# Por qué los Filósofos debieran Preocuparse de la Complejidad Computacional

**Scott Aaronson***

**Traducción: Jorge Portilla**

**Resumen**

Podría pensarse que una vez que se sabe que algo es computable, es una cuestión práctica de poca importancia filosófica ulterior qué tan *eficientemente* puede computarse. En este ensayo, ofrezco un caso detallado de que se estaría en un error. En particular, sostengo que la *teoría de la complejidad computacional* – el campo que estudia los recursos (como tiempo, espacio y aleatoriedad) necesarios para resolver problemas computacionales – conduce a nuevas perspectivas sobre la naturaleza del conocimiento matemático, el debate de la inteligencia artificial fuerte, el computacionalismo, el problema de la omnisciencia lógica, el problema de la inducción de Hume, el acertijo grue de Goodman, los fundamentos de la mecánica cuántica, la racionalidad económica, las curvas temporales cerradas, y varios otros temas de interés filosófico. Concluyo discutiendo aspectos de la misma teoría de la complejidad que podrían beneficiarse del análisis filosófico.

**Contenido**



* MIT. Email: aaronson@csail.mit.edu. El material original, *Why Philosophers Should Care About Computational Complexity*, está basado en el trabajo apoyado por la National Science Foundation bajo la Subvención No. 0844626. También apoyado por una subvención YFA de DARPA, la Sloan Foundation y una Cátedra de TIBCO.

## 1. Introducción

> *La visión de que las máquinas no pueden dar lugar a sorpresas se debe, creo yo, a una falacia a la que filósofos y matemáticos están particularmente sujetos. Esta es la suposición de que en cuanto un hecho se presenta a una mente todas las consecuencias de ese hecho saltan en la mente simultáneamente con él. Es una suposición muy útil bajo muchas circunstancias, pero se olvida muy fácilmente que es falsa. Alan M. Turing [130]*

La teoría de la computación, creada por Alan Turing, Alonzo Church, Kurt Gödel y otros en la década de 1930, no sólo cambió la civilización; también tuvo un impacto duradero en la filosofía. En efecto, aclarar cuestiones filosóficas fue el *punto* original de su trabajo; ¡los pagos tecnológicos llegaron más tarde! Hoy sería difícil imaginar una discusión seria sobre (digamos) la filosofía de la mente, los fundamentos de la matemática o las perspectivas de la inteligencia de máquinas, que no estuviera informada por esta revolución en el conocimiento humano desde hace tres cuartos de siglo.

Sin embargo, a medida que las computadoras se hicieron ampliamente disponibles a partir de la década de 1960, los científicos de la computación llegaron a ver cada vez más que la teoría de la computabilidad no estaba haciendo bastantes preguntas correctas. Porque casi *todos* los problemas que realmente queremos resolver resultan ser computables en el sentido de Turing; la pregunta real es cuáles problemas son *eficientemente* o *factiblemente* computables. Esta última cuestión dio lugar a un nuevo campo, llamado teoría de la complejidad computacional (que no debe confundirse con la teoría de la "otra" complejidad, que estudia sistemas complejos como autómatas celulares). Desde la década

de 1970, la teoría de la complejidad computacional ha sido testigo de algunos descubrimientos espectaculares que incluyen la completitud de NP, la criptografía de clave pública, nuevos tipos de demostración matemática (tales como las demostraciones probabilísticas, interactivas y de conocimiento cero), y los fundamentos teóricos del aprendizaje de máquinas y la computación cuántica. Para quienes trabajan en estos tópicos, el trabajo de Gödel y Turing puede verse en retrospectiva, simplemente como un calentamiento para las "grandes" preguntas sobre computación.

Debido a esto, me parece sorprendente que la teoría de la complejidad *no* haya influido en la filosofía a alguna cosa como la magnitud que tiene la teoría de la computabilidad. Se plantea la pregunta: ¿*por qué no la tiene*? Surgen en la mente varias respuestas posibles: quizás la teoría de la computabilidad tuvo implicaciones filosóficas más ricas (aunque, como veremos, se puede hacer un caso fuerte para exactamente lo opuesto). Tal vez la complejidad tiene esencialmente las mismas implicaciones filosóficas que la computabilidad, y ésta llegó allí primero. Tal vez los extraños se asustan de aprender la teoría de la complejidad por la "barrera matemática". Tal vez la explicación es social: el mundo donde Gödel, Turing, Wittgenstein y Russell participaron en la misma conversación intelectual, se desvaneció con la Segunda Guerra Mundial; después de eso, la ciencia de la computación teórica vino a ser manejada por la tecnología y perdió contacto con sus orígenes filosóficos. Quizá los recientes adelantos en la teoría de la complejidad simplemente no han tenido suficiente tiempo para entrar en la conciencia filosófica.

Sin embargo, sospecho que esa parte de la respuesta es sólo el *fracaso de los teóricos de la complejidad para comunicar* lo que pueden agregar al arsenal conceptual de la filosofía. De aquí este ensayo cuya modesta meta es ayudar a corregir ese fracaso, examinando algunos aspectos de la teoría de la complejidad que podría interesar a los filósofos, así como algunos problemas filosóficos que pienso que una perspectiva de la complejidad puede clarificar.

Para anticipar malas interpretaciones, permítanme agregar una nota de humildad antes de ir más allá. Este ensayo tocará muchos problemas que los filósofos han debatido por generaciones, como la inteligencia artificial fuerte, el problema de la inducción, la relación entre sintaxis y semántica, y la interpretación de la mecánica cuántica. *En ninguno de estos casos* afirmaré que la teoría de la complejidad computacional "disuelve" el problema filosófico – sólo que contribuye con perspectivas e intuiciones útiles. Con frecuencia mencionaré explícitamente rompecabezas filosóficos que pienso que un análisis de complejidad o los deja intactos o de lo contrario se presenta a sí mismo. Pero incluso donde no lo hago, no debe presumirse que piense que no haya tales rompecabezas. De hecho, una de mis esperanzas para este ensayo es que científicos de la computación, matemáticos, y otras personas técnicas que lo lean, se marchen con una mejor apreciación de la sutileza de algunos de los problemas considerados en la filosofía analítica moderna.[1]

### 1.1 Lo Que Este Ensayo *No* Cubrirá

No intentaré discutir toda *posible* conexión entre la complejidad computacional y la filosofía, o incluso cada conexión que ya se haya hecho. Un pequeño número de filósofos ha invocado hace mucho tiempo las ideas de complejidad computacional en su trabajo; de hecho, el "philpapers archive" lista 32 trabajos bajo el título Computational Complexity.[2] La mayoría de esos trabajos demuestra teoremas sobre las complejidades computacionales

1. Cuando yo use la palabra "filosofía" en este ensayo, me referiré a la filosofía dentro de la tradición analítica. No comprendo suficientemente bien la filosofía continental u oriental para decir si tiene alguna conexión interesante con la teoría de la complejidad computacional.
2. Ver philpapers.org/browse/computational-complexity

de varios sistemas lógicos. De los restantes trabajos, algunos usan "complejidad computacional" en un sentido diferente que yo lo hago – por ejemplo, abarcan la teoría de la computabilidad – y algunos invocan el *concepto* de complejidad computacional, pero ningún resultado particular del *campo* dedicado a él. Quizás los más cercanos en espíritu a este ensayo sean los interesantes artículos de Cherniak [40] y Morton [98]. Además, muchos escritores han hecho alguna versión de las observaciones en la Sección 4, sobre la complejidad computacional y el Test de Turing: ver, por ejemplo, Block [30], Parberry [102], Levesque [88] y Shieber [118].

Decidiendo qué conexiones incluir en este ensayo, adopté las reglas básicas siguientes:

(1) La conexión debe implicar un problema "propiamente filosófico" – por ejemplo, la justificación por inducción o la naturaleza del conocimiento matemático – y no sólo un problema técnico en lógica o teoría de modelos.

(2) La conexión debe delinearse sobre *intuiciones específicas* del campo de la teoría de la complejidad computacional: no sólo la *idea* de complejidad, o el *hecho* de que existen problemas difíciles.

Hay muchas ideas filosóficamente interesantes en la teoría de la complejidad moderna que este ensayo sólo menciona brevemente o nada en absoluto. Un ejemplo son los generadores pseudo-aleatorios (ver Goldreich [64]): funciones que convierten una corta "semilla" aleatoria en una cuerda larga de bits que, si bien no es verdaderamente aleatoria, es tan "parecida al azar" que ningún algoritmo eficiente puede detectar alguna regularidad en ella. Aunque en este sentido no ha sido demostrado que los generadores pseudo aleatorios existan,[3] hay muchos candidatos plausibles, y la creencia de que al menos alguno de ellos funciona es central para la criptografía moderna. (La sección 7.1 invocará el concepto relacionado de *funciones* pseudo aleatorias). Un segundo ejemplo es *cifrado totalmente homomórfico*: una nueva clase sumamente excitante de métodos, el primero de los cuales fue anunciado por Gentry [61] en 2009, para realizar cómputos arbitrarios en datos cifrados *sin nunca descifrarlos*. El resultado de tal cómputo le parecerá una jerga sin sentido a la persona que lo computó, pero, sin embargo, puede ser comprendido (y hasta reconocido como resultado correcto) por alguien que conoce la clave para descifrarlo. ¿Cuáles son las implicaciones de los generadores pseudo-aleatorios para los fundamento de la probabilidad, o del cifrado totalmente homomórfico para debates sobre el significado semántico de los cómputos? Espero mucho que este ensayo inspire a otros para que aborden éstas y similares preguntas.

Fuera de de complejidad computacional, hay al menos tres otros puntos mayores de intersección entre la filosofía y la ciencia de la computación teórica moderna. El primero es la *semántica de los lenguajes de programación* que tienen grandes y obvias conexiones con la filosofía del lenguaje.[4] El segundo es la *teoría de los sistemas distribuidos* que provee un área de aplicación y una rica fuente de ejemplos para el trabajo filosófico sobre razonamiento acerca del conocimiento (ver Fagin *et al*. [54] y Stalnaker [126]). El tercero es la *complejidad de Kolmogorov* (ver Li y Vitanyi [90]) que estudia la *longitud* del programa de computación más corto que logra alguna funcionalidad, sin tomar en cuenta tiempo, memoria, y otros recursos usados por el programa.[5]

---

3. La conjetura de que existen generadores pseudo-aleatorios, implica la conjetura de $\mathsf{P} \neq \mathsf{NP}$ (sobre lo cual se hablará más adelante), pero podría ser más fuerte aún: la implicación inversa es desconocida.
4. La entrada "*The Philosophy of Computer Science*" en la *Stanford Encyclopedia of Philosophy*, plato.stanford.edu/entries/computer-ciencia, dedica la mayoría de su espacio a esta conexión.
5. Una variante, "la complejidad limitada por recursos de Kolmogorov", toma en cuenta el tiempo y la memoria, y es parte de la teoría de la complejidad computacional.

En este ensayo no discutiré ninguna de estas conexiones, excepto al pasar (por ejemplo, la Sección 5 toca la lógica del conocimiento en el contexto del "problema de la omnisciencia lógica", y la Sección 7 lo hace sobre la complejidad de Kolmogorov en el contexto del aprendizaje PAC). En defensa de estas omisiones, permítanme ofrecer cuatro excusas. Primero, estas otras conexiones caen fuera de mi tópico declarado. Segundo, harían que este ensayo fuera aun más largo de lo que ya es. Tercero, carezco de los antecedentes necesarios. Y cuarto, mi impresión es que los filósofos – al menos algunos de ellos – están ya más conscientes de estas otras conexiones que de las conexiones de la complejidad computacional que quiero explicar.

## 2 Complejidad 101

La teoría de la complejidad computacional es un campo grande y extendido; naturalmente este ensayo sólo mencionará partes pequeñas de él. Se insta a los lectores que quieran ahondar más profundamente en el asunto, a que consulten uno de muchos sobresalientes libros de texto, como los de Sipser [124], Papadimitriou [101], Moore y Mertens [96], Goldreich [63], o Arora y Barak [15]; o los artículos de apreciación de Wigderson [137, 138], Fortnow y Homer [59], o Stockmeyer [127].

Podría pensarse que, una vez que conocemos que algo es computable, que tome 10 o 20 segundos para computar es obviamente más preocupación de ingenieros que de filósofos. ¡Pero esa conclusión no sería tan obvia si la cuestión fuera 10 segundos versus $10^{10^{10}}$ segundos! Y de hecho, en la teoría de la complejidad, los huecos cuantitativos de los que nos preocupamos son usualmente tan vastos que se los debe considerar también huecos cualitativos. Piénsese, por ejemplo, en la diferencia entre leer un libro de 400 páginas y leer *cada posible* libro de tal tamaño, o entre escribir un número de mil dígitos y contar hasta ese número.

Más precisamente, la teoría de la complejidad hace la pregunta: cómo escalan los recursos necesarios para resolver un problema con alguna medida $n$ del tamaño del problema: ¿"razonablemente" (como $n$ o $n^2$, digamos), o "irrazonablemente" (como $2^n$ o $n!$)? Como ejemplo, dos enteros del $n$-dígitos pueden multiplicarse usando $n^2$ pasos computacionales (por el método de la escuela primaria), o hasta $\sim n \log n \log\log n$ pasos (por métodos más avanzados [114]). Cualquier método se considera eficiente. En contraste, el método más rápido conocido para la operación inversa – *factorizar* un entero de $n$-dígitos en primos – usa $\sim 2^{n^{\frac{1}{3}}}$ pasos, lo cual es considerado ineficiente.[6] Célebremente, esta brecha conjeturada entre dificultades inherentes a multiplicar y factorizar, es la base para la mayoría de la criptografía usada actualmente en la Internet.

Los científicos de la computación teórica generalmente llaman a un algoritmo "eficiente" si su tiempo de ejecución puede ser limitado superiormente por cualquier función polinomial de $n$, e "ineficiente" si su tiempo de ejecución puede ser limitado inferiormente por cualquier función exponencial de $n$.[7] Estos criterios tienen la gran ventaja de la conveniencia teórica. Si bien la complejidad exacta de un problema podría depender de los "detalles de la codificación de bajo nivel", tales como si nuestra máquina de Turing tiene

---

6. Este método se denomina la *criba del campo de números*, y el tiempo citado para la ejecución depende de conjeturas plausibles pero no demostradas en teoría de números. El mejor tiempo de ejecución demostrado es $\sim 2^{\sqrt{n}}$. De éstos, ambos representan mejoras no triviales al método ingenuo de tratar todos los posibles divisores, lo cual toma $\sim 2^n$ pasos. Ver Pomerance [106] para un buen estudio de los algoritmos de factorización.

7. En algunos contextos, "exponencial" significa $c^n$ para alguna constante $c > 1$, pero en la mayor parte de los contextos de complejidad teórica también puede significar $c^{n^d}$ para constantes $c > 1$ y $d > 0$.

una o dos cintas de memoria, o cómo son codificadas las entradas como cadenas binarias, donde un problema cae en la dicotomía polinomial/exponencial, puede demostrarse que es independiente de casi todas estas opciones.[8] Igualmente importantes son las *propiedades de cierre* de tiempo polinomial y exponencial: un algoritmo de tiempo polinomial que llama a una subrutina de tiempo polinomial todavía produce un algoritmo de tiempo polinomial, mientras que un algoritmo de tiempo polinomial que llama a una subrutina de tiempo exponencial (o viceversa) produce un algoritmo de tiempo exponencial. Hay también razones más sofisticadas por las que los científicos de la computación teórica se enfocan en el tiempo polinomial (en lugar de, digamos, tiempo $n^2$ o tiempo $n^{\log n}$); exploraremos algunas de esas razones en la Sección 5.1.

