PRINCIPIA **29**(3): 315–355 (2025) doi: 10.5007/1808-1711.2025.e96501
Published by NEL — Epistemology and Logic Research Group, Federal University of Santa Catarina (UFSC), Brazil.

COMPUTADORA HUMANA E HIPERCOMPUTABILIDAD RELATIVISTA: UNA DISCUSIÓN DE LA TESIS DE TURING FÍSICA

HUMAN COMPUTER AND RELATIVISTIC HYPERCOMPUTABILITY: A DISCUSSION OF THE PHYSICAL TURING THESIS

CARLOS VILLACÍS
Universidad Complutense de Madrid, ESPAÑA
cvillaci@ucm.es
https://orcid.org/0000-0003-2339-4463

Abstract. The Turing Thesis states that any effectively computable function is Turing-computable or computable by some Turing machine. Analogously, the physical Turing Thesis states that any physically computable function is Turing-computable. This thesis involves a notion of physical computation distinct from the mathematical version of the definition of computation. First, we discuss the distinction based on the condition of the medium-independent vehicle and Turing's idea of the human computer. Secondly, we discuss the physical Turing Thesis based on the thought experiment of Andréka *et al.* (2018) in which a relativistic hypercomputer can be built operating in the context of a slowly rotating black hole. With this we obtain a counterexample to the physical Turing Thesis since there would be a computer capable of physically computing non-Turing-computable functions.

Keywords: Turing machine • human computer • physical Turing Thesis • hypercomputability • relativistic hypercomputer

RECEIVED: 28/09/2023 REVISED: 03/08/2024 ACCEPTED: 03/02/2025

1. Introducción

La teoría de la computabilidad es una de las ramas de la lógica junto con la teoría de la demostración y la semántica formal (teoría de modelos en su versión matemáticamente más desarrollada). Esta rama se ocupa de las funciones efectivamente computables. Una función es recursiva si existe un algoritmo que la computa efectivamente. Del mismo modo, un conjunto es decidible si existe un algoritmo que permite determinar si cumple la propiedad que lo define. Se han planteado diversas definiciones formales de la noción informal de computabilidad efectiva, todas las cuales han resultado ser equivalentes: el cálculo-λ de Church, las funciones recursivas generales de Gödel, las máquinas de Turing, las funciones recursivas por minimalización de Kleene, etc. Estos distintos métodos definen las mismas funciones.

Las funciones que definen los mencionados métodos son efectivamente computables. Ahora bien, este resultado no es suficiente para demostrar que un conjunto

es indecidible. Para ello necesitamos su conversa, a saber: que toda función efectivamente computable es computable por una máquina de Turing (u otro método equivalente). Nótese que se establece una relación entre una noción informal y un método matemático. Este es precisamente el lugar que ocupa la tesis de Turing en la teoría de la computabilidad, puesto que dicho enunciado es justamente lo que afirma la llamada tesis de Turing. Así, pues, la tesis de Turing pertenece al ámbito de la teoría de la computabilidad. No obstante, esta tesis permite comprender conceptualmente las nociones mismas de máquina y de procedimiento mecánico. Por consiguiente, la tesis de Turing cae también bajo el campo del análisis filosófico, el cual es necesario para aclarar conceptos fundamentales relacionados con la computación y someterlos a crítica racional.

A estas definiciones abstractas hay que añadir el problema de la implementación, es decir, el problema de aclarar qué quiere decir que un sistema físico implemente un sistema de computación formal. Dicho de otro modo, cómo se explica la ejecución de una computación abstracta por parte de un sistema físico. La tesis de Turing física se formula de modo análogo estableciendo que toda función físicamente computable es Turing-computable.

En síntesis, nuestra propuesta principal consiste en defender que no existe un criterio de distinción conceptual entre la noción de computación física y la noción de computación abstracta. La relación entre la computación abstracta y lo que en la literatura actual se considera computación física reside en que la computación física es una idea intuitiva que sirve para defender la tesis matemática de Turing. La defensa de nuestra propuesta se apoya en dos argumentos principales. Primero, con el "argumento del colapso" incidimos sobre la condición del vehículo independiente del medio para mostrar que lo relevante para la computación física sigue siendo el nivel de la descripción abstracta. Lo importante para el mecanismo físico que computa es la manipulación del vehículo descrito a un nivel abstracto siguiendo una regla formal. Segundo, con la idea de "computadora humana" mostramos que dicha idea encaja en las condiciones de la computación física, que en la propuesta de Turing es un argumento intuitivo en favor de la tesis matemática de Turing, por lo que podemos interpretar que la relación de la computación física con la computación abstracta es la de operar como un argumento intuitivo en favor de la tesis matemática de Turing y no como un tipo de computación distinto al matemático. Nuestra propuesta secundaria consiste en discutir la posibilidad de una hipercomputadora relativista a partir del experimento mental de Andréka *et al*. (2018). No se trata de reproducir el resultado, sino de analizar la constructibilidad física a partir de la condición de fiabilidad de Piccinini (2015) y de causalidad local de Gandy (1980).

Para conseguir los objetivos antedichos estructuramos el artículo en las siguientes cuatro secciones. En primer lugar, realizamos un análisis de la Turing-computabilidad y establecemos el marco teórico de referencia, a saber, el de las máquinas de Turing.

En segundo lugar, examinamos el concepto de computación física a partir del problema de la implementación y la tesis de Turing física diferenciándola de su versión matemática. En tercer lugar, discutimos la tesis de Turing física a partir de la idea de la computadora humana y ofrecemos un planteamiento alternativo en lo concerniente a la relación entre la computación abstracta y la concreta. En cuarto lugar, analizamos un experimento mental (Andréka *et al*. 2018) sobre la viabilidad de una hipercomputadora física determinista y discreta que pueda sostenerse como contraejemplo a la tesis de Turing física.

2. Computación abstracta: Turing-computabilidad

2.1. Máquinas de Turing

El objetivo del trabajo con máquinas de Turing reside en definir la clase de funciones efectivamente computables. Para tales funciones hay un algoritmo mediante el cual podemos computar sus valores. La idea general consiste en que si existe un algoritmo, entonces es posible construir una máquina capaz de realizar esa tarea efectiva. Turing (1936, §2) establece que una máquina automática o máquina-a[1] es aquella cuyas acciones están determinadas cada una por un conjunto finito de condiciones (denominadas configuraciones-m).

Es posible dar una caracterización formal de tales objetos abstractos a los que se los denomina *máquinas de Turing*. Una máquina de Turing es capaz computar cualquier tarea efectiva, sin embargo, en un principio dicha capacidad se especificó para funciones numéricas, de ahí el título de su artículo seminal. El concepto de máquina de Turing, en definitiva, ofrece una noción formal y precisa de función computable. Decimos de esas funciones que son *Turing-computables*.[2] Una máquina de Turing es una abstracción que, para que tenga contenido intuitivo, se puede imaginar compuesta del siguiente soporte material y funciones:

1. Cinta de cálculo: cinta de longitud infinita (en las dos direcciones) dividida en celdas, las cuales son las unidades discretas en las que la cinta se divide.
2. Cabeza lectora: dispositivo capaz de leer el contenido de cada celda (de una en una).
3. Seguir y ejecutar instrucciones mediante las siguientes acciones:
    - La cabeza lectora puede escribir y borrar los símbolos de las celdas de tal modo que en las celdas puede haber 1 o 0 (o nada).
    - Moverse a la celda de la izquierda o a la celda de la derecha de la celda en la que se encuentra la cabeza lectora.

- La instrucción siguiente a cada paso del proceso de ejecución de la tarea depende del contenido de la celda que actualmente está leyendo la cabeza lectora.

Se trata de una idealización que simplifica en sus elementos mínimos la ejecución de una tarea con un conjunto finito de instrucciones. Lo fundamental en la computación es, por tanto, la descripción abstracta de la ejecución de la tarea. De este modo, ejecutar una tarea según determinadas instrucciones consiste en escribir y borrar los signos de las celdas y leerlos para concretar lo que se debe hacer a continuación. El número de configuraciones de la máquina es finito. Cada acción de la máquina está determinada por su correspondiente configuración y por los símbolos que en ese momento aparecen en la cinta. Realizamos las siguientes definiciones:

1. Una *instrucción* $i := \langle n, X, Y, m \rangle$, donde
   - n y m son números enteros positivos y $n \neq 0$. m es el estado siguiente que se ejecutará.
   - X puede ser 1 o 0. Este segundo símbolo es aquello que la cabeza lectora encuentra en una celda.
   - Y puede ser 1, 0, L o R. Este tercer símbolo indica lo que la cabeza lectora debe realizar en la celda en función del contenido que esta tenga: imprimir 1, borrar el contenido 0, desplazarse a la izquierda L o a la derecha R.[3]
2. Una *máquina de Turing* es un conjunto finito de instrucciones que cumple:
   - $\forall i i' (n = n' \wedge X = X' \supset i = i')$, lo cual expresa que no hay dos instrucciones cuyos dos primeros elementos sean los mismos.
   - Existe una instrucción para la que $m = 0$.
3. Un *estado* es el conjunto formado como máximo por las dos instrucciones tales que su primer símbolo es el mismo. Tal conjunto puede ser unitario o vacío. El concepto de estado es el núcleo de una máquina de Turing.[4]

Dadas las condiciones de contorno[5] y dada una máquina de Turing T, una función numérica se define como la función n-aria f tal que sus valores son los outputs de T cuando trabaja sobre cada input (n-tupla) satisfaciendo las condiciones de contorno. Si la máquina no se para o realiza una parada no estándar, entonces f es indefinida para el correspondiente input.

2.2. Máquinas universales

La definición de las máquinas de Turing supone un resultado teórico de gran importancia, puesto que describe formalmente la idea de computabilidad. Ahora bien, con dicha descripción se puede alcanzar un segundo resultado de forma sencilla. Este segundo resultado es la *enumerabilidad efectiva de las máquinas de Turing*. Que un conjunto sea efectivamente numerable quiere decir que se pueden listar con cierto orden los elementos que pertenecen a tal conjunto.

Una máquina de Turing se puede describir como la sucesión de instrucciones que la definen. Una máquina se define por un conjunto finito de instrucciones que podemos poner en una secuencia una tras otra. Cada instrucción es, a su vez, una sucesión de cuatro símbolos. Se denomina *palabra* al conjunto finito de instrucciones que definen a una máquina expuestas en una sucesión horizontal. Como puede observarse, el vocabulario V de las palabras es $\{0, L, R, n \in \omega\}$, siendo ω el conjunto de los números naturales y suponiendo que 0 no pertenece a ω. El conjunto V es enumerable ya que existe una función biyectiva h con los números naturales tal que:

- $h(0) = 1$
- $h(L) = 2$
- $h(R) = 3$
- Para cada $n \in \omega$, $h(n) = n + 3$

Por consiguiente, el número de máquinas no es mayor que el de los números naturales. Lo que se muestra es que si se aplica h a la *palabra* de una máquina "$h(P_T)$", entonces obtenemos una transcripción numérica de la máquina correspondiente. Para simplificar este número se puede emplear el teorema de factorización única, en virtud del cual obtenemos un único número primo para cada transcripción numérica de cada palabra o un único producto de números primos. Simplemente se descompone en las potencias de factores primos. Con todo ello conseguimos un código para cada máquina T representado por la función cod-$h(T)$. Con el concepto de función numérica computable y con la función cod-$h(T)$, se puede demostrar la enumerabilidad del conjunto de todas las máquinas de Turing y la enumerabilidad del conjunto de todas las funciones numéricas computables.

El siguiente gran resultado es el de la existencia de Máquinas Universales de Turing. La idea que expresa es aquella que entiende que pueden existir máquinas capaces de operar sobre los códigos de otras máquinas. Ahora bien, para poder expresarlo (y demostrarlo) necesitamos un elemento más en nuestro formalismo, a saber, los *índices*. Puesto que no todo entero positivo es por necesidad el código de la trascripción numérica de una máquina, necesitamos una condición de normalización

para conseguir una uniformidad. Al ser enumerable, los elementos del conjunto de las máquinas de Turing siguen el orden de los números naturales y con la normalización podemos concebir a cualquier entero positivo como la representación codificada de una máquina. Por consiguiente, tenemos una función de los códigos sobre los números naturales. Podemos identificar, por medio de esta asignación, a las máquinas y a las funciones. Lo expresamos del siguiente modo:

- T^i: el *índice* de una máquina T es i si el código normalizado de T es i.
- f^i: el *índice* de una función numérica computable f es i si f es una función numérica computable por la máquina T^i.

Con los índices podemos establecer la siguiente definición formal: una *Máquina Universal de Turing* es una máquina de Turing $U(x, y)$ tal que $U(x, y_1, \ldots,_n) = T^x(y_1, \ldots, y_n)$. Donde (x, y) es una $n + 1$-tupla de enteros positivos que desempeñan el rol de inputs. Dada la definición, U toma al primer entero de la tupla (x) como el código de una máquina cuyos inputs serían los restantes enteros de la tupla $(y_1, \ldots, y_n)$. Por tanto, el índice de T es el primer entero (x) de los inputs de U. De este modo, es claro que el valor de la máquina universal U con inputs $(x, y_1, \ldots, y_n)$ es el mismo que obtiene la máquina T^x con los inputs $(y_1, \ldots, y_n)$. Es importante aclarar que una máquina universal no opera sobre los símbolos de V, sino que está diseñada para imitar a otra máquina leyendo su código e interpretando el valor que indica la acción que debe realizar. Lo que hace, en definitiva, es incorporar la correspondencia entre los factores del código de la máquina a la que imita y las acciones que esa máquina puede realizar en su cinta.

