiento concreto definido en el tiempo. De este modo es posible acoplar unas máquinas a otras permitiendo que actúen de forma combinada. Las ATM's sí tienen, por el contrario, un comportamiento temporal definido: todas ellas finalizan en k segundos. Por tanto, no es posible combinar ATM's de modo que una adopte como input el output de otra de ellas. Una máquina que procediera de ese modo rompería la convención de terminar su rutina en k segundos, ya que ni siquiera podrían haberla iniciado. Siempre se puede simular esta composición mediante máquinas de Turing, pero –y esta es la conjetura no exenta de dificultades– hay aspectos de lo que se ha dicho más atrás sobre la posible enumeración de las funciones ATM-computables que parece depender esencialmente de la posibilidad de combinar objetos –máquinas– del *mismo* tipo y no de una mera simulación.

Me gustaría terminar este apartado y con ello este trabajo dejando claro que no creo que este argumento y otros que se puedan aportar se encuentren libres de toda dificultad. Hay que tener en cuenta que nos enfrentamos a situaciones extremadamente abstractas definidas en la mayoría de los casos sin la debida precisión

24. Cada máquina de Turing puede ser concebida como un pequeño programa junto con una porción de hardware que ahora no nos interesa tomar en consideración. Cada programa, al ser una lista finita de instrucciones –no hay programas infinitos ni recetas que no consten de un número finito de instrucciones– puede diferenciarse de cualquier otro por mera inspección. En otras palabras, se le puede asignar un número de serie o código que lo identifique perfectamente.

y rigor. Cambios que en apariencia sólo afectarían al comportamiento superficial o accidental de una máquina de Turing puede suponer, como se acaba de ver, cambios más que sustanciales con respecto a las propiedades formales subyacentes.

La conclusión que se puede obtener de esta suerte de análisis inverso es que el problema de parada y otras limitaciones aparentes no debe ser visto como especie de maldición bíblica lanzada contra el alcance y potencia de algunas de nuestras habilidades cognitivas más elementales. Este problema, como otros, sólo parece el precio a pagar por disponer de la capacidad para nombrar de manera efectiva los objetos de una cierta clase. La enumerabilidad efectiva de los miembros de un conjunto es, en cierto modo, una condición necesaria para hablar con propiedad de la representación completa de una clase de objetos por parte de nuestra intuición matemática. Resolver el tipo de limitación que entraña este problema por los medios que sean puede ser bueno, pero también podría comprometer seriamente la seguridad que suministra saber que nuestra descripción no deja fuera nada que deba estar dentro. En definitiva, no parece posible alcanzar el paraíso matemático por lo que hace a nuestra capacidad para describir tareas de forma efectiva. Pero, ¿quién quiere un paraíso inmóvil y totalmente cierto?

Bibliografía

Alonso, E., *Lógica y Computabilidad*, Summa Logicae en el siglo XXI. http://logicae.usal.es, 2004.

Alonso, E. y Manzano, M., «Diagonalisation and Church's Thesis: Kleene's Homework», *History and Philosophy of Logic* 26(2), 2005, pp. 91-113.

Boolos, G. S. y Jeffrey, R. C., *Computability and Logic*, New York, Cambridge University Press, 1989.

Bringsjord, S. y Arkoudas, K., «The modal argument for hypercomputing minds», *Theoretical Computer Science* 317, 2004, pp. 167-190.

Bringsjord, S.; Bello, P. y Ferruci, D., «Creativity, the Turing Test, and the (better) Lovelace Test», 2005. Preprint disponible en http://www.rpi.edu/~faheyj2/SB/SELPAP/DARTMOUTH/lt3.pdf.

Church, A., «An Unsolvable Problem of Elementary Number Theory», *The American Journal of Mathematics*, vol. 58, 1936, pp. 345-363.

Cleland, C. E., «The concept of computability», *Theoretical Computer Science* 317, 2004, pp. 209-225.

Davis, E. B., «Building Infinite Machines», *British Journal of Philosophy of Science* 52, 2001, pp. 671-682.

Davis, M. (ed.), *The Undecidable*, Raven Press, New York, 1965.

Copeland, B. J., «Hypercomputation: philosophical issues», *Theoretical Computer Science* 317, 2004, pp. 251-267.

Feferman, S., «Transfinite recursive progressions of axiomatic theories», *Journal of Symbolic Logic*, vol. 27, 1962, pp. 259-316.

Gold, E. M., «Limiting Recursion», *Journal of Symbolic Logic*, vol. 30, nº 1, 1965, pp. 28-48.

HAMKIN, J. D. y LEWIS, A., «Infinite Time Turing Machines», *Journal of Symbolic Logic*, vol. 65, 2000, pp. 567-604.

KIEU, T. D., «Computing the non-computable», *Contemporary Physics*, vol. 44, nº 1, 2003, pp. 51-71.

MACLENNAN, B. J., «Natural computation and non-Turing models of computation», *Theoretical Computer Science* 317, 2004, pp. 115-145.

MANZANO, M., «Alonzo Church: His Life, His Work and Some of His Miracles», *History and Philosophy of Logic* 18(4), 1997, pp. 211-232.

MANZANO, M. y HUERTAS, A., *Lógica para Principiantes*, Madrid, Alianza Editorial, 2004.

ORD, T. y KIEU, T. D., «The Diagonal Method and Hypercomputation», *British Journal of Philosophy of Science*, vol. 56, 2005, pp. 147-156.

ROGERS, H., *Theory of Recursive Functions and Effective Computability*, New York, McGraw-Hill, 1967.

SHAGRIR, O., «Super-tasks, accelerating Turing machines and uncompuability», *Theoretical Computer Science* 317, 2004, pp. 105-114.

TURING, A., «On Computable Numbers, with an Application to the Entscheidungsproblem», *Proceedings of the Mathematical Society*, vol. 42, 1936, pp. 230-265.

WANG, H., *A Logical Journey. From Gödel to Philosophy*, Cambridge, Massachusets, The Mit Press, 1996.