La distinción polinomial/exponencial está abierta a obvias objeciones: ¡un algoritmo que tomó $1.00000001^n$ pasos sería mucho más rápido en la práctica que un algoritmo que tomó $n^{10000}$ pasos! Además, hay muchas tasas de crecimiento que caen entre polinomial y exponencial, como el $n^{\log n}$ y $2^{\sqrt{\log n}}$. Pero empíricamente, el tiempo polinomial *resultó* corresponder tan frecuentemente a "eficiente en la práctica", y el tiempo exponencial "ineficiente en la práctica", que los teóricos de la complejidad se sintieron cómodos haciendo la identificación. ¿*Por qué* la identificación funciona?, es una pregunta interesante en su propio derecho, una a la que volveremos en la Sección 12.

*A priori*, insistir que los programas terminan después de razonables montos de tiempo, que usan razonables montos de memoria, etc. podría sonar como enmiendas relativamente menores a la noción de computación de Turing. En la práctica, sin embargo, estos requerimientos conducen a una teoría con un carácter completamente diferente que la teoría de la computabilidad. Primeramente, la complejidad tiene conexiones mucho más cercanas con las *ciencias*: nos permite formular preguntas sobre (por ejemplo) evolución, mecánica cuántica, física estadística, economía, o adquisición del lenguaje humano que serían sin sentido desde un punto de vista de la computabilidad (puesto que *todos* los problemas relevantes son computables). La complejidad también difiere de la computabilidad en la diversidad de *técnicas* matemáticas usadas: mientras que inicialmente la complejidad (como la computabilidad) utilizó principalmente la lógica matemática, hoy utiliza probabilidad, teoría de números, combinatoria, teoría de la representación, análisis de Fourier, y casi todo otro asunto sobre el que se escriben libros amarillos. Por supuesto, esto no sólo contribuye a la profundidad de la teoría de la complejidad sino también a su percibida inaccesibilidad.

En este ensayo, argumentaré que la teoría de la complejidad tiene relevancia directa para cuestiones mayores de filosofía, incluyendo sintaxis y semántica, el problema de la inducción y la interpretación de la mecánica cuántica. O que, por lo menos, ¡si la teoría de la complejidad *tiene o no tiene* tal relevancia es una pregunta importante para la filosofía! Mi visión personal es que la complejidad se demostrará últimamente más relevante para la filosofía que lo fue la computabilidad, precisamente debido a las ricas conexiones con las ciencias mencionadas antes.

## 3 Relevancia del Tiempo Polinomial

8. Esto no quiere decir que no importa *ningún* detalle del modelo computacional: ¡por ejemplo, se sabe que algunos problemas son resolubles en tiempo polinomial en una computadora cuántica, pero *no* se sabe si lo son en tiempo polinomial en una computadora clásica! Pero en mi visión, el hecho que la distinción polinomial/exponencial pueda "notar" una opción de modelado de esta magnitud es una característica de la distinción, no un error (*bug*).

Cualquiera que dude de la importancia de la distinción polinomial/exponencial sólo necesita ponderar cuántas intuiciones básicas en matemática, ciencia, y filosofía se basan ya, implícitamente, en esa distinción.

En esta sección daré tres ejemplos.

### 3.1 El Entscheidungsproblem Revisitado

El *Entscheidungsproblem* fue el sueño, enunciado por David Hilbert en la década de 1920, de diseñar un procedimiento mecánico para determinar la verdad o falsedad de cualquier declaración matemática bien formada. Según el relato usual, el sueño de Hilbert se destruyó irrevocablemente por el trabajo de Gödel, Church y Turing en la década de 1930. Primero, el Teorema de la Incompletitud demostró que ningún sistema formal recursivamente axiomatizable puede codificar *todas y solamente* las declaraciones matemáticas verdaderas. Segundo, los resultados de Church y Turing demostraron que, aun si nos conformamos con un sistema incompleto $F$, no hay *todavía* ningún procedimiento mecánico para ordenar las declaraciones matemáticas en las tres categorías "demostrable en $F$", "indemostrable en $F$", e "indecidible en $F$".

Sin embargo, hay una trampa en el relato anterior, la cual fue primeramente señalada por el propio Gödel, en una carta a John von Neumann en 1956, que se hizo famosa en la ciencia de la computación teórica desde su redescubrimiento en la década de 1980 (ver Sipser [123] para una traducción en inglés). Dado un sistema formal $F$ (como la teoría de conjuntos de Zermelo-Fraenkel), Gödel escribió, considerar el problema de decidir si una declaración matemática $S$ tiene una prueba en $F$ *con n símbolos o menos*. A diferencia del problema original de Hilbert, este "Entscheidungsproblem truncado" es claramente decidible. Porque simplemente, si nada más, siempre podríamos programar una computadora para buscar a través de todas los $2^n$ posibles cadenas de bits con $n$ símbolos, y verificar si cualquiera de ellas codifica una prueba válida en $F$ de $S$. La dificultad es "meramente" que este enfoque toma un monto astronómico de tiempo: si (digamos) $n = 1000$, ¡entonces el universo habrá degenerado en agujeros negros y radiación mucho antes de que una computadora pueda verificar $2^{1000}$ pruebas!

Pero como también señaló Gödel, está lejos de ser obvio cómo demostrar que no hay un enfoque mucho mejor: un enfoque que evitaría la búsqueda por fuerza bruta, y encontrar pruebas de tamaño $n$ en tiempo polinomial en $n$. Además:

> Si realmente hubiera una máquina con [tiempo de ejecución] ~ $Kn$ (o incluso sólo con ~ $Kn^2$) [para algún $K$ constante independiente de $n$], esto tendría consecuencias de la mayor magnitud. Es decir, indicaría claramente que, a pesar de la insolubilidad del Entscheidungsproblem, el esfuerzo mental del matemático en el caso de preguntas por sí o no, podría ser completamente [agregado en una nota a pie de página: aparte de la postulación de axiomas] reemplazado por máquinas. De hecho, uno simplemente tendría que seleccionar un $n$ tan grande que, si la máquina no arroja ningún resultado, entonces tampoco habría razón alguna para pensar más en el problema.

Si reemplazamos "~ $Kn$ o ~ $Kn^2$" en el reto de Gödel por ~ $Kn^c$ para una constante arbitraria $c$, entonces conseguimos precisamente lo que la ciencia de la computación conoce ahora como el problema de P contra NP. Aquí P (Tiempo Polinomial) es, hablando aproximadamente, la clase de problemas que son solucionables por un algoritmo de tiempo polinomial. Entretanto, NP (Tiempo Polinomial No Determinístico) es la clase de problemas computacionales para los cuales puede *reconocerse* una solución en tiempo polinomial, aunque una solución podría ser muy difícil de encontrar.[9] (Piénsese, por

[9] Contrariamente a un común concepto erróneo, NP no significa "No polinomial". Hay problemas computacionales que se sabe que requieren más que tiempo polinomial (ver Sección 10), pero los

ejemplo, en la factorización de un número grande, o en la resolución de un rompecabezas o de un Sudoku). Claramente, $P \subseteq NP$, así que la pregunta es si la inclusión es estricta. Si $P = NP$, entonces la habilidad de *verificar* eficientemente las soluciones a los enigmas implicaría la habilidad para *encontrar* soluciones eficientemente. ¡Una analogía sería si cualquiera capaz de *apreciar* una gran sinfonía también podría componer una de ellas!

Dada la intuitiva implausibilidad de tal escenario, esencialmente todos los teóricos de la complejidad proceden (en mi opinión, razonablemente) a suponer que $P \neq NP$, aun cuando públicamente reclamen apertura de mente sobre la cuestión. Demostrar o refutar $P \neq NP$ es uno de los problemas del premio Clay Millennium de siete millones de dólares[10] (junto a la Hipótesis de Riemann, la Conjetura de Poincaré demostrada en 2002 por Perelman, etc.), lo cual debiera dar alguna indicación de la dificultad del problema.[11]

Ahora regresemos al problema de si una declaración matemática *S* tiene una prueba con *n* símbolos o menos, en algún sistema formal *F*. Una formalización adecuada de este problema se ve fácilmente en NP. Porque *encontrar* una prueba podría ser intratable, pero si se nos *da* una supuesta prueba, podemos verificar ciertamente en tiempo polinomial en *n* si cada línea de la prueba sigue por una simple manipulación lógica de líneas anteriores. De hecho, este problema resulta ser NP-*completo*, lo que significa que pertenece a una enorme clase de problemas de NP, por primera vez identificados en la década de 1970, que "capturan la entera dificultad de NP". Otros pocos ejemplos de problemas NP-completos son Sudoku y rompecabezas, el Problema del Vendedor Ambulante, y el de la satisfabilidad para fórmulas proposicionales.[12] Preguntar si $P = NP$ es equivalente a preguntar si *cualquier* problema NP-completo puede resolverse en tiempo polinomial, y también es equivalente a preguntar si *todos* ellos pueden ser.

Entonces, en términos modernos, Gödel está diciendo que si $P = NP$, entonces siempre que un teorema tenga una prueba de longitud razonable, podríamos *encontrar* esa prueba en un monto razonable de tiempo. En tal situación, podríamos decir que "para todo propósito práctico", el sueño de Hilbert de mecanizar las matemáticas ha prevalecido, a pesar de los resultados de indecidibilidad de Gödel, Church y Turing. Si aceptas esto, entonces parece justo decir que hasta que P versus NP se resuelva, la narración del Entscheidungsproblem de Hilbert – su ascenso, su caída, y las consecuencias para la filosofía – todavía no termina.

---

problemas NP no están entre esos. De hecho, las clases NP y "No polinomial" tienen una intersección no vacía exactamente si $P \neq NP$.

Para definiciones detalladas de P, NP, y varios centenares de otras clases de complejidad, véase mi sitio web Complexity Zoo: www.complexityzoo.com.

10. Para más información ver www.claymath.org/millennium/PvsNP/

Mi propia opinión es que P versus NP es manifiestamente el *más importante* de los siete problemas. Porque si $P = NP$, entonces por el argumento de Gödel, hay una excelente oportunidad de que pudiéramos programar nuestras computadoras para resolver también los otros seis problemas.

11. Uno podría preguntarse: ¿podemos explicar qué hace tan difícil al problema $P = NP$, en vez de señalar simplemente que muchas personas listas han intentado resolverlo y fracasaron? Después de cuatro décadas de investigación, *tenemos* explicaciones parciales para la dificultad del problema, en forma de "barreras" formales que descartan grandes clases de técnicas de prueba. Tres barreras identificadas hasta ahora son la *relativización* [21] (qué excluye la diagonalización y otras técnicas con sabor a "computabilidad"), *algebrización* [8] (que excluye la diagonalización, incluso cuando se combina con las principales técnicas no relativizantes conocidas hoy), y las *pruebas naturales* [110] (que muestran que muchas técnicas "combinatorias", si funcionaran, podrían revertirse para conseguir algoritmos más rápidos para distinguir funciones aleatorias de pseudoaleatorias).

12. Por contraste, y contrariamente a una idea equivocada común, hay fuerte evidencia que la factorización de enteros no es NP-completa. Es sabido que si $P \neq NP$, entonces habrán problemas NP que ni son P ni NP-completos [86], y la factorización es un candidato para tal problema. El punto llegará a ser relevante cuando discutamos la computación cuántica.

### 3.2 Evolucionabilidad

Los creacionistas frecuentemente reclaman la evolución darwiniana es tan vacua como explicación para las adaptaciones complejas como "un tornado que arma un avión 747 mientras pasa a través de un depósito de chatarra". ¿Por qué es falso este reclamo? Hay varios modos relacionados de contestar la pregunta, pero para mí, uno de los más esclarecedores es el siguiente. En principio, se *podría* ver un 747 armándose a sí mismo en un depósito de chatarra propenso a tornados – pero antes de que suceda, se necesitaría aguardar por un número esperado de tornados que creciera *exponencialmente* con el número de piezas de chatarra auto-ensambladas (esto es similar a cómo, en termodinámica, $n$ partículas de gas en una caja se *congregarán* eventualmente en una esquina de la caja, pero sólo después de un tiempo $\sim c^n$ para alguna constante $c$). Por contraste, pueden observarse frecuentemente procesos evolutivos en simulaciones – y en algunos casos, incluso demostrado teóricamente [108] – para encontrar soluciones interesantes a problemas de optimización después de un número de pasos que crece sólo polinomialmente con el número de variables.

Interesantemente, en una carta de 1972 a Hao Wang (ver [134, pág. 192]), Kurt Gödel expresó sus propias dudas sobre la evolución como sigue:

> Creo que ese mecanismo en biología es un prejuicio de nuestro tiempo que será refutado. En este caso, una refutación, en mi opinión, consistirá en un teorema matemático al efecto que la formación dentro del tiempo geológico de un cuerpo humano por las leyes de la física (o de cualquier otra ley de naturaleza similar), comenzando desde una distribución aleatoria de las partículas elementales y el campo, es tan improbable como una separación de la atmósfera en sus componentes por casualidad.

Personalmente, no veo ninguna razón para aceptar la intuición de Gödel en esta materia sobre el consenso general de la biología moderna. Pero prestemos atención a la redacción característicamente cuidadosa de Gödel. Él no pregunta si la evolución puede *eventualmente* formar un cuerpo humano (porque sabe que puede, dado un tiempo exponencial); en cambio, pregunta si puede hacerlo en una escala de tiempo "meramente" geológica. Así como la carta de Gödel a von Neumann anticipó el problema de P versus NP, así podría decirse que la carta de Gödel a Wang se anticipa a un reciente esfuerzo, por el celebrado científico de la computación Leslie Valiant, para construir una "teoría cuantitativa de la evolucionabilidad" [132]. Elaborando sobre el trabajo más temprano de Valiant en la teoría del aprendizaje computacional (discutido en la Sección 7), la evolucionabilidad intenta formalizar y responder preguntas sobre la velocidad de la evolución. Por ejemplo: "¿qué clases de conductas adaptativas pueden evolucionar, con alta probabilidad, después de sólo un número polinomial de generaciones? ¿Qué clases de conductas pueden aprenderse en tiempo polinomial, pero no vía evolución?" Aunque existen algunos resultados tempranos interesantes, nadie debe sorprenderse que esa evolucionabilidad no esté cerca, en ninguna parte, de poder calcular, desde primeros principios, si cuatro mil millones años son una longitud "razonable" o "irrazonable" de tiempo para que el cerebro humano evolucione desde la sopa primordial.

Como lo veo, esta dificultad refleja un punto general sobre la cuestión de la "evolucionabilidad" de Gödel. A saber, incluso suponiendo que Gödel estaba en lo cierto, que la visión mecanística del mundo de la biología moderna era "tan improbable como una separación de la atmósfera en sus componentes por casualidad", la teoría de la complejidad computacional parece estar lejos, sin esperanzas, de poder demostrar algo de esa clase. En 1972, se podría argumentar que esto meramente reflejaba la novedad del asunto: nadie había pensado muy profundamente todavía sobre cómo demostrar los *límites inferiores* en

tiempo de computación. Pero ahora, las personas *han* pensado profundamente sobre ello, y han identificado grandes obstáculos para demostrar incluso tales "obvias" y bien definidas conjeturas, como P ≠ NP.[13] (La Sección 4 elaborará un punto relacionado, sobre la dificultad de demostrar los límites inferiores no triviales en el tiempo o la memoria necesarios para que un programa de computación pase el Test de Turing).