3. Computación física: noción mecanicista de computación

3.1. El problema de la implementación

Piccinini[6] (2015, p.4) distingue claramente entre computación abstracta y computación concreta. La primera se refiere a sistemas formales de computación y la segunda se refiere a la computación de sistemas físicos. Como ejemplos de tales sistemas físicos concretos Piccinini (2015, p.4) pone el ejemplo de las computadoras y los cerebros: "computación en sistemas físicos concretos como computadoras y cerebros".[7] En este sentido, la computación concreta consiste en que un sistema físico implemente un sistema de computación formal como pueda ser una máquina de Turing. La explicación de lo antedicho comporta el denominado *problema de la implementación computacional*, esto es, explicar cómo un sistema físico ejecuta una computación abstracta. Piccinini (2015, p.6) lo expresa del siguiente modo: "¿Bajo qué condiciones un

sistema físico concreto realiza una computación cuando la computación está definida por un formalismo matemático abstracto?". La especificación de tales condiciones supone establecer una noción de computación concreta o física. El problema de la implementación puede formularse de tal modo que lo que se pretende esclarecer es el criterio de distinción entre un sistema físico que sea un sistema computacional o que realice computaciones[8] y un sistema físico que no. El problema de la implementación también concierne a qué computación realizan tales sistemas concretos.

El criterio que da Piccinini (2015, p.10 y capítulo 7) para solucionar el problema de la implementación es la noción mecanicista o mecánica de computación física (*The Mechanistic Account of Computation*, en adelante MAC) que define del siguiente modo:

> Un sistema de computación física es un mecanismo cuya función teleológica es realizar una computación física. Una computación física es la manipulación (por un mecanismo funcional) de un vehículo independiente del medio de acuerdo con una regla. Un vehículo independiente del medio es una variable física definida únicamente en términos de sus grados de libertad (por ejemplo, si su valor es 1 o 0 durante un intervalo de tiempo determinado), en contraposición a su composición física específica (por ejemplo, si es un voltaje y qué valores de voltaje corresponden a 1 o 0 durante un intervalo de tiempo determinado). Una regla es un mapeo de inputs y/o estados internos a estados internos y/o outputs.

Descomponemos las siguientes condiciones de MAC:[9] (i) un sistema computacional físico es un mecanismo con una función teleológica, (ii) es la función la que realiza la computación, (iii) la computación física consiste en la manipulación de un vehículo independiente del medio (*medium-independent vehicle*), (iv) la manipulación del vehículo sigue una regla, (v) el vehículo es una variable física y lo que define la variable son los valores que determina el grado de libertad del vehículo (y no su composición física) y (vi) la regla que sigue el vehículo es un mapeo o función del conjunto de inputs y/o de estados internos al conjunto de estados internos y/o de outputs.

Tal como el propio Piccinini (2015, p.8) observa, un mismo sistema físico puede satisfacer distintas descripciones abstractas. Las descripciones informativamente más económicas pueden expresar las mismas propiedades con menos información omitiendo determinados detalles del sistema en cuestión. Cuanta menos información, más abstracta es la descripción. Con esto simplemente nos interesa destacar un aspecto que, si bien es una perogrullada, será relevante en la discusión precisamente debido a que no siempre se tienen en consideración sus efectos. Este aspecto es la distinción clara entre una descripción abstracta de un sistema concreto y el mismo sistema concreto. Dicha consideración nos permitirá discutir las condiciones MAC(i)-(vi) a partir del argumento que ofrece Turing sobre la *computadora humana* para

apoyar su tesis. Las condiciones MAC(i)-(vi) deben diferenciarse de los argumentos de Turing ya que las primeras son relativas a la computación física y las segundas a la computación formal, y la manipulación es de un vehículo y simbólica, respectivamente. De lo contrario, las definiciones colapsarían.

Existen nociones de computación física rivales a MAC. Piccinini (2015, capítulos 2 y 3) argumenta contra la noción del mapeo (*mapping account*) y la noción semántica (*semantic account*). Ambas nociones admiten diversas formulaciones a su vez. A continuación, mencionamos solamente la noción del mapeo simple (*simple mapping account*) puesto que nos será de utilidad en la discusión. Reformulamos la definición de Piccinini (2015, p.17) del siguiente modo: el sistema físico S realiza la computación C si existe una aplicación $g : E_f \rightarrow E_c$, donde E_f es el conjunto de estados físicos que la *descripción física* de S atribuye a S y E_c es el conjunto de estados computacionales de C. g es tal que para toda transición de estado $e_1 \curvearrowright e_2$ entre elementos de E_c, si S está en el estado de E_f que mapea sobre e_1, entonces S pasa al estado de E_f que mapea sobre e_2. Esto quiere decir que las transiciones de estado de S reflejan (*mirror*) las transiciones de estado de C.

Al no estipular restricciones en la admisibilidad de los mapeos, la noción del mapeo simple cae en cierta arbitrariedad. En este sentido Piccinini (2015, p.18) afirma que "La noción del mapeo simple resulta ser muy liberal: atribuye muchas computaciones a muchos sistemas". Si admitimos la carencia absoluta de restricciones, entonces nos aproximamos a un pancomputacionalismo extremo según el cual cualquier sistema concreto implementa cualquier computación. Esta argumentación puede comprenderse como una reducción al absurdo en tanto que se estaría violando la asunción de objetividad[10] según la cual la computación física es una cuestión de hecho. Por el contrario, la arbitrariedad del establecimiento de un mapeo convierte a la implementación en una cuestión subjetiva relativa a quien establezca dicho mapeo entre cualquier sistema físico y cualquier computación. No hay un criterio empírico para determinar si un sistema físico realiza o no una computación. Así, pues, Piccinini (2015, p.19) expresa que "La noción del mapeo simple es inadecuada precisamente porque trivializa la afirmación de que un sistema físico realiza un cálculo".

Es importante destacar que para Piccinini (2015, p.19) la arbitrariedad supone un problema para esta noción de computación física y no para la existencia real de computaciones físicas. Esto se debe a que Piccinini (2015, pp.141-142) argumenta que MAC sí satisface el desiderátum de objetividad. A pesar de que un mismo componente pueda ser considerado como parte de distintos mecanismos según la perspectiva, una vez establecido el mecanismo que debe ser explicado el componente ofrece una respuesta objetiva a la explicación. Para MAC las propiedades funcionales de los componentes de los mecanismos son una cuestión de hecho, "Por ejemplo, algo es un registro de memoria o no" (Piccinini 2015, p.142).

3.2. Tesis de Turing física

Siguiendo a Piccinini (2015, capítulo 15), descomponemos la equivalencia de la tesis de Turing en dos relaciones de inclusión. Denomina tesis de Turing a aquella en virtud de la cual toda función intuitivamente computable es Turing-computable y tesis de Turing conversa a aquella que afirma que toda función Turing-computable es intuitivamente computable. Para Piccinini (2015, p.244) el argumento de la computadora humana (infra §4.2.1.) permite mostrar la conversa de la tesis de Turing. Lo que nos interesa destacar[11] es la distinción que Piccinini (2015, capítulo 15) asume en lo que respecta al extremo intuitivo de la tesis. En este sentido, entiende que "intuitivamente computable" puede especificarse en términos matemáticos-formales o en términos físicos. La idea intuitiva-matemática de computable es la de *procedimiento efectivo*. La idea intuitiva-física de computable es la de un *proceso físico* o la de una *función computable físicamente*. Por consiguiente, la tesis física de Turing puede formularse de los siguientes modos: todo proceso físico es un proceso Turing-computable o toda función físicamente computable es Turing-computable. A la primera formulación de la tesis física de Turing Piccinini (2015, p.246) la denomina audaz (*bold*) y a la segunda la llama modesta (*modest*). Las tesis conversas serían, respectivamente: toda función Turing-computable es computable por un procedimiento efectivo, todo proceso Turing-computable es físicamente realizable y toda función Turing-computable es físicamente computable. Esquematizamos el planteamiento de Piccinini de la siguiente manera:

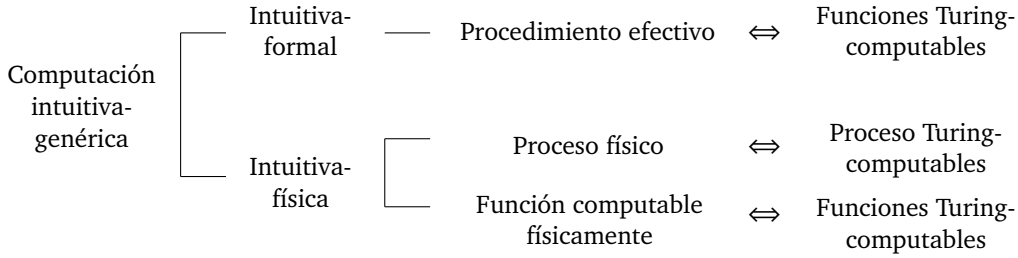

Tabla 1

3.2.1. Tesis de Turing matemática

Es menester señalar que la tesis matemática de Turing se desarrolló en el contexto teórico del programa formalista hilbertiano para la fundamentación de la matemática, en particular con el *Entscheidungsproblem* o problema de decisión. No se estableció para describir sistemas físicos. Dar una solución al *Entscheidungsproblem* era el objetivo teórico que se pretendía alcanzar a partir de los modelos formales de computación. El caso de las máquinas de Turing no era una excepción. El *Entschei-*

dungsproblem consiste en preguntarse si hay un algoritmo general capaz de decidir si cualquier fórmula del lenguaje de primer orden es un teorema o no. Una teoría es decidible sii hay un procedimiento efectivo con capacidad para determinar si una fórmula cualquiera es o no una fórmula de la teoría en un tiempo finito.

El hecho de que las funciones definidas a través de las máquinas de Turing sean efectivamente computables no es suficiente para demostrar que un conjunto es indecidible. Para ello se necesita la conversa, esto es, que las funciones efectivamente computables son aquellas que las máquinas de Turing define. Es una investigación próxima a los métodos de prueba, puesto que se trata de determinar si de Σ se puede derivar A en lógica de primer orden o si de Σ no se puede derivar A en lógica de primer orden, donde Σ es un conjunto finito de fórmulas que sirven de premisas en un lenguaje lógico de primer orden y A es una fórmula bien formada. El objetivo era construir un *procedimiento efectivo* con esta capacidad de decisión[12]. En el planteamiento de Piccinini (2015, *passim*) este procedimiento efectivo se corresponde con la idea intuitiva de computación formal.

Piccinini (2015, p.247) describe los procedimientos efectivos a partir de cuatro notas características. La ejecutabilidad (*Executability*), por la cual el procedimiento está determinado por un conjunto finito de instrucciones descritas a partir de un conjunto finito de símbolos y se llega al resultado en un número finito de pasos. La automaticidad (*Automaticity*), por la cual el procedimiento no exige agregar inventiva para ejecutar la tarea. La uniformidad (*Uniformity*), por la cual el procedimiento sigue una rutina de la misma manera para cada tarea. La fiabilidad (*Reliability*), por la cual al terminar la rutina el procedimiento arroja el output correcto. Se trata, en definitiva, de un algoritmo para realizar una tarea efectiva. En adelante nos referiremos a estas cuatro características simplemente como *características de los procedimientos efectivos*. Las máquinas de Turing pretenden cumplir estas condiciones para ser un procedimiento efectivo formal. Piccinini (2015, p.249) concibe la tesis de Turing matemática como una tesis epistemológica y la circunscribe a las funciones definidas sobre dominios enumerables: "La CT matemática se formula en términos de funciones definidas sobre dominios efectivamente enumerables. (...) es la noción de lo que se puede establecer siguiendo procedimientos efectivos definidos sobre dominios efectivamente enumerables".

3.2.2. Tesis de Turing física

La tesis de Turing física debe ser tal que sirva para la definición de computación física. La tesis de Turing matemática gira en torno a la noción de *procedimiento efectivo*, la cual noción era la idea intuitiva de computación en su versión formal, según el planteamiento de Piccinini antedicho. En el caso de la tesis de Turing física nos debemos ocupar de la noción de *proceso físico* en tanto que idea intuitiva de computación

en su versión física. Por consiguiente, la formulación de la tesis de Turing física se hace por analogía a la tesis matemática, esto es, trasladando las características de los procedimientos efectivos a los procesos físicos. La propuesta de Piccinini (2015, p.249) es formular esta idea en términos epistemológicos en forma de restricción a la noción de computación física. Lo que plantea es una restricción de usabilidad (*Usability Constraint*) en virtud de la cual para que un proceso físico pueda considerarse como una computación, este proceso debe poder ser usado por un observador finito para arrojar los outputs buscados de una determinada función. La formulación de Piccinini (2015, p.250) es la siguiente:

> **Restricción de usabilidad**: si un proceso físico es una computación, puede ser usado por un observador finito para obtener los valores deseados de una función.

