COMPUTABILIDAD, ALEATORIEDAD Y LA TESIS DE CHURCH-TURING FÍSICA

Valentín Muro / Universidad de Buenos Aires

I. Introducción

Adam Olszewski (1999) propone que la Tesis de Church-Turing puede ser usada para refutar el platonismo matemático[1]. Para ello postula una máquina que al lanzar una moneda define una función que "computa el valor (0 o 1) para la moneda *n* que se haya lanzado" y dice que dicha función es efectivamente computable pero no Turing-computable (computable por una máquina de Turing). Entonces, dado que según el platonismo matemático (PM), toda función de enteros positivos a enteros positivos ya existe, existiría una función efectivamente computable pero no Turing-computable. Esto contradice a la Tesis de Church-Turing (CT) que dice que toda función efectivamente computable es Turing-computable. Por contraposición, si CT es verdadera, entonces PM es falso. Rafał Urbaniak (2011) critica este argumento desafiando la afirmación de que lo que esta máquina de hecho realiza es una computación. Revisaré dicha crítica y detallaré algunos puntos de la misma. Finalmente, haré una breve introducción a la Tesis de Church-Turing Física.

II. El argumento de Olszewski

Tomando en cuenta PM y CT, Olszewski (1999) propone un importante vínculo filosófico entre ambos: que ambas tesis se excluyen mutuamente. El argumento, tal como lo recupera Urbaniak (2011), es:

> Supongamos que CT es verdadera. También supongamos que los hechos son como los describen los platónicos (…) Uno puede inferir que todas las funciones en el conjunto de los números naturales con cero ya existen. Supongamos que tenemos una máquina a nuestra disposición, basada en una computadora, que usa un dispositivo periférico para lanzar una moneda aleatoriamente. Cada lanzamiento sucesivo (su número) y su resultado son almacenados (…) (el conjunto de valores sería 0 y 1). Si ese proceso (…) continuara infinitamente, una cierta función se determinaría. Sin embargo, en tanto objeto platónico, siempre existe. Asimismo, para cada número natural (el número de lanzamiento) uno puede computar el valor para el argumento. Es difícil,

[1] El platonismo matemático ha sido uno de los temas más debatidos en las últimas décadas en la filosofía de las matemáticas (Linnebo, 2009). Se trata de la postura metafísica de que existen entidades abstractas u objetos cuya existencia es independiente de nosotros y nuestro lenguaje, y que las afirmaciones matemáticas obtienen sus valores de verdad en virtud de que esos objetos tengan ciertas propiedades y cumplan con ciertas relaciones. Según la versión adoptada por Olszewski, toda función matemática en el conjunto de números naturales con el cero ya está definida platónicamente, es decir, ya existe.

> de todas formas, imaginar una prueba matemática de su computabilidad.[2] (Urbaniak, 2011:2 del manuscrito)

De acuerdo a Olszewski, si PM es verdadero existe una función intuitivamente[3] computable que no es Turing-computable.

III. Refutación de Urbaniak

Urbaniak traza su refutación indicando ciertas cuestiones que deben revisarse en el argumento de Olszewski. Primero, indica que el supuesto de que ninguna secuencia generada aleatoriamente es Turing-computable es problemático y describe una manera de llevar a cabo el argumento sin él. En segundo lugar discute la afirmación de que la función que describe Olszewski es efectivamente computable a pesar de no ser Turing-computable. En tercer lugar explica por qué la máquina que describe Olszewski no cuenta como una máquina que realiza computaciones. Finalmente trata la confusión conceptual responsable de la plausibilidad inicial del argumento.

Desglosando el argumento de Olszewski, Urbaniak reconstruye de la siguiente manera la descripción de su máquina:

> Una máquina, llamémosla M, genera aleatoriamente una secuencia infinita de 0s y 1s (o está en proceso de generar tal secuencia). (Urbaniak, 2011:3 del ms.)

Luego, destaca que esta definición se presta a ciertos inconvenientes, como por ejemplo que debemos confiar en la no Turing-computabilidad de la secuencia generada sin contar con buenos motivos para hacerlo.[4] Propone, entonces, redefinir M de la siguiente manera:

> Una máquina, llamémosla M, genera aleatoriamente una secuencia infinita de 0s y 1s (o está en proceso de generar tal secuencia), y resulta que la función que le corresponde a este proceso no es Turing-computable. (Urbaniak, 2011:5 del ms.)

La ventaja de este acercamiento es que uno no se envuelve en discusiones acerca de la noción de aleatoriedad[5] utilizada teniendo que argumentar que la secuencia generada por la máquina la satisface (dar tal argumento sería complicado ya que matemáticamente no sabemos nada de la función antes de encender la máquina).

---

[2] El trabajo original de Olszewski fue escrito en polaco. Traduje la versión en inglés ofrecida por Urbaniak.

[3] Se alude a lo intuitivo de una función en referencia a la simplicidad de la misma. Usaré sin distinción intuitivamente computable y efectivamente computable al referirme a funciones.

[4] Por ejemplo, podría darse el caso de que M describiera una función *f* que generara una secuencia tal que para los lanzamientos impares devolviera 0 y para los pares devolviera 1, pero dicha función es claramente Turing-computable.

[5] Por ejemplo, Chaitin (2001:112) da al menos seis en su estudio de la teoría de algoritmos. Sin embargo, para nuestros fines, podemos simplemente definir a un objeto como aleatorio "si es desorganizado y no tiene patrones ni regularidades" (Nies, 2008:97).

Esta nueva definición de M apoya la no Turing-computabilidad de la función descripta por ella.
Volvamos al argumento de Olszewski. La máquina M genera una secuencia aleatoria que no es Turing-computable. Si PM es verdadero, entonces la función que asocia el número de lanzamientos con cara o ceca de la moneda lanzada ya existe. Por lo tanto, tenemos una secuencia generada por una máquina que no es Turing-computable, pero dado que podemos encender nuestra máquina para descubrir el valor de la función para cualquier valor ingresado, según Olszewski, la función es efectivamente computable. Pero esto significa que el platonismo implica la existencia de una función efectivamente computable que no es Turing-computable, algo que CT niega. Por lo tanto, por CT el platonismo matemático es falso.
¿Pero por qué deberíamos aceptar que la función descripta por M es efectivamente computable a pesar de no ser Turing-computable? Olszewski no argumenta explícitamente pero pareciera ser que la intuición subyacente es que "por supuesto que *f* es computable. Para cualquier número finito x de pasos puedo descubrir cuál es el valor que la máquina produce luego de x pasos, sólo debo esperar y ver" (Urbaniak, 2011:6 del ms.). Si este es el fundamento para que *f* sea efectivamente computable, entonces resulta ser que esta no es la noción empleada en CT. Es prudente entonces volver a su definición. Según Copeland (2004), CT concierne a la noción de método efectivo o mecánico en lógica y matemática[6]. Un método es considerado efectivo o mecánico[7] siempre que:

> a) el método puede, en la práctica o en principio, ser llevado a cabo por un computador humano trabajando con papel y lápiz;
> b) el método puede ser dado al computador humano en forma de un número finito de instrucciones;
> c) el método no requiere ni perspicacia ni ingenio por parte del ser humano llevándolo a cabo;
> d) el método definitivamente funcionará si se lo lleva a cabo sin error;
> e) el método produce el resultado deseado en un número finito de pasos; o, si el resultado es una secuencia infinita de símbolos (como la expansión decimal de π), entonces el método produce cada símbolo individual en un número finito de pasos. (Copeland, 2004:42)

Urbaniak, con notable precisión, ubica el problema de la noción de computabilidad efectiva de Olszewski en primer lugar en la condición a) y dice:

> Esperar y ver qué valor devolverá una máquina, aunque en un sentido débil se trate de un procedimiento efectivo (luego de un número finito de pasos el resultado se obtiene), no es una computación. (Urbaniak, 2011:8 del ms.)

[6] Puede ubicarse el fundamento de gran parte de esta definición de computabilidad en (Turing, 1936).

[7] Los términos 'efectivo' y su sinónimo 'mecánico' en estas disciplinas son técnicos, es decir, no poseen su significado cotidiano. (Copeland, 2004)

Sostiene este argumento en el hecho de que deben mediar cuestiones epistémicas o extra-matemáticas que "esencialmente van más allá del uso de papel y lápiz" (Urbaniak, 2011:8). También indica problemas de índole epistémica con las condiciones b) y c), pero por falta de espacio no nos detendremos en ellos. En cuanto a d) y e) no hay conflicto alguno. La argumentación de Urbaniak da un giro cuando propone dejar de lado las definiciones históricas de computabilidad, cuyos conflictos con el argumento de Olszewski hemos visto, y en cambio enfocarnos en qué significa que una máquina realice una computación.[8] Tomamos el concepto de *usabilidad* de Piccinini (2011) como el criterio para considerar a ciertos procesos físicos como computaciones. Esta noción es tratada en detalle por Piccinini para lograr una definición adecuada de la Tesis de Church-Turing Física (ver más abajo). Destacamos las siguientes condiciones de usabilidad:

> **Regla de independencia de los procesos.** Para que una máquina sea usable, es decir, realice una genuina computación, el problema siendo resuelto debe ser definible independientemente del proceso que lo resuelve o computa. (Piccinini, 2011:25 del ms.)

La máquina M (o cualquier máquina aleatoria) no cumple esta condición porque:

> Primero, no hay manera de definir la función f: $\mathbf{N} \rightarrow \{0,1\}$ cuyos valores son generados por P [el proceso aleatorio] sin hacer referencia a P mismo. Por supuesto, f existe en el sentido de la teoría de conjuntos, dado que es el conjunto de los pares resultantes de P. Pero en este contexto, definir f significa ciertamente especificar la relación que hay entre los argumentos y los valores de f, como hacemos cuando definimos, por ejemplo, la función detención [halting] para máquinas de Turing. Si P es genuinamente aleatorio, entonces no existe manera de especificar f sin generar los valores de f haciendo correr P. (Piccinini, 2011:16 del ms.)

La función definida por la máquina M no es *independiente del proceso generador* ya que sólo quedará definida para el argumento n una vez que se hayan ejecutado los n pasos o lanzamientos de moneda. Otras condiciones de usabilidad son:

> **Repetibilidad.** Para que una máquina sea usable debe ser posible usarla varias veces para computar el valor de salida para un mismo valor de entrada.
>
> **Configurabilidad.** Para que una máquina sea usable debemos poder regresarla al estado original, para que cuando la pongamos en marcha nuevamente produzca el mismo resultado para el mismo valor de entrada. (Urbaniak, 2011:9 del ms.; Piccinini, 2011:26 del ms.)

Claramente, M no es *repetible*: si la detenemos por un rato y luego la arrancamos de nuevo es poco probable (y de hecho no está garantizado) que el fragmento inicial de la secuencia generada coincida con el fragmento inicial de la secuencia generada previamente a su detención. M tampoco es *configurable*: no tenemos

[8] Cabe mencionar un punto importante respecto de la función que supuestamente genera la máquina M: por varios motivos es posible negar que un proceso físico genuinamente aleatorio pueda contar como un proceso computacional. Para un análisis detallado de lo que sucede con la computación y los procesos físicos ver (Piccinini, 2011).

manera de asegurarnos de que esté en el exacto mismo estado inicial dos veces y que si la arrancáramos produciría la misma secuencia. Por lo tanto, M fracasa en la prueba de usabilidad y no puede ser considerada una máquina computacional o computadora.

Podemos ahora, entonces, detenernos en la noción de computabilidad que Olszewski utilizó, explicar por qué esta no es la misma utilizada en la formulación de CT e indagar por qué esta diferencia no es inmediatamente obvia. Pareciera que en vez de utilizar la noción de computabilidad efectiva, la afirmación de Olszewski de que *f* es efectivamente computable se apoya en una noción relacionada pero que tiene importantes diferencias: aquella de que *hay una manera de descubrir el resultado*. Si este es el sentido que asociamos con la computabilidad efectiva, entonces la afirmación de Olszewski es correcta: "una vez que encendemos la máquina, hay una manera de descubrir cuál es el valor que la función devolverá para determinado argumento. Sin embargo, esta no es la noción involucrada en CT." (Urbaniak, 2011:11 del ms.)

Ante la pregunta que podría quedar pendiente de si puede construirse una función efectivamente computable que no sea Turing-computable, ya contamos con herramientas para dar una respuesta negativa. La noción misma de *computabilidad efectiva* fue definida por varios autores usando distintos formalismos como las máquinas de Turing, funciones recursivas generales (Gödel) y λ-definibilidad (Church), para lograr capturar de forma precisa la noción informal de lo que puede ser calculado mediante procedimientos efectivos (Piccinini, 2011). Es por esto que sugerir una función efectivamente computable que no sea Turing-computable representa por definición una contradicción. Sin embargo, tal como muestra Urbaniak, el problema no se resuelve con sólo revisar las definiciones, sino que debe revisarse con rigor la noción de computabilidad efectiva utilizada en cada caso y debe verificarse si la misma es correcta o no.

IV. Tesis de Church-Turing Física

Durante las últimas dos o tres décadas se trabajó en la distinción entre la Tesis de Church-Turing propiamente dicha y su versión Física (CTF). Mientras que CT concierne a las funciones que son efectivamente computables en el sentido intuitivo, CTF concierne a aquellas funciones cuyos valores son generados por sistemas físicos. Piccinini (2007) recomienda la distinción entre dos nociones de CTF diferentes: la Fuerte y la Moderada.

> **Tesis de Church-Turing Física Fuerte:** Todo proceso físico es Turing-computable. (Piccinini, 2011:3 del ms.)

Para revisar esta primera definición consideraremos que ciertos procesos físicos son genuinamente aleatorios[9]. Si consideramos, entonces, que un proceso físico aleatorio define una función, entonces dicha función obtendrá sus resultados a partir de un proceso del que es completamente dependiente. Y, tal como vimos, esa

[9] Un ejemplo de *proceso genuinamente aleatorio* que encuentro por demás interesante es el del decaimiento atómico, un proceso estocástico o indeterminista a partir del cual se pueden producir secuencias de números genuinamente aleatorios. En oposición, un proceso *pseudoaleatorio* es aquel que aparenta ser aleatorio pero no lo es. Las secuencias pseudoaleatorias por lo general muestran aleatoriedad estadística a pesar de ser generados por un proceso completamente determinista.

función no podrá ser considerada efectivamente computable. Un segundo problema, que ya habíamos notado con la máquina de Olszewski, es que los procesos físicos no siempre son repetibles, justamente, como en el caso de nuestro genuino proceso aleatorio descripto por la máquina M. Un tercer motivo para desestimar la posibilidad de que todo proceso físico sea computable, retomando la noción de usabilidad, es que un usuario de este proceso no puede elegir el valor que desea que su función compute. Supongamos que la máquina M tarda 1 segundo por moneda para computar el valor. Si queremos saber el valor para el número 1 millón tendremos que esperar 11 días, dado que debe lanzar ese número de monedas para llegar al valor solicitado, pero dicho proceso no podría ser acelerado de ninguna manera.

> A diferencia de las computaciones propiamente dichas, los procesos aleatorios no pueden usarse para generar el valor deseado de una función o solucionar las instancias deseadas de un problema general. Los procesos aleatorios pueden ser *aprovechados* por una computación, por supuesto—hay importantes técnicas computacionales que dependen de elecciones aleatorias o pseudoaleatorias en alguna etapa de la computación. (…) Pero ninguna técnica computacional puede reducirse a una mera secuencia de elecciones aleatorias. Es por esto que cualquier tesis que sería falseada por una secuencia generada por un proceso aleatorio, tal como CTF Fuerte, es demasiado fuerte como para capturar la noción de computabilidad *física* (…). En oposición a lo que algunos autores parecen suponer, CTF Fuerte es demasiado fuerte como para ser la tesis física equivalente a la Tesis de Church-Turing Matemática. (Piccinini, 2011:18 del ms., itálica del autor)

Las restricciones mencionadas alcanzan para vislumbrar el perfil que debe tener un proceso considerado útil para la computación física, aunque, como muestra Piccinini, esta lista podría ser ampliada. Según él, el punto importante es que nos interesa una computación por lo que podemos aprender de ella, si no podemos aprender nada de una supuesta computación, entonces ese proceso no es relevante a CTF. Antes de introducirnos a la noción de CTF Moderada, expone algunas motivaciones:

> Si CTF se fundamenta en la noción epistemológica de computación, evitando así los problemas de CTF Fuerte, necesita cubrir menos que lo que puede *realizarse físicamente*. Lo que CTF debe cubrir es todo proceso de *computación física* genuina, donde la computación física es explicada en términos de nuestras restricciones de usabilidad (…). (Piccinini, 2011:31 del ms., itálica del autor)

Inmediatamente podemos empezar a notar que hay ciertas preocupaciones epistemológicas que motivan el adoptar una versión de CTF frente a la otra. En otras palabras, CTF Moderada trata de los procesos que pueden ser usados por un observador finito[10] para generar los valores de la función deseada.

> **Tesis de Church-Turing Física Moderada:** Toda función que es físicamente computable es Turing-computable. (Piccinini, 2011:4 del ms.)

[10] Respecto de la finitud del observador, recomiendo (Piccinini, 2011).

CTF Fuerte es una noción más amplia que la de computación por un procedimiento efectivo, en cambio, como CTF Moderada está restringida por criterios epistemológicamente relevantes y no presenta los problemas que surgen con CTF Fuerte, concretamente, que es demasiado fácil de falsear e irrelevante frente a la noción de computabilidad que motiva a CT, es por estos motivos que esta noción es la más adecuada para CTF.

V. Platonismo matemático y computabilidad efectiva

Hemos visto que la noción de Olszewski de computabilidad efectiva es inapropiada y es por ello que su argumento no funciona para refutar al platonismo matemático. Sin embargo, el análisis de estos argumentos sirve de disparador para la discusión sobre las nociones de computabilidad y su relación con los procesos físicos. Es remarcable el estudio de Piccinini sobre este último tema, en tanto nos provee de herramientas conceptuales para tratar asuntos como la hipercomputación[11], de gran interés para las ciencias de la computación. Es importante destacar que el pasaje que uno haga de una tesis matemática como CT a otro campo (como el de la física, o, si se quiere, el de la computación física) debe ser lo suficientemente riguroso como para no caer en discusiones como la que revisamos.

[11] Un hipercomputador computa funciones que no son Turing-computables.

Bibliografía

Chaitin, G. J. (2001). *Exploring randomness*. Londres: Springer-Verlag.

Copeland, B. J. (2004). *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life: Plus The Secrets of Enigma*. Nueva York: Oxford University Press.

Linnebo, Ø. (2009). Platonism in the Philosophy of Mathematics. *The Stanford Encyclopedia of Philosophy (Fall 2011 Edition).* Recuperado noviembre 21, 2011 de http://plato.stanford.edu/archives/fall2011/entries/platonism-mathematics.

Nies, A. (2009). *Computability and Randomness*. Nueva York: Oxford University Press.

Olszewski, A. (1999). Teza Churcha a platonizm. *Zagadnienia Filozoficzne w Nauce*, 24, 96–100.

Piccinini, G. (2007). Computationalism, the Church-Turing Thesis, and the Church-Turing Fallacy. *Synthese*, 154(1), 97–120.

Piccinini, G. (2011). The physical Church-Turing thesis: Modest or bold. *British Journal of Philosophy of Science*. (inédito).

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Maths. Soc. (2)*, 42, 230–265.

Urbaniak, R. (2011). How Not To Use the Church-Turing Thesis Against Platonism. *Philosophia Mathematica*, 19(1), 74–89.