Revista de Filosofía ISSN: 0034-8244
2001, 26: 139-164

Mentalismo, mecanicismo: el nuevo argumento de Penrose

Enrique ALONSO

(Universidad Autónoma de Madrid)

Resumen

Este ensayo ofrece un análisis crítico del último argumento que el matemático y filósofo Roger Penrose ofrece a favor de la tesis según la cual hay habilidades de la mente humana que nunca podrán ser igualadas por ingenio mecánico alguno. Al mismo tiempo se ofrece una descripción general de los últimos episodios del eterno enfrentamiento entre mentalismo y mecanicismo y se concluye con una sugerencia acerca de los puntos en los que cabe esperar nuevas situaciones de tensión entre estos dos grandes paradigmas. Este trabajo pretende, además, mostrar la vigencia de un tipo de filosofía elaborada a partir de las herramientas de la ciencia moderna cuyos problemas son, no obstante, los del pensamiento filosófico tradicional.

Palabras clave: Filosofía de la mente; Mecanicismo; Mentalismo; Computabilidad; Lógica.

Abstract

In this paper we analize the last argument given by Penrose to defend metalistic thesis against the positions represented by mechanism. We also give a general description of some of the most relevant events which conform the present state in the discussion between mentalism and mechanism. This work also tries to show how to make good philosophy making use of scientifc tools to answer the questions possed by traditional thougt in humanities.

Keywords: Philosophy of Mind; Mechanism; Mentalism; Computability; Logic.

Un viejo problema

A. Mentalismo y mecanicismo no son etiquetas que correspondan exactamente a sendas doctrinas filosóficas presumiblemente enfrentadas. Se trata, más bien, de dos grandes *actitudes filosóficas* en las que se *cae* sin esfuerzo aparente, de algo que quizá no se es nunca hasta que alguien hace la pregunta oportuna. Es entonces cuando adoptamos las tesis de uno u otro bando reproduciendo un debate que de formas muy distintas siempre ha estado ahí formando parte de la gran trama de la historia de las ideas.

Pero, ¿qué cabe entender por una *actitud mentalista* y qué por una *actitud mecanicista*? Ciertamente que no se es mentalista o mecanicista de una sola forma. Podemos encontrarnos, de hecho, con formas de mentalismo enfrentadas en todo lo demás, y mecanicistas que nunca aceptarían fácilmente ser representantes de una misma posición. Así las cosas, cualquier intento de definir estas actitudes debería ser entendido como la respuesta provisional, afirmativa o negativa, que cada cual puede dar a la siguiente pregunta, insistiendo en que cualquier matiz debe dejarse para más adelante:

> [1] ¿Hay actos objetivos de la mente humana que no pueden ser reproducidos por ingenio mecánico alguno, del tipo que fuere?

Se adopta una posición mecanicista si la repuesta es negativa, siendo mentalista cuando se responde afirmativamente.

Soy consciente de que al proceder de este modo ignoro deliberadamente distingos a los que se han dedicado muchas páginas y considerable esfuerzo. No pretendo ofender con ello, sino invitar al lector a que tome las cosas desde el principio. Por lo que hace al contenido de este ensayo es mejor así.

B. Como ya he dicho, el enfrentamiento entre estas dos actitudes fundamentales puede encontrarse a lo largo de la práctica totalidad de la historia intelectual que es propia de nuestra cultura. Es fácil entender que una pugna de tan larga tradición no siempre se puede mantener con la misma intensidad. Esto explica la existencia de una sucesión de etapas de tensión y cambio junto a otras de profundo letargo. Durante cada una de estas crisis periódicas una de las dos posiciones consigue alcanzar una situación de clara ventaja.

Todo parece a punto de terminar y durante algún tiempo resulta difícil encontrar argumentos que oponer al avance incontenible del adversario. La experiencia demuestra, finalmente, que antes o después siempre surge un dato, una teoría, o un contraargumento de eficacia similar al responsable de la última crisis. El *equilibrio* termina por restablecerse bajo la forma de un silenciamiento de las posiciones enfrentadas, quedando, por lo general, nuevos conceptos e ideas útiles por sí mismos y gracias a los cuales podemos ver en esta historia de tensiones algo más que un estéril enfrentamiento de escuelas.

Con este ensayo no pretendo dar cuenta más que del último de los episodios de esta historia. En esta ocasión la ofensiva procede del frente del mentalismo, pero para ello quizá sea preciso decir algo de un pasado reciente en el que el mecanicismo experimentó avances ante los que parecía inútil cualquier resistencia.

El pasado reciente

Una de las razones que suelen influir en la reactivación del debate mentalismo-mecanicismo es la aparición de nuevos conceptos, de nuevos lenguajes con los que expresar cada una de estas posiciones. Es un hecho obvio que el mecanicismo del siglo xvii y xviii no emplea el mismo lenguaje, ni se sirve de las mismas teorías que el mecanicismo del siglo xx. Las causas que más directamente influyen en la reactivación de las tensiones a lo largo del siglo pasado han de buscarse en dos tipos de resultados fuertemente ligados a la configuración de la Lógica contemporánea. Me refiero a

[2] i. La formulación de los principios básicos de la Teoría de la computación,

y

ii. La aparición de ciertos resultados metamatemáticos caracterizados genéricamente como *teoremas de limitación*.

La formulación de los principios básicos de la Teoría de la Computación produce un fuerte desequilibrio a favor de la hipótesis mecanicista. Alan Turing es, a parte de uno de los responsables materiales más directos de la fundación de dicha disciplina, el líder intelectual de esta nueva ofensiva mecanicista. Los términos de su desafío aparecen formulados en el artículo "Computing Machinery and Intelligence", *Mind*, vol. 59 (1950) y se concretan en un argumento ciertamente robusto: el *Test de Turing*.

[3] *Test de Turing*. Si bajo las condiciones más generales que quepa imaginar no es posible establecer una diferencia apreciable entre la conducta de un ser humano y la de ordenador convenientemente programado, entonces nos veremos racionalmente obligados a reconocer que se trata de entidades del mismo tipo en todo sentido relevante.

La contrarréplica a este argumento llegaría tiempo más tarde de la mano de Searle y su experimento mental de la *Habitación China* –"Minds, Brains, and Programs" (1980)–. La tesis que anima este argumento podría formularse del siguiente modo:

[4] *Tesis de la Habitación China*. Reproducir una rutina y encarnar un a algoritmo constituyen situaciones por completo distintas. La primera no entraña comprensión, mientras que la segunda sí.

La razón de tan peculiar nombre es el modelo que Searle considera en su experimento mental. Dentro de una habitación cerrada, salvo por una rendija, hay un operario humano que dispone de diccionarios exhaustivos de la lengua china, idioma que, por otra parte, ignora. Por la rendija se introducen textos en chino que al tanto son devueltos en castellano por el ocupante de la habitación siguiendo un procedimiento de traducción bien establecido que previamente se le ha suministrado. Según Turing, sostiene Searle, el operario que está en la habitación china *sabe* chino. Conclusión tan absurda, añade, que basta como contraejemplo de dicha tesis. Sólo este argumento ha sido capaz de restaurar el equilibrio entre mentalismo y mecanicismo tras la dura ofensiva lanzada por Turing y la moderna filosofía mecánica.

La ofensiva mentalista, independiente de los intentos por derribar los argumentos dados por el contrario, parte, a su vez, del análisis de ciertos resultados que conocemos como teoremas de limitación. De entre estos hay dos que ocupan una posición sin duda excepcional:

[5] i. Los Teoremas de Incompletitud de Gödel y
ii. El Problema de Parada.