Como puede observarse, la restricción de usabilidad tiene dos componentes que exigen ser explicados, a saber: la usabilidad y el concepto de observador finito. Por un lado, Piccinini (2015, p.250) admite dos definiciones de observador finito, una general y otra restringida. La general se refiere a "cualquier sistema organizado funcionalmente cuyo comportamiento está influenciado por una computación de manera apropiada". La restringida se ciñe a "los seres humanos y cualquier otro ser inteligente con capacidades limitadas de manera similar". Por otro lado, para explicar la noción de usabilidad, Piccinini (2015, pp.250-251) traslada las características de los procedimientos efectivos a los procesos físicos que computan, pero reformulándolos en términos de computación física. La característica de mayor importancia en lo concerniente a su traslación a los procesos físicos es la ejecutabilidad. La ejecutabilidad de un sistema físico exige que el observador finito en cuestión pueda iniciar su actividad dirigida a producir resultados legibles o arrojar los valores legibles de una función. En este sentido, el observador puede observar cuál es la función que el proceso físico computa, siendo legibles los inputs y los outputs del proceso físico. Además, el sistema que ejecuta el proceso debe poder construirse físicamente y debe permitir que la ejecución se repita.

Las características de uniformidad y automaticidad en los procesos físicos no cambian con respecto a los procedimientos efectivos. Respecto a la característica de la fiabilidad específicamente en los procesos físicos, hay que señalar que los sistemas construidos no pueden operar perpetuamente por lo que se requiere que arrojen resultados correctos al menos algunas de las veces. Ahora bien, tal como hemos presentado, Piccinini (2015, p.251) define la característica de la ejecutabilidad de los procesos físicos a partir de cinco condiciones: legibilidad de los inputs y outputs, regla independiente del proceso, cualidad de repetible, capacidad de asentamiento (*settability*) y que se pueda construir físicamente.

La condición de legibilidad la especifica Piccinini (2015, p.251) con relación a

la posibilidad de que un observador finito pueda usar una cantidad medible como input y como output. Además, el sistema físico debe tener un estado de parada claro. La condición de la regla independiente del proceso alude a que la tarea que debe ejecutarse sea independiente del proceso mismo de ejecución. La condición de repetibilidad hace referencia a la posibilidad de que un observador finito pueda repetir la concatenación de transiciones de estados del proceso y alcanzar el mismo resultado. La capacidad de ajuste es una cualidad concerniente a la posibilidad de que el usuario del sistema lo ajuste a su estado inicial e introduzca los argumentos pertinentes. La constructibilidad física es necesaria para que un sistema de computación física sea relevante respecto a la tesis de Turing física. Piccinini (2015, p.253) define esta condición del siguiente modo: "El sistema es físicamente construible en el sentido actual solo en el caso de que los materiales físicos pertinentes puedan organizarse para exhibir las propiedades pertinentes. Los materiales pueden incluir entidades que sean muy grandes, pequeñas o distantes, siempre que puedan hacerse para exhibir las propiedades pertinentes"[13]. Por último, la fiabilidad de los sistemas físicos de computación no puede hacer referencia a la imposibilidad de que el sistema se averíe y siempre funcione. No obstante, para que el sistema sea usable debe operar y arrojar los resultados correctos en un porcentaje alto. Además, las interferencias externas no deben generar problemas constantes. Como puede observarse, la característica de fiabilidad mantiene cierta ambigüedad e indefinición que se ajusta en la práctica del uso.

4. Discusión

En esta sección discutimos los criterios de distinción entre los conceptos analizados de computación física y matemática. Para ello confeccionamos dos argumentos. Por un lado, el argumento del colapso muestra que la condición MAC(iii) del vehículo independiente del medio pone de manifiesto (contra las pretensiones de Piccinini 2015) que lo relevante en la computación es la descripción abstracta de la tarea ejecutada. Por otro lado, mostramos cómo Turing emplea la idea de *computadora humana* como argumento intuitivo para fortalecer la tesis de Turing matemática y cómo esta misma idea encaja en la noción de computadora física. De modo que es razonable concebir la computación física como un argumento intuitivo en favor de la tesis de Turing matemática. Considerar la computación física como concepto independiente exigiría mayor justificación.

Primero, manteniendo la analogía de Piccinini entre los procedimientos efectivos y los procesos físicos, lo que obtenemos como resultado es que la usabilidad no añade nada a la tesis de Turing, de tal modo que la versión física y la matemática colapsan. Segundo, Piccinini (2015, p.250) afirma que un humano puede seguir un

procedimiento efectivo debido a las características de los procedimientos efectivos: "un observador finito puede seguir un procedimiento efectivo porque un procedimiento efectivo es ejecutable, automático, uniforme y fiable". En su comprensión epistemológica, las características de los procedimientos efectivos permiten al observador finito en cuestión conocer los valores de las funciones matemáticas definidas sobre dominios enumerables. Consideramos que esta interpretación no se adecúa a la tesis sostenida por Turing, dado que en su defensa de la tesis de Turing la figura de la computadora humana es un recurso explicativo para reforzar la equivalencia entre la idea formal de computación y la intuitiva. Tal como mostraremos, el objetivo que se tiene al emplear la figura de la computadora humana no es revelar lo que conoce quien manipula los símbolos, sino dar cuenta de la propia manipulación simbólica. Si el objetivo fuera revelar lo que conoce quien usa el procedimiento, entonces no se estaría argumentando en favor de la tesis de Turing, sino que se estaría abordando una tesis de otra índole.

4.1. Argumento del colapso

Tal como expusimos (supra §3.1.), las MAC(i)-(vi) constituyen la propuesta de Piccinini (2015) para explicar los sistemas que realizan computaciones concretas. Con esta propuesta pretende resolver el problema de la implementación. Es una comprensión de los sistemas computacionales concretos en tanto que mecanismos con funciones teleológicas. A Piccinini (2015, p.118) le interesa diferenciar su propuesta mecanicista de las nociones semánticas de computación concreta puesto que el vehículo computacional no involucra una representación y la individuación de las funciones computadas no depende del contenido semántico. El *leitmotiv* es "la computación no presupone representación" (Piccinini 2015, p.118). Ahora bien, para que un sistema sea un mecanismo funcional debe satisfacer las siguientes condiciones:

- El sistema es: (i) un conjunto de partes que lo componen, (ii) unas propiedades entendidas como poderes causales que posibilitan las funciones teleológicas del sistema y (iii) una organización.
- Las funciones del sistema dependen de la organización de las partes y las propiedades.

Ante esta definición, cabe hacer dos observaciones. Por un lado, la definición matemática a partir de las máquinas de Turing satisface las condiciones para ser un mecanismo funcional. Por este motivo Piccinini (2015, p.120) responde que la diferencia se encuentra en las propiedades que no son físicamente implementables y en la infalibilidad de su funcionamiento. Por otro lado, no todo sistema funcional es un mecanismo de computación. Esto exige especificar una noción de computación como

criterio para distinguir a los mecanismos de computación de los demás mecanismos funcionales. Por tanto, Piccinini (2015, pp.120-125) desarrolla una noción genérica de computación. La definición que ofrece es la siguiente:

> **Computación genérica:** el procesamiento de vehículos por un mecanismo funcional de acuerdo con reglas que son sensibles únicamente a las diferencias entre diferentes porciones (es decir, partes espaciotemporales) de los vehículos. (Piccinini 2015, p.121)

Así, pues, se denomina computación (en sentido genérico) a aquella función teleológica de un mecanismo funcional consistente en manipular vehículos, tal que dicho procesamiento sigue determinadas reglas que responden con exclusividad a las diferencias entre las partes de los vehículos en cuestión. Una regla de este tipo no es más que un mapeo del conjunto de inputs y/o del conjunto de estados internos al conjunto de outputs. Dado que tomamos como referencia la versión modesta de la tesis de Turing física y las MAC(i)-(vi), se sigue que un sistema físico de computación es un mecanismo funcional tal que alguna de sus funciones consiste en computar funciones matemáticas del conjunto de inputs al conjunto de outputs. Uno de los aspectos que nos interesa destacar para la discusión (y que Piccinini (2015, p.121) enfatiza) es que la función computada es una *descripción abstracta* del comportamiento del mecanismo. Lo cual quiere decir que su descripción abstracta es exactamente lo mismo que la regla que el mecanismo sigue para manipular sus vehículos. De hecho, la computación física es la misma manipulación de los vehículos siguiendo unas reglas determinadas. Por otro lado, es menester aclarar que Piccinini (2015, p.121) entiende por "vehículo" un estado con capacidad para cambiar de valor, esto es, una variable. Ahora bien, un vehículo está compuesto por partes espaciotemporales. El propio Piccinini (2015, p.121) propone el ejemplo de una cadena de dígitos: "una cadena de dígitos es un tipo de vehículo que está hecho de dígitos concatenados entre sí. (...). Normalmente, un dígito es un vehículo binario; puede tomar solo uno de dos estados".

Piccinini (2015, p.121) aclara que no debemos confundir el hecho de que un sistema siga una regla con el hecho de que el sistema represente dicha regla. Lo segundo no es necesario para que el mecanismo funcional compute. Para realizar la computación el sistema no necesita enumerar el conjunto de pares ordenados de inputs y outputs. En este sentido distingue entre seguir una regla y dar el algoritmo. Sin embargo, seguidamente Piccinini (2015, p.122) afirma que puede darse la regla estableciendo la relación entre el conjunto de inputs y outputs: "puede darse especificando la relación que se debería obtener entre inputs y outputs".

Frente a esto debemos recordar que en términos conjuntistas una función o mapeo es un tipo de relación o conjunto de pares ordenados tal que para todo elemento $x \in domR$ hay un único y tal que $\langle x, y \rangle \in R$, donde R es una relación y dom es

el dominio. Teniendo esto en consideración, desde nuestra perspectiva podemos argumentar en contra de la distinción tajante entre computación física y matemática, puesto que la noción de regla que sigue el mecanismo físico y la noción abstracta de algoritmo colapsan, ya que ambos son una función matemática. Dicho de otro modo, al seguir una regla se está proporcionando el algoritmo.

El procesamiento del vehículo es un ejercicio de manipulación por parte del mecanismo en tanto que consiste en transformar una parte del vehículo en otra parte: "El procesamiento o manipulación de un vehículo es cualquier transformación de una parte de un vehículo en otra" (Piccinini 2015, p.122). La computación que realiza el mecanismo es una función teleológica que consiste en manipular vehículos siguiendo determinadas reglas. Ahora bien, contrario a lo que pretende Piccinini, lo relevante en la computación física es la manipulación simbólico-abstracta, al igual que en el caso de la computación matemática. Mostramos esta condición a partir de una crítica interna de la propia caracterización de la computación física de Piccinini y no desde una concepción alternativa. En particular, traemos a colación la condición del vehículo de ser "independiente del medio" y la distinción entre descripción abstracta de un sistema concreto y el propio sistema concreto (supra §3.1.).

La condición MAC(iii) establece que el vehículo es independiente del medio (*medium-independent vehicle*). Esto quiere decir que la definición del vehículo en cuestión no depende del medio físico en el que se implementa la computación. Por tanto, se está apelando a la distinción entre la descripción abstracta del sistema físico y el propio sistema. Más aún, se necesita el nivel de la descripción abstracta para que el vehículo sea independiente del medio físico de implementación: "Las descripciones computacionales de sistemas físicos concretos son lo suficientemente abstractas como para ser independientes del medio" (Piccinini 2015, p.122). Como podemos observar, para la noción mecanicista de computación física, lo determinante sigue siendo el nivel de la descripción abstracta. Si no recurrimos a un nivel de abstracción entonces el vehículo no es independiente del medio, lo cual viola una de las condiciones para la computación física: si un proceso físico no es independiente del medio, entonces no hay computación alguna. Lo determinante para la computación física es la manipulación del vehículo descrito a un nivel abstracto de acuerdo a una regla también formal. La manipulación abstracta es precisamente lo que caracteriza la noción del procedimiento efectivo de la computación matemática. En este punto central ambas nociones colapsan en el nivel de la descripción abstracta del sistema físico.

La regla que sigue el sistema físico solo considera los valores que toma el vehículo, esto es, determinado tipo de trasformaciones de las partes de un vehículo en determinados intervalos de tiempo. Las restantes propiedades físicas no son consideradas por la regla. Esto permite la abstracción para que el vehículo sea independiente del medio de tal modo que sea posible implementar una misma computación en medios

físicos heterogéneos. Esta consideración remarca la relevancia del nivel abstracto-formal para la computación. Lo único que se requiere es que el medio tenga ciertas dimensiones de variación que la regla pueda considerar como valores de una variable. Lo relevante no son los diferentes mecanismos que realizan la computación, sino la computación misma que opera en el nivel de la descripción abstracta de los mecanismos. La realizabilidad en múltiples medios no es lo mismo que la independencia del medio. Piccinini (2015, p.123) exige la segunda propiedad, la cual es más fuere que la primera.

En definitiva, la noción genérica de computación no apela y no puede apelar al medio físico de implementación. La computación física definida en las MAC(i)-(vi) para solucionar el problema de la implementación también sitúa a la computación en el nivel formal de la descripción abstracta del sistema. No puede evitar que lo relevante para la computación siga siendo el concepto formal de procedimiento efectivo. El colapso lo observamos particularmente en las condiciones (iii) y (vi). Piccinini (2015, p.123) afirma que lo relevante para la computación "No dependen de propiedades físicas específicas del medio, sino solo de la presencia de grados de libertad relevantes". Es decir, depende de una noción abstracta de variabilidad independiente del medio físico de implementación.

4.2. La defensa de Turing de la tesis de Turing

Con la defensa de Turing de la tesis de Turing ponemos de manifiesto que la idea de computadora human cumple las condiciones de la computación concreta y, a su vez, opera como argumento intuitivo de la tesis matemática, de modo que no son nociones independientes. Lo que proponemos es una lectura de Turing en términos de la discusión de la computación física actual.

El objetivo de Turing con la máquina homónima es representar lo que hacen los humanos cuando realizan una tarea efectiva. Representa el análisis de la capacidad de resolver problemas. Se trata del procedimiento efectivo en la ejecución de una tarea. Turing (1936, §9) ofrece específicamente dos argumentos para defender que cualquier tarea efectiva puede ser realizada por una máquina de Turing. En un primer momento formula la tesis del siguiente modo: "Aún no se ha hecho ningún intento de mostrar que los números 'computables' incluyen todos los números que naturalmente se considerarían como computables" (Turing 1936, p.249), y él pretende llevar a cabo ese intento. Es importante considerar que la primera aparición de "computable" hace referencia a lo que puede computar una máquina de Turing y "lo que naturalmente se consideraría como computable" hace referencia a la noción intuitiva de computable. Esto quiere decir que la noción intuitiva o natural de lo que es computable está incluida en lo que es Turing-computable.

Como hemos mencionado, se trata de una tesis que pone en relación una no-

ción intuitiva con un concepto matemático, de tal modo que los argumentos que la sostienen no pueden ser más que de carácter intuitivo:[14] "Todos los argumentos que se puedan dar están destinados a ser, fundamentalmente, apelaciones a la intuición y, por esta razón, matemáticamente bastante insatisfactorios" (Turing 1936, p.249). Asimismo, comprende que para abordar el núcleo del problema con mayor rigor debemos centrarnos en los procesos mediante los cuales se puede computar: "La verdadera pregunta en cuestión es '¿Cuáles son los posibles procesos que pueden llevarse a cabo para computar un número?' ".

Dado este planteamiento, Turing ofrece tres tipos de argumentos: a) razones de carácter intuitivo, b) probar la equivalencia de dos definiciones, una de las cuales tiene mayor impacto intuitivo, y c) mostrar mediante casos concretos la amplitud de la cantidad de números que son Turing-computables. A continuación, analizamos los argumentos tipo a), b) y un tercer argumento que puede servir para poner en relación a los dos previos y mostrar su fuerza. A estos argumentos Turing los denomina I, II y III. Los argumentos tipo c) muestran el éxito de la tesis pero no hacen referencia al núcleo sistemático de la misma. Por este motivo nos centramos en los argumentos anteriores.[15]

4.2.1. Argumento I

Los argumentos del primer tipo hacen referencia a una serie de nociones intuitivas acerca de la capacidad humana de realizar tareas efectivas, a lo que se puede denominar *computadoras humanas*. Diferenciamos siete suposiciones acerca de las computadoras humanas en el texto, las cuales se ajustarían a aquello que realiza una máquina de Turing (1936, p.249-251):

1. Una computadora humana puede realizar su tarea de forma efectiva empleado una cinta unidimensional dividida en celdas, tal como lo hace una máquina de Turing. No es necesaria una hoja bidimensional.
2. Una computadora humana puede realizar su tarea de forma efectiva siendo capaz de reconocer sólo un número finito de símbolos diferentes.
3. Una computadora humana puede realizar su tarea de forma efectiva siendo capaz de leer al mismo tiempo un número finito de celdas de la cinta. De modo que para leer más celdas debe llevar a cabo observaciones distintas de forma sucesiva.
4. Una computadora humana puede realizar su tarea de forma efectiva pudiendo hacer lecturas sucesivas de celdas que se encuentren a una distancia limitada. Esto quiere decir que no caben los saltos ilimitados de una celda a otra de la cinta.

5. Una computadora humana puede realizar su tarea de forma efectiva cambiando el contenido de las celdas de una en una. Sólo puede cambiar el contenido de una celda a la vez, de manera que para cambiar varios debe llevar a cabo cambios diferentes de forma sucesiva.
6. Una computadora humana puede realizar su tarea de forma efectiva realizando las siguientes acciones simples: a) cambiar los símbolos de la celda que se observa y b) pasar de una celda a otra que se encuentre a una distancia determinada respecto a la celda leída.
7. Una computadora humana puede realizar su tarea de forma efectiva determinando sus acciones por los símbolos que va leyendo en las celdas. Se puede sustituir *salva veritate* los "estados mentales" de una computadora humana por "configuraciones de símbolos". Por consiguiente, a cada estado mental de la computadora humana le corresponde una particular configuración de una máquina automática. De este modo, el símbolo leído y la configuración determinan la operación de la computadora y su siguiente configuración.

Con esto se expresa que cualquier tarea efectiva que un humano pueda realizar, la puede hacer sin necesidad de más acciones que estas siete, las cuales las hace una máquina de Turing. Por lo tanto, la computabilidad humana es equivalente a lo que una máquina de Turing hace. Cualquier secuencia de símbolos que puede computar una computadora humana también la puede computar una máquina de Turing. En síntesis: por un lado, la idea intuitiva de computabilidad (o la capacidad humana de realizar una tarea efectiva o la computadora humana) puede describirse en las siete condiciones simples expuestas y, por otro lado, es posible construir una máquina (si hubiese una cinta infinita) que realice las operaciones descritas de la computadora humana. Luego, la máquina de Turing recoge la idea intuitiva y natural de computabilidad.

> Las máquinas que acabamos de describir [computadora humana] no difieren esencialmente de las máquinas de computación tal como se definen en el § 2 [máquinas de Turing], y correspondiente a cualquier máquina de este tipo se puede construir una máquina de computación para computar la misma secuencia, es decir, la secuencia computada por la computadora. (Turing 1936, p.252)

4.2.2. Argumento II

El segundo argumento consiste en observar que se puede construir una máquina de Turing capaz de probar cualquier fórmula que se pueda probar en un cálculo de primer orden.[16]

> Una vez que se acepta que los números computables son todos "computables", se deducen otras proposiciones del mismo carácter. En particular, se deduce que, si existe un proceso general para determinar si una fórmula del cálculo de funciones de Hilbert es demostrable, entonces la determinación puede ser realizada por una máquina. (Turing 1936, p. 249)

El método (Turing 1936, p.252) es sencillo:

1. s es una secuencia de símbolos.
2. $G_a(x)$ es una función que indica que el símbolo correspondiente a la posición x de la secuencia s es 1.
3. $\neg G_a(x)$ indica que el símbolo correspondiente a la posición x de la secuencia s es 0.
4. $N(x)$ indica que x es un entero no negativo.
5. $F(x, y)$ indica que$y = x + 1$.

El conjunto de propiedades de s se puede encontrar mediante las funciones presentadas, de tal modo que al juntarlas obtenemos una fórmula φ que define a s. φ incluye los axiomas de Peano P. Al establecer que φ define a s, se afirma que:

1. $\neg\varphi$ no es una fórmula que se pueda probar.
2. Para cada n se puede probar una de estas dos fórmulas:

$$A_n := \varphi \wedge F^n \supset G_a(i^n)$$
$$B_n := \varphi \wedge F^n \supset \neg G_a(i^n)$$

En las que $N(i)$ y $F^n := F(i^{i-1}, i^n)$.

Alterando una máquina de Turing T podemos obtener una máquina T_s que compute s. En ese caso se dice que la secuencia s es computable. Siguiendo a Turing (1936, pp.253), el funcionamiento de T_s es el siguiente:

1. Las acciones de T_s están divididas en secciones.
2. La sección n se encarga de encontrar el símbolo en la posición n de las secuencia s.
3. Tras acabar la sección $n - 1$ se imprime una marca detrás de cada uno de los símbolos.[17] A partir de entonces las acciones se realizan sólo en las celdas situadas a la derecha de ::.
4. En este momento T_s empieza a trabajar del mismo modo que T. Ahora bien, lo característico de T_s consiste en que cuando encuentra una fórmula que puede ser probada la máquina compara dicha fórmula con A_n y con B_n. Caben tres posibilidades:

- Si coincide con A_n, entonces imprime el símbolo 1 y finaliza la sección n.
- Si coincide con B_n, entonces imprime el símbolo 0 y finaliza la sección n.
- Si no coincide con A_n o B_n, entonces la máquina sigue trabajando hasta que encuentre una fórmula que coincida con A_n o B_n y dé fin a la sección n.

Dada la hipótesis de que φ define a s y el funcionamiento de T, la sección n va a finalizar en algún momento determinado. Esto quiere decir que la secuencia s es computable o que T_s no incurre en bucles.

Destacando el aspecto conceptual, debemos aclarar que el argumento II es importante puesto que muestra cómo una máquina en sentido ingenieril podría fabricarse (si hubiese cinta infinita) para probar los teoremas del cálculo lógico de primer orden. El nivel abstracto se encuentra en los teoremas y el nivel intuitivo en la máquina. El argumento II también tiene su relevancia en otro sentido, a saber: si con las máquinas de Turing podemos computar el conjunto de teoremas de la lógica de primer orden y consideramos (o Turing considera) que este sistema es el más adecuado a nuestro razonamiento, entonces tenemos un motivo sólido en favor de la tesis de Turing.

4.2.3. Argumento III

Turing ofrece un tercer argumento (que no debe confundirse con los argumentos tipo c)) que pone en relación a los dos anteriores dado que consiste en una modificación del argumento I o en un corolario del argumento II. En este caso Turing incide en evitar la noción de *estado mental* "al considerar una contraparte más física y definida de ella" (Turing 1936, p.253). Mantenemos la idea de que la tarea efectiva se realiza sobre una cinta, ahora bien, la computadora humana puede realizar su trabajo a partir de un conjunto de instrucciones apuntadas en una nota. Este conjunto de instrucciones es precisamente la contraparte bien definida de los estados mentales.

Asimismo, se mantiene la idea de que el trabajo de la computadora humana se puede simplificar realizando acciones de forma discreta, de tal modo que una instrucción le lleva a realizar una acción y a apuntar la siguiente instrucción. Así, pues, cada etapa en el avance en el proceso de la computación está determinada por un conjunto de instrucciones y por los símbolos que estén escritos en la cinta sobre la que se trabaja. Por consiguiente, podemos formular una secuencia de símbolos que describa un estado particular de la computadora. Dicha expresión es la fórmula de estado (*state formula*), la cual está compuesta, en este orden, por: los símbolos de la cinta, la letra Δ y una nota de instrucciones. Cada fórmula de estado está determinada por la fórmula de estado del estado anterior de la computadora. Además, podemos usar un cálculo de primer orden para expresar la relación que hay entre una fórmula de estado y su predecesora (o su sucesora).

Dada esta descripción, Turing afirma que se puede asumir que es posible axiomatizar en lógica de primer orden las reglas que dirigen el funcionamiento de la computadora. Lo cual se entiende en tanto que la relación entre dos fórmulas de estado se puede expresar en un lenguaje de primer orden. Finalmente, Turing (1936, p.254) infiere "Si esto es así, podemos construir una máquina para escribir las sucesivas fórmulas de estado y, por lo tanto, computar el número requerido". Esto sirve tanto para una computadora humana como para una máquina de computación. En definitiva, lo que se hace es "comparar a un hombre en el proceso de calcular un número real con una máquina que sólo es capaz de cumplir un número finito de condiciones" (Turing 1936, p.231). En los argumentos II y III le interesa remarcar su relación con el lenguaje lógico de primer orden para ofrecer una imagen más cercana a la idea intuitiva de computación.

4.3. Computación concreta y abstracta: planteamiento alternativo

Nos interesa destacar que las máquinas de Turing pueden presentarse como pertenecientes a un nivel físico (salvo por la cinta infinita) y su logro es precisamente "mecanizar" el razonamiento abstracto. Esto contrasta con la comprensión que tiene Piccinini de las máquinas de Turing como idea exclusiva del nivel matemático-abstracto. Es fácil observar que los siete rasgos característicos de la computadora humana del argumento I no son más que reformulaciones de las condiciones generales de un procedimiento efectivo. Se trata de formular en términos ordinarios lo que es una sucesión discreta y finita de pasos que conducen a una conclusión. No es más que la expresión informal de un procedimiento constructivo. Esta conexión se acentúa en el argumento III, puesto que se eliminan las referencias a los estados mentales y se ciñe a la manipulación simbólica.

Ante esto, destacamos dos cuestiones. Por un lado, recordamos que el propio formalismo de las máquinas de Turing son ya expresiones intuitivas ingenieriles de los procesos constructivos. Para comprobarlo, basta comparar el marco teórico de las máquinas de Turing con formalismos equivalentes como el cálculo lambda. Salvo por la cinta infinita, una máquina de Turing puede fabricarse. De hecho, gran parte de su éxito procede de esta cuestión. Por otro lado, la computadora humana es un sistema de computación física, a partir de lo cual afirmamos que la computación física puede comprenderse como una expresión informal de un procedimiento efectivo y no como una versión alternativa de una idea de computación más primitiva. Esto último no quiere decir que los sistemas físicos no puedan realizar computaciones, sino que lo relevante para la computación se encuentra en el terreno abstracto y algunos sistemas físicos tienen capacidad para seguir dicha abstracción.

La idea de la computadora humana no supone una concepción antropológica ni una afirmación sobre la naturaleza de los procesos cognitivos humanos que pretenda

identificarlos con definiciones en teoría de la computabilidad. No estamos tratando sobre limitaciones de facultades cognitivas humanas relacionadas con el cálculo. Antes bien, se trata de un modelo mínimo a partir del cual derivar y expresar intuitivamente cualquier procedimiento efectivo. El argumento muestra una equivalencia procedimental entre esta noción y la de máquina automática. Si la tesis de Turing es cierta, entonces podemos decir que una máquina de Turing explica qué hacen los humanos cuando realizan una tarea efectiva. No es una tesis sobre el funcionamiento de los procesos cognitivos de los seres humanos, sino que se trata de la definición de la ejecución de una tarea, sea esta física o matemática. La Turing-computabilidad es equivalente a la computadora humana, lo que quiere decir que también es relativa a fenómenos físicos.

Con ello queda clara nuestra perspectiva sobre la relación entre la computación abstracta y la concreta. Los argumentos de Turing encajan bajo la idea de computación concreta. Ahora bien, Turing los emplea para mostrar el carácter intuitivo de la Turing-computabilidad, de la cual no cabe duda de que es computación abstracta. Lo que pretendemos poner de manifiesto con la idea de la computadora humana de la defensa de Turing de la tesis homónima es que la noción mecanicista de computación física que propone Piccinini (2015) pertenece a un orden conceptual distinto al que el autor presupone: el mecanismo funcional no es una definición de la idea intuitiva de computación física, sino que es otra definición de la idea intuitiva de computación matemática, puesto que no puede escapar de referirse en último término a la idea de procedimiento efectivo abstracto (por la independencia del medio físico y la dependencia de un mapeo, tal como explicamos en §4.1.). El procedimiento efectivo es también de la descripción abstracta del sistema físico. La computadora humana sería otra definición de computación física, pero Turing (1936) la emplea para dotar de contenido intuitivo a la idea de procedimiento efectivo en general para defender su equivalencia con la máquina de Turing. Lo mismo ocurre con la definición de mecanismo funcional: dota de contenido intuitivo a la idea de procedimiento efectivo en general, por lo que refuerza la tesis matemática de Turing. Así, pues, frente al esquema presentado con anterioridad (supra §3.2., tabla 1), proponemos uno alternativo para esquematizar la relación entre la computación abstracta y la concreta (tabla 2).

El MAC como definición de computación física, dado el argumento del colapso, puede comprenderse como un argumento intuitivo más en favor de la equivalencia entre la Turing-computabilidad y la idea de procedimiento efectivo. Parte de la confusión puede deberse a que Piccinini (2015, p.254) considera que las instrucciones de una máquina de Turing son el procedimiento efectivo que determina las computaciones que realiza dicha máquina. Dada la tesis (matemática) de Turing, es cierto que podemos entenderlo de ese modo. No obstante, nuestro análisis inicial (supra §2.1.) muestra que una máquina de Turing se define precisamente como un conjunto finito de instrucciones. Para no caer en una tautología, lo que consideramos que

Computación intuitiva
$=$ $\Leftrightarrow$ Funciones Turing-computables
Procedimiento efectivo

$\uparrow$

Argumentos en favor de la tesis:
- Argumentos de Turing
- Resistencia al argumento diagonal
- Falta de contraejemplos
- Equivalencia de los diferentes formalismos para definir lo que es calculable de forma efectiva
- **Computación física**
- Etc.

Tabla 2

la tesis de Turing afirma es que para cada tarea efectiva puede establecerse un conjunto finito de instrucciones (que sigan el formalismo de las máquinas de Turing) tal que la computen. Consideramos que esta confusión es la que le lleva a malinterpretar el orden conceptual en el que se sitúa la computadora humana. Lo relevante para la computación no son las capacidades de un humano competente para seguir instrucciones, sino las propias instrucciones. La computadora humana es un procedimiento efectivo que realiza las tareas que una máquina de Turing ejecuta. La idea de computadora humana simplemente dota de contenido intuitivo para reforzar la tesis según la cual la Turing-computabilidad computa todo lo que puede ser calculado por procedimientos efectivos.

Bajo la perspectiva de Piccinini, Turing se mueve en el terreno exclusivo de la computación matemática. No obstante, en contra del planteamiento de Piccinini, lo que desde nuestra perspectiva nos interesa poner de manifiesto es (i) que la idea de computadora humana cumple la idea de computación concreta, (ii) el concepto de máquina de Turing es una noción matemática de computación y (iii) que, tal como muestra Turing (1936), ambas nociones son equivalentes. Entendemos que la tesis de Turing, tal como Turing la defiende, hace colapsar la computación concreta y la computación abstracta en la noción lógico-abstracta de algoritmo. Lo relevante tanto para la computación abstracta como la concreta es el procedimiento mismo o la descripción abstracta y no *quién* (computadora humana) o *qué* (máquina física) lleva a cabo la manipulación. En síntesis, consideramos que la crítica del colapso expuesta adquiere mayor robustez al analizar la defensa que realiza Turing (1936) de lo que hoy conocemos con el nombre de Tesis de Turing. Turing emplea recursos informales e intuitivos para apoyar la tesis según la cual todo procedimiento efectivo es Turing-computable. En este sentido la noción de *computadora humana* da contenido intuitivo al objeto abstracto de la máquina de Turing.

Por todo ello entendemos que la relación entre la noción de computación física y matemática consiste en que la física no es una noción independiente de la matemática, sino una idea intuitiva para fortalecer la tesis de Turing matemática, puesto que no puede probarse formalmente. Ahora bien, el conjunto de funciones que computa una computadora física depende de los límites de la caracterización de computación física. Si solo computa funciones Turing-computables, entonces se suscribe la tesis de Turing física. Esta segunda cuestión es la que analizaremos a continuación.

5. Contraejemplo de la tesis de Turing física

En esta sección nos proponemos analizar y discutir un contraejemplo particular de la tesis de Turing física en su versión modesta, a saber, la viabilidad de una hipercomputación relativista. Ahora bien, con carácter previo debemos aclarar en qué sentido la hipercomputación física supondría la existencia de funciones físicamente computables pero no Turing-computables. Esto exige un análisis del concepto de hipercomputación. Tomamos como paradigma de tarea no Turing-computable al *Entscheidungsproblem* (o problema de decisión[18]) que formulamos en términos del *problema de parada*,[19] dado que nos movemos dentro del marco teórico de las máquinas de Turing. En definitiva, lo que nos proponemos mostrar en esta sección es que, incluso si admitimos la noción de computación física que discutimos y rechazamos en la sección anterior (supra §4.), existe al menos un contraejemplo que convierte en falsa la tesis física de Turing. Esta tarea exige la constructibilidad física de una hipercomputadora. Piccinini (2015, capítulo 16) defiende que no existen tales contraejemplos. Nosotros, por el contrario, admitimos la propuesta de Andréka *et al*. (2018) sobre la posibilidad de una computación relativista que "presenta un desafío a la tesis de Church-Turing física".

5.1. Los límites de la computación física

El principal antecedente de la idea de computación física fue el desarrollado por Gandy (1980), quien generalizó la noción de mecanismo y probó su equivalencia con una máquina de Turing. Lo que resulta pertinente para nuestros objetivos es la caracterización que propuso de computación física para definir el extremo físico de la equivalencia. Se trata de una definición idealizada a partir de cuatro principios con el objetivo de mostrar que la clase de estas máquinas físicas de computación es una generalización que agota la clase de todas las máquinas físicas de computación. Esto demarca un límite claro a la idea de computación física, a saber, la Turing-computabilidad. Dicho de otro modo, la propuesta de Gandy impide la existencia de una hipercomputadora física, discreta y determinista. Como puede observarse,

la generalización de Gandy supuso el establecimiento de lo que en el debate actual denominamos tesis de Turing física.

Lo que Gandy (1980) pretendió probar es que no puede existir una máquina discreta y determinista que ejecute tareas no Turing-computables. Para ello tuvo que generalizar la idea de mecanismo y lo hizo con cuatro principios físicos idealizados. Dicha generalización es una caracterización de la computación física. La tesis de Gandy (1980, p.124) establece que "Lo que puede calcular una máquina es computable", donde "máquina" alude a su idea generalizada de computación física y "computable" hace referencia a la Turing-computabilidad. Como puede observarse, se trata de la tesis de Turing física (supra §3.2.). Gandy (1980, p.126) aclara que se refiere a computadoras digitales, a partir de lo cual Copeland y Shagrir (2007, p.218) formulan la tesis de Gandy del siguiente modo: "Cualquier función que pueda computarse mediante un dispositivo mecánico determinista discreto es Turing-computable". Se trata, por tanto, de la tesis de Turing física modesta y se refiere a máquinas discretas y deterministas. Gandy (1980) define a una computadora física, entendida como una generalización de un dispositivo mecánico discreto y determinista, a partir de cuatro principios[20]:

G1. Una máquina de Gandy[21] G es un par $\langle S, F\rangle$ donde:

1. S es un conjunto de estados.[22]
2. F es una operación de transición de estados de s_i a $s_j \in S$ tal que $F(s_0)$, $F(F(s_0))$, $F(F(F(s0)))$,... son los estados subsiguientes del estado inicial s_0.

G2. Existe un límite en la jerarquía del ensamblaje de estados. Todo estado está compuesto por partes que, a su vez, están compuestas por partes, pero esta construcción está limitada en tanto que los estados pertenecen a una sección[23] determinada inicialmente.

G3. Existe un límite en los tipos de partes de las que se componen los estados.

G4. Existe una causalidad local en las transiciones de estado. Una operación de transición realiza un reensamblaje de un estado cambiando partes. La causación local establece que las partes con las que se reensambla F(x) dependen de las partes de x.

Gandy (1980, p.126) (i) muestra que un dispositivo mecánico discreto y determinista (*discrete deterministic mechanical device*) satisface los principios G1-4 y (ii) prueba que lo que computa una máquina de Gandy[24] (dispositivo caracterizado por G1-4) es Turing-computable. De lo que se sigue que (iii) lo que computa un dispositivo mecánico discreto y determinista es Turing-computable.

Ante este planteamiento, Copeland y Shagrir (2007) ofrecieron ejemplos de máquinas físicas que exceden los límites establecidos por Gandy (1980). Copeland y Shagrir (2007) mostraron la posibilidad de máquinas físicas no deterministas y no discretas que computan funciones no Turing-computables. Además, discutieron la

posibilidad de máquinas que computan funciones no recursivas realizando supertareas[25] tanto en física newtoniana como en un espacio-tiempo relativista.[26] A este respecto, se considera la propuesta de Davies (2001) sobre la posibilidad de construir en un contexto newtoniano una serie infinita de máquinas que no necesita superar la velocidad de la luz en los cambios de las transiciones de estado. Sin embargo, el requerimiento que exige es un universo continuo que no tenga un límite inferior en el tamaño de sus componentes que al dividirse conserven sus propiedades, puesto que cada máquina de la serie es más pequeña que su predecesora.[27]

A continuación, llevamos a cabo una discusión análoga a la de las máquinas de Gandy y los contraejemplos de Copeland y Shagrir (2007), pero empleando el modelo de Piccinini (2015) y el experimento mental de Andréka *et al*. (2018). Esta discusión es relevante por dos motivos. Por un lado, las condiciones MAC(i)-(vi) de Piccinini (2015) constituyen una generalización más amplia en la definición de la computación física que la de los principios de Gandy. En este sentido, la propuesta de Piccinini es la caracterización más relevante en la literatura actual y la definición más general para dar cuenta de la computación física. Asimismo, el experimento mental de Andréka *et al*. (2018) permite pensar un contraejemplo más interesante de la tesis de Turing física en tanto que muestra una manera más detallada de construir una máquina física que compute funciones no recursivas sin renunciar al carácter determinista y discreto de la misma.

La versión audaz de la tesis física de Turing no es una buena candidata para establecerse como la tesis física análoga a la tesis matemática de Turing. Piccinini (2015, capítulo 15.3.) argumenta que existen procesos propios de la versión audaz que no satisfacen la restricción de usabilidad. Lo cual quiere decir que existen contraejemplos que falsan la tesis física de Turing en su versión audaz: todo proceso físico es Turing-computable.[28] En definitiva, se pone de manifiesto que la noción de computación física involucrada en la versión audaz no es usable. Esto se debe precisamente a su audacia: al ranguear sobre todo proceso físico, lo que admite es que todo proceso físico es una computación.[29] Dada esta circunstancia, Piccinini (2015, capítulo 16.) defiende que la tesis física adecuada a la tesis matemática de Turing es la versión modesta de la tesis física homónima: toda función físicamente computable es Turing-computable. Sin embargo, la viabilidad de la hipercomputación física supondría un contraejemplo que falsaría la versión modesta de la tesis física de Turing ya que existirían funciones físicamente computables que no son Turing-computables.

5.2. Hipercomputabilidad relativista

Haciendo abstracción de los antecedentes y de la trayectoria[30] de los modelos abstractos de hipercomputación, nos centramos en el concepto de Máquinas Aceleradas de Turing (*Accelerating Turing Machines*) de Copeland (2002 y 2004). La rutina que

ejecuta una máquina está compuesta de pasos finitos, lo que en términos temporales exige que su ejecución completa se dé en un tiempo finito. La idea básica de las máquinas aceleradas consiste en que cada paso de la máquina se ejecuta en un lapso de tiempo que dura la mitad de la duración del paso anterior. El sumatorio de esta sucesión "$1/2^n + 1/2^{n+1} \ldots$" nunca llega a la unidad de tiempo. El resultado de la aceleración es una máquina de Turing capaz de realizar una tarea de infinitos pasos discretos en un tiempo finito (o *supertarea*). El estado final es un *límite* matemático. Ahora debemos pasar de este modelo matemático de hipercomputación a la viabilidad de la construcción de hipercomputadoras físicas.

Para Andréka *et al*. (2018, p.195) una función es físicamente realizable si puede idearse un experimento mental en el que se compute dicha función sin violar el conocimiento actual que tenemos de las leyes físicas: "En términos generales, [la función] f es físicamente realizable si hay un experimento mental físico en el que f es computada de acuerdo con las leyes actuales de la naturaleza". Apoyándose en Hogarth (1994)[31] y en Etesi y Németi (2002), Andréka *et al*. (2018, p.196) diseñan un experimento mental en el que se construye físicamente una computadora que opera en condiciones de un espacio-tiempo relativista general tal que puede decidir todo conjunto recursivamente enumerable. Esto quiere decir que la máquina relativista puede computar tareas que una máquina de Turing en términos clásicos no puede computar (paradigmáticamente, el problema de parada). La computadora relativista que Andréka *et al*. (2018, p.196) pretenden construir en el experimento mental está compuesta de:

1. Una máquina de Turing con capacidad para trabajar infinitamente.
2. Un programador con capacidad para extraer en tiempo finito los outputs que arroja la máquina de Turing.

La estrategia fundamental está ligada a la idea de la máquina acelerada, que si bien no es físicamente construible, con el experimento mental de Andréka *et al*. (2018) nos podemos aproximar a su funcionamiento.[32] La estrategia consiste en dos momentos: (i) el programador programa la máquina de Turing de modo que están juntos y (ii) el programador examina los resultados de la máquina de forma independiente y alejada de ella en un tiempo finito. En el espacio y el tiempo clásicos esto no es viable puesto que la tarea que la máquina de Turing ejecuta es infinita y requiere el trabajo infinito de la propia máquina. El programador es el observador finito de la restricción de usabilidad y debe examinar los outputs que la máquina arroja tras su trabajo infinito. Esto quiere decir que el envío de los resultados al observador debe llegar antes del segundo momento en el que el propio observador los evalúa. Por tanto, el segundo momento es posterior al tiempo infinito de la ejecución de la tarea por parte de la máquina. Para que sea viable Andréka *et al*. (2018) proponen que la

computadora relativista funcione en un espacio-tiempo relativista general.

El fenómeno fundamental para el funcionamiento de la computadora relativista es el efecto de la dilatación gravitacional del tiempo en el campo gravitatorio de objetos masivos.[33] Este efecto hace que el tiempo transcurra más lento en regiones en potenciales gravitatorios más bajos. En síntesis, la gravedad hace que los relojes se ralenticen. Con este efecto se puede conseguir que el tiempo del trabajo de la máquina de Turing vaya más rápido que el del observador. Al aproximarse al horizonte de eventos de un agujero negro la atracción gravitatoria se acerca a infinito. Lo que Andréka *et al*. (2018, p.200) proponen es una aceleración infinita del tiempo de trabajo de la máquina de Turing mediante el acercamiento progresivo del observador al horizonte de eventos. De este modo, tras un lapso de tiempo en la máquina, el observador se acerca al horizonte de eventos hasta determinado punto en el que su tiempo es la mitad de rápido que el de la máquina, y así sucesivamente en cada paso. Lo que obtenemos es una máquina acelerada que permite completar la rutina infinita de la máquina en un tiempo finito para el observador.

Como puede observarse, la exigencia fundamental del experimento mental reside en la posibilidad de que exista un momento posterior a un tiempo infinito, puesto que solo así el observador puede leer los resultados de la rutina infinita de la máquina. Desde una perspectiva clásica nos encontramos ante una representación contraintuitiva. Earman (1995) denominó "evento Malament-Hogarth"[34] a este tipo de eventos que contienen una línea del universo con futuro infinito en su pasado causal. Para la hipercomputadora relativista se necesita que "El pasado causal de un evento E es el conjunto de todos los eventos O tales que existe una curva causal con O y E en ella tal que O es anterior a E en la curva" (Andréka *et al*. 2018, p.206), donde E es el evento en el que el programador lee los resultados del trabajo infinito de la máquina y O es el evento en el que el programador programa la maquina (está junto a ella).

A continuación, reconstruimos la definición[35] de hipercomputadora relativista que establecieron Andréka *et al*. (2018, p.206) de tal manera que cumpla las antedichas condiciones. Una hipercomputadora relativista es una tupla $\langle M, g, \gamma_C, \gamma_P, O, E\rangle$, donde:

1. O y E son los eventos previamente mencionados.
2. $\langle M, g\rangle$ es una estructura espaciotemporal,[36] donde:
    a) M es un conjunto de eventos para el que hay un sistema de funciones inyectivas $f : M_i \to R \times R \times R \times R$, donde R es el conjunto de números reales.
    b) g es un tensor métrico sobre M. Con g se determina qué subconjuntos de M son localmente transitables[37] ("viajables") en cada evento por un observador y por los fotones.[38] Esto permite el envío de información y

se dice que son causales.

3. γ_C es una línea de universo infinita que tiene a O como elemento y se encuentra en el pasado causal de E.
4. γ_P es una línea de universo que tiene a O y a E como elementos y en la que el evento E es posterior a O.

Siguiendo la caracterización de Andréka *et al*. (2018, p.205) y enfocándonos solo en los aspectos pertinentes, tenemos que el conjunto de todos los eventos en los que está presente un cuerpo u observador es su línea de universo; y un espaciotiempo es un conjunto estructurado de eventos. Esto quiere decir que una línea de universo (de un observador) es un subconjunto del espaciotiempo en cuestión. Lo que define las líneas de universo posibles de los observadores es la estructura espaciotemporal. En definitiva, el evento E es un evento Malament-Hogarth.

5.3. Constructibilidad

Copeland y Shagrir (2007, p.220) aclaran que es una cuestión empírica abierta el problema de resolver si sus ejemplos ideales de computadoras físicas que computan funciones no Turing-computables es viable dado el conocimiento actual de la física. Lo que nosotros consideramos es que el experimento mental de Andréka *et al*. (2018) es un ejemplo de hipercomputadora determinista viable debido a los agujeros negros de Kerr. Los criterios de constructibilidad que empleamos son el principio G4 de causalidad local que tomamos de Gandy (1980) y la ejecutabilidad y fiabilidad de los sistemas físicos de computación refiriéndonos principalmente a la restricción de usabilidad que tomamos de Piccinini (2015).

En lo que respecta a la constructibilidad, es necesario distinguir entre dificultades no superables por principios físicos y dificultades no superables por el estado actual de la tecnología. Esto quiere decir que basta con un experimento mental que no viole el conocimiento actual de las leyes físicas. Las críticas relativas a nuestra imposibilidad actual de alcanzar un agujero negro en rotación no se aplican. Lo relevante en este caso es el contexto físico en el que opera el conjunto formado por la máquina y el programador. A este respecto, lo relevante reside en que el tiempo que el observador tarda en llegar al horizonte de eventos interno es finito, pero el tiempo que transcurre para la máquina de Turing que envía las señales es infinito. Además, lo que se usa para ralentizar la caída del observador es el propio efecto repelente del horizonte de eventos interno.[39]

El principio G4 es el más relevante para nuestro análisis de la constructibilidad física. Gandy (1980, p.126) explica que el principio de la causación local incluye dos presuposiciones físicas. Primero, existe un límite inferior en el tamaño de las partes

atómicas de la máquina. Segundo, existe un límite superior en la velocidad de los cambios de las transiciones de estado, el cual es la velocidad de la luz. Por ejemplo, la propuesta de Davies (2001) no transgrede el segundo límite, pero sí el primero. El propio Davies (2001, p.680) aclara que "nuestro universo no es un universo newtoniano continuo, y sólo hemos dado un esbozo de cómo construir una máquina de este tipo".

Ahora bien, en lo que respecta al experimento mental de Andréka *et al*. (2018), el primer límite es ajeno a la hipercomputadora $\langle M, g, \gamma_C, \gamma_P, O, E\rangle$. El segundo límite no es transgredido por la existencia de una línea de universo con futuro infinito en su pasado causal ya que el tensor métrico g sobre M determina los subconjuntos localmente transitables. En cuanto a la restricción de usabilidad, el programador es el observador finito que debe poder leer los resultados que arroja la máquina. Lo que debemos evitar es, por tanto, la destrucción del usuario antes de que se puedan conocer los resultados del proceso. Esta condición es la que Piccinini (2015, p.255) denomina fiabilidad. La constructibilidad física y la fiabilidad son las condiciones que se ponen en cuestión en lo que respecta a la ejecutabilidad del sistema físico de computación que estamos tratando.[40] Es importante destacar que estas dos condiciones son precisamente aquellas que convierten a un sistema en usable en la práctica. La constructibilidad física y la fiabilidad son las que permiten pasar de un mero sistema de notación para computar funciones a un sistema físico usable para realizar computaciones. Esto quiere decir que la discusión sobre la viabilidad de la hipercomputabilidad física es relevante para la tesis física de Turing tanto como contraejemplo como en lo relativo a las características esenciales de los sistemas físicos de computación en general.

El problema de la fiabilidad se puede resolver en el experimento mental de Andréka *et al*. (2018) con el funcionamiento de la computadora en un agujero negro en rotación. Tal como observan Andréka *et al*. (2018, p.200), por un lado, para que sea usable la máquina en un contexto de relatividad especial se necesita una aceleración ilimitada del observador lo cual provocaría su muerte y, por otro lado, en un agujero negro de Schwarzschild (o agujero negro estático) el observador se encuentra en la siguiente disyuntiva: o bien la atracción gravitatoria lo destruye, o bien la atracción gravitatoria impide que le lleguen las señales que la máquina le envía. Por este motivo Andréka *et al*. (2018, p.200) proponen un agujero negro de Kerr en rotación lenta.[41] Un agujero negro de Kerr tiene dos horizontes de eventos. El primero se encuentra en el interior del segundo. La hipersuperficie externa es la propia de la atracción gravitacional y la hipersuperficie interna es la concerniente al efecto repelente de la rotación del agujero negro tal que es superior a la atracción gravitacional. Esto es lo que permitiría (en el experimento mental) que el observador se introduzca en la esfera exterior y recibir la totalidad de los resultados del trabajo infinito que envía la máquina de Turing antes de que (el observador) se introduzca en la esfera interior.

Respecto al determinismo, seguimos la distinción que realizaron Copeland y Shagrir (2007, pp.227-228) entre determinismo y Gandy-determinismo. Dado el funcionamiento de las máquinas de Gandy los procesos con pasos infinitos no paran. Por tanto, siguiendo G1, el *Gandy-determinismo* exige que el estado de cada paso del proceso esté determinado de forma exclusiva por la configuración del estado del paso inmediatamente anterior. El *determinismo*, sin embargo, solo exige que la descripción del estado inicial de la máquina determine el trabajo de la máquina. Si empleamos este sentido de determinismo (que debilita G1), entonces se pueden construir hipercomputadoras deterministas. Copeland y Shagrir (2007, p.229) concluyen que "la noción estrecha de determinismo de Gandy falla en incluir algunos sistemas físicos que normalmente se diría que obedecen leyes físicas deterministas". Ahora bien, un espacio-tiempo de un agujero negro de Kerr permite pensar una máquina que satisfaga la restricción de usabilidad.

Por último, cabe mencionar que Gandy (1980, p.130) defiende sus principios afirmando que, si estos se debilitan, entonces cualquier función es computable. Lo anterior parece una trivialización de la teoría de la computabilidad, debido a lo cual no se deben debilitar los principios de Gandy. Con ello se está defendiendo los límites de la computación física. Contra esta idea, Copeland y Shagrir (2007, p.229) argumentan que no se incurre en trivialidad si se aclara que la computabilidad de una función es relativa a un índice que remite a las capacidades y recursos de la máquina en cuestión. En este sentido se puede decir que una función no es computable por una máquina de Turing, pero sí lo es por otro tipo de máquina. Podemos establecer un argumento análogo respecto al pancomputacionalismo, a saber, no se incurre en trivialidad si se especifican los límites de las computaciones que puede realizar cada sistema físico.

Emparentada con la observación crítica sobre los límites, encontramos la crítica de Davis (2006) según la cual no puede existir la hipercomputación "porque todo lo que experimentamos es finito" (Davis 2006, p.4). Davis remarca que se trataría de un "milagro" en el modelo cuántico, sin embargo, admite que en el contexto relativista general un evento con una línea de universo con futuro infinito en su pasado causal puede observarse en el universo actual (por eventos Malament-Hogarth). El problema para Davis (2006, pp.5-6) se encuentra en este caso en la cuestión sobre la posibilidad de una máquina que aproveche este fenómeno para realizar supertareas. Esta cuestión es precisamente la que hemos pretendido responder en esta sección con el problema de la constructibilidad entendida a partir de la condición de la ejecutabilidad y el principio de causación local.

6. Conclusión

Dados los resultados principales obtenidos en las discusiones anteriores (§4 y §5), extraemos dos conclusiones. En primer lugar, argumentamos contra la independencia de la noción de computación física en el siguiente sentido: la noción de computación física no es una definición teórica de la idea intuitiva de computación efectiva alternativa a la definición matemática. La idea de computación física la comprendemos no como una versión distinta a la versión matemática, sino como uno de los posibles argumentos en favor de la tesis de Turing. Esto lo mostramos mediante dos argumentos. Por un lado, poniendo de manifiesto su equivalencia con la idea de la computadora humana. Las versiones matemática y concreta de computación no son dos formas de explicar la idea intuitiva de computación efectiva, sino que la equivalencia entre la idea intuitiva de computación efectiva y la definición matemática de la Turing-computabilidad puede defenderse a partir de diversos motivos intuitivos entre los que se encuentra la noción de computación física.

Es revelador que la generalización de Gandy (1980) sea una formulación de una computación física y, a su vez, constituya una generalización de la noción de computación abstracta de Turing. Esto se debe a que la definición del propio Turing (1936) es de hecho ingenieril, pero lo que destaca es la descripción abstracta del proceso. En el marco conceptual de las máquinas de Turing lo central es el análisis de los procesos simbólicos, al igual que con la λ-definibilidad. Ahora bien, lo que se añade en el modelo de Turing es que dicho análisis debe poder realizarlo un computador. Turing propone la idea de la computadora humana y Gandy generaliza las condiciones de esa idea con sus principios para dispositivos mecánicos deterministas. Lo que queremos clarificar es que denominar a la segunda como "computación física" no quiere decir que se trate de una noción independiente de la noción formal. Lo que pretendemos poner de manifiesto es que la computación consiste en analizar procesos simbólicos. El núcleo conceptual de la computación consiste en analizar operaciones mecánicas sobre configuraciones simbólicas, sean realizadas dichas operaciones por una computadora humana o un dispositivo mecánico determinista.[42] La diferencia técnica —que no conceptual— estriba en que las máquinas de Gandy permiten computaciones paralelas.

Por otro lado, el colapso entre ambas versiones no quiere decir que los sistemas físicos no realicen computaciones, sino que computar es un procedimiento abstracto que determinados sistemas físicos pueden seguir del mismo modo que una computadora humana. La implementación se lleva a cabo al seguir una regla en la manipulación de un vehículo, pero lo relevante para la computación se encuentra en la descripción abstracta del sistema físico, puesto que el vehículo debe ser independiente del medio para computar una función. La implementación no es un problema añadido que se tenga que resolver con una nueva definición de computación física.

Un sistema físico como las computadoras digitales puede realizar manipulaciones simbólicas siguiendo un algoritmo del mismo modo que lo hace una computadora humana.

En segundo lugar, incluso mantenido la especificidad de la tesis de Turing física se pueden encontrar contraejemplos que muestren el fallo de la tesis. Para elaborar un contraejemplo se necesita mostrar la constructibilidad física de una hipercomputadora capaz de computar funciones no Turing-computables. El punto central consistió en mostrar que el experimento mental de la hipercomputadora relativista de Andréka *et al*. (2018) resiste (i) a las condiciones de la ejectuabilidad de los sistemas físicos de computación, (ii) a la condición de causación local y (iii) es un contraejemplo determinista. Esta computadora es capaz de calcular efectivamente tareas como el problema de parada (que tomamos como prototipo de función no Turing-computable). Lo cual se debe a que se aproxima a la realizabilidad física de una máquina acelerada gracias a la posibilidad de eventos que contienen una línea de universo con futuro infinito en su pasado causal en un espaciotiempo relativista general de agujero negro de Kerr en rotación lenta.

La idea de computación física estandarizada en la literatura se concibe como siendo independiente de la versión matemática y posteriormente se discute su relación mediante la tesis de Turing física. Copeland y Shagrir (2007, p.230) consideran que los ejemplos de computadoras físicas que computan funciones no Turing-computables muestran que la computación física es distinta de la matemática. Debemos hacer explícitos los presupuestos de esta afirmación. Primero, que las máquinas de Gandy siguen el modelo de la computación matemática al generalizar el modelo de Turing y al ceñirse a la Turing-computabilidad. Segundo, que los principios de Gandy no describen la clase de todas las máquinas físicas de computación, puesto que existen contraejemplos. Tercero, que al existir máquinas físicas no Gandy-deterministas y no discretas, la computación física es distinta de la computación abstracta puesto que la primera podría computar funciones que la segunda no. Sin embargo, habría un error en este razonamiento, a saber: también existen hipercomputadoras abstractas. El propio Copeland propuso las máquinas aceleradas[43] y Copeland y Shagrir (2011) mostraron que una máquina acelerada de Turing puede computar la función de parada en un sentido externo si se mantiene una concepción realista de las máquinas de Turing.

En definitiva, lo que proponemos es que una máquina de Turing es ya una descripción de un proceso mecánico, pero lo relevante para la computación es el procedimiento abstracto de la ejecución de la tarea. Gandy generalizó esta noción lógico-matemática de proceso mecánico y debemos tener en cuenta que la causación local procede del propio modelo de Turing. En ambos ejemplos se realiza una tarea abstracta. Esto se puede ver de manera clara en las críticas realizadas a la noción mecanicista de Piccinini (2015). Dicha generalización supuso un cambio de la idea de la

computadora humana a la de una máquina física. Pero ambos son referencias intuitivas para fortalecer la tesis de Turing. Lo que mostramos con el argumento del colapso y la idea de la computadora humana es que lo relevante para la computación es la ejecución de la tarea, la cual es una operación lógico-matemática. Nuestra conclusión es, por tanto, la siguiente: las máquinas físicas, sean estas computadoras humanas o dispositivos mecánicos, ejecutan procedimientos efectivos los cuales son operaciones simbólico-formales. La computación concierne a la descripción abstracta de estas operaciones.

Apéndice: el problema de parada

Mostramos cómo una máquina de Turing no puede calcular determinadas tareas. Nos centramos en el problema de parada. En términos informales, el resultado negativo al problema de parada quiere decir que no existe un procedimiento efectivo capaz de determinar si una tarea efectiva termina o no en tiempo finito. Hay tareas respecto a cuyos valores no se puede saber si a) existe un final de la tarea pero no se ha llegado a él aún o si b) no existe un final de la tarea y la rutina no acabará nunca. Turing (1936, §11) demostró que no puede existir un proceso general para decidir si una fórmula es o no un teorema de lógica de primer orden, mediante la demostración de que no puede existir una máquina tal que teniendo a una fórmula como dato de entrada pueda determinar si se puede probar o no. Como mostraremos a continuación, se trata de una prueba de la respuesta negativa al *Entscheidungsproblem*.

El teorema de la existencia de máquinas universales de Turing (supra §2.2.) pone de manifiesto que realizar una tarea efectiva está intrínsecamente ligado a la autorreferencialidad o recursión, esto es, un proceso que se basa en su misma definición. Teniendo esto en consideración y que las máquinas de Turing calculan las funciones numéricas, nos topamos con el método de *diagonalización*[44] al cuestionarnos cuántas funciones de este tipo definidas sobre los números naturales existen. Para simplificar el esquema del método sólo consideramos las funciones totalmente definidas, esto es, las funciones cuyos dominios no tienen elementos que no tengan un valor de llegada. Procedemos por reducción al absurdo y suponemos que existe un número infinito de funciones totales que no sea mayor al conjunto de números naturales. Esto quiere decir que partimos de la hipótesis de que el conjunto de las funciones numéricas totales es infinito enumerable. Ahora necesitamos representar la clase F de funciones numéricas totales (con un argumento) definidas sobre los números naturales en una *matriz de diagonalización*. Siguiendo el análisis de Alonso (2004, p.36) de la aplicación del proceso diagonal que realiza Turing (1936, §8), definimos la matriz D^F:

	0	1	2	3	4	… n
f_0	X_{00}	X_{01}	X_{02}	X_{03}	X_{04}	…
f_1	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	…
…	…	…	…	…	…	…
f_i	X_{i0}	X_{i1}	X_{i2}	X_{i3}	X_{i4}	…
…	…	…	…	…	…	…

Tabla 3

Donde $f_i \in F$ y $n \in \omega$. A partir de la matriz, podemos obtener:

1. El *objeto diagonal* $\delta := \langle X_{00}, X_{11}, \ldots, X_{ii}, \ldots \rangle$ (es la línea diagonal).
2. El *objeto antidiagonal* $\delta^* := \langle X^*_{00}, X^*_{11}, \ldots, X^*_{ii}, \ldots \rangle$, tal que X^*_{ii} es un entero diferente a su correspondiente en δ.

Tenemos dos resultados importantes:

1. Dado que podemos obtener cualquier X^*_{ii} a partir de algún procedimiento como sumar 1 a X_{ii}, entonces es claro que $\delta \in F \Rightarrow \delta^* \in F$.
2. Si δ^* está en la matriz, entonces es una función tipo f_i. De aquí se sigue que habrá un X_{ii} que le corresponda como función. Pero esto entraría en contradicción con la misma definición del objeto antidiagonal, ya que X_{ii} pertenece al objeto diagonal. Por tanto, $\delta^* \notin D^F$.

De estos resultados se sigue que si $\delta \in F$, entonces la matriz D^F no es una representación completa de F. Esto se entiende en tanto que si existe un objeto diagonal, entonces $\delta^* \in F$ pero $\delta^* \notin D^F$. Por tanto, hay funciones que son de la clase F pero no son representables en la matriz de F. Además, lo anterior tiene como corolario que si el objeto diagonal es una función de la clase F, entonces no se pueden enumerar de forma efectiva los elementos de F, ya que la existencia de D^F depende de la hipótesis de partida de que F es enumerable. Por consiguiente, se concluye que el conjunto de las funciones numéricas totales no es enumerable de forma efectiva. Ahora bien, la cuestión es la siguiente: tenemos que el conjunto de máquinas de Turing es enumerable, tenemos que el conjunto de las funciones numéricas Turing-computables es enumerable, pero también tenemos que no todas las funciones numéricas que se pueden calcular de modo efectivo son computables. Los objetos diagonal y antidiagonal muestran que existen funciones calculables que no pueden pertenecer de forma simultánea a un mismo lenguaje.

Sin embargo, se puede pensar que los objetos diagonal y antidiagonal pertenecen a una clase de funciones en la que no todas sean calculables de la misma manera.

Algunas funciones son susceptibles de una descripción finita en el mismo lenguaje pero no arrojan valores para determinados argumentos. El problema estriba en que no se puede diseñar un procedimiento para identificar a tales funciones. Esto muestra cierto desajuste entre la descripción finita de tareas efectivas, la descripción completa de la clase de tareas efectivas y el concepto informal de tarea efectiva. ¿Es aceptable la idea intuitiva de una tarea que siendo efectiva no tenga un resultado? Esta cuestión es recogida en el formalismo de las máquinas de Turing como el *problema de parada*. En caso de resolverse se demostraría la inexistencia de un procedimiento que impide la descripción completa del conjunto de tareas efectivas.

El problema de parada consiste en construir un procedimiento efectivo tal que, dada una máquina de Turing y dado un número n, determine si la máquina va a terminar su rutina al introducir n como input. Se trata de determinar si la función de parada que tiene como argumentos el código de la máquina y n es Turing-computable. Podemos formular el problema de parada del siguiente modo: se debe probar si puede o no existir una máquina $U_p(x, y_1, \ldots, y_n)$ tal que:

> $= 1$ si $T^x(y_1, \ldots, y_n)$ realiza una parada estándar emitiendo un valor para los datos de entrada $\langle y_1, \ldots, y_n \rangle$,
>
> $= 0$ si ocurre cualquier otra cosa.

Seguimos el análisis de Alonso (2004, pp.44-45) para mostrar que el esquema de la demostración es similar al del argumento diagonal:

- Tomamos como hipótesis de partida la existencia de $U_p(x, y)$.
- Sea la máquina $D^*(x)$ tal que $(U_p(x, x) = 0) \Rightarrow (D^*(x) = 1)$; y si $U_p(x, x) \neq 0$, entonces $D^*(x)$ entra en bucle.
- $D^*(x)$ existe: dada la existencia de $U_p(x, x)$, añadimos la instrucción de escribir 1 y borrar 0 cuando se encuentre 0 en la cinta y desplazarse a uno de los lados cuando encuentre 1.
- $D^*(x)$ tiene un índice puesto que el conjunto de las máquinas de Turing es enumerable. Si su índice es i, entonces $D^*(x) = T^i(x)$.
- En el caso de que $x = i$ hay dos posibilidades:

    1. $D^*(i) = 1$. Por la definición de $D^*(x)$ tenemos que $U_p(i, i) = 0$. Por la definición de $U_p(x, x)$ tenemos que $T^i(i)$ no está definida. Esto nos conduce a una contradicción dado que $D^*(x) = T^i(x)$, lo cual implica que $D^*(i)$ no está definida.

2. $D^*(i)$ entra en bucle. Del mismo modo $T^i(i)$ entra en bucle, lo cual implica que $U_p(i,i) = 0$. Esto conduce a una contradicción dado que si tenemos $U_p(i,i) = 0$, entonces por definición $D^*(i) = 1$.

Con $x = i$ sólo llegamos a contradicción. Por lo tanto, $D^*(x)$ no se puede definir, de lo cual se sigue que $U_p(x,x)$ tampoco es definible, lo que lleva a establecer que $U_p(x,y)$ tampoco es definible.

Referencias

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

Alonso, E. 2006. De la computabilidad a la hipercomputación. *Azafea* 8: 121-146.

Andréka, H.; Madarász, J.; Németi, I.; Németi, P.; Székely, G. 2018. Relativistic Computation. In: M. Cuffaro; S. Fletcher (ed.), *Physical Perspectives on Computation, Computational Perspectives on Physics*, p.195-216. Cambridge: Cambridge University Press.

Copeland, B. J. 2002. Accelerating Turing Machines. *Minds and Machines* 12(2): 281-301.

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

Copeland, B. J. & Shagrir, O. 2007. Physical Computation: How General are Gandy's Principles for Mechanisms?. *Minds & Machines* 17: 217-231.

Copeland, B. J. & Shagrir, O. 2011. Do Accelerating Turing Machines Compute the Uncomputable?. *Minds and Machines* 21(2): 221-239.

Davies, E.B. 2001. Building infinite machines. *British Journal for the Philosophy of Science* 52(4): 671-682.

Davis, M. 1982. *Computability and unsolvability*. Nueva York: Dover.

Davis, M. 2006. Why there is no such discipline as hypercomputation. *Applied Mathematics and Computation* 178(1): 4-7.

Deutsch, D. 1985. Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. *Proceedings of the Royal Society of London* 400: 97-117.

Earman, J. 1995. *Bangs, Crunches, Whimpers, and Shrieks: Singularities and Acausalities in Relativistic Spacetimes*. Oxford: Oxford University Press.

Earman, J. & Norton, J. 1993. Forever is a day: Supertasks in Pitowsky and Malament-Hogarth spacetimes. *Philosophy of Science* 60(1): 22-42.

Etesi, G. & Németi, I 2002. Non-Turing Computations via Malament-Hogarth Spacetimes. *International Journal of Theoretical Physics* 41: 342-70.

Gandy, R. 1980. Church's thesis and the principles for mechanisms. In: J. Barwise,; H. J. Keisler; K. Kunen (ed.), *The Kleene symposium*. Ámsterdam: North-Holland.

Hogarth, M. 1994. Non-Turing computers and non-Turing computability. *Proceedings of the Biennial Meeting of the Philosophy of Science Association* 1: 126-138.

Kripke, S. 2013. The Church-Turing 'Thesis' as a Special Corollary of Gödel's Completeness Theorem. In: B. J. Copeland; C. Posy; O. Shagrir (ed.), *Computability: Turing, Gödel, Church, and Beyond*, p.77-104. Cambridge: The MIT Press.

Manchak, J. 2010. On the Possibility of Supertasks in General Relativity. *Foundations of Physics* 40(3): 276-288.

Manchak, J. 2020. Malament-Hogarth Machines. *British Journal for the Philosophy of Science* 71(3): 1143-1153.

Piccinini, G. 2015. *Physical Computation. A Mechanistic Account*. Oxford: Oxford University Press.

Piccinini, G. & Maley, C. 2021. Computation in Physical Systems. In: E. Zalta (ed.), *The Stanford Encyclopedia of Philosophy*. Edición de verano 2021. https://plato.stanford.edu/archives/sum2021/entries/computation-physicalsystems.

Pitowsky, I. 1990. The Physical Church Thesis and Physical Computational Complexity. *Iyyun* 39: 81-99.

Shagrir, O. & Pitowsky, I. 2003. Physical hypercomputation and the church-turing thesis. *Minds and Machines* 13(1): 87-101.

Sieg, W. 2002. Calculations by man and machine: Mathematical presentation. In: P. Gardenfors; J. Wolenski; K. Kijania-Placek (ed.), *In the Scope of Logic, Methodology and Philosophy of Science*, vol.I, p.247-262. Países Bajos: Kluwer Academic Publishers.

Sieg, W. 2008. Church Without Dogma: Axioms for Computability. In: S. B. Cooper; B. Löwe; A. Sorbi (ed.), *New Computational Paradigms*, p.139-152. Nueva York: Springer.

Thomson, J. F. 1954 Tasks and super-tasks. *Analysis* 15(1): 1-13.

Turing, A. 1936. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the Mathematical Society* 42: 230-265.

Wolfram, S. 1985. Undecidability and Intractability in Theoretical Physics. *Physical Review Letters* 54: 735-738.

Notas

[1]Naturalmente, Turing (1936) no empleó las expresiones "máquinas de Turing" o "tesis de Turing". Sin embargo, dada la actual estandarización de este modelo, en adelante se usarán dichas expresiones para evitar ambigüedades. Asimismo, es importante señalar que nos centramos en la noción de máquina automática o máquina-*a* presente en el artículo original de Turing (1936).

[2]Para el análisis de la Turing-computabilidad de esta sección nos apoyamos en Turing (1936) y en la exposición de Alonso (2004).

[3]Puede comprobarse que 1 y 0 no significan lo mismo si se encuentran en X o si se encuentran en Y.

[4]Con estas definiciones podemos interpretar la instrucción 1011. Dicha instrucción expresa que si la cabeza lectora se encuentra en una celda con el símbolo 0 (o vacía), entonces debe marcar en ella el símbolo 1 y después activar el estado 1. El último símbolo indica el estado que se debe activar a continuación y no se refiere a una instrucción. Esto explica que n y m puedan tener el mismo número sin entrar en bucle.

[5]Para calcular una función numérica debemos tener en cuenta las *condiciones de contorno*, a las cuales distinguimos en tres tipos. En primer lugar, los requerimientos sobre los inputs de la máquina. Tales inputs son los argumentos de la función de la cual se computan sus valores. Dichos requerimientos son básicamente tres: a) los enteros positivos son representados mediante el contenido de una sucesión de celdas, b) los argumentos de la función son representados mediante el contenido de varias sucesiones de celdas divididas por una celda

vacía y c) los límites de una sucesión de inputs se identifican por un conjunto previamente estipulado de celdas vacías seguidas. En segundo lugar, existen determinadas condiciones sobre la situación inicial en la que la máquina se encuentra: a) cuando se activa la máquina la cinta no tiene celdas ocupadas salvo las que contienen el input y b) cuando se activa la máquina la cabeza lectora está ubicada en la celda ocupada inmediatamente a la izquierda del input. En tercer lugar, hay que tener en cuenta las condiciones de parada: a) la máquina tiene el estado 0 y b) la cabeza lectora está ubicada de forma estática sobre la celda inmediatamente a la izquierda de una sucesión de celdas ocupadas y limitadas por el conjunto estipulado de celdas vacías que indican los límites de una sucesión de inputs. No obstante, es también posible que la máquina se detenga sin llegar al estado 0 y que la cabeza lectora se detenga en una situación de la cinta diferente a la descrita en la condición b). En estos casos ocurre una parada no estándar.

[6]Tomamos a Piccinini (2015) como referencia principal puesto que su propuesta de la *The Mechanistic Account of Computation* es la de mayor relevancia sobre la computación física en la actualidad. Sus condiciones plantean un marco conceptualmente más amplio que el del artículo seminal de Gandy (1980) que analizaremos más adelante (infra §5.1.). La noción mecanicista o mecánica de computación física de Piccinini (2015) es la generalización más amplia actualmente y hace explícitas asunciones irrenunciables. Puede encontrarse una síntesis de las principales tesis de Piccinini (2015) en Piccinini y Maley (2021).

[7]En adelante, todas las citas literales son traducciones propias de sus respectivos originales en inglés.

[8]Es importante diferenciar entre realizar computaciones y ser computacionalmente modelado. Un sistema físico puede ser modelado computacionalmente, pero no realizar él mismo computaciones. La afirmación según la cual todo sistema físico computa es la tesis metafísica del pancomputacionalismo.

[9]En adelante, nos referiremos a estas condiciones como MAC(i)-(vi).

[10]Piccinini (2015, capítulo 1.4.) presenta seis desiderata que una noción de computación concreta debe cumplir, el primero de los cuales es la objetividad. Asimismo, Piccinini (2015, p.18) apela al desiderátum de objetividad para criticar la noción del mapeo simple.

[11]Ya que será nuestro objeto principal de discusión.

[12]El problema de la decisión se puede formular en términos del *problema de parada* dentro del marco teórico de las máquinas de Turing. La conexión se constata al expresar el problema de parada en lógica de primer orden. Con estos recursos Turing (1936) probó una solución negativa al problema de la decisión. Esta cuestión se analizará en apéndice y es la base para la comprensión de la noción de la hipercomputación.

[13]Esta condición será la que nos permitirá discutir la posibilidad de la hipercomputabilidad física más adelante (infra §5.3.).

[14]Kripke (2013) argumentó que la tesis de Turing sí puede probarse formalmente. No obstante, la discusión de esta propuesta excede los límites de nuestra investigación.

[15]Los argumentos tipo c) se encuentran en Turing (1936, §10).

[16]Es importante destacar que es diferente *probar* un teorema y *decidir* si una fórmula es un teorema o no. El *problema de la decisión* tiene una respuesta negativa, ya que no se pueden enumerar recursivamente los no-teoremas de la lógica de primer orden.

[17]Turing propone que la marca sean los dos puntos duplicados "::".

[18]Existen diversos problemas de decisión que se preguntan por la existencia de un algo-

ritmo que determine la verdad o la falsedad de un conjunto de enunciados. En sentido general, siguiendo la explicación de Davis (1982, p.69), sea P un predicado y dados los números $a_1, \ldots, a_n$, el problema de decisión asociado al predicado P consiste en determinar si $P(a_1, \ldots, a_n)$ es verdadero o falso. El problema de decisión para $P(a_1, \ldots, a_n)$ se puede resolver recursivamente si P es recursivo.

[19]Véase el apéndice para una explicación del problema de parada.

[20]Nos apoyamos en la formulación informal de Copeland y Shagrir (2007, pp.218-219) para destacar el aspecto conceptual de los principios.

[21]En adelante llamaremos "máquina de Gandy" a un dispositivo mecánico discreto y determinista definido por los principios de Gandy G1-4.

[22]S es una "clase estructural" en el siguiente sentido: "subclases de conjuntos finitos hereditarios HF sobre un conjunto potencialmente infinito de átomos que está cerrado bajo estructuras isomorfas" (Copeland y Shagrir 2007, p.218).

[23]Es una sección de HF, véase nota anterior.

[24]Es importante señalar que a nivel técnico, a diferencia de la computadora humana de Turing, una máquina de Gandy determina la evolución de sistemas dinámicos en paralelo. El trabajo de una máquina de Gandy permite la computación paralela. Para un análisis de la computación paralela véase Sieg (2002).

[25]Thomson (1954) introdujo el concepto de "supertarea" (*supertask*) para referirse a tareas de pasos infinitos cuya rutina termina en un tiempo finito.

[26]Copeland y Shagrir (2007, p.227) dejaron esta posibilidad como una cuestión abierta físicamente. Esta es la posibilidad que analizamos en la siguiente sección.

[27]Para el detalle de la construcción y el funcionamiento de las máquinas de Davies véase Davies (2001, §§ 2 y 3).

[28]Es menester destacar que esta es la formulación presente en los trabajos de Piccinini. No obstante, en la literatura relacionada existen otras formulaciones de la versión audaz. Por ejemplo, Wolfram (1985) y Deutsch (1985) sostienen que todo proceso físico puede simularse por alguna máquina de Turing.

[29]Esta tesis es la que afirma el pancomputacionalismo, a saber, que todo sistema físico es computacional.

[30]La trayectoria pasa por el concepto de máquinas-*oráculo* de Turing, la *lámpara* de Thomson, la *máquina Zeus* de Boolos o las máquinas de Turing de tiempo infinito de Hamkin y Lewis.

[31]Pitowsky (1990) propuso la idea original de una máquina relativista que Hogarth (1994) desarrolló.

[32]No es exactamente una máquina acelerada porque esta exige que la velocidad de las sucesivas operaciones de transición sea arbitrariamente alta. El experimento mental, en cambio, logra este funcionamiento con una duración infinita en su pasado causal.

[33]El teorema de la dilatación gravitacional del tiempo es de la relatividad especial y se da en relatividad general por el principio de equivalencia.

[34]Earman y Norton (1993) investigaron las propiedades físicas de los especiotiempos Malament-Hogarth y analizaron si un observador podría registrar el resultado de una computación de infinitos pasos.

[35]Manchak (2020) analiza formalmente diversas definiciones y la existencia de máquinas Malament-Hogarth.

[36]Véase Manchak (2010) para un desarrollo formal de un espaciotiempo Malament-Hogarth con relación a la posibilidad de ejecutar supertareas.

[37]Se trata de las *curvas* que son transitables.

[38]En inglés "locally timelike" y "locally lightlike" si es transitable por cuerpos o por fotones, respectivamente.

[39]Cabe mencionar que con su experimento mental Andréka *et al*. (2018) dan respuesta a las dos dificultades fundamentales que observa Piccinini (2015, capítulo 16.3.) frente la constructibilidad de una hipercomputadora relativista, a saber: (i) debido al trabajo infinito la probabilidad de error de la máquina es 1 y (ii) en los espaciotiempos que permiten hipercomputadoras relativistas hay singularidades y las singularidades las entiende como límite de la relatividad general antes que como una posibilidad para la hipercomputabilidad. No obstante, se trata de los detalles del debate cruzado. Nosotros nos hemos centrado en los aspectos fundamentales.

[40]Debemos tener como referencia lo expuesto en §3.2.2.

[41]Además, los espaciotiempos relativistas especial de Minkowski y general de los agujeros negros estáticos no contienen eventos Malament-Hogarth y el espaciotiempo de un agujero negro de Kerr sí.

[42]Este argumento se ve reforzado si consideramos la axiomática de "sistemas dinámicos discretos" que establece Sieg (2008), puesto que (i) permite representar las computaciones de la computadora humana y de dispositivos mecánicos y (ii) modelos de esta axiomática pueden reducirse a máquinas de Turing.

[43]Es importante aclarar que con esto simplemente nos referimos a la posibilidad de una hipercomputadora abstracta y no a que dicha hipercomputadora abstracta suponga una refutación de la tesis de Turing matemática. Tal como observan Shagrir y Pitowsky (2003, §3) las máquinas en el contexto de eventos Malament-Hogarth son un contraejemplo a la "tesis de Gandy" (tesis de Turing física para máquinas de Gandy), pero no a la tesis de Turing matemática ya que se computan funciones no Turing-computables con métodos no efectivos por usar pasos infinitos.

[44]La diagonalización es un método empleado por Cantor en la demostración de que el conjunto de los números reales no es numerable.