### 3.3 Enteros Conocidos

Mi último ejemplo de la relevancia filosófica de la distinción polinomial/exponencial concierne al concepto de "conocimiento" en matemáticas.[14] Para 2011, el "mayor número primo conocido", como reporta la GIMPS (Great Internet Mersenne Prime Search),[15] $p := 2^{43112609} - 1$. Pero reflexionando, ¿qué significa decir que $p$ es conocido? ¿Queremos decir que, si lo deseáramos, podríamos imprimir literalmente sus dígitos decimales (usando alrededor de 30.000 páginas)? Eso parece ser un criterio demasiado restrictivo. Porque, dado un entero positivo $k$ junto con una prueba de que $q = 2^k - 1$ es primo, dudo que la mayoría de los matemáticos vacilaran en llamar a $q$ un primo "conocido",[16] aun si $k$ fuera tan grande que imprimir sus dígitos decimales (o guardarlos en una memoria de computación) estuviera más allá de la capacidad de la Tierra. ¿Debemos llamar a $2^{2^{1000}}$ una "potencia desconocida de 2", sólo porque tiene demasiados dígitos decimales para listar antes de que el sol se enfríe?

Se percibe que todo lo que realmente debe importar es que

(a) la expresión "$2^{43112609} - 1$" selecciona un único entero positivo, y

(b) ha sido demostrado (en este caso, vía computadora, por supuesto) que ese entero es primo.

¡Pero espera! Si esos son los criterios, ¿entonces por qué no podemos registrar así el mayor primo conocido?

$$p^l = \text{El primer primo mayor que } 2^{43112609} - 1.$$

Claramente $p^l$ existe, está inequívocamente definido y es primo. Si queremos, podemos escribir un programa que garantice encontrar $p^l$ e incluso dar salida a sus dígitos decimales, usando un número de pasos que pueda ser limitado superiormente *a priori*.[17] Todavía nuestra intuición insiste obstinadamente que $2^{43112609} - 1$ es un primo "conocido" en un sentido que $p^l$ no lo es. ¿Hay alguna base de principios para tal distinción?

La base más clara que puedo sugerir es la siguiente. Conocemos un algoritmo que toma como entrada un entero positivo $k$, y que produce los dígitos decimales de $p = 2^k - 1$ usando un número de pasos que es polinomial – de hecho, lineal – en el número de dígitos de $p$. Pero no conocemos ningún algoritmo similarmente eficiente que demostradamente dé salida al primer primo mayor que $2^k - 1$.[18]

---

13. Reconocidamente, podría demostrarse que la selección natural darwiniana requeriría tiempo exponencial para producir alguna funcionalidad, sin demostrar por eso que cualquier algoritmo requeriría tiempo exponencial.
14. Esta sección estuvo inspirada por una pregunta de A. Rupinski en el website MathOverflow. Ver mathoverflow.net/questions/62925/philosophical-question-related-to-largest-known-primes/
15. www.mersenne.org
16. Aquí queremos decir implícitamente: "conocido" por toda la comunidad matemática, en lugar de conocido por algún individuo particular cuando se pregunta por ello. La Sección 5 considerará el último tipo de conocimiento.
17. Por ejemplo, se podría usar el Teorema de Chebyshev (también llamado Postulado de Bertrand) que dice que para todo N > 1 existe un primo entre N y 2N.
18. La *Conjetura de Cramer* indica que el espaciado entre dos primos consecutivos de $n$ dígitos nunca excede $n^2$. Esta conjetura parece asombrosamente difícil: suponiendo incluso la Hipótesis de Riemann,

### 3.4 Resumen

El punto de estos ejemplos fue ilustrar que, más allá de su utilidad para la ciencia de la computación teórica, la brecha polinomial/exponencial es también un territorio fértil para la filosofía. Pienso en la brecha polinomial/exponencial como ocupando un "punto medio" entre otras dos clases de brechas: por una parte, pequeñas brechas cuantitativas (como la brecha entre $n$ pasos y $2^n$ pasos); y por otra parte, la brecha entre un número finito de pasos y un número infinito. La dificultad con pequeñas brechas cuantitativas es que son demasiado sensibles a las opciones de modelado "mundanas" y a los detalles de la tecnología. Pero la brecha entre finito e infinito tiene el problema opuesto: es tranquilamente insensible a las distinciones que realmente nos interesan, tales como la que existe entre encontrar una solución y verificarla, o entre la física clásica y la cuántica.[19] La brecha polinomial/exponencial evita ambos problemas.

## 4 Complejidad Computacional y el Test de Turing

*Puede pensar una computadora*? Por casi un siglo, las discusiones sobre este asunto han combinado frecuentemente dos cuestiones. La primera es la cuestión "metafísica":

> Suponiendo que un programa de computación haya pasado el Test Turing (o una variante fuerte dicho test como se desee definir,[20] ¿sería correcto atribuirle "conciencia", "*qualia*", "*aboutness*", "intencionalidad", "subjetividad", "personeidad"**,** o cualquier otra condición encantada que deseemos atribuir a otros seres humanos y a nosotros mismos?[*]

La segunda es la cuestión "práctica":

> ¿Podría escribirse un programa de computadora que pasara (una versión fuerte del) Test de Turing? ¿Hay alguna razón fundamental por la que no se pudiera?

Por supuesto, fue precisamente en un intento de separar estas cuestiones que Turing propuso en primer lugar el test que lleva su nombre. Pero a pesar de sus esfuerzos, una característica familiar del hasta hoy en día argumento contra la inteligencia artificial, afirma primero la imposibilidad metafísica de ésta, y luego trata de reforzar esa posición con afirmaciones acerca de las dificultades prácticas de la AI: "Seguro", dicen, "un programa de computadora podría imitar unos minutos de burla ingeniosa, pero a diferencia de un ser

---

sólo se conoce cómo deducir el límite superior mucho más débil $\sim n2^{n/2}$. Pero interesantemente, si se demuestra la Conjetura de Cramer, expresiones como "el primer primo mayor que $2^k - 1$" definirán entonces "primos conocidos" según mi criterio.

19. En particular, es fácil verificar que el conjunto de funciones computables no depende de si definimos computabilidad con respecto a una máquina clásica o cuántica de Turing o a una determinística o no determinística. A lo sumo, estas opciones pueden cambiar un tiempo de ejecución de máquina de Turing por un factor exponencial, lo cual es irrelevante para la teoría de la computabilidad.

20. El Test de Turing, propuesto por Alan Turing [130] en 1950, es una prueba donde un juez humano interactúa con otro humano o un programa de computadora de conversación, tipeándose los mensajes de un lado y del otro. El programa "pasa" el Test si el juez no puede confiablemente distinguir el programa del interlocutor humano.
Por una "variante fuerte" del Test de Turing, quiero decir que además de la conversación usual de teletipo, se pudieran agregar pruebas adicionales que requirieran visión, audición, tacto, olfato, oralidad, escritura, expresiones faciales, danza, práctica de deportes y ejecución de instrumentos musicales, etc. – aunque muchos humanos perfectamente inteligentes pudieran ser incapaces de pasar dichas pruebas.

*** Nota del traductor.** Se dejan sin traducir: **aboutness**, que "es la relación que sostienen los elementos significantes con o de lo que sea en lo que estén o que aborden o les incumba" [Yablo Stephen (2014). *Aboutness*. Princeton University Press], y **qualia**, que son propiedades "de datos sensoriales" o "intrínsecas no representacionales" o "intrínsecas, no físicas, inefables", según la revisión de 2015 de la *Standford Encyclopedia of Philosophy*. Se creó el neologismo en español **personeidad** para *personhood*, es decir, "lo que es ser una persona", según la *Standford Encyclopedia of Philosophy*.

humano, nunca mostraría miedo o enojo o celos, o compondría sinfonías, o envejecería, o se enamoraría ...”

La pregunta de seguimiento obvia – ¿y si un programa *hiciera* todas esas cosas? – frecuentemente se deja sin contestar, o bien se responde listando más cosas que un programa de computadora nunca evidentemente podría hacer. Debido a esto, sospecho que muchas personas que *dicen* considerar a la inteligencia artificial una imposibilidad metafísica, realmente la consideran sólo una imposibilidad práctica: simplemente no han llevado el experimento mental requerido lo suficiente lejos para ver la diferencia entre las dos.[21] Incidentalmente, este es un caso bien claro, hasta donde sé, del cual la gente se beneficiaría si estudiara más filosofía.

Por lo tanto, los argumentos anti-AI que más me interesan han sido siempre los que se dirigen a la cuestión práctica desde la salida, proponiendo pruebas empíricas de “espada en la piedra” (en la frase de Daniel Dennett [46]) que, se afirma, los humanos pueden pasar pero las computadoras no. La prueba más famosa es probablemente la que se basa en el teorema de incompletitud de Gödel, tal como propuso John Lucas [92] y elaboró Roger Penrose en sus libros *The Emperor’s New Mind* [103] y *Shadows of the Mind* [104].

Brevemente, Lucas y Penrose argumentaron que, según el Teorema de Incompletitud, una cosa que una computadora nunca puede llevar a cabo haciendo deducciones vía reglas formales fijas, es “ver” la consistencia de sus propias reglas. Todavía esto, afirman, es algo que los matemáticos humanos *pueden* hacer, vía alguna clase de percepción intuitiva de realidad platónica. Luego, los humanos (o al menos, los matemáticos humanos) nunca pueden ser simulados por máquinas.

Los críticos señalaron numerosos huecos en este argumento,[22] a lo que Penrose respondió extensamente en *Shadows of the Mind*, inconvincentemente en mi opinión. Sin embargo, incluso antes que analicemos alguna propuesta de prueba de espada en la piedra, me parece que hay una pregunta mucho más básica. A saber, ¿qué se quiere decir cuando se dice que se tiene una tarea que los “humanos pueden realizar pero las computadoras no”?

### 4.1 El Argumento de la Tabla de Búsqueda

Hay una dificultad fundamental aquí que fue notada por otros en un contexto ligeramente diferente [30, 102, 88, 118]. Permítanme explicar primero la dificultad, y luego discutir la diferencia entre mi argumento y los anteriores.

---

21. Una excepción famosa es John Searle [115], quién ha dejado claro que, si (digamos) su mejor amigo resultara ser controlado por un microchip en lugar que por un cerebro, entonces consideraría que su amigo nunca había sido una persona en absoluto.

22. Ver Dennett [46] y Chalmers [37] por ejemplo. Para resumir:
(1) ¿Por qué deberíamos suponer que una computadora opera dentro de un sistema formal cognitivamente-sólido? Si concedemos a una computadora la misma libertad para cometer errores ocasionales que concedemos a los humanos, entonces el Teorema de Incompletitud ya no es relevante.
(2) ¿Por qué deberíamos suponer que los matemáticos humanos tienen percepción directa de “realidad Platónica”? Matemáticos humanos (como Frege) han estado antes equivocados sobre la consistencia de sistemas formales
(3) Una computadora podría, por supuesto, programarse para emitir “creo que el sistema formal *F* es consistente” e incluso emitir las respuestas a varias preguntas de seguimiento sobre por qué cree esto. Así que al argumentar que tales armamentos realmente no “contarían realmente” (porque no reflejarían verdadera comprensión”), críticos de la AI como Lucas y Penrose se ven obligados a retirarse de su visión de una “prueba de espada-en-la-piedra” empírica, y recurrir a otros criterios no especificados, relacionados con la estructura interna de la AI. ¿Pero entonces por qué poner la espada en la piedra en primer lugar?

En la práctica, las personas juzgan que son mutuamente conscientes luego de interactuar durante un tiempo muy corto, quizás tan pequeño como unos segundos. Esto sugiere que podemos poner un límite superior finito – para ser generosos, digamos $10^{20}$ – al número de bits de información que dos personas A y B intercambiarían realísticamente, antes que A haya acumulado suficiente evidencia para concluir que B era consciente.[23] Ahora imagina una tabla de búsqueda que almacena cada posible historia H de la conversación A y B, y próxima a H, la acción $f_B(H)$ que B *tomaría* dada esa historia. Por supuesto, como en La Biblioteca de Babel de Borges, la tabla de búsqueda consistiría casi completamente de sinsentido carente de significado, y sería también demasiado grande para caber dentro del universo observado. Pero todo lo que nos importa es que la tabla de búsqueda sería *finita*, por la suposición de que hay un límite superior finito en la longitud de la conversación. Esto implica que la función $f_B$ es computable (de hecho, puede ser reconocida por un autómata finito). De estas simples consideraciones, concluimos que si *hay* un obstáculo fundamental para que las computadoras pasen el Test de Turing, entonces no será encontrado en la teoría de la computabilidad. [24]

En *Shadows of the Mind* (Sombras de la Mente) [104, pág. 83], Penrose reconoce este problema, pero da una respuesta enigmática e insatisfactoria:

> Uno podría igualmente bien imaginar computadoras que contengan nada más que listas de 'teoremas' matemáticos totalmente falsos, o listas que contengan mezclas aleatorias de verdades y falsedades. ¿Cómo vamos a decir en qué computadora confiar? Los argumentos que estoy tratando de hacer aquí no dicen que es imposible una simulación efectiva de la producción de actividad humana consciente (aquí matemática), puesto que por pura casualidad podría 'suceder' que la computadora la consiguiera correctamente – incluso sin ningún tipo de comprensión. Pero las probabilidades contra esto son absurdamente enormes, y las cuestiones que están siendo tratadas aquí, a saber, cómo se decide cuáles expresiones matemáticas son verdaderas y cuáles falsas, ni siquiera están siendo tocadas...

La dificultad con esta respuesta es que de nuevo equivale a un retroceso de la prueba de la espada-en-la-piedra, a criterios internos más oscuros. Si, al final, vamos a tener de cualquier modo que mirar dentro de la computadora para determinar si verdaderamente "entiende" sus respuestas, *entonces, ¿por qué no prescindir de la teoría de la computabilidad desde el comienzo?* Porque la teoría de la computabilidad sólo consigna si, o no, existen máquinas de Turing para resolver varios problemas, y ya hemos visto que ese no es el asunto relevante.

A mi entender, hay una dirección que Penrose podría tomar desde este punto para evitar la incoherencia – aunque decepcionantemente, no es la dirección que escoge. Es decir, podría señalar que, mientras que la tabla de búsqueda "funciona", requiere recursos

---

23. Las personas que interactúan sobre Internet, vía correo electrónico o mensajes instantáneos, regularmente se juzgan mutuamente como seres humanos en lugar de spam-bots, después de intercambiar un número mucho menor de bits. En cualquier caso, consideraciones cosmológicas sugieren un límite superior de aproximadamente $10^{122}$ bits en cualquier proceso observable [34].

24. Algunos lectores pueden notar una tensión aquí: Expliqué en la Sección 2 que los teóricos de la complejidad se preocupan por el comportamiento asintótico cuando el tamaño $n$ del problema va a infinito. ¿Por qué ahora estoy diciendo que, para los propósitos del Test de Turing, debemos restringir la atención a valores finitos de $n$ tales como $10^{20}$? Hay dos respuestas para esta pregunta. La primera es que, a diferencia de los problemas matemáticos como la factorización o el problema de la detención, no está claro si tiene sentido generalizar el Test de Turing a longitudes de conversación arbitrarias: porque el Test de Turing se define en términos de seres humanos y la capacidad conversacional humana es finita. La segunda respuesta es que, en la medida en que tiene sentido generalizar el Test de Turing a longitudes de conversación arbitrarias $n$, me interesa saber si la complejidad asintótica de pasar el test crece polinomialmente o exponencialmente con $n$ (como lo explica el resto de la sección).

computacionales que crecen exponencialmente con la duración de la conversación. Esto conduciría a la siguiente especulación:

> (*) *Cualquier* programa de computadora que pasase el Test de Turing tendría que ser exponencialmente ineficiente en la longitud de la prueba – medida en algún recurso como tiempo, uso de memoria o el número de bits necesarios para escribir el programa. En otras palabras, la astronómica tabla de búsqueda es esencialmente lo mejor que se puede hacer.[25]

De ser verdadera, la especulación (*) haría lo que quiere Penrose: implicaría que el cerebro humano ni siquiera puede ser *simulado* por computadora, dentro de las restricciones de recursos del universo observable.