Los teoremas de Incompletitud de Gödel son un tópico dentro de un terreno que algunos empiezan a interpretar como una genuina *filosofía natural* contemporánea. Pese a ello, o quizá por eso mismo, su enunciado ha sufrido con frecuencia de interpretaciones y formulaciones poco o nada rigurosas. Ciertamente que no es fácil ofrecer versiones informales aceptables de estos teoremas, pero quizá sea esa la tarea que nos corresponde afrontar ahora. En

lo que sigue, PA simboliza una teoría axiomática cuyos teoremas son verdades de la aritmética elemental –aunque no contenga necesariamente todas las verdades de la aritmética–. Se trata, por tanto, de una teoría que intenta reducir a unos pocos principios y reglas las verdades relativas a las operaciones elementales que realizamos con números naturales. La denominación PA, por *Peano Arithmetic*, se debe a que fue el matemático italiano Giuseppe Peano el primero en ofrecer los axiomas y reglas que acabo de mencionar. El lenguaje en que se formulan esos axiomas y reglas es un lenguaje formal propio de la Lógica del que no necesitamos ahora saber más.

[6] *Teoremas de Incompletitud de Gödel*.

i. *Primer Teorema de Incompletitud*. Si PA es consistente, entonces existe un enunciado G tal que ni él ni su negación son demostrables en PA.
ii. *Segundo Teorema de Incompletitud*. Si PA es consistente, entonces el enunciado que representa en PA la consistencia de PA no es demostrable en PA.

En breve volveremos a hablar de ellos. El segundo de los teoremas mencionados en [5], el Problema de Parada, pese a ser fácilmente identificable por muchos como uno de esos resultados de impacto de los generados recientemente por las ciencias formales, es menos conocido en sus términos precisos. Su enunciado hace referencia a un concepto que va a aparecer constantemente a lo largo de este ensayo, el de función computable. Una función computable es cualquier función cuyos valores se pueden calcular mediante el uso de algún algoritmo. La suma, el producto, decidir si un número dado es primo son ejemplos de funciones computables. Aunque puede parecer que al hablar así nos comprometemos con un problema con interés exclusivo para el matemático, nada hay más lejos de la realidad. El uso de funciones debe ser entendido aquí como un pretexto útil para hablar de lo que nos interesa, la capacidad del ser humano para realizar tareas de manera efectiva, mecánica, y el alcance que esa facultad posee. El manejo de funciones numéricas sólo contribuye a ofrecer un marco experimental idóneo. Así, por ejemplo, hace posible hablar de la x-ésima función computable –f_x, en símbolos- según una cierta enumeración –listado– de todas las funciones que poseen la notable característica de ser computables. El Problema de Parada dice, entonces, lo siguiente:

[7] *Problema de Parada*. No hay una función computable H(x,y) que permita determinar si la x-ésima función computable f_x finaliza arrojando un resultado cuando computa el input y.

Si reemplazamos el término "función computable" por lo que es su correlato intuitivo, la noción de tarea efectiva o algoritmo, el Problema de Parada adquiere un aspecto algo más amenazador. Lo que este viene a decir es, ni más ni menos, que no hay un procedimiento efectivo para determinar si una tarea, igualmente efectiva, finaliza –de ahí el nombre de este *problema*– con éxito una vez iniciada. Se mire como se mire, parece un duro golpe a nuestras intuiciones más elementales acerca de aquello que creemos hacer cuando procedemos a resolver un problema de manera efectiva. Es difícil encajar sin más nuestra incapacidad para establecer algo tan simple como si una tarea termina o no cuando ésta es, o decimos de ella que es *efectiva*. ¿Qué tipo de efectividad es esa que no nos brinda condiciones de control en principio tan obvias? Para aceptar tan severa consecuencia del *Problema de Parada* es preciso, no obstante, conceder antes que el término "función computable" o si se quiere "tarea computable por un programa" equivale al de "tarea efectiva". En lo que sigue entenderemos que el uso del adjetivo "computable" supone la existencia de un lenguaje formal –no muy distinto de un lenguaje de programación– en el cual se define la tarea en cuestión, mientras que el uso del término "efectiva", predicada de una tarea, simplemente dice de ella que sus pasos pueden ser ejecutados de manera mecánica por cualquier ser humano. Oponemos, pues, un concepto formal y cerrado, el de "tarea computable", a uno intuitivo y abierto, "tarea efectiva".

Consideremos ahora conjuntamente los dos teoremas de limitación –ambos hablan acerca de cosas que no podemos hacer– citados en [5] y que hemos desarrollado en [6] y [7]. ¿Qué hay en este tipo de resultados que los hace tan interesantes a ojos del mentalismo? Un resultado de limitación parece ser ante todo la manifestación de la imposibilidad lógica de resolver un problema legítimo con los medios que le corresponden (es el ser humano quien se plantea el problema y es el ser humano quien reconoce la imposibilidad de resolverlo), pero el mentalismo ve en ellos la evidencia de que el ser humano es capaz de identificar las limitaciones de ciertos formalismos consiguiendo en el proceso soluciones a los problemas propuestos que van más allá de esas limitaciones (al establecer una limitación acerca de las posibilidades de un formalismo automáticamente nos situamos fuera, por encima de ella).

El caso paradigmático de uso de uno de estos resultados en beneficio de la hipótesis mentalista es lo que se ha llegado a conocer como *Argumento de Lucas* –por J.R. Lucas– expuesto por vez primera en "Minds, Machines and Gödel", *Philosophy*, XXXVI, 1961, pp.112-127, y con más detalle en *The Freedom of the Wil,* Oxford Univ. Press, 1970. Su argumento, que toma como

punto de partida el *Primer Teorema de Incompletitud de Gödel*, podría resumirse del siguiente modo:

> [8] *Argumento de Lucas*: Cualquier formalismo S que contenga PA es tal que al razonar sobre él, podemos establecer la existencia de una fórmula verdadera con respecto a la interpretación estándar de S pero indemostrable en S. Por tanto, esa fórmula será aceptable desde nuestro punto de vista, por ser verdadera, pero inaceptable para S –por ser indemostrable–, con lo que ningún cálculo será capaz de encapsular las habilidades formales del ser humano.

La réplica del mecanicismo a este argumento se basa en una observación bien simple. Lo que el *Primer Teorema de Incompletitud* muestra es *si* PA es consistente, *entonces* no es demostrable en PA un cierto enunciado G (ni su negación). Es más, una versión formal de este mismo hecho puede ser demostrada como un genuino teorema de PA dando lugar, de paso, al Segundo Teorema de Incompletiud. El problema está en que ni PA, ni ningún ser humano, disponen de una demostración aceptable –y elemental– de que PA es, de hecho, consistente.

Esta replica restituye el equilibrio dando lugar a un nuevo movimiento en el frente mentalista:

> [9] *Argumento de Lucas extendido*: La consistencia de un sistema formal tan elemental como PA puede tomarse como una verdad *incuestionable*.

Esta respuesta y el intento de convertir las condiciones antecedentes de los resultados de limitación –la consistencia de PA en este caso– en *verdades incuestionables* es una de las características más evidentes del pensamiento de Penrose y dicho sea de paso, un componente recurrente en más de un autor reputado por aportaciones independientes a la que ahora nos ocupa–cfr.: R.Thom "The Hylemorphic Schemata in Mathematics", *Philosophy of Mathematics today*. Kluwer,1997–.

El debate entorno al uso de los teoremas de limitación de Gödel como defensa de las posiciones mentalistas parece haber alcanzado, no obstante, un cierto punto de equilibrio y la opinión generalizada es que poco más queda hacer salvo pronunciarse a favor o en contra de cada posición: no existen razones últimas que permitan inclinarse justificadamente por una u otra opción.

La ofensiva que presenta mayor novedad es la que toma como punto de partida el segundo de los resultados de limitación mencionados. Vista la estrategia seguida por Lucas al analizar los teoremas de Gödel no es muy

difícil anticipar cuál va a ser el núcleo del argumento ahora. El ser humano puede establecer mediante algún razonamiento básicamente correcto y dadas ciertas condiciones antecedentes un valor para H(x,y), la función mencionada en el Problema de Parada, que ningún procedimiento computable puede establecer consistentemente. Este germen de argumento es lo que va a ser desarrollado por Penrose en *Shadows of the Mind*, Oxford Univ. Press, 1994 hasta llegar a obtener un razonamiento que ya ha sido bautizado por algunos como el *Nuevo Argumento de Penrose* (NAP) –P. Lindström, "Penrose's New Argument", *Journal of Philosophical Logic* 30, pp. 241-250, 2001–.

El siguiente apartado de este ensayo constituye un análisis en profundidad de los derechos del argumento de Penrose a favor de las tesis mentalistas.

El nuevo argumento de Penrose.

El razonamiento que Penrose desarrolla en *Shadows of the Mind* arranca de lo que el propio autor denomina el *argumento de Turing-Gödel*. Se trata, en realidad, de un peculiar género de demostración más conocido bajo el nombre de *diagonalización*, o también *Diagonal de Cantor*, en honor al primer matemático que hizo uso de ella de manera explícita. Esta técnica, paradójica en apariencia, y siempre sorprendente, ha tenido muchas consecuencias dentro del dominio de la matemática moderna. Una de ellas, y no la menor, es el enunciado del Problema de Parada. Alan Turing se sirvió de la técnica de diagonalización para establecer en un artículo absolutamente fundamental para la moderna Teoría de la Computación –"On computable Numbers with an Aplication to the Entscheidungsproblem", *Proceedings of the London Mathematical Society*, vol. 42, 1936-37, pp. 230-265– una respuesta negativa al *Problema de Parada* de forma sorprendentemente directa y clara. Es más dudoso que Gödel, como parece conceder Penrose, tenga parte en este asunto, ya que la intervención de la técnica de diagonalización en sus trabajos fundamentales es mucho menos directa que en caso de Turing, pero siempre podemos considerar su mención aquí en términos de un sentido homenaje.

En la medida en que el propio argumento de Penrose depende mucho de la estructura interna de la demostración diagonal que lleva al Problema de Parada habré de emplear algún tiempo en exponer en qué consiste. Para ello se precisan unas pocas definiciones:

[10] Definiciones:

i. Sea A la clase formada por todas las funciones numéricas computables.

ii. Sea *e* una enumeración efectiva de A

iii. H(x,y) es la siguiente función numérica:
H(x,y) = 1 si fx(y) está definida
0 en otro caso

iv. g(x) = H(x,x) (función diagonal)

v. g*(x) =1, si g(x) = 0
Indefinida, g(x)=1.

Estos son los *dramatis personae* que bastan y sobran para entender todo lo que sigue. Y lo que sigue es, entiéndase bien, uno de los resultados de mayor impacto epistemológico habido dentro del pensamiento filosófico contemporáneo. La clase A agrupa todas las funciones numéricas para las que es posible encontrar un programa que calcule sus valores –aunque ello no significa que siempre lo consigan–. Por un resultado independiente que no vale la pena analizar aquí, sabemos que dicha clase es *efectivamente enumerable*. En otras palabras, podemos colocar cada una de estas funciones en una lista exhaustiva –aunque admitiendo repeticiones– e identificar así cada función computable en A por su número de orden en la misma. Esta posibilidad es la que permite definir la función H(x,y) de manera coherente y plantearnos si es o no una función computable, es decir, una función en A. Si no fuera posible identificar de manera efectiva cuál es la i-ésima función computable en A, no tendría mucho sentido preguntarse si H(x,y) lo es a su vez: sería obvio que no. Como ya hemos dicho, H(x,y) es la función asociada al *Problema de Parada* y parte imprescindible para definir las funciones asociadas g(x) y g*(x) responsables inmediatas del argumento que sigue. Para establecer que H(x,y) no pertenece a A voy a servirme de un razonamiento por reducción al absurdo que expongo de manera semiformal –adoptando la mecánica propia del Cálculo de Deducción Natural de la Lógica clásica– para comentarlo acto seguido con mayor detalle:

[11] *Problema de Parada*: H(x,y) no es una función computable.

1. $H(x,y) \in A$
2. Si $H(x,y) \in A$, entonces $g(x) \in A$ [por la definición de g(x)]

3. Si $g(x)\in A$, entonces $g^*(x)\in A$	[por la definición de $g^*(x)$]
4. $g^*(x)\in A$	[MP 1,2 y 2,3]
5. $g^*(x)=f_i(x)$	[por 4 y la enumeración *e*]
6. $g^*(i)=1$	
7. $f_i(i)=1$	[reemplazando en 6 g^* por f_i]
8. $f_i(i)$ está definida	[por 7]
9. $g(i)=1$	[por la definición de g]
10. $g^*(i)$ no está definida	[por la definición de g^*]
11. $\neg(g^*(i)=1)$	[reductio 6-10]
12. $g^*(i)$ no está definida	[por 11 y la def. de g^*]
13. $f_i(i)$ no está definida	[reemplazando en 12 g^* por f_i]
14. $g(i)=0$	[por 13 y la def. de g]
15. $g^*(i)=1$	[por la def. de g^* y 14]
16. $\neg(H(x,y)\in A)$	[reductio 1-15]

Como es bien sabido, toda demostración por reducción la absurdo empieza por suponer lo contrario –la negación– de aquello que se desea establecer. En la línea 1 se supone, por tanto, que la función H(x,y) sí es una función computable, es decir, pertenece a A. De ahí se obtiene inmediatamente que g(x) es computable, lo que en definitiva lleva a sostener que $g^*(x)$ también lo es, aunque quizá este punto requiera alguna aclaración. Puesto que estamos interpretando que la computabilidad de una rutina o tarea equivale a la existencia de un programa que la traduce, que $g^*(x)$ sea computable debe interpretarse como la existencia de un cierto programa en el que g(x) contará de algún modo especial. Para obtenerlo basta con tomar el programa que traduce g(x) y añadir un pequeño fragmento que borra el resultado de g(x) si este es *0* y lo reemplaza por *1* y entra en un bucle si el resultado de g(x) es *1*. Al entrar $g^*(x)$ en un bucle hacemos que su valor quede indeterminado. Generar ese bucle es, por otra parte, algo totalmente trivial: basta ordenar al programa que vuelva al principio o cualquier cosa por el estilo.

Puesto que las funciones en A tienen todas ellas un número de orden que las identifica –por la existencia de una enumeración efectiva– podemos suponer que $g^*(x)$ es la i-ésima función en esa lista, es decir, aceptamos –línea 5– que $g^*(x)=f_i(x)$. *i* es un número natural más y, por tanto, es del todo legítimo preguntarse qué valor adoptará $g^*(i)$. No se trata de obrar malintencionadamente –como en ocasiones se sugiere para poner de manifiesto el carácter truculento de este tipo de demostraciones–, sino de analizar algo por completo inevitable: el comportamiento que $g^*(x)$ adopta cuando x=i. Dada la definición de $g^*(x)$, sólo caben dos opciones: que $g^*(i)=1$ o que $g^*(i)$ quede indefinida –es decir, entre en un bucle–. Veamos qué pasa si suponemos que

g*(i)=1 –línea 6–. Decir que g*(i)=1 es lo mismo, por la línea 5, que decir que $f_i(i)=1$. Así pues, resulta que la i-ésima función computable está definida –adopta un valor, en este caso *1*- cuando x=i. Eso hace que g(i)=1 –línea 9– ya que la función g(x) adopta precisamente el valor *1* en aquellos casos en los que la x-ésima función en la enumeración está definida cuando procede a calcular el valor x, es decir, el mismo que corresponde a su posición en la lista[1]. Pero si g(i)=1, ello basta para que g*(i) entre en un bucle y quede indefinida –compruébese la definición de g*(x) ofrecida en [10.v]–. Concluimos, pues, que g*(i) no puede valer *1* –línea 11–. Pero si es así, g*(i) no puede sino quedar indefinida –línea 12–. O lo que es lo mismo, $f_i(i)$ queda indefinida –línea 13–. Eso basta, a su vez, para que g(i)=0, lo que por la definición de g*(x) se resuelve en que g*(i)=1 –líneas 14 y 16– alcanzándose una contradicción que sólo depende del supuesto inicial, esto es, del hecho de afirmar que H(x,y) pertenecía a A.

Este es, grosso modo, el razonamiento que subyace a la respuesta negativa al *Problema de Parada*. Tal y como lo he expuesto sólo se deduce de él que la función encargada de analizar la conducta del resto de las funciones computables –averiguar en qué casos están definidas y en qué casos no– no es ella misma computable. H(x,y) no puede ser en ningún caso expresable por los mismos medios que sirven para dar cuenta de las funciones computables en A. Si tenemos en cuenta que hoy por hoy no conocemos funciones, o tareas, efectivamente calculables que no puedan ser expresadas en términos de un programa, el resultado anterior adquiere mayor radicalidad: afirma que no hay procedimiento efectivo alguno capaz de determinar para cualquier otro procedimiento efectivo si este concluye su objetivo con éxito o no[2].

Sea como fuere, es este argumento, apasionante como pocos, el que sirve como punto de partida a Penrose para concluir algo tan poco inocente como la superioridad de la mente humana frente a cualquier ingenio mecánico que este pueda llegar a diseñar.

El análisis general de la posición táctica del mentalismo lleva a localizar con bastante exactitud el tipo de operación que Penrose debe ejecutar sobre el *Problema de Parada*, Pp a partir de ahora. Se trata de hallar un procedimiento suficientemente próximo al descrito en H(x,y) –cfr. supra [10.iii]– como para que el argumento diagonal tenga aplicación, pero permitiendo

---

[1] El nombre de *técnica de diagonalización* procede de un cierto tipo de representación de este punto del cálculo.

[2] Esto no supone que para cierto procedimiento efectivo o función no padamos demostrar en qué casos está definido y en cuales no. Lo que debemos aceptar es la existencia de problemas de este tipo que no podemos resolver ofreciendo una respuesta afirmativa o negativa.

mostrar una forma de calcular *por otros medios* lo que una función computable no puede. Es decir, se trata de *torcer* el argumento diagonal original de forma que podamos determinar *racionalmente* que un cierto procedimiento adquiere un valor que no puede adquirir si suponemos que se trata de un procedimiento representable por medio de una tarea computable. El punto es sutil y, por desgracia, no siempre aparece expuesto con la suficiente claridad en el original de Penrose. De hecho, mucho de lo que sigue puede ser entendido como una reconstrucción racional del argumento que le lleva a adoptar una posición tan favorable al mentalismo, argumento sugerido en su obra más que establecido de forma tajante.

Lo primero que voy a analizar es aquella parte del argumento diagonal precedente –el descrito en [11]– que a mi juicio despierta la imaginación de Penrose. Para ello voy a recortar un fragmento de esa demostración introduciendo un ligerísimo cambio en su estructura. H(x,y), g(x) y g*(x) son como antes.

[12] Fragmento del Problema de Parada.

1. H(x,y)A	
2. Si H(x,y)A, entonces g(x)∈ A	[por la definición de g(x)]
3. Si g(x)∈ A, entonces g*(x)∈ A	[por la definición de g*(x)]
4. g*(x)∈ A	[MP 1,2 y 2,3]
5. g*(x)=f_i(x)	[por 4 y la enumeración *e*]
6. f_i(i)=1	
7. g*(i)=1	[reemplazando en 6 f_i por g*]
8. f_i(i) está definida	[por 7]
9. g(i)=1	[por la definición de g]
10. f_i(i) no está definida	[por la definición de g*: si g(i)=1, entonces g*(i) no está definida, y reemplazando g* por f_i, $f_i(i)\uparrow$]
11. ¬(f_i(i)=1)	[reductio 6-10]
12. f_i(i) no está definida	[por 11 y la def. de g*]
.....	

Como se puede ver, este fragmento es en todo idéntico al anterior salvo por el hecho de que en la línea 6 hablamos de f_i(x) y no de g*(x). La diferencia es poca, pero importante. f_i(x) es una función que suponemos computable, pues ocupa la i-ésima posición en la enumeración de A. Tras varios pasos que el lector puede revisar si quiere, se llega en la línea 12 a establecer

de forma absolutamente elemental que $f_i(i)$ no está definida. Para Penrose este proceso constituye el germen de una demostración genuina de que esa función computable no está definida para un cierto valor, conclusión que, no obstante, no puede ser alcanzada por función computable alguna salvo contradicción –si así fuera, simplemente completaríamos el fragmento expuesto en [12] hasta obtener algo muy similar a [11]. Hay pues algo que mi mente puede establecer acerca de $f_i(x)$ que ningún programa es capaz de establecer de manera consistente. La idea de Penrose expuesta en el NAP no es sino una forma de dotar de solidez a esta observación o intuición fundamental preservándola de todos los obstáculos que aún le quedan por salvar. El principal de ellos consiste en hacer que esa prueba, que bastaría para que yo pudiera reconocer que $f_i(i)$ no está definida, sea realmente aceptable desde un punto de vista intuitivo. El análisis lógico de las condiciones que hacen posible que tal demostración resulte aceptable es algo que Penrose no hace explícito en su argumento. Pero su estudio permite, no sólo entender las razones a favor de sus tesis, sino también aquellas que a mi juicio se pueden esgrimir en contra.

Una condición necesaria para aceptar una demostración, del tipo que sea, es que resulte consistente. Y la anterior aún no lo es. Siendo $H(x,y)$, $g(x)$ y $g^*(x)$ como antes, parece inevitable que el fragmento ofrecido en [12] se prolongue del modo siguiente:

[13] Prolongación de [12]

13. $g(i)=0$	[por 12 y la def. de g]
14. $g^*(i)=1$	[por 13 y la def. de g*]
15. $f_i(i)=1$	[reemplazando en 14 g* por f_i]

En la línea 15 se afirma que la i-ésima función computable está definida tomado el valor *1* cuando procede a calcular que asigna a *i*. Resulta así que las razones que eventualmente pudiera tener para convencerme racionalmente de que $f_i(i)$ no está definida tienen como consecuencia lógica que $f_i(i)=1$. Malas razones son esas que me permiten asegurar una cosa tanto como su contraria. El análisis lógico del problema empieza, no obstante, cuando nos preguntamos acerca de la posibilidad de afirmar consistentemente que $f_i(i)$ no está definida, es decir, cuando nos preguntamos por la posibilidad de aislar [12] de [13], o mejor dicho, lo que serían sus correlatos abstractos, aquello que queda cuando se elimina todo lo que es superfluo.

¿Cuál es el material realmente imprescindible en el planteamiento que

subyace al NAP? Parece claro que lo que anda en juego es la relación existente entre un cierto algoritmo o tarea, que posee una definición explícita independiente de formalismo alguno, y la función computable o programa que presumiblemente representa el algoritmo anterior. Es decir, volvemos a enfrentar un término informal, un cierto procedimiento o algoritmo, a un programa. Nos referiremos al primer término del problema, es decir al algoritmo, mediante $\chi(x)$ y a la función computable asociada mediante $f_i(x)$ de nuevo. Por el momento, no necesito decir nada acerca de la definición explícita de $\chi(x)$. Asumo que se trata de alguna tarea que se efectúa sobre números, pero como ya he dicho antes esto no nos compromete con nada y en particular, no implica que nuestras conclusiones sólo afecten al dominio de la aritmética. En lo que sigue, asumiremos que A continúa representando la clase de todas las funciones numéricas definibles por medio de un programa, y que dicha clase en enumerable. Implícitamente nos comprometemos además con el hecho de que $f_i(x)$ sólo puede, o adoptar el valor *1*, o quedar indefinida, pero eso es todo. De lo que se trata es de reproducir la estructura de [12] sobre $\chi(x)$ y $f_i(x)$, entendiendo que $f_i(x)$ representa esta vez a $\chi(x)$, es decir, constituye un programa asociado a la tarea que en términos informales se propone ejecutar $\chi(x)$. Eso implica demostrar que $f_i(i)$ no está definida partiendo tan sólo del supuesto de que $\chi(x) \in A$ –es computable– y de que $f_i(x)$ representa a $\chi(x)$. Bien, pues hagámoslo limitándonos a ver qué hace falta para llegar de una cosa a la otra:

[14] Demostrando que $f_i(i)$ no está definida

1. $\chi(x) \in A$
2. $\chi(x)=f_i(x)$ [por 1]
3. $f_i(i)=1$

Puesto que no podemos suponer que de $f_i(i)=1$ se siga definicionalmente que $f_i(i)$ está indefinida y todo lo que sabemos es que $\chi(x)$ es representado por $f_i(x)$,

4. $\chi(i)=1$ [por 2]

Es obvio que de $\chi(i)=1$ debe seguirse que $f_i(i)$ está indefinida, lo cual determina ya en parte la definición de $\chi(x)$, que por ahora hemos obviado.

5. $f_i(i)$ está indefinida
6. $\neg (f_i(i)=1)$ [reductio 3-4]
7. $f_i(i)$ está indefinida [por 6]

Parece haber, en definitiva, dos pasos críticos en la demostración –inconclusa– anterior, el que permite pasar de la línea 3 a la línea 4 y el que lleva de ésta a la línea 5.

[15] Condiciones críticas sobre la demostración [14]

i. Si $f_i(x)$ está definida, entonces el algoritmo representado por ella $\chi(x)$ está definido y toma el valor que $f_i(x)$ adopta en ese caso.
ii. Si $\chi(x)=1$, entonces $f_x(x)$ no está definida.

La primera de estas condiciones garantiza que de $f_i(i)=1$ se pueda pasar a $\chi(i)=1$, paso sin el cual no se podría llegar a afirmar que esa misma función computable queda indefinida, es decir, que $f_i(i)$ está indefinida. La segunda condición permite pasar de $\chi(i)=1$ a afirmar que $f_i(i)$ queda, de hecho, indefinida. Si se piensa un instante, pronto apreciamos que todo lo que se ha hecho en [14] y [15] es establecer una manera de pasar de $f_i(i)=1$ a $\neg(f_i(i)=1)$ –$f_i(i)$ está indefinida– que no resulte directamente autocontradictoria. Para ello basta con que nos sirvamos de lo que la tradición denominaría un *término medio* que en este caso pasa por hacer uso de $\chi(x)$ y de la condición [15.ii]. Los requisitos reunidos en [15] constituyen, simplemente, la expresión más general que puede darse a las condiciones mínimas imprescindibles para alcanzar nuestro objetivo.

Ahora bien, queda aún por finalizar la demostración iniciada en [14] y hay que hacerlo, además, de modo que no se pueda concluir algo contradictorio acerca de la conducta de $f_i(i)$. Téngase en cuenta que lo que necesitamos es una demostración consistente de que $f_i(i)$ permanece indefinida, pero una que valga, claro está, para afirmar que, por eso mismo, $\chi(i)=1$. Pero no adelantemos acontecimientos.

La forma más obvia de finalizar la demostración iniciada en [14] pasa por considerar la línea 7 como causa suficiente para sostener que $\chi(i)$ queda igualmente indefinida. Esto es así en la medida en que hemos supuesto que $f_i(x)$ es la función que representa el algoritmo expresado en $\chi(x)$ y por tanto, que todo lo que se pueda decir de una se podrá decir de la otra. Si, por las razones que fueren, se consiguiera añadir más adelante alguna línea conteniendo $\chi(i)=1$, la presunta equivalencia entre función computable –$f_i(x)$– y el algoritmo correspondiente –$\chi(x)$– permitiría pasar a afirmar que $f_i(i)=1$, lo que arruinaría cualquier intento de demostrar consistentemente que $f_i(i)$ permanece indefinida.

La cuestión parece haber alcanzado un punto que nos obliga a revisar la libre substitución entre $f_i(x)$ y $\chi(x)$. El hecho de que $f_i(x)$ *represente* el algoritmo expresado por $\chi(x)$ podría no llevar necesariamente a admitir cualesquiera substituciones de uno de estos términos por el otro con independencia del contexto en que puedan ocurrir. Resultaría entonces que lo único con que nos comprometeríamos al decir que una cierta función computable, o programa, representa este o aquel algoritmo no iría más allá de la condición descrita en [15.i]. Es decir, que cuando la función computable que representa un cierto algoritmo adopta un valor el propio algoritmo adopta ese valor.

Soy consciente de que esta propuesta puede desconcertar a muchos lectores, incluso a aquellos que hayan conseguido llegar a este punto sin excesivos problemas. ¿Qué otra cosa se puede querer decir cuando se afirma que un programa *representa* un algoritmo distinta a que programa y algoritmo son intercambiables –son lo mismo– en cualquier contexto? Uno de los puntos peor explicados por Penrose en la exposición de su argumento son las razones que, de hecho, le obligan a romper los términos de esa identidad. Mi impresión es que Penrose ve en ello una genuina *petitio principii* que descarta cualquier análisis ulterior del problema. Si cuando decimos poseer un programa que representa un algoritmo creemos estar diciendo que ese algoritmo *es*, de hecho, ese programa, entonces estamos dando por sentado que cualquier tarea que podamos describir en términos de un algoritmo es computable. O lo que es lo mismo, que no existen tareas que puedan ser descritas de manera objetiva por la mente humana que no puedan ser ejecutadas igualmente por un ingenio mecánico, o programa, no humano.

Aunque parezca extraño lo que se propone aquí, todos sabemos que cuando decimos que este o aquel programa describe esta o aquella tarea efectiva, lo que sostenemos es nuestra disposición a aceptar los resultados que arroja ese programa cuando concluye con éxito su tarea. Pero, posiblemente, no estamos tan dispuestos a aceptar que nuestra tarea no arroje resultado alguno en aquellos casos en que lo único que vemos es un programa que ejecuta pasos sin aparente término. Es posible que en tal circunstancia estemos más dispuestos a ensayar otras opciones que a continuar con el programa en cuestión. Opciones que, eventualmente, pueden llevar a conclusiones contrarias a las que se desprenderían de la operación del programa en curso.

No pretendo salvar a Penrose de lo que son dudas más que razonables acerca de lo que se esconde en el fondo de su argumento. Mi objetivo es concederle todo cuanto sea posible para continuar analizando su prueba hasta el final. El resultado es la necesidad de admitir, junto a las condiciones ya descritas en [15], lo siguiente:

[16] Relación entre un algoritmo y el programa que lo representa:

– Sea $\varphi(x)$ un cierto algoritmo y sea $f_k(x)$ la k-ésima función computable en la enumeración de todas las funciones de ese mismo tipo. Decir que $f_k(x)$ *representa* a $\varphi(x)$ no permite en general substituir libremente una entidad por otra en cualesquiera contextos.

La reunión de [15] y [16] permite imaginar una prueba puramente lógica en la que resulta posible establecer simultáneamente que: i. $f_i(x)$ representa a $\chi(x)$, ii. $f_i(i)$ está indefinida y, finalmente, que $\chi(i)=1$, y todo ello a partir tan sólo del supuesto según el cual $\chi(x)$ es representable en términos de algún programa. Veámoslo:

[17] Esquema lógico del NAP

1. $\chi(x)$ es representable en A	
2. $f_i(x)$ representa a $\chi(x)$	[por 1]
3. Si $\chi(x)=1$, entonces $f_x(x)$ está indefinida	[único requisito a exigir en χ]
4. $f_i(i)=1$	
5. $\chi(i)=1$	[por R1]
6. $f_i(i)$ está indefinida	[por 5 y 3]
7. $\neg(f_i(i)=1)$	[reductio 4-6]
8. $f_i(i)$ está indefinida	[por 7]
9. $\chi(i)=1$	[desiderata]
10. $f_i(i)\neq\chi(i)$	[por 8 y 9]

La línea 9, que marca la diferencia entre $f_i(x)$ y $\chi(x)$, se introduce ahora como un simple *desiderata* que queda pendiente de justificación, pues realmente no hay nada aún que permita comprender las razones por las cuales el material disponible ente las líneas 1 y 8 pueda llevar a concluir que $\chi(i)=1$.

Lo que queda por hacer es, entonces, presentar un algoritmo que satisfaga explícitamente la condición [15.ii] y que permita dar el paso que se contiene en la línea 9. Con esto regresamos a la obra de Penrose ofreciendo la definición del procedimiento que según ese autor no es posible representar adecuadamente mediante ningún programa.

[18] Algoritmo *J(x,y)* de Penrose.

– J(x,y) encapsula todos los métodos aceptables –consistentes– habilitados para demostrar que la rutina que inicia la función con índice *x* cuando procede a calcular el input *y* no finaliza. Si alguna de estas pruebas finaliza mostrando que $f_x(y)$ está indefinida, entonces, y sólo entonces, J(x,y)=1.

Este procedimiento resulta, como poco, llamativo. No obstante, tiene una cierta tradición en la literatura sobre mecanicismo. Su plausibilidad surge de la evidencia aportada por las innumerables demostraciones en las que se establece que un cierto problema carece de solución. No voy a entrar en ello ahora, pero son muchos y muy conocidos los problemas matemáticos clásicos que han dado lugar a ese género de demostraciones. El propio Penrose dedica un buen número de páginas de su obra a exponer y comentar algunos de ellos.

Que J(x,y) tiene el carácter de un algoritmo es obvio: sólo considera *demostraciones consistentes* como razón para admitir un cierto hecho o proposición. Y una demostración es, proceda de donde proceda, un argumento finito que parte de unas premisas o axiomas y en el que cada paso es consecuencia inmediata de alguno o algunos de los precedentes. El último de estos pasos constituye la propia conclusión de la demostración. Por otra parte, es cierto que J(x,y) no invoca directamente programa alguno: admite *cualquier procedimiento* de prueba admisible como tal en un momento dado. Se trata de un procedimiento cuya definición parece suficientemente clara al tiempo que deja abierto el rango de recursos de los que puede servirse. Parece, en definitiva, un buen ejemplo a tener en cuenta a la hora de evaluar la existencia de algoritmos perfectamente definidos cuya traducción a los términos precisos de un lenguaje de programación resulta, no obstante, imposible.

El argumento que Penrose emplea para intentar convencernos de que J(x,y) se encuentra en ese caso puede ser entendido ahora como una instancia del esquema que he descrito en [17]. De hecho, tengo la impresión de que esa estructura es común a muchos tipos de argumentos empleados en la misma dirección que el NAP[3]. Todo lo que es preciso hacer para que el NAP adopte ese formato es definir $\chi(x)$ en términos de J(x,y) y ver entonces si el algoritmo así obtenido satisface las condiciones que se precisan para concluir con éxito la demostración. Es decir, la condición indicada en [15.ii] y la justificación de la línea 9 en la demostración dada en [17].

[19] Definición de $\chi(x)$
  – $\chi(x)=J(x,x)$

$\chi(x)$ resulta ser, por tanto, un algoritmo que contiene todas las demostraciones consistentes que establecen que un cierto programa no concluye su rutina con éxito cuando computa el número que casualmente coincide con

[3] Pienso, en concreto, en el conocido argumento de Kalmar al cual Penrose le debe mucho aquí aunque no lo haga explícito.

aquel que ese programa posee bajo una enumeración previamente establecida. La autorreferencia aquí es inevitable. Que $\chi(x)$ satisface [15.ii] resulta ser simplemente inmediato: $\chi(x)$ adopta el valor *1* sólo en aquellos casos en que $f_x(x)$ no está definida. Es importante entender que $\chi(x)$ no adopta ese valor en todos los casos en que de hecho tal cosa sucede. Si tuviéramos que sostener tal cosa, entonces estaríamos de nuevo ante una versión del algoritmo asociado al *Problema de Parada*, y por tanto, ante algo que sabemos que no tiene sentido analizar. La función descrita por Penrose es, en definitiva, bastante razonable: no se exige en ningún momento que sea capaz de determinar si toda función computable no definida sea demostrablemente no definida, se limita a reunir todas las técnicas habilitadas o reconocibles como consistentes para establecer que una función computable no alcanza finalizar su rutina arrojando un resultado.

El cuidado puesto en que el esquema argumentativo anterior sea consistente permite dotar de una justificación realmente robusta al paso 9 marcado hasta ahora como desiderata. Obsérvese que 1-8 es una demostración consistente, bajo ciertos supuestos, de que $f_i(i)$ no está definida, lo cual nos lleva a incluirla entre una de las que haría que (x), por su definición, adoptase el valor 1. Esto lleva a traducir el esquema lógico anterior en una prueba genuina:

[20] *Prueba de Penrose*:

1. Si $\chi(x)=1$, entonces $f_x(x)=1$	[por la definición de J(x,y)]
2. $\chi(x)$ es representable en A	
3. $f_i(x)$ representa a $\chi(x)$	[por 2]
4. $f_i(i)=1$	
5. $\chi(i)=1$	[por 4]
6. $f_i(i)$ no está definida	[por 5 y 3]
7. $\neg(f_i(i)=1)$	[reductio 4-6]
8. $f_i(i)$ no está definida	[por 7]
9. $\chi(x)=1$	[1-8 constituye una demostración consistente de $f_i(i)$ no está definida]
10. $f_i(i)\neq\chi(i)$	[por 8 y 9]
11. $\chi(x)$ no es representable en A	

Restaurando el equilibrio

Mi análisis formal del NAP parece concluir dando la razón a las posiciones de Penrose y con ello a las del mentalismo tal y como se ha descrito líneas atrás. Se ha conseguido diseñar un argumento diagonal parte del cual actúa como una demostración capaz de establecer que el algoritmo encarnado por $\chi(x)$ no finaliza cuando computa un cierto input. Se trata de una prueba que yo, como sujeto racional, puedo reconocer e incorporar entre aquellas que me permiten establecer algo con entera certeza. Pero, por la misma naturaleza del razonamiento, no se puede afirmar que esa misma demostración pueda ser incorporada como parte del programa que presuntamente representaría el algoritmo encarnado en $\chi(x)$. Se trata de un tipo de demostración que no está disponible para ese programa, una a la cual ningún programa puede acceder.

Es imposible pretender que este estudio, o el propio argumento de Penrose, resulten claros e inteligibles sin dedicar algún tiempo a reflexionar sobre ello. No es el tipo de problema que resulte fácil de entender a primera vista ni con el que se disfrute de una agradable lectura. Se trata, como casi siempre que nos movemos cerca de los dominios de la autorreferencia[4], de asuntos de carácter sutil más parecidos a torcidos juegos de palabras que a auténticos argumentos. Sus consecuencias no son, sin embargo, triviales. De la adecuada evaluación del NAP depende, por ejemplo, la corrección del objetivo último del mecanicismo. O dicho en unos términos algo más populares hoy en día, depende la viabilidad del programa de la Inteligencia Artificial fuerte (IA fuerte). No merece la pena insistir en las consecuencias prácticas que el reconocimiento del NAP tendría sobre innumerables investigaciones que de un modo u otro asumen o aceptan que la diferencia entre la mente humana y los actuales ordenadores es meramente de grado. Y por tanto, eliminable en un futuro más o menos cercano. La confirmación del NAP supondría, sencillamente, tener que empezar a mirar en otra dirección.

Pese a que no ha transcurrido en realidad mucho tiempo desde que Penrose diese a conocer este argumento, sí podemos suponer que se trata del suficiente como para sospechar, dada la importancia de sus consecuencias, que no ha convencido a la comunidad científica que está en condiciones de juzgarlo. Si, como me parece evidente, el NAP no es sino una variante algo más depurada del viejo argumento de Kalmar, la existencia de dificultades entorno a este tipo de planteamientos parece la norma y no la excepción.

---

[4] La autorreferencia se produce aquí en la medida en que admitimos que una función computable sea representada por un número, su código, al tiempo que procede a calcuklra números a partir de otros dados como argumentos.

Llegados a este punto, no creo que deba, ni pueda, prolongar por más tiempo mi posición ecléctica ante el NAP. El análisis lógico precedente no estaba destinado al puro goce estético, como es evidente, sino a suministrar las razones que me van a permitir sostener mi *rechazo* al argumento de Penrose. Veo en ello la recuperación del equilibrio entre mentalismo y mecanicismo afectada, quizá, por lo que parecía un prometedor intento de reclamar la debilitada primacía del ser humano sobre algunos de los productos de su ingenio.

De todas las posibles formas de restaurar los derechos del mecanicismo hay una que no aceptaré por más obvia que parezca. Se trata de aquella por la cual el propio argumento racional diseñado por Penrose puede ser incorporado, en alguna instancia superior, entre aquellos que son representables por medio de un programa. El carácter racional y objetivo del procedimiento seguido garantizaría que, *una vez construido este*, buscásemos, y de hecho hallásemos, una función computable $f_i(x)$ capaz de incorporarlo entre sus recursos. Estaríamos así ante una solución muy bienvenida desde las posiciones mentalistas. Y ello por una razón obvia: comunica claramente la impresión de que el ser humano es siempre capaz de *ir un paso por delante* –cfr.: J.R. Lucas, op. cit.–. Nuestra mente poseería un tipo de plasticidad que le permite idear criterios racionales de admisibilidad que son ajenos a cualquier formalismo definido de una vez por todas, por más que estos puedan dar cuenta a posteriori de lo que nuestra mente es capaz de ingeniar.

Mi crítica al argumento de Penrose me obliga a revisar la relación que establecemos entre un algoritmo y una función computable determinada cuando afirmamos que la función computable –un programa– $f_k(x)$ *representa* el algoritmo $\varphi(x)$. Parece claro que en esa relación de representabilidad, a diferencia de lo que sucede en el caso de la mera identidad, algoritmo y función representante conservan su autonomía. La cuestión que me intriga consiste en tener claro qué se supone que representamos cuando aceptamos construir un programa para $\chi(x)$.

$\chi(x)$, por su definición a partir de $J(x,y)$ encapsula –el término es de Penrose– *todas* las demostraciones consistentes destinadas a establecer que una función computable no se detiene, no finaliza arrojando un resultado, cuando ejecuta una cierta rutina. Mi duda hace referencia al modo en que debemos interpretar esa cuantificación universal –*todas las demostraciones*– en el momento de conceder un código a $\chi(x)$. Claramente hay dos opciones.

[21] **Opción 1**. El cuantor universal que figura en la definición informal de $\chi(x)$ hace referencia a cualquier posible demostración existente alguna vez. Esto

sugiere un dilema:

a) el programa responde también a esa interpretación del cuantor,

b) el programa que $f_i(x)$ encarna no responde a una interpretación tan amplia del cuantor.

Si optamos por a), entonces nos vemos forzados a reconocer que $f_i(x)$ posee recursos suficientes como para romper el bucle que se alcanza al afirmar que $f_i(i)$ no está definida –es decir, que ejecuta pasos sin término ni objetivo alguno–. Este bucle se revela, realmente, como un falso bucle y por tanto como una mera etapa del cómputo de $f_i(x)$. Una que además permite afirmar que $f_i(i)=1$. La prueba entera degenera –resulta inconsistente– y el argumento queda invalidado.

Si optamos por b), difícilmente podremos ver en $f_i(x)$ una representación de $\chi(x)$, ya que estamos aceptando que $\chi(x)$ no es una entidad dada de una vez por todas, sino en permanente cambio.

[22] **Opción 2**. El cuantor universal que figura en la definición informal de $\chi(x)$ se refiere a cualesquiera demostraciones consistentes que puedan ser aceptadas como tales *justo antes* de asignar a $\chi(x)$ un programa que lo representa. O, alternativamente, todas aquellas que sean independientes del hecho de suponer que *i* es el código de la función computable o programa que representa a $\chi(x)$ en A.

La opción 2 es, sin duda, la más interesante. Las demostraciones matemáticas no suceden realmente en el tiempo. No hay una relación de temporalidad independiente o autónoma que permita decir que esta demostración es anterior a aquella, salvo que exista una relación de dependencia lógica. Una demostración aún no construida en el momento de asignar a $\chi(x)$ el código *i* puede contar como recurso disponible para $\chi(x)$ siempre que no dependa lógicamente de nada previamente establecido o supuesto para ese algoritmo. Pero, como hemos visto, ese no es, ni puede ser el caso en el NAP. La demostración que se desarrolla de 1-8 depende, precisamente de ese dato, con lo cual, deja automáticamente de ser una demostración accesible para $\chi(x)$.

La corrección mecanicista al argumento de Penrose tiene lugar en forma de una restricción acerca del rango de recursos disponibles para programar una tarea supuestamente efectiva, restricción que en este caso se aplica a las demostraciones con que $\chi(x)$ puede contar a la hora de admitir o rechazar algo. Esto no significa que la demostración 1-8 no pueda ser reconocida por

mi ingenio como una prueba correcta de que $f_i(i)$ no está definida. Lo único que supone es que no puedo reconocerla como parte de $\chi(x)$ porque ha sido generada a partir de la hipótesis de que $\chi(x)$ estaba dado de una vez por todas en el programa representado por el código i.

La reconducción mecanicista del argumento lógico de Penrose daría lugar a lo siguiente:

[23] *Corrección del NAP*

1. Si $\chi(x)=1$, entonces $f_x(x)=1$	[por la definición de J(x,y)]
2. $\chi(x)$ es representable en A	
3. $f_i(x)$ representa a $\chi(x)$	[por 2]
4. $f_i(i)=1$	
5. $\chi(i)=1$	[por R1]
6. $f_i(i)$ no está definida	[por 5 y 3]
7. $\neg(f_i(i)=1)$	[reductio 4-6]
8. $f_i(i)$ no está definida	[por 7]
9. **$\chi'(i)=1$**	[1-8 es una prueba aceptable de que $f_i(i)$ no está definida]

.....
.....

Es decir, el algoritmo que reconoce el hecho de que $f_i(i)$ no está definida no puede ser el propio $\chi(x)$, ya que hemos admitido tenerlo completamente dado ante nosotros antes de iniciar la demostración que permite que reconozcamos que $f_i(i)$ no está definida. Eso sólo implica contar con un procedimiento $\chi'(x)$ que sería en todo igual a $\chi(x)$ salvo por el hecho de que incorpora entre sus recursos una demostración como la precedente. $\chi'(x)$ será tan representable en términos de algún programa como el propio $\chi(x)$ dando lugar, de este modo, a lo que en lógica se denomina una *jerarquía inductiva.*

Sea como fuere, debe quedar claro que lo único que hay de incorrecto en el NAP, tal y como se refleja en [20], es el uso en la línea 9 del mismo algoritmo que hemos supuesto dado al principio de la demostración. Cambiar $\chi(x)$ por un algoritmo $\chi'(x)$ impide concluir que $f_i(i)\neq\chi(i)$ en la línea 10 de [20], con lo que el formato lógico del NAP se transforma en el que acabo de exponer en [23]. Este esquema no sirve a los propósitos de Penrose, ni a los del mentalismo en general, aunque tampoco inclina la balanza ni un sólo milímetro más a favor de las posiciones mecanicistas. Simplemente detiene el golpe dejando las cosas como ya estaban, ni más, ni menos.

La línea argumentativa que he seguido conduce, como ya he dicho, al sutil territorio de las jerarquías inductivas. Diré sólo una palabra al respecto. Es obvio que lo que se consigue con esta reinterpretación del NAP es dar lugar a una jerarquía de algoritmos χ^0, χ^1, ... y una réplica en términos de funciones computables que hace corresponder a cada nuevo algoritmo en la serie una nueva función computable. Esto no ofrece, en principio, nada que permita mostrar lo que es la clave de bóveda del NAP, es decir, que $\chi(i) \neq f_i(i)$ para algún i.

Llegados a este punto resulta inevitable recordar un misterioso pronunciamiento de Gödel en el que se sustentan sus considerables dudas y reticencias acerca de las posiciones mecanicistas. Los fragmentos que cito a continuación proceden de *A logical Journey. From Gödel to Philosophy*, Mit Press, 1996, de Hao Wang, aunque ya aparecen en su mayoría en *From Mathematics to Philosophy*, de 1974.

> "Mind, in its use, is not static, but constantly developing.[...] Althought at each stage of the mind's developement the number of its posible states is finite, there is no reason why this number should not converge to infinity in the course of its developement. [...] Now there may exists systematic methods of acelerating, specializing, and uniquely determining this developement, for example, by asking the right questions on the basis of a mechanical procedure. But it must be admitted that the precise definition of a procedure of this kind would require a substantial deepening of our understanding of the basic operations of the mind. Vaguely defined procedures of our understanding of the basic operations of the mind. Vaguely defined procedures of this kind, however, are known, for example, the process of defining recursive well-orderings of integers representing larger and larger ordinals or the process of forming stronger and stronger axioms fo infinity." p. 199-200.

El proceso inductivo que acabo de describir se asemeja mucho al modelo que Gödel parece tener en mente. Si profundizamos aún un paso más en la descripción de la situación anterior nos damos cuenta de que ahora queda a nuestro alcance un peculiar objeto: el límite de la serie χ^0, χ^1,...χ^n,... Representaremos ese límite como χ^ω. χ^ω parece encapsular todos los mecanismos de prueba destinados a demostrar que una cierta función computable $f_x(x)$ no está definida, junto con todas las demostraciones que se inician suponiendo que existe una función computable que representa en A ese algoritmo. La peculiaridad de este objeto es que en virtud de su definición informal, parece formar un *punto fijo*, es decir: $\chi^\omega(x) = \chi^{\omega+1}$. No hay, en otras palabras, un algoritmo siguiente distinto de él mismo. Tengo la impresión de que en tal

caso la opción que se impone es aquella que hace que la función computable $f_n(x)$ introducida en el supuesto de la demostración, la cual representaría ahora al propio $\chi^{\omega}(x)$, *vea* también el resto de la prueba, haciendo entonces que a partir de $\chi^{\omega}(n)=1$ se concluya que $f_n(n)=1$.

Con estos comentarios nos aproximamos claramente al punto donde muy posiblemente van a tener lugar las próximas situaciones de tensión entre mecanicismo y mentalismo. El problema reside en que no creo que dispongamos aún de una idea precisa del equilibrio existente entre tres de las características presentes en el modelo computacional vigente. Me refiero a: i. la plasticidad o capacidad de un programa para admitir alteraciones de su diseño sin cambiar él mismo, ii. el hecho de que un programa es una entidad dada de una vez por todas, es decir, mantiene su identidad en todo momento, y iii. la forma en que los programas se refieren a otros programas en el curso de sus operaciones. No digo que estos tres aspectos no estén tratados en la literatura, ni que, por separado, no estén claros. Lo que vengo a sostener es que no poseemos una idea del todo precisa de lo que sucede cuando interactuan todos ellos a la vez.

Es posible, sólo posible, que entre alguno de los lectores que hayan conseguido sortear con éxito los obstáculos que tiene la lectura de este texto cunda ahora la impresión de haber llegado a un lugar en el que de algún modo ya se había estado antes. Los tres problemas que acabo de mencionar tienen un cierto aire familiar que llama la atención en un contexto aparentemente tan alejado del quehacer del filósofo profesional del siglo xxi. Pienso, en concreto, en la considerable similitud que esas cuestiones abiertas tienen con un problema de tan rancia tradición como el de la *identidad personal*. Me cuesta trabajo creer que lo que pueda decirse a continuación acerca de la constante pugna entre mentalismo y mecanicismo no tenga consecuencias, así las cosas, para ese viejo problema. Pero a cambio también creo con toda firmeza que es conveniente y productivo repasar lo que la filosofía tradicional ha dicho sobre el problema del sujeto con el fin de avanzar un paso más en la comprensión de los recursos que la moderna filosofía mecánica nos brinda.

Esta demanda de ayuda, que asumo con todas sus consecuencias, sólo puede sorprender a aquellos que vean en este texto un trabajo de lógica más y por tanto, algo por completo alejado del estilo y las preocupaciones de la genuina filosofía. Para aquellos que están dispuestos a aceptar la existencia de una filosofía hecha con las herramientas de la ciencia moderna, lo que aquí se propone sólo puede ser visto como una oportunidad.

Referencias bibliográficas

Davis, M. (ed.) (1965): *The Undecidable. Basic Papers on Undecidable Propositions, Unsolvable Problems, and Computable Functions.* Raven Press, Hewlett, Nueva York.

Gödel, K. (1931): "On Formally Undecidable Propositions of Principia Mathematica and Related Systems I" en *The Undecidable*, M. Davis (ed.). Raven Press. Hewlett, Nueva York, 1965.

Kalmar, L. (1957): "An Argument against the plausibility of Church's Thesis", *Constructivity in Mathematics, Proceedings of the colloquium held at Amsterdam, 1957*, A. Heyting (ed.). North-Holland Publishing Company. Amsterdam 1959, pp.72-80.

Kleene, S.C. (1987): "Reflections on Church's Thesis", *Notre Dame Journal of Formal Logic*, vol.28, nº4, pp.490-498.

Lucas, J.R. (1961): "Minds, Machines and Gödel", *Philosophy* 36, pp.120-124.

Lucas, J.R. (1970): *The Freedom of the Wil,* Oxford Univ. Press.

Lindström P. (2001): "Penrose's New Argument", *Journal of Philosophical Logic* 30, pp. 241-250.

Penrose, R. (1989): *La Nueva Mente del Emperador*. Mondadori. Madrid, 1991.

Penrose, R. (1994): *Shadows of the Mind*, Oxford Univ. Press.

Searle, J. (1989): "Mentes y Cerebros sin Programas". Ed. castellana de E. Rabosi. *Filosofía de la Mente y Ciencia Cognitiva*. E. Rabosi (ed.). Ediciones Paidós. Barcelona, 1996.

Thom, R. (1997): "The Hylemorphic Schemata in Mathematics", *Philosophy of Mathematics today*. Kluwer.

Turing, A.N. (1936): "On Computable Numbers, with an Applications to the Entscheidungsproblem", en *The Undecidable*, M. Davis (ed.). Raven Press. Hewlett, Nueva York, 1965.

Turing, A (1950): "Computing Machinery and Intelligence", *Mind*, vol. 59. Trad. castellana como "¿Puede Pensar una Máquina?" en *Mentes y Máquinas*. Tecnos. Madrid, 1985.

Wang, Hao (1974): en *From Mathematics to Philosophy*, New York: Humanities Press.

Wang, Hao (1996): *A logical Journey. From Gödel to Philosophy*, Mit Press.