Además, a diferencia de la exigencia de computabilidad más temprana, (*) tiene la ventaja de no ser trivialmente falsa. Por otro lado, para decirlo suavemente, (*) tampoco es trivialmente verdadera. Para los defensores de la AI, la falta de evidencia convincente para (*) es escasamente sorprendente. Después de todo, si crees que el cerebro *en sí* es básicamente una máquina de Turing clásica eficiente[26], ¡entonces tienes una explicación simple de por qué nadie ha demostrado que el cerebro no puede ser simulado por una máquina así! Sin embargo, la teoría de la complejidad también deja en claro que, aun si *supusiéramos que (*) se mantiene*, habría pocas esperanzas de *demostrarlo* en nuestro estado actual de conocimiento matemático. Después de todo, ni siquiera podemos demostrar conjeturas plausibles bien-definidas tales como $\mathsf{P} \neq \mathsf{NP}$.

### 4.2 Relación con Trabajo Previo

Como se mencionó antes, estoy lejos de ser la primera persona en preguntar acerca de los recursos computacionales usados para pasar el Test de Turing, y si ellos escalan polinomialmente o exponencialmente con la longitud de la conversación. Mientras muchos escritores ignoran esta distinción crucial, Block [30], Parberry [102], Levesque [88], Shieber [118] y algunos otros la discutieron explícitamente. La diferencia principal es que las discusiones anteriores tuvieron lugar en el contexto del argumento del Salón Chino de Searle [115].

Brevemente, Searle propuso un experimento mental – los detalles no nos conciernen aquí – pretendiendo mostrar que un programa de computadora podría pasar el Test de Turing, aunque le faltara manifiestamente algo que una persona razonable llamaría "inteligencia" o "entendimiento" (de hecho, Searle arguye que ningún sistema físico puede entender algo "puramente en virtud de" los cómputos que implementa). En respuesta, muchos críticos dijeron que el argumento de Searle estaba profundamente desencaminado, porque nos animaba implícitamente a imaginarnos un programa de computadora que era simplista en sus operaciones internas – algo así como la tabla de búsqueda gigante descrita en la Sección 4.1. Y aunque era verdad, continuaron los críticos, que una tabla de búsqueda gigante no "comprendería verdaderamente" sus respuestas, ese punto también es *irrelevante*. Porque la

---

25. Como me señaló Gil Kalai, podría especularse en cambio, que existe un programa de computadora eficiente para pasar el Test de Turing, pero encontrar tal programa requeriría recursos computacionales exponenciales. En esa situación, el cerebro humano podría de hecho ser simulado eficientemente por un programa de computadora, ¡pero quizá no por un programa que los humanos pudieran alguna vez escribir!

26. Aquí, por una máquina de Turing M "eficiente", queremos decir que el tiempo de ejecución, uso de memoria y tamaño del programa de M son suficientemente modestos para que no haya ningún problema real de principio para entender como M pudiera ser simulada por un sistema físico clásico consistiendo en $\sim 10^{11}$ neuronas y $\sim 10^{14}$ sinapsis. Por ejemplo, una máquina de Turing que contenga una tabla de búsqueda de tamaño $10^{10^{20}}$ no sería eficiente en ese sentido.

tabla de búsqueda gigante es, de todos modos, una ficción filosófica: ¡algo que ni siquiera cabe en el universo observable! Si imaginamos en cambio un programa de computadora *compacto y eficiente* pasando el Test de Turing, entonces la situación cambia drásticamente. Porque ahora, para *explicar* cómo el programa puede ser tan compacto y eficiente, necesitaremos postular que incluye representaciones de conceptos abstractos, capacidades para aprendizaje y razonamiento, y toda clase de otro mobiliario interno que esperaríamos encontrar en una mente.

Personalmente, encuentro esta respuesta a Searle sumamente interesante – puesto que si es correcta, sugiere que la distinción entre la complejidad polinomial y la exponencial tiene importancia metafísica. De acuerdo a esta respuesta, una tabla de búsqueda de tamaño exponencial que pasara el Test de Turing no sería sintiente (o consciente, inteligente, auto-consciente, etc.), pero un programa limitado polinomialmente con exactamente el mismo comportamiento de entrada/salida podría ser sintiente. Además, el último programa sería sintiente porque fue limitado polinomialmente. Sin embargo, aun cuando ese criterio para la sintiencia halaga mi orgullo de teórico de la complejidad, me encuentro renuente a tomar una posición sobre una materia tan pesada. Mi punto, en la Sección 4.1, era más simple y (esperanzadamente) menos controversial: a saber, que si quieres afirmar que pasar el Test de Turing es *totalmente imposible*, entonces, guste o no, *debes* hablar sobre complejidad en vez de computabilidad solamente. En otros términos, los escritores anteriores [30, 102, 88, 118] y yo estamos todos interesados en los recursos computacionales necesarios para pasar un Test de Turing de longitud $n$, pero por razones diferentes. Donde otros invocaron consideraciones de complejidad para discutir con Searle sobre la cuestión metafísica, yo estoy invocándolas para discutir con Penrose sobre la cuestión práctica.

### 4.3 ¿Pueden los Humanos Resolver Eficientemente Problemas NP-Completos?

En ese caso, ¿qué podemos realmente *decir* sobre la cuestión práctica? ¿Hay algunas razones para aceptar la afirmación que yo llamé (*) – la afirmación de que los humanos *no* son simulables eficientemente por máquinas de Turing? Considerar esta cuestión nos conduce inmediatamente a algunas posibilidades especulativas. Así por ejemplo, si resultara que los humanos pudieran resolver instancias arbitrarias de problemas NP-completos en tiempo polinomial, entonces eso constituiría ciertamente excelente evidencia empírica para (*).[27] Sin embargo, a pesar de las ocasionales afirmaciones en contrario, no veo personalmente ninguna razón para creer que los humanos pueden resolver problemas NP-completos en tiempo polinomial, y excelentes razones para creer lo opuesto.[28]

Recordar, por ejemplo, que el problema de factorizar enteros está en NP. Así, si los humanos pudieran resolver problemas NP-completos, entonces presumiblemente debiéramos poder factorizar también números enormes. Pero factorizar no parece ser

---

27. ¡Y divertidamente, si pudiéramos resolver problemas NP-completos, entonces presumiblemente encontraríamos mucho más fácil demostrar que las computadoras *no podrían* resolverlos!

28. ¡De hecho, ni siquiera está claro para mí si debiéramos pensar en los humanos como capaces de resolver todos los problemas P eficientemente, dejemos aparte los problemas NP-completos! Recordar que P es la clase de problemas que son resolubles en tiempo polinomial por una máquina de Turing determinística. Se conocen muchos problemas que pertenecen a P por razones bastante sofisticadas: dos ejemplos son probar si un número es primo (¡aunque no lo factorice!) [9] y probar si un gráfico tiene una perfecta coincidencia. En principio, por supuesto, un humano podría ejecutar laboriosamente los algoritmos de tiempo polinomial para tales problemas usando lápiz y papel. ¿Pero es legítimo el uso de lápiz y papel, donde el uso de una computadora no lo sería? ¿Cuál es el poder computacional del intelecto humano "sin ayuda"? El reciente trabajo de Drucker [51] que muestra cómo usar una colección de fotografías de stock para incrementar la "memoria efectiva" disponible para cálculos mentales, proporciona una perspectiva empírica fascinante sobre estas cuestiones.

exactamente el candidato más prometedor para una prueba de espada-en-la-piedra: es decir, una tarea que es fácil para los humanos pero difícil para las computadoras. ¡Hasta donde cualquiera sabe hoy, factorizar es difícil para los humanos y las computadoras (clásicas) por igual, aunque con una ventaja definida del lado de las computadoras!

El punto básico apenas puede enfatizarse bastante: cuando teóricos de la complejidad hablan sobre problemas "intratables", generalmente quieren decir problemas matemáticos que toda nuestra experiencia nos lleva a creer que son al menos tan difíciles para humanos como para computadoras. Esto sugiere que, *aún si* los humanos no fueran eficientemente simulables por máquinas de Turing, la "dirección" en la que fueron difíciles de simular, sería casi con certeza, diferente de las direcciones usualmente consideradas en la teoría de la complejidad. Veo dos modos (hipotéticos) de que esto pudiera suceder.

Primero, las tareas en las que los humanos fueran singularmente buenos – como pintar o escribir poesía – podrían ser *incomparables* con tareas matemáticas como resolver problemas NP-completos, en el sentido de que ninguna fuera eficientemente reducible a la otra. Esto significaría, en particular, de que no podría haber ningún algoritmo de tiempo polinomial incluso para *reconocer* arte mayor o poesía (puesto que si tal algoritmo existiera, entonces la tarea de *componer* arte mayor o poesía podría estar en NP). Dentro de la teoría de la complejidad, es conocido que allí existen pares de problemas que son incomparables en ese sentido. Como un ejemplo plausible, nadie sabe actualmente cómo reducir la simulación de computadoras cuánticas para la solución de problemas NP-completos *o* viceversa.

Segundo, los humanos podrían tener la habilidad de resolver interesantes *casos especiales* de problemas NP-completos más rápido que cualquier máquina de Turing. Así por ejemplo, aún cuando las computadoras fueran mejores que los humanos en factorizar grandes números o en resolver enigmas de Sudoku generados al azar, los humanos todavía podrían ser mejores en explorar problemas con "alto nivel de estructura" o "semántica", tales como demostrar el Último Teorema de Fermat o (irónicamente) diseñando algoritmos de computadora más rápidos. De hecho, incluso en dominios limitados como el de la solución de enigmas, mientras las computadoras pueden examinar soluciones millones de veces más rápido, los humanos (por ahora) son vastamente mejores en notar *patrones globales* o *simetrías* en el enigma que hacen que una solución sea trivial o imposible. Como un ejemplo entretenido, considérese el *Principio del Palomar* que dice que no pueden ser colocadas $n + 1$ palomas en $n$ agujeros, con a lo sumo una paloma por agujero. No es difícil construir una fórmula booleana preposicional $\varphi$ que codifica el Principio del Palomar para algún valor fijo de $n$ (digamos, 1000). Sin embargo, si entonces alimentas $\varphi$ a los algoritmos de satisfabilidad booleanos actuales, ellos se pondrán a trabajar asiduamente probando posibilidades: "Veamos, si yo pusiera esta paloma aquí, y esa allí... ¡diablos, *todavía* no funciona! "Y ellos continuarán probando las posibilidades para un número exponencial de pasos, olvidados de la razón "global" por la cual la meta nunca puede lograrse. De hecho, empezando en la década de 1980, el campo de *complejidad de prueba* – un primo cercano de la complejidad computacional – ha podido mostrar que grandes clases de algoritmos requieren tiempo exponencial para demostrar el Principio del Palomar y tautologías proposicionales similares (ver Beame y Pitassi [24] para un estudio).

Todavía, si queremos construir nuestra prueba de la espada-en-la-piedra sobre la habilidad de detectar "patrones del más alto nivel" en problemas de búsqueda combinatoria, entonces la carga está en nosotros para explicar lo que queremos decir por patrones-de-alto-nivel, y por qué pensamos que ninguna máquina de Turing en tiempo polinomial – incluso mucho más sofisticadas que las que podemos imaginar hoy – podría detectar también esos

patrones. Para un esfuerzo inicial por entender problemas NP-completos desde una perspectiva de la ciencia cognitiva, ver Baum [22].

### 4.4 Resumen

Mi conclusión es que, si te opones en principio a la posibilidad de la AI, entonces

(i) Puede tomar la "ruta metafísica" (como hace Searle [115] con el Salón Chino), concediendo la posibilidad de un programa de computadora de pasar toda prueba concebible para inteligencia, pero arguyendo que no es suficiente, o

(ii) Puede conjeturar un astronómico límite inferior en los recursos necesarios para ejecutar tal programa o escribirlo en primer lugar – pero aquí hay poca cuestión de prueba para el futuro previsible.

Crucialmente, debido al argumento de la tabla de búsqueda, una opción que no tienes es afirmar la imposibilidad llana de un programa de computadora de pasar el Test de Turing, sin mención de límites cuantitativos de complejidad.

## 5 El Problema de la Omnisciencia Lógica

Una de las preocupaciones centrales de la filosofía analítica moderna es dar cuenta formal del *conocimiento*; la literatura incluso es demasiado vasta para examinarla aquí (no obstante, ver Fagin *et a*l. [54] para una visión general amigable de la ciencia de la computación).

Típicamente, las cuentas formales del conocimiento implican axiomas "lógicos" convencionales, como

- Si conoces $P$ y conoces $Q$, entonces también conoces $P \wedge Q$

Complementadas por axiomas "modales" que tienen que ver con el conocimiento mismo, tales como

- Si conoces $P$, entonces también conoces que conoces $P$
- Si no conoces $P$, entonces conoces que no conoces $P$[29]

Mientras los detalles difieren, lo que la mayoría de las cuentas formales del conocimiento tienen en común es que tratan el conocimiento de un agente como *cerrado* bajo la aplicación de varias reglas de deducción como las de arriba. En otros términos, los agentes son considerados *lógicamente omniscientes*: si conocen ciertos hechos, entonces también conocen todas las consecuencias lógicas posibles de esos hechos.

Tristemente y obviamente, ningún mortal ha alcanzado o incluso aproximado a esta clase de omnisciencia (recordar la cita de Turing al comienzo de la Sección 1). Así por ejemplo, yo puedo conocer las reglas de la aritmética sin conocer el Último Teorema de Fermat, y puedo conocer las reglas del ajedrez sin saber si las blancas tienen una victoria forzada. Además, la dificultad *no* está (como a veces se exigió) limitada a unos dominios, como matemática y juegos. Como señaló Stalnaker [126], si suponemos omnisciencia lógica, entonces ya no podríamos dar cuenta de *ninguna* contemplación de hechos ya conocidos por nosotros – y por lo tanto, para la actividad principal y uno de los asuntos principales de la filosofía misma.

Ahora podemos expresar flojamente lo que Hintikka [73] llamó el *problema de la omnisciencia lógica*:

---

29. No es sorprendente que estos axiomas particulares hayan generado controversia: no dejan ninguna posibilidad para Rumsfeldianos "desconocidos desconocidos" o (una categoría que él dejó de lado) "conocidos desconocidos

> ¿Podemos dar alguna cuenta formal de "conocimiento" capaz de acomodar gente que aprende cosas nuevas sin dejar sus sillones?

¡Por supuesto, una "solución" vacua sería declarar que tu conocimiento es simplemente una lista de todas las oraciones verdaderas[30] que "conoces" – y que sea así, si sucede que la lista no es cerrada bajo deducciones lógicas! Pero esta "solución" no es de ayuda en absoluto para explicar *cómo* o *por qué* conoces cosas. ¿No podemos hacerlo mejor?

Intuitivamente, queremos decir que su "conocimiento" consiste en varios hechos ("la grama es verde"), junto con *algunas* consecuencias lógicas simples de esos hechos ("la grama no es rosa"), pero no necesariamente *todas* las consecuencias, y ciertamente no todas las consecuencias que implican razonamiento matemático difícil. Desgraciadamente, tan pronto como intentemos formalizar esta idea, nos metemos en problemas.

El problema más obvio es la falta de un límite afilado entre los hechos que conoces inmediatamente, y aquéllos que podrías "conocer", pero sólo después de pensamiento significante (recordar la discusión de los "primos conocidos" de la Sección 3.3). Un problema relacionado es la falta de un límite afilado entre los hechos que conoces "sólo si te preguntan por ellos", y aquéllos que conoces aun cuando *no* te pregunten. Interesantemente, estos dos límites parecen cortarse entre sí. Por ejemplo, mientras probablemente ya has encontrado el hecho que 91 es compuesto, podría tomarte algún tiempo recordarlo; mientras que probablemente nunca te has encontrado con el hecho de que 83190 es compuesto, una vez que te preguntan, probablemente puedes asentir inmediatamente a ello.

Pero como discutió Stalnaker [126], hay un tercer problema que parece mucho más serio que cualquiera de los dos anteriores. ¡A saber, podrías "conocer" un hecho particular si se te pregunta por él de un modo, pero no si te preguntan de un modo diferente! Para ilustrar esto, Stalnaker usa un ejemplo que podemos reconocer inmediatamente de la discusión del problema de P versus NP en la Sección 3.1. Si yo te preguntara si 43 x 37 = 1591, probablemente podrías contestar fácilmente (por ejemplo, usando $(40 + 3)(40 - 3) = 40^2 - 3^2$). Por otra parte, si te preguntara en cambio cuáles fueron los factores primos[31] de 1591, probablemente *no podrías* contestar tan fácilmente.

> Pero las respuestas a las dos preguntas tienen el mismo contenido, incluso en una noción de contenido de grano muy fino. Supón que fijamos el umbral de accesibilidad para que la información que 43 y 37 son los primeros factores de 1591 sea accesible en respuesta a la segunda pregunta, pero no accesible en respuesta a la primera. ¿Sabes o no cuáles son los factores primos de 1591? ... Nuestro problema es que no sólo estamos tratando de decir lo que un agente conocería al hacerle ciertas preguntas; en su lugar, estamos tratando de usar los hechos acerca de las capacidades de respuesta a preguntas de un agente para llegar a lo que el agente conoce, aun si las preguntas no se hacen. [126, pág. 253]

Para agregar otro ejemplo: ¿"sabe" una niña típica de cuatro años que la suma de números reales es conmutativa? Ciertamente no si le hacemos la pregunta con esas palabras – y si tratáramos de *explicar* las palabras, ella probablemente no nos entendería. Sin embargo, si le mostráramos una pila de libros, y le preguntáramos si podría hacer la pila más alta barajando los libros, probablemente no cometería un error que implicara imaginar que la suma no era conmutativa. En ese sentido, podríamos decir ella "implícitamente" sabe lo que sus clases de matemática harán explícito más adelante.

---

30. Si no requerimos que las oraciones sean verdaderas, entonces presumiblemente estamos hablando acerca de creencia en lugar de conocimiento.
31. Si careces del *concepto* de un factor primo entonces simplemente podría en cambio pedirte dos números de dos dígitos que den un resultado de 1591 cuando se multiplican.

Desde mi óptica, éstos y otros ejemplos sugieren fuertemente que sólo una parte pequeña de lo que queremos significar por "conocimiento" es conocimiento sobre la verdad o falsedad de proposiciones individuales. Y crucialmente, esto permanece así aun si restringimos nuestra atención a conocimiento "puramente verbalizable" – de hecho, el conocimiento usado para contestar preguntas factuales – y no (digamos) conocimiento de cómo montar una bicicleta o hacer swing con un palo de golf, o conocimiento de una persona o de un lugar.[32] Muchos usos cotidianos de la palabra "conocer" soportan esta idea:

¿Conoces cálculo?
¿Conoces español?
¿Conoces las reglas del bridge?

Cada una de las preguntas anteriores podría interpretarse como preguntar: *¿posees un algoritmo interior, por el cual puedes responder de alguna forma a un gran (y posiblemente ilimitado) conjunto de preguntas?* Mientras esto raramente se hace explícito, los ejemplos de esta sección y de la Sección 3.3 sugieren agregar la salvedad: … *responder en una razonable cantidad de tiempo.*

Pero supón que aceptamos que "conocer cómo" (o "conocer un buen algoritmo para") es un concepto más fundamental que "conocer qué". ¿Cómo eso nos ayuda *en absoluto* a resolver el problema de la omnisciencia lógica? Podrías preocuparte de que regresamos justo adonde comenzamos. Después de todo, si tratamos de dar una cuenta formal de "conocer cómo", entonces simplemente como en el caso de "conocer qué", sería tentador escribir axiomas como el siguiente:

> Si sabes como computar $f(x)$ y $g(x)$ eficientemente, entonces también sabes como computar eficientemente $f(x) + g(x)$.

Naturalmente, querremos tomar el cierre lógico de esos axiomas. Pero entonces, antes de que lo conozcamos, ¿no habremos conjurado en nuestras imaginaciones un súper-ser computacionalmente omnisciente que pudiera computar algo?

### 5.1 Los Axiomas de Cobham

Como bien se sabe, en la década de 1930, Church y Kleene propusieron definiciones de la "funciones calculables" que resultaron precisamente equivalentes a la definición de Turing, pero que difirieron de ésta en no hacer ninguna mención explícita a máquinas. En lugar de analizar el *proceso* de computación, el enfoque Church-Kleene fue simplemente listar *axiomas* que las funciones computables de números naturales $f$: $\mathbb{N} \rightarrow \mathbb{N}$ debían satisfacer – por ejemplo, "si $f(x)$ y $g(x)$ son ambas computables, entonces así es $f(g(x))$" – y luego definir "las" funciones computables como el conjunto más pequeño que satisfacía esos axiomas.

En 1965, Alan Cobham [42] preguntó si lo mismo podría hacerse para funciones eficientemente o factiblemente computables. Como respuesta, ofreció axiomas que caracterizan precisamente lo que hoy llamamos FP, o Función de Tiempo Polinomial (aunque Cobham la llamó $\mathcal{L}$). La clase FP consiste en todas las funciones de números naturales $f : \mathbb{N} \rightarrow \mathbb{N}$ que son computables en tiempo polinomial por una máquina de Turing determinística. Note que FP es "moralmente" lo mismo que la clase P (Tiempo Polinomial) definido en la Sección 3.1: sólo difieren en que P es una clase de problemas de *decisión* (o

---

32. Porque "conocer" a una persona sugiere haber encontrado realmente a la persona, mientras que "conocer" un lugar sugiere que se ha visitado el lugar. Interesantemente, en hebreo se usa un verbo completamente diferente para "conocer" en el sentido de "estar familiarizado con" (makir) que para "conocer" en el sentido intelectual (yodeya).

equivalentemente, funciones $f: \mathbb{N} \to \{0, 1\}$, mientras que FP es una clase de funciones con rango entero.

Lo notable sobre la caracterización del tiempo polinomial de Cobham fue que no incluyó *ninguna* mención explícita a dispositivos de computación o límites en sus tiempos de ejecución. Permítaseme ahora listar una versión de los axiomas de Cobham, adaptada de Arora, Impagliazzo y Vazirani [16]. Cada uno de los axiomas habla acerca de cuales funciones de números naturales $f: \mathbb{N} \to \mathbb{N}$ son "eficientemente computables."

(1) Toda función constante $f$ es eficientemente computable, así como toda función que es no cero sólo finitamente frecuente.

(2) **Apareamiento**: Si $f(x)$ y $g(x)$ son eficientemente computables, entonces lo es $\langle f(x), g(x)\rangle$, donde $\langle , \rangle$ es alguna función de apareamiento normal para los números naturales.

(3) **Composición**: Si $f(x)$ y $g(x)$ es eficientemente computable, entonces lo es $f(g(x))$.

(4) **Bolsa de Sorpresas**: Las siguientes funciones son todas eficientemente computables:

- las funciones aritméticas $x + y$ y $x \times y$
- $[x] = [\log_2 x] + 1$ (el número de bits en la representación binaria de $x$)
- las funciones de proyección $\Pi_1(\langle x, y\rangle) = x$ y $\Pi_2(\langle x, y\rangle) = y$
- bit $(\langle x, y\rangle)$ (el $i^{ésimo}$ bit de la representación binaria de $x$, o 0 si $i > |x|$)
- diff $(\langle x, y\rangle)$ (el número obtenido de $x$ por inversión de su $i^{ésimo}$ bit)
- $2^{|x|^2}$ (llamada la "función de rotura")

(5) **Recursión Limitada**: Supón que $f(x)$ es eficientemente computable, y $|f(x)| < |x|$ para todo $x \in \mathbb{N}$.

Entonces la función $g(\langle x, k\rangle)$, definida por

$$g(\langle x, k\rangle) = \begin{cases} f(g(\langle x, \lfloor k/2 \rfloor\rangle)) & \text{si } k > 1 \\ x & \text{si } k = 1 \end{cases},$$

Unos pocos comentarios sobre los axiomas de Cobham podrían ser útiles. Primero, el axioma que "hace la mayor parte del trabajo" es (5). Intuitivamente, dado cualquier número natural $k \in \mathbb{N}$ que podemos generar comenzando desde la entrada original $x \in \mathbb{N}$, el axioma de la Recursión Limitada nos permite preparar un "proceso computacional" que se ejecuta para $\log_2 k$ pasos. Segundo, el rol de la "función de rotura", $2^{|x|^2}$, es permitirnos asignar enteros de $n$ bits a enteros de $n^2$ bits a enteros de $n^4$ bits y así sucesivamente, y por eso (en combinación con el axioma de la Recursión Limitada) fijar procesos computacionales que se ejecutan para números de pasos *polinomiales* arbitrarios. Tercero, aunque la suma y multiplicación están incluidas como "funciones eficientemente computables", es crucial que la exponenciación *no* está incluida. De hecho, si $x$ e $y$ son enteros de $n$ bits, entonces $x^y$ podría requerir exponencialmente muchos bits sólo para escribir.
El resultado básico es entonces el siguiente:

**Teorema 1** ([42, 112]) *La clase* FP*, de funciones* $f: \mathbb{N} \to \mathbb{N}$ *computable en tiempo polinomial por una máquina de Turing determinística, satisface los axiomas (1)-(5), y es la clase más pequeña que lo hace.*

Para demostrar el Teorema 1, es necesario hacer dos cosas, ninguna de ellas difícil: primero, mostrar que cualquier función *f* que puede definirse usando los axiomas de Cobham también puede computarse en tiempo polinomial; y segundo, mostrar que los axiomas de Cobham son suficientes para simular cualquier máquina de Turing de tiempo polinomial.

Un inconveniente de los axiomas de Cobham es que parecen "infiltrarse por la puerta trasera en el concepto de tiempo polinomial" – tanto por medio de la "función de rotura", como a través de la condición de búsqueda arbitraria $|f(x)| \leq |x|$ en el axioma (5). En la década de 1990, sin embargo, Leivant [87] y Bellantoni y Cook [25], dieron caracterizaciones lógicas más "elegantes" de FP que evitaban este problema. Así por ejemplo, Leivant mostró que una función *f* pertenece a FP, si y sólo si *f* se computa por un programa que puede demostrarse correcto en lógica de segundo orden, con comprensión restringida a fórmulas positivas de cuantificador libre. Resultados como estos proporcionan evidencia adicional – si alguna fuese necesaria – que la computabilidad de tiempo polinomial es una noción extremadamente natural: un "amplio blanco en espacio conceptual" que hasta se acierta mientras se apunta en direcciones puramente lógicas.

Durante unas pocas décadas pasadas, la idea de definir las clases de complejidad como P y NP en modos "lógicos, libres de máquinas" ha dado lugar a un campo entero llamado *teoría de la complejidad descriptiva* que tiene conexiones profundas con la teoría del modelo finito. Mientras una mayor discusión de la teoría de la complejidad descriptiva nos llevaría demasiado lejos de nuestro asunto, para una introducción definitiva ver el libro de Immerman [78], o Fagin [52] para un estudio.

### 5.2 Omnisciencia Versus Infinidad

Volviendo a nuestro tema original, exactamente, ¿cómo las teorías axiomáticas tales como la de Cobham (o la de Church y la de Kleene, por lo demás) escapan al problema de la omnisciencia? Una respuesta directa es que, a diferencia del conjunto de oraciones verdaderas en algún lenguaje formal, que es sólo *contablemente* infinito, el conjunto de funciones $f: \mathbb{N} \rightarrow \mathbb{N}$ es *incontablemente* infinito. Y por lo tanto, aunque definamos las funciones $f: \mathbb{N} \rightarrow \mathbb{N}$ "eficientemente computables" tomando un cierre lógico contablemente infinito, estamos seguros de perder *algunas* funciones *f* (¡en efecto, casi todas ellas!).

La observación anterior sugiere una estrategia general para domesticar el problema de la omnisciencia lógica. A saber, podríamos rechazar definir el "conocimiento" de un agente en términos de cuáles preguntas individuales puede responder rápidamente, e insistir en hablar sobre qué *familias* infinitas de preguntas puede responder rápidamente. En forma de eslogan, queremos "combatir omnisciencia con infinito".

Veamos cómo, al tomar esta ruta, podemos dar respuestas semi-creíbles a los rompecabezas sobre el conocimiento discutidos anteriormente en esta sección. En primer lugar, la razón por la que puede "saber" que 1591 = 43 x 37, pero al mismo tiempo no "conocer" los factores primos de 1591, es que, cuando hablamos de conocer las respuestas a estas preguntas, realmente queremos decir sabiendo *cómo* responderlas. Y como vimos, no tiene que haber ninguna contradicción en conocer un algoritmo de multiplicación rápido, pero *no* un algoritmo de factorización rápido, incluso si modelamos su conocimiento sobre algoritmos como deductivamente cerrado. Dicho de otro modo, al integrar las dos preguntas

> Q1 = "¿Es 1591 = 43 x 37?"
> Q2 = "¿Cuáles son los factores primos de 1591?"

*en infinitas familias de preguntas relacionadas*, podemos romper la simetría en medio del conocimiento implicado en responderlas.

Similarmente, podríamos pensar en un niño como poseedor de un algoritmo interno que, dada cualquier oración de la forma $x + y = y + x$ (para valores específicos de $x$ e $y$), inmediatamente produce *verdadero*, sin siquiera examinar $x$ e $y$. Sin embargo, el niño todavía no tiene la capacidad de procesar oraciones *cuantificadas*, como "$\forall x, y \in \mathbb{R}\ x + y = y + x$". En ese sentido, todavía carece del conocimiento explícito de que la adición es conmutativa.

Aunque la "cura" para la omnisciencia lógica esbozada arriba resuelve algunos rompecabezas, no es sorprendente que plantee muchos otros. Permítaseme terminar esta sección discutiendo tres objeciones principales a la "cura infinita".

La primera objeción es que simplemente hemos empujado el problema de la omnisciencia lógica a otro lugar. Porque supongamos que un agente "conoce" cómo computar todas las funciones en alguna clase restringida como FP. Entonces, ¿cómo podemos darle sentido a que el agente *aprende un nuevo algoritmo*? Una respuesta natural es que, incluso si tienes la "capacidad latente" para computar una función $f \in \mathsf{FP}$, es posible que no *sepas* que tienes la capacidad – ya sea porque no conoces un algoritmo adecuado, o porque *sí* conoces un algoritmo, pero no sabes que es un algoritmo para $f$. Por supuesto, si quisiéramos perseguir las cosas hasta el fondo, tendríamos que contar una historia sobre el *conocimiento de los algoritmos*, y cómo allí se evita la omnisciencia lógica. Sin embargo, ¡afirmo que esto representa progreso! Para advertir que, incluso sin tal historia, ya podemos explicar *algunos* fracasos de la omnisciencia lógica. Por ejemplo, la razón por la que no conoces los factores de un gran número podría *no* ser tu ignorancia de un método de factorización rápida, sino más bien que no existe tal método.

La segunda objeción es que, cuando abogaba por enfocarse en familias infinitas de preguntas en lugar de preguntas únicas en aislamiento, nunca especifiqué *qué* familias infinitas. La dificultad es que la misma pregunta podría generalizarse de modos diferentes descabellados. Como ejemplo, considera la pregunta

Q = "¿Es 432150 compuesto?"

Q es una instancia de un problema computacional que los humanos encuentran muy difícil: "dado un entero grande $N$, $N$ es compuesto?" Sin embargo, Q también es un caso de un problema computacional que los humanos encuentran muy fácil: "dado que un gran entero $N$ terminado en 0, ¿$N$ es compuesto?" Y de hecho, esperaríamos que una persona supiera la respuesta a Q si notó que 432150 *termina* en 0, pero no de otro modo. Para mí, lo que este ejemplo demuestra es que, si queremos discutir el conocimiento de un agente por lo que se refiere a las preguntas individuales como Q, entonces el problema pertinente será si allí *existe* una generalización G de Q, tal que el agente conoce un algoritmo rápido para responder preguntas de tipo G, y también reconoce que Q es de tipo G.

La tercera objeción es simplemente la normal sobre la relación entre complejidad asintótica y oraciones finitas. Por ejemplo, si modelamos el conocimiento de un agente usando los axiomas de Cobham, entonces podemos de hecho explicar por qué el agente no sabe jugar ajedrez perfecto en un tablero de $n$ x $n$, para valores *arbitrarios* de $n$.[33] Pero en un tablero normal de 8 x 8, jugar ajedrez perfecto requeriría "meramente" (digamos) $\sim 10^{60}$ pasos computacionales, lo cual es una constante y por consiguiente ciertamente polinomial. Así, estrictamente en base a los axiomas de Cobham, ¿qué explicación podríamos ofrecer posiblemente de por qué un agente racional que conocía las reglas de ajedrez 8 x 8, tampoco sabía como jugarlo óptimamente? Mientras esta objeción podría parecer

33. Porque se sabe que un ajedrez en tablero n x n es EXP-completo, y también se conoce que $\mathsf{P} \neq \mathsf{EXP}$. Ver Sección 10, y particularmente nota a pie 64, para más detalles.

devastadora, es importante entender que no es diferente de la objeción usual formulada contra argumentos teoréticos de complejidad, y puede darse la respuesta usual. A saber: las oraciones asintóticas *siempre* son vulnerables a volverse irrelevantes, si los factores constantes resultaran ser ridículos. Sin embargo, la experiencia ha demostrado que, por las razones que sean, raramente ocurre que se pueda suponer normalmente que una conducta asintótica "tiene fuerza explicativa hasta que se demuestre lo contrario" (la Sección 12 dirá más sobre la fuerza explicativa de las demandas asintóticas, como un problema que requiere el análisis filosófico).

### 5.3 Resumen

Debido a las dificultades señaladas en la Sección 5.2, mi propia visión es que la teoría de la complejidad computacional todavía no está cerca de "resolver" el problema de omnisciencia lógica, en el sentido de dar una cuenta formal satisfactoria del conocimiento que también evite hacer predicciones absurdas.[34] No tengo idea si tal cuenta siquiera es posible.[35] Sin embargo, lo que he intentado mostrar en esta sección es que la teoría de la complejidad proporciona un "caso límite" bien definido en el que el problema de la omnisciencia lógica *es* solucionable, tanto como pudiera esperarse. El caso límite es donde el tamaño de las preguntas crece sin el límite, y la solución se da allí por los axiomas de Cobham: los "axiomas de saber cómo", cuyo cierre lógico puede tomarse sin por eso invitar a la omnisciencia.

En otras palabras, cuando contemplamos el problema de la omnisciencia, afirmo que estamos frecuentemente en una situación similar a la que enfrentamos en física – donde podríamos estar sin poder entender algún fenómeno (digamos, la entropía gravitacional), excepto en casos límite tales como el de los agujeros negros. En epistemología como en física, los casos límite que más o menos entendemos ofrecen un punto de partida obvio para aquellos que desean asir el caso general.

## 6 Computacionalismo y Cascadas

Durante las últimas dos décadas, un cierto argumento sobre computación – que llamaré el *argumento de la cascada* – ha sido ampliamente discutido por filósofos de la mente.[36] Como el argumento del famoso Salón Chino de Searle [115], el argumento de la cascada busca mostrar que los cómputos son "inherentemente sintácticos", y nunca pueden ser "sobre" algo – y que por esta razón, la doctrina del "computacionalismo" es falsa.[37] Pero a diferencia del Salón Chino, el argumento de la cascada suplementa la escasa apelación a la intuición con una afirmación adicional: a saber, que el "significado" de un cómputo, en cualquier magnitud en que tenga uno, es siempre *relativo a algún observador externo*.

---

34. Es cierto que no he hecho justicia en esta sección a la gran literatura sobre omnisciencia lógica; los lectores que deseen cavar más profundamente deben ver Fagin y Halpern [53] por ejemplo.

35. Compra el pesimismo expresado por Paul Graham [69] sobre la representación de conocimiento en términos más generales:

> En la práctica, la lógica formal no es de mucha utilidad, porque a pesar de algún progreso en los últimos 150 años, todavía sólo somos capaces de formalizar un pequeño porcentaje de oraciones. Tal vez nunca lo hagamos mucho mejor, por la misma razón que la "representación de conocimiento" estilo década de 1980 nunca pudo haber funcionado; muchas oraciones pueden no tener ninguna representación más concisa que un enorme estado analógico del cerebro.

36. Ver Putnam [107, apéndice] y Searle [116] para dos instanciaciones del argumento (aunque los detalles formales de cualquiera de ellas no nos conciernen aquí).

37. "Computacionalismo" se refiere a la visión de que la mente es literalmente una computadora, y que el pensamiento es literalmente un tipo de cómputo.

Más concretamente, considera una cascada (aunque cualquier otro sistema físico con un espacio de estados suficientemente grande también serviría). No me refiero aquí a una cascada diseñada especialmente para realizar computaciones, sino a una cascada natural: digamos, las Cataratas de Niágara. Gobernadas por las leyes de la física, la cascada implementa alguna asignación (*mapping*) $f$ de un conjunto de posibles estados iniciales a un conjunto de posibles estados finales. Si aceptamos que las leyes de la física son reversibles, entonces $f$ también debe ser inyectiva. Ahora supón que restringimos la atención a algún subconjunto finito de posibles estados iniciales de $S$, con $|S| = n$. Entonces $f$ es exactamente una asignación uno a uno de $S$ a algún conjunto de salida $T = f(S)$ con $|T| = n$. Ahora la "observación crucial" es ésta: dada *cualquier* permutación $\sigma$ del conjunto de enteros $\{1,..., n\}$ a sí mismo, hay algún modo de etiquetar los elementos de $S$ y $T$ por enteros en $\{1,..., n\}$, tal que podemos interpretar $f$ como implementando $\sigma$. Por ejemplo, si permitiéramos $S = \{s_1,..., s_n\}$ y $f(s_i) = t_i$, entonces es suficiente para etiquetar el estado inicial $s_i$ por $i$ y el estado final $t_i$ por $\sigma(i)$. Pero la permutación $\sigma$ podría tener cualquier "semántica" que nos guste: podría representar un programa para jugar ajedrez, o factorizar enteros, o simular una cascada diferente. Por consiguiente, el "mero cómputo" no puede dar lugar a un significado semántico. Aquí es cómo Searle [116, pág. 57] expresa la conclusión:

> Si somos consistentes adoptando el Test de Turing o algún otro "criterio objetivo" para la conducta inteligente, entonces la respuesta a preguntas tales como "¿Pueden porciones de materia no inteligente producir conductas inteligentes?" e incluso, "¿Cómo lo hacen exactamente?" son absurdamente obvias. Cualquier termóstato, calculadora de bolsillo o cascada produce "conducta inteligente", y nosotros sabemos en cada caso cómo funciona. Se diseñan ciertos artefactos para comportarse como si fueran inteligentes, y puesto que todo sigue leyes de la naturaleza, entonces, todo tendrá alguna descripción bajo la cual se comporta como si fuera inteligente. Pero este sentido de "conducta inteligente" no tiene relevancia psicológica alguna en absoluto.

El argumento de la cascada ha sido criticado sobre numerosos fundamentos: ver Haugeland [72], Block [30], y sobre todo Chalmers [37] (quién parodió el argumento demostrando que una receta de torta, siendo meramente sintáctica, nunca puede dar lugar al atributo semántico de migacidad). Para mi mente, sin embargo, el modo más fácil de demoler el argumento de la cascada es por medio de consideraciones de complejidad computacional.

De hecho, supón que realmente quisimos usar una cascada para ayudarnos a calcular los movimientos del ajedrez. ¿Cómo haríamos eso? En términos de complejidad, lo que queremos es una *reducción* del problema del ajedrez al problema de la simulación de cascada. Es decir, queremos un algoritmo eficiente que de algún modo *codifique* una posición de ajedrez $P$ en un estado inicial $s_P \in S$ de la cascada, de tal modo que un buen movimiento de $P$ pueda ser leído eficientemente desde el estado final correspondiente de la cascada, $f(s_P) \in T$.[38] ¿Pero cómo sería el algoritmo? No podemos decir con seguridad – ciertamente no sin el conocimiento detallado sobre $f$ (es decir, la física de cascadas), así como los medios por los que los elementos $S$ y $T$ se codifican como cuerdas binarias. ¡Pero para cualquier opción razonable, parece abrumadoramente probable que *cualquier* algoritmo de reducción *resuelve el problema de ajedrez mismo*, sin usar en absoluto la cascada de modo esencial! Con algo más de precisión, conjeturo que, dado cualquier algoritmo del juego de ajedrez $A$ que acceda a un "oráculo de cascada" $W$, hay un algoritmo

---

**38.** Técnicamente, esto describe una clase restricta de reducciones, llamadas reducciones no adaptativas. Una reducción adaptativa de ajedrez a cascadas podría resolver un problema de ajedrez por algún procedimiento que implique inicializar una cascada y observar su estado final, entonces usar los resultado de esa computación acuática para inicializar una segunda cascada y observar su estado final, y así sucesivamente por algún número polinomial de repeticiones.

del juego de ajedrez *A'* igualmente bueno, con similares requerimientos de tiempo y espacio que no accede a *W*. Si esta conjetura se sostiene, entonces nos da un modo absolutamente independiente del observador de formalizar nuestra intuición que la "semántica" de cascadas no tiene nada que ver con ajedrez.[39]

### 6.1 "Reducciones" Que Hacen Todo El Trabajo

Interesantemente, el problema de las reducciones "triviales" o "degeneradas" también surge dentro de la teoría de la complejidad, así que podría ser instructivo ver cómo se maneja allí. Recuerde de la Sección 3.1 que, hablando flojamente, un problema es NP-*completo* si es "máximamente difícil entre todos los problemas NP" (siendo NP la clase de problemas para los cuales las soluciones pueden verificarse en tiempo polinomial). Más formalmente, decimos que *L* es NP-completo si

(i) $L \in \mathsf{NP}$, y

(ii) dado cualquier *otro* problema *L'* de NP, existe un algoritmo de tiempo polinomial para resolver *L'* usando el acceso a un oráculo que resuelve *L*. (O más sucintamente, $L' \in \mathsf{P}^L$ dónde $\mathsf{P}^L$ denota la clase de complejidad P aumentada por un oráculo *L*).

El concepto de completitud NP tuvo increíble poder explicativo: mostró que *miles* de problemas aparentemente no relacionado de física, biología, optimización industrial, lógica matemática y otros campos eran todos *idénticos* desde el punto de vista de la computación en tiempo polinomial, y que ninguno de estos problemas tenía una solución eficiente a menos que P = NP. Así, era natural para los científicos teóricos de la computación querer definir un concepto análogo de completitud P. En otros términos: entre todos los problemas que son resolubles en tiempo polinomial, ¿cuáles son "maximalmente difíciles?".

Pero incluso ¿cómo debiera definirse la completitud P? Para ver la dificultad, supón que, por analogía con la completitud NP, decimos que *L* es P-completo si

(i) $L \in \mathsf{P}$ y

(ii) $L' \in \mathsf{P}^L$ para todo $L' \in \mathsf{P}$.

Entonces es fácil ver que la segunda condición es vacua: ¡todo problema P es P-completo! Porque "reduciendo" *L'* a *L*, un algoritmo de tiempo polinomial puede simplemente ignorar siempre al oráculo *L* y resolver *L'* por sí mismo, tanto como nuestro hipotético programa de ajedrez ignoró su oráculo de cascada. Debido a esto, la condición (ii) debe reemplazarse por una más fuerte; una elección popular es

(ii') $L' \in \mathsf{LOGSPACE}^L$ para todo $L' \in \mathsf{P}$.

Aquí LOGSPACE significa, informalmente, la clase de problemas resolubles por una máquina de Turing con una memoria de lectura/escritura consistiendo sólo en log *n* bits,

---

39. El lector perceptivo podría sospechar que contrabandeamos nuestra conclusión en la suposición de que los estados de la cascada $s_P \in S$ y $f(s_P) \in T$ fueron codificados como cuerdas binarias de un modo "razonable" (y no, por ejemplo, en un modo que codifica la solución al problema de ajedrez). Pero una lección crucial de la teoría de la complejidad es que, cuando discutimos los "problemas computacionales", siempre hacemos, de todos modos, un compromiso implícito sobre las codificaciones de entrada y salida. Así por ejemplo, si se dieran enteros positivos como entrada vía sus factorizaciones primas, entonces el problema de la factorización sería trivial (sólo aplicar la función identidad). ¿Pero a quién le preocupa? Si, definiendo matemáticamente el problema de la simulación de cascada, requerimos codificaciones de entrada y salida que ocasionan la resolución de problemas de ajedrez, entonces no sería razonable en absoluto, llamar a nuestro problema (solamente) un "problema de simulación de cascada".

dada una entrada de tamaño $n$.[40] No es difícil mostrar que LOGSPACE $\subseteq$ P, y hay una fuerte creencia de que esta contención sea estricta (aunque tal como con P $\neq$ NP, no hay aún ninguna prueba). El punto clave es que, si queremos una noción *no vacua* de completitud, entonces la clase de complejidad reductora necesita ser más *débil* (demostrablemente o conjeturalmente) que la clase a ser reducida. De hecho, clases de complejidad aun menores que LOGSPACE casi siempre bastan en la práctica.

En mi visión, hay aquí una lección importante para debates sobre computacionalismo. Por ejemplo, supón que queremos afirmar que una computación que juega ajedrez es "equivalente" a alguna otra que simula una cascada. Entonces nuestra afirmación sólo es no vacua si es posible *exhibir* la equivalencia (es decir, dar las reducciones) dentro de un modelo de computación que no es en sí mismo bastante poderoso para resolver los problemas de ajedrez o de cascada.

## 7 Aprendizaje-PAC y el Problema de la Inducción

Hace siglos, David Hume [77] señaló célebremente que parece lógicamente imposible aprender del pasado (y, por extensión, ciencia). Por ejemplo, si en una muestra de 500 cuervos cada uno es negro, ¿por qué eso nos da *algunos* fundamentos – aunque sea probabilísticos – para esperar que el cuervo 501 también sea negro? Cualquier respuesta moderna a esta pregunta probablemente se referirá a la navaja de Occam, el principio de que las hipótesis "más simples" consistentes con los datos son correctas con mayor probabilidad.

Así por ejemplo, la hipótesis que todos los cuervos son negros es "más simple" que la que la mayoría son verdes o púrpura, y que sólo los 500 que vimos eran negros. Intuitivamente, parece que la navaja de Occam debe ser parte de la solución al problema de Hume; la dificultad es que tal respuesta lleva a preguntas propias:

(1) ¿Qué queremos decir por "más simple"?

(2) ¿Por qué las explicaciones simples son probablemente correctas? O, menos ambiciosamente: ¿qué propiedades debe tener la realidad para que Navaja Occam "funcione"?

(3) ¿Cuántos datos debemos coleccionar antes que podamos encontrar una "hipótesis simple" que probablemente predecirá datos futuros? ¿Cómo hacemos para encontrar tal hipótesis?

En mi visión, la teoría de *Aprendizaje PAC* (*Probablemente Aproximadamente Correcto*), iniciada por Leslie Valiant [131] en 1984, ha hecho tan grandes adelantos en todas estas preguntas que merece ser estudiada por cualquiera interesado en la inducción.[41] En esta teoría, consideramos a un "aprendiz" idealizado al que se le presentan los puntos $x_1,...,\ x_m$ extraídos al azar de algún $S$ fijo grande, junto con "clasificaciones" $f(x_1),...,\ f(x_m)$ de esos puntos. La meta del aprendiz es inferir la función $f$, suficientemente bien para poder predecir $f(x)$ para la mayoría de los futuros puntos $x \in S$. Como ejemplo, el aprendiz podría ser un banco, $S$ podría ser un conjunto de personas (representadas por sus historias de crédito), y $f(x)$ podría representar si o no una persona $x$ le fallará al banco en un préstamo.

---

40. ¡Nota que una máquina LOGSPACE no tiene siquiera suficiente memoria para almacenar su cuerda de entrada! Por esta razón, pensamos en la cuerda de entrada como provista en una cinta especial de *sólo lectura*.

41. Ver Kearns y Vazirani [83] para una excelente introducción a aprendizaje PAC, y de Wolf [140] para trabajo anterior aplicando aprendizaje PAC a la filosofía y la lingüística: específicamente, a descarnar el argumento de Chomsky de la "pobreza del estímulo". De Wolf también discute varias formalizaciones de la Navaja de Occam distinta de la basada en aprendizaje PAC.

Por simplicidad, frecuentemente suponemos que $S$ es un conjunto de cuerdas binarias, y que la función $f$ asigna a cada $x \in S$ un único bit, $f(x) \in \{0, 1\}$. Sin embargo, ambas suposiciones pueden removerse sin cambiar significativamente la teoría. Las suposiciones importantes son las siguientes:

(1) Cada uno de los puntos de muestra $x_1,..., x_m$ se obtiene *independientemente* de alguna "distribución de muestreo" (posiblemente desconocida) $\mathcal{D}$ sobre $S$. Además, los puntos futuros $x$ sobre los que el aprendiz necesita predecir $f(x)$ se obtienen de la misma distribución.

(2) La función $f$ pertenece a un conocida "clase de hipótesis" $\mathcal{H}$. Esta $\mathcal{H}$ representa "el conjunto de posibilidades que el aprendiz está dispuesto a tomar en consideración" (y es típicamente mucho más pequeño que el conjunto de todas las $2^{|S|}$ posibles funciones de $S$ a $\{0, 1\}$.

**Teorema 2 (Valiant [131])** *Considera una clase de hipótesis finita $\mathcal{H}$, una función booleana $f : S \rightarrow \{0, 1\}$ en $\mathcal{H}$, y una distribución muestral $\mathcal{D}$ sobre $S$, así como una tasa de error $\varepsilon > 0$ y una probabilidad de falla $\delta > 0$ que el aprendiz está dispuesto a tolerar. Llama a una hipótesi $h : S \rightarrow \{0, 1\}$ "buena" si*

$$\Pr_{x \sim \mathcal{D}}[h(x) = f(x)] \geq 1 - \varepsilon.$$

*También, llama "confiables" a los puntos de muestra $x_1,\ldots, x_m$ si cada hipótesis $h \in \mathcal{H}$ satisface que $h(x_i) = f(x_i)$ para todo $i \in \{1,\ldots, m\}$ es buena. Entonces*

$$m = \frac{1}{\varepsilon} \ln \frac{|\mathcal{H}|}{\delta}$$

*puntos de muestra $x_1,\ldots, x_m$ obtenidos independientemente de $\mathcal{D}$ serán confiables con probabilidad al menos $1 - \delta$.*

Intuitivamente, el Teorema 2 dice que la conducta de $f$ en un número pequeño de puntos escogidos aleatoriamente determina su conducta en la mayoría de los puntos restantes. En otras palabras, si, por medios no especificados, el aprendiz se las arregla para encontrar alguna hipótesis $h \in \mathcal{H}$ que hace predicciones correctas en todos sus puntos de datos pasados $x_1,\ldots, x_m$, entonces suponiendo que $m$ sea suficientemente grande (y como sucede, $m$ no necesita ser muy grande), el aprendiz puede confiar estadísticamente que $h$ también hará predicciones correctas en la mayoría de los puntos futuros.

La parte del Teorema 2 que lleva la inequívoca marca de la teoría de la complejidad es el límite en el tamaño de la muestra, $m = \frac{1}{\varepsilon} \ln \frac{|\mathcal{H}|}{\delta}$. Este límite tiene tres notables implicaciones. Primero, aun cuando la clase $\mathcal{H}$ contiene exponencialmente muchas hipótesis (digamos, $2^n$), todavía se puede aprender una función arbitraria $f \in \mathcal{H}$ que usa una cantidad *lineal* de datos de muestra, puesto que $m$ sólo crece logarítmicamente con $|\mathcal{H}|$: en otras palabras, como el número de bits necesarios para *escribir* una hipótesis individual. Segundo, se puede hacer *exponencialmente pequeña* la probabilidad de que la hipótesis $h$ fallará para generalizar (digamos, $\delta = 2^{-n}$), a costa de aumentar el tamaño de la muestra $m$ por sólo un factor lineal. Tercero, suponiendo que la hipótesis generaliza, su tasa de error $\varepsilon$ disminuye inversamente con $m$. No es difícil mostrar que cada una de estas dependencias es firme, así por ejemplo, si requerimos $\varepsilon = 0$ o $\delta = 0$ entonces ningún $m$ finito es suficiente. Éste es el

origen del nombre del aprendizaje PAC: lo más que puede esperarse es producir una hipótesis que sea "probablemente, aproximadamente" correcta.

La prueba de Teorema 2 es fácil: considere cualquier hipótesis $h \in \mathcal{H}$ que es mala, significando que

$$\Pr_{x \sim \mathcal{D}}[h(x) = f(x)] < 1 - \varepsilon.$$

Entonces por la suposición de independencia,

$$\Pr_{x_1,\ldots, x_m \sim \mathcal{D}}[h(x_1) = f(x_1) \wedge \ldots \wedge h(x_m) = f(x_m)] < (1 - \varepsilon)^m.$$

Ahora, el número de hipótesis no malas es más que el número total de hipótesis, $|\mathcal{H}|$. Así por el límite de la unión, la probabilidad de que *exista* una hipótesis mala que concuerde con $f$ sobre todo $x_1$,...., $x_m$ puede ser a lo sumo $|\mathcal{H}| \cdot (1 - \varepsilon)^m$. Por consiguiente $\delta \leq |\mathcal{H}| \cdot (1 - \varepsilon)^m$, y todo lo que resta es resolver para $m$.

La relevancia del Teorema 2 al problema de inducción de Hume es que describe una clase de situaciones no triviales donde la inducción está garantizada para trabajar con alta probabilidad. El Teorema 2 también ilumina el papel de la Navaja de Occam en la inducción. Para aprender usando un número "razonable" de puntos de muestra $m$, la clase de hipótesis $\mathcal{H}$ debe tener una cardinalidad suficientemente pequeña. Pero eso es equivalente a decir que cada hipótesis $h \in \mathcal{H}$ debe tener una descripción sucinta – puesto que el número de bits necesarios para especificar una hipótesis arbitraria $h \in \mathcal{H}$ es simplemente $[\log_2 |\mathcal{H}|]$. Si el número de bits necesarios para especificar una hipótesis fuese demasiado grande, entonces $\mathcal{H}$ siempre será vulnerable al problema de la *sobre adaptación*: algunas hipótesis $h \in \mathcal{H}$ sobreviviendo sólo por casualidad el contacto con los datos de la muestra.

Como me indicaba Agustin Rayo, hay varias interpretaciones posibles de la Navaja de Occam que no tienen nada que ver con la complejidad descriptiva: por ejemplo, podríamos querer que nuestras hipótesis fuesen "simples" en términos de sus compromisos ontológicos o ideológicos. Sin embargo, a cualquier magnitud que interpretemos la Navaja de Occam, diciendo que son preferibles las hipótesis *cortas* o *de baja complejidad*, el Teorema 2 llega más cerca de lo que pudiera haberse pensado posible para una justificación matemática de por qué la Navaja funciona.

Muchos filósofos podrían estar familiarizados con enfoques formales alternativos a la Navaja de Occam. Por ejemplo, dentro de la estructura bayesiana, se puede escoger una anterior sobre todas las posibles hipótesis que dan el mayor peso a las hipótesis "más simples" (donde la simplicidad está medida, por ejemplo, por la longitud del programa más corto que computa las predicciones). Sin embargo, mientras el aprendizaje PAC y el enfoque bayesiano están relacionados, el enfoque PAC tiene la ventaja de requerir sólo una decisión *cualitativa* acerca de cuáles hipótesis se quiere considerar, en lugar de una anterior cuantitativa sobre hipótesis. Dada la clase de hipótesis $\mathcal{H}$, entonces se puede buscar métodos de aprendizaje que funcionen para *cualquier* $f \in \mathcal{H}$. (Por otra parte, el enfoque PAC requiere una suposición sobre la distribución de probabilidad acerca de las *observaciones*, mientras que el enfoque bayesiano no).

### 7.1 Desventajas del Modelo PAC Básico

Me gustaría ahora discutir tres desventajas del Teorema 2, desde que pienso que las desventajas iluminan aspectos filosóficos de la inducción tanto como lo hacen las ventajas.

La primera desventaja es que el Teorema 2 funciona para las clases finitas de hipótesis. En la ciencia, sin embargo, las hipótesis frecuentemente involucran parámetros continuos de los cuales hay infinidad incontable. Por supuesto, se podría resolver este problema simplemente discretizando los parámetros, pero entonces el número de hipótesis (y por consiguiente la relevancia de Teorema 2) dependería de qué tan fina fue la discretización. Afortunadamente, podemos evitar tales dificultades dándonos cuenta que el aprendiz sólo se preocupa acerca de "diferencias" entre dos hipótesis en la medida en que conduzcan a las diferentes predicciones. Esto conduce a la noción fundamental de dimensión VC (por sus creadores, Vapnik y Chervonenkis [133]).

**Definición 3 (VC-dimension)** *Una clase de hipótesis $\mathcal{H}$ fragmenta los puntos muestrales $\{x_1,..., x_k\} \subseteq S$ si para todas las $2^k$ posibles disposiciones de $h(x_1,..., h(x_k)$, existe una hipótesis $h \in \mathcal{H}$ compatible con esas disposiciones. Entonces VC-dim ($\mathcal{H}$), dimensión VC de $\mathcal{H}$, es el k más grande para el que existe un subconjunto $\{x_1,..., x_k\} \subseteq S$ que $\mathcal{H}$ fragmenta* (*o si ningún máximo finito existe, entonces VCdim* ($\mathcal{H}$) = $\infty$).

Claramente cualquier clase de hipótesis finita tiene dimensión VC finita: de hecho, VCdim ($\mathcal{H}$) $\leq \log_2 |\mathcal{H}|$. Sin embargo, incluso una clase de hipótesis infinita puede tener la dimensión VC finita si es "suficientemente simple". Por ejemplo, sea $\mathcal{H}$ la clase de todas las funciones $h_{a,b}: \mathbb{R} \to \{0, 1\}$ de forma

$$h_{a,b}(x) = \begin{cases} 1 & \text{si } a \leq x \leq b \\ 0 & \text{de otro modo.} \end{cases}$$

Entonces es fácil de verificar que VCdim ($\mathcal{H}$) = 2.

Con la noción de dimensión VC en mano, podemos formular una poderosa (¡y difícil de demostrar!) generalización del Teorema 2, debida a Blumer et al. [31].

**Teorema 4 (Blumer et al. [31])** *Para alguna constante universal K* > 0, *el límite sobre m en el Teorema 2 puede reemplazarse por*

$$m = \frac{K \, \mathrm{VCdim}(\mathcal{H})}{\varepsilon} \ln \frac{1}{\delta\varepsilon},$$

*con el teorema ahora manteniéndose para cualquier clase de hipótesis $\mathcal{H}$, finita o infinita.*

Si $\mathcal{H}$ tiene dimensión VC-infinita, entonces es fácil construir una distribución de probabilidades $\mathcal{D}$ sobre puntos muestrales tal que *ningún número finito m de muestras de $\mathcal{D}$ sea suficiente para PAC-aprender una función $f \in \mathcal{H}$*: realmente se está en la infortunada situación descrita por Hume, de no tener en absoluto fundamentos para predecir que el próximo cuervo será negro. En algún sentido, entonces, el Teorema 4 está diciéndonos que la dimensión VC finita es una condición necesaria y suficiente para que la inducción científica sea posible. Una vez más, el Teorema 4 también tiene una interpretación en términos de la Navaja de Occam, con la pequeñez de la dimensión VC jugando ahora el papel de la simplicidad.

La segunda desventaja del Teorema 2 es que no nos da ninguna pista acerca de cómo *encontrar* una hipótesis $h \in \mathcal{H}$ consistente con los datos de la muestra. Todo lo que dice es que, *si* encontramos tal *h*, entonces *h* probablemente estará cerca de la verdad. Esto ilustra que, incluso en el simple arreglo previsto de aprendizaje PAC, la inducción no puede ser meramente materia de ver bastantes datos y entonces "generalizar" de ello, ¡porque podrían necesitarse inmensos cómputos para encontrar una generalización conveniente! De hecho, siguiendo el trabajo de Kearns y Valiant [82], ahora sabemos que muchos problemas de aprendizaje natural – por ejemplo, inferir las reglas de un lenguaje regular o libre de contexto de ejemplos aleatorios de oraciones gramaticales y no gramaticales – son computacionalmente intratables en un sentido extremadamente fuerte:

> *Cualquier algoritmo de tiempo polinomial para encontrar una hipótesis consistente con los datos implica un algoritmo de tiempo polinomial para quebrar criptosistemas extensamente usados como RSA.*[42] [Nota del traductor: Rivest, Shamir y Adleman]

La aparición de criptografía en la manifestación anterior está lejos de ser accidental. En cierto sentido que puede hacerse preciso, aprendizaje y criptografía son problemas "duales": un aprendiz quiere encontrar patrones en los datos, mientras un criptógrafo quiere generar datos cuyos patrones sean difíciles de encontrar. Más concretamente, una de las primitivas básicas en criptografía se llama familia de funciones pseudo aleatorias. Ésta es una familia de funciones booleanas eficientemente computables $f_s$: $\{0, 1\}^n \rightarrow \{0, 1\}$, parametrizadas por una corta "semilla" aleatoria *s*, que son *virtualmente indistinguibles de las funciones aleatorias* por un algoritmo de tiempo polinomial. Aquí, imaginamos que el algoritmo que estaría distinguiendo puede preguntar a la función $f_s$ en varios puntos *x*, y también que conoce la asignación (*mapping*) de *s* a $f_s$, y así sólo es ignorante de la semilla *s* misma. Hay fuerte evidencia en criptografía que las familias de funciones pseudo aleatorias, existen: de hecho, Goldreich, Goldwasser, y Micali [65] mostraron cómo construir una a partir de cualquier *generador* pseudo aleatorio (el último fue mencionado en la Sección 1.1).[43]

Ahora, dada una familia de funciones pseudo aleatoria, imagina un aprendiz PAC cuya clase de hipótesis $\mathcal{H}$ consiste en $f_s$ para toda posible semilla *s*. Al aprendiz se le proporciona alguna muestra escogida al azar de puntos $x_1,..., x_m \in \{0; 1\}^n$, junto con los valores de $f_s$ en esos puntos: $f_s(x_1),..., f_s(x_m)$. Dados estos "datos de entrenamiento", la meta del aprendiz es deducir cómo computar $f_s$ por sí misma – y por eso predice los valores de $f_s(x)$ en nuevos puntos *x*, no puntos en la muestra de entrenamiento. Desgraciadamente, es fácil ver que si el aprendiz pudiera hacer eso, entonces, por lo mismo, distinguiría $f_s$ de una función verdaderamente aleatoria – e, igualmente, contradeciría nuestra suposición de partida que $\{fs\}$ era pseudo aleatoria. Nuestra conclusión es que, si las suposiciones básicas de la criptografía moderna se mantienen (y en particular, si existen generadores pseudo aleatorios), entonces deben haber situaciones donde aprender es imposible debido puramente a la complejidad computacional (y no a causa de datos insuficientes).

---

42. En la disposición de "aprendizaje apropiado" – donde el aprendiz necesita producir una hipótesis en algún formato especificado – incluso es conocido que muchos problemas naturales de aprendizaje PAC son NP-completos (ver Pitt y Valiant [105] por ejemplo). Pero en la disposición "impropia" – donde el aprendiz puede describir su hipótesis usando cualquier algoritmo de tiempo polinomial – sólo se sabe cómo mostrar que los problemas de aprendizaje PAC son difíciles bajo suposiciones criptográficas, y parece ser que hay razones inherentes para esto (ver Applebaum, Barak, y Xiao [14]).

43. Además, Hastad et al. [125] mostraron cómo construir un generador pseudo aleatorio de cualquier función de *dirección única*: hablando mal y pronto, una función *f* que es fácil de computar pero difícil de invertir hasta en una entrada aleatoria.

La tercera desventaja del Teorema 2 es la suposición que la distribución $\mathcal{D}$ de la cual se examina al aprendiz, es la misma que la distribución de la que fueron extraídos los puntos muestrales. Para mí, esta es la desventaja más seria, puesto que nos dice que el aprendizaje PAC modela el "aprendizaje" realizado por un estudiante preparándose para un examen resolviendo problemas del año anterior, o un empleador que usa un modelo de regresión para identificar las características de contrataciones exitosas, o un criptoanalista que quiebra un código desde una colección de textos llanos y cifrados. No modela, sin embargo, el "aprendizaje" de un Einstein o un Szilard, que hacen predicciones acerca de fenómenos que son diferentes en clase de algo ya observado. Como David Deutsch enfatiza en su reciente libro *The Beginning of Infinity* [49], la meta de la ciencia no es meramente resumir las observaciones, y consecuentemente permitirnos hacer predicciones acerca de observaciones similares. Al contrario, la meta es descubrir explicaciones con "alcance", significando la habilidad de predecir lo que sucedería incluso en situaciones noveles o hipotéticas, como que el sol desapareciera de repente o se construyera una computadora cuántica. A mi ver, desarrollar un modelo matemático obligante de aprendizaje *explicativo* – un modelo que "sea a la explicación lo que el modelo de PAC es a la predicción" – es un excelente problema abierto.[44]

### 7.2 Complejidad Computacional, Bleen y Grue

En 1955, Nelson Goodman [68] propuso lo que llamó el "nuevo enigma de la inducción" que sobrevive a la respuesta de la Navaja de Occam al problema original de la inducción de Hume. En el enigma de Goodman se nos pide considerar la hipótesis "Todas las esmeraldas son verdes". ¿La pregunta es, por qué favorecemos *esa* hipótesis por encima de la siguiente alternativa, que es igualmente compatible con toda nuestra evidencia de esmeraldas verdes?

> "Todas las esmeraldas son verdes antes del 1 de enero de 2030, y entonces azules después".

La respuesta obvia es que la segunda hipótesis agrega complicaciones superfluas y por consiguiente es desfavorecida por la Navaja de Occam. A eso, Goodman responde que las definiciones de "simple" y complicado dependen de nuestro lenguaje. En particular, supón que no hubiera ninguna palabra para verde (*green*) o azul (*blue*), pero sí una palabra *grue*, que significa "verde antes del 1 de enero de 2030, y azul después", y una palabra *bleen*, que significa "azul antes del 1 de enero de 2030, y verde después". En ese caso, sólo podríamos expresar la hipótesis "Todas las esmeraldas son verdes" diciendo

> "Todas las esmeraldas son grue antes del 1 de enero de 2030, y entonces bleen después".

– ¡una hipótesis manifiestamente más complicada que la simple "Todas las esmeraldas son grue"!

Confieso que, cuando contemplo el acertijo grue, no puedo evitar la broma sobre los Anti-Inductivistas, quienes, cuando les preguntan por qué continúan creyendo que el futuro *no se parecerá* al pasado, cuando esa falsa creencia ha traído a su civilización nada más que pobreza y miseria, responden "¡porque la anti-inducción nunca ha funcionado antes!" Sí, si artificialmente definimos nuestros conceptos primitivos "a contra-corriente del mundo", entonces no debemos sorprendernos si la conducta real del mundo se hace más embarazosa

---

44. ¡Este problema no es tan desesperado como pudiera parecer! El importante progreso incluye el trabajo de Angluin [11] sobre autómata finito aprendiendo de preguntas y contraejemplos, y de Angluin et al. [12] sobre aprendizaje de un circuito inyectando valores. Ambos trabajos estudian modelos de aprendizaje naturales que generalizan el modelo PAC, permitiendo "experimentos científicos controlados", cuyos resultados confirman o refutan una hipótesis y por eso proporcionan una guía acerca de qué experimentos hacer luego.

para describir, o si hacemos predicciones erróneas. Sería como si estuviéramos usando un lenguaje de programación que no tenga una función incorporada para la multiplicación, sino sólo para $F(x, y) := 17x - y - x^2 + 2xy$. ¡En ese caso, el primer instinto de una persona normal sería o cambiar los lenguajes de programación, o de lo contrario definir la multiplicación en términos de $F$, y olvidarse acerca de $F$ desde ese punto en adelante![45] Ahora, hay aquí un problema filosófico genuino: ¿por qué grue, bleen y $F(x, y)$ *sí* van "a contra corriente del mundo", mientras que verde, azul y multiplicación van con la corriente? Pero para mí, ese problema (como la confusión de Wigner sobre "la irrazonable efectividad de la matemática en las ciencias naturales" [139]) es más sobre el mundo mismo que sobre los conceptos humanos, así que no debiéramos esperar algún análisis puramente lingüístico para resolverlo.

¿Qué entonces, acerca de la complejidad computacional? En mi visión, aun cuando la complejidad computacional no resuelve el enigma grue, contribuye con una intuición útil. A saber, que cuando hablamos acerca de la simplicidad o complejidad de las hipótesis, debemos distinguir dos problemas:

(a) El *escalamiento asintótico* del tamaño de la hipótesis, como el "tamaño" $n$ de nuestro problema de aprendizaje va a infinito.
(b) Los sobrecostos (*overheads*) de factor constante.

En términos del modelo PAC básico en la Sección 7, podemos imaginar un "parámetro oculto" $n$ que mide el número de bits necesario para especificar un punto individual en el conjunto $S = S_n$ (otros modos de medir el "tamaño" de un problema de aprendizaje también funcionarían, pero este modo es particularmente conveniente). Por conveniencia, podemos identificar $S_n$ con el conjunto $\{0, 1\}^n$ de cuerdas del $n$ bits, así que $n = \log_2 |S_n|$. Necesitamos entonces considerar no *sólo* una sola clase de hipótesis, sino una *familia* infinita de clases de hipótesis $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, ...\}$ una para cada entero positivo $n$. Aquí $\mathcal{H}_n$ consiste en funciones de hipótesis $h$ que asignan $S_n = \{0, 1\}^n$ a $\{0, 1\}$.

Ahora, sea $L$ un lenguaje para especificar las hipótesis en $\mathcal{H}$: en otras palabras, una asignación de (algún subconjunto de) cuerdas binarias $y \in \{0; 1\}$ a $\mathcal{H}$. También, dada una hipótesis $h \in \mathcal{H}$, sea

$$\mathcal{K}L(h) := min\{|y| : L(y) = h\}$$

la longitud de la descripción *más corta* de $h$ en el lenguaje $L$. (Aquí $|y|$ significa solamente el número de bits en $y$). Finalmente, sea

$$\mathcal{K}L(n) := \max\{\mathcal{K}L(h) : h \in \mathcal{H}_n\}$$

el número de bits necesario para especificar una hipótesis arbitraria en $\mathcal{H}_n$ usando el lenguaje $L$.

Claramente $\mathcal{K}L \geq [\log_2 |\mathcal{H}_n|]$, con igualdad si y sólo si $L$ es "óptimo" (es decir, si representa cada hipótesis $h \in \mathcal{H}_n$ usando tan pocos bits como sea posible). La pregunta que

---

45. Supón que nuestro lenguaje de programación proporciona sólo multiplicación por constantes, adición y la función $F(x, y) := ax^2 + bxy + cy^2 + dx + ey + f$. Podemos suponer sin pérdida de generalidad que $d = e = f = 0$. Entonces provisto que $ax^2 + bxy + cy^2$ se factoriza en dos términos lineales independientes, $px + qy$ y $rx + sy$, podemos expresar el producto $xy$ como

$$\frac{F(sx - qy, -rx + py)}{(ps - qr)^2}$$

nos concierne es qué tan rápido crece $L$ ($n$) como una función de $n$, para varias elecciones de lenguaje $L$.

¿Qué tiene esto que ver con el acertijo grue? Bien, podemos pensar en los detalles de $L$ (su sintaxis, vocabulario, etc.) afectando la conducta de "bajo orden" de la función $L$ ($n$). Así por ejemplo, supón que somos tan desafortunados que $L$ contiene las palabras *grue* y *bleen*, pero no *azul* y *verde*. Eso podría incrementar $L$ ($n$) por un factor de diez o algo así, –puesto que ahora, cada vez que queremos mencionar "verde" cuando especificamos nuestra hipótesis $h$, necesitamos una circunlocución verbosa como "grue antes del 1 de enero de 2030, y entonces bleen después", y similarmente para azul.[46] Sin embargo, una lección crucial de la teoría de la complejidad es que la conducta de orden superior de $L$ ($n$) – por ejemplo, si crece polinomialmente o exponencialmente con $n$ – es casi completamente inafectada por los detalles de $L$. La razón es que si los lenguajes $L_1$ y $L_2$ difieren sólo en sus "detalles de bajo orden", entonces *traducir* una hipótesis de $L_1$ a $L_2$ o viceversa, incrementará la longitud de la descripción por no más que un factor polinomial. De hecho, como en nuestro ejemplo de grue, hay normalmente una "constante de traducción universal" $c$ tal que $\mathcal{K}_{L1}(h) \leq c\mathcal{K}_{L2}(h)$ o incluso $\mathcal{K}_{L1}(h) \leq \mathcal{K}_{L2}(h) + c$ para toda hipótesis $h \in \mathcal{H}$.

La única excepción a la regla anterior es si los lenguajes $L_1$ y $L_2$ tienen *poderes expresivos* diferentes. Por ejemplo, quizá $L_1$ permite sólo anidar expresiones a profundidad dos, mientras $L_2$ permite hacerlo a profundidades arbitrarias; o $L_1$ sólo permite conectivas proposicionales, mientras $L_2$ también permite cuantificadores de primer orden. En esos casos, $L_1(h)$ podría de hecho ser mucho más grande que $\mathcal{K}_{L2}(h)$ para algunas hipótesis $h$, posiblemente hasta exponencialmente mayor ($L_1(h) \approx 2^{\mathcal{K}L_2(h)}$). Una gruesa analogía sería esto: supón que no hubieses aprendido lo que son ecuaciones diferenciales, y no tuvieras ninguna idea de cómo resolverlas incluso aproximadamente o numéricamente. En ese caso, la mecánica de Newton podría haberte parecido tan complicada como la teoría ptolemaica con epiciclos, ¡si no *más* complicada! Porque el único modo de que pudieras hacer predicciones con la mecánica newtoniana sería usando una gran tabla de soluciones de ecuaciones diferenciales "precomputadas" – y *a ti* esa tabla te parecería tan pesada e inelegante como una tabla de epiciclos. Pero nota que en este caso, tu percepción sería el resultado, no de alguna elección arbitraria de vocabulario, sino de un hueco *objetivo* en tus poderes expresivos matemáticos.

Para resumir, nuestra elección de vocabulario – por ejemplo, si tomamos verde/azul o bleen/grue como conceptos primitivos – podría de hecho importar si queremos usar la Navaja de Occam para predecir el color futuro de las esmeraldas. Pero pienso que la teoría de la complejidad nos justifica en tratar grue como un "efecto $n$ pequeño": algo que llega a ser menos importante en el límite asintótico de problemas de aprendizaje más y más complicados.

---

**46.** Aunque nótese que, si el lenguaje $L$ es suficientemente expresivo para permitir esto, podemos simplemente definir verde y azul en términos de bleen y grue *una vez*, y luego referirnos a esas definiciones siempre que sea necesario. En ese caso, tomar bleen y grue (en lugar de verde y azul) como conceptos primitivos, incrementaría a $L$ ($n$) sólo por una constante *aditiva*, en lugar de una multiplicativa. El hecho anterior se relaciona a un resultado fundamental de la teoría de la complejidad de Kolmogorov (vea Li y Vitanyi [90] por ejemplo). A saber, si $P$ y $Q$ son dos lenguajes de programación universales de Turing cualesquiera, y si $\mathcal{K}_P(x)$ y $\mathcal{K}_Q(x)$ son respectivamente las longitudes de los programas más cortos en el $P$ y $Q$ que producen una cuerda dada $x \in \{0, 1\}^*$ entonces existe allí una "constante de traducción" universal $c_{PQ}$, tal que $|\mathcal{K}_P(x) - \mathcal{K}_Q(x)| \leq c_{PQ}$ para *todo* $x$. Esta $c_{PQ}$ es simplemente el número de bits necesario para escribir un $P$-intérprete para $Q$-programas o viceversa.

## 8 Computación Cuántica

La *computación cuántica* es una propuesta para usar la mecánica cuántica para resolver ciertos problemas computacionales mucho más rápido de lo que sabemos resolverlos hoy.[47] Para eso se necesitaría construir un nuevo tipo de computadora, capaz de explotar los efectos cuánticos de superposición e interferencia. Construyendo tal computadora – una suficientemente grande para resolver problemas interesantes – resta un enorme desafío para los físicos e ingenieros, debido a la fragilidad de los estados cuánticos y a la necesidad de aislarlos de su ambiente externo.

Entretanto, sin embargo, los científicos de computación teóricos han estudiado extensivamente lo que podríamos y no podríamos hacer con una computadora cuántica si tuviéramos una. Para ciertos problemas, se conocen notables algoritmos cuánticos resolubles en tiempo polinomial, aunque los algoritmos clásicos mejor conocidos requieren tiempo exponencial. Más célebremente, en 1994 Peter Shor [119] proporcionó un algoritmo cuántico en tiempo polinomial para factorizar enteros, y como subproducto, romper la mayoría de los códigos criptógraficos usados hoy en Internet. Además de las implicaciones prácticas, el algoritmo de Shor proveyó también una pieza clave de evidencia de que cambiando de computadoras clásicas a cuánticas se agrandaría la clase de problemas resolubles en tiempo polinomial. Para los científicos de computación teóricos, esto tenía una profunda lección: si queremos conocer los límites de la computación eficiente, podemos necesitar "dejar nuestros sillones" e incorporar hechos reales acerca de la física (como mínimo, la verdad o falsedad de la mecánica cuántica).[48]

Se construyan o no computadoras cuánticas escalables en cualquier momento, mi propia visión (sesgada) es que la computación cuántica representa uno de los grandes adelantos científicos de nuestro tiempo. Pero aquí quiero hacer una pregunta diferente: ¿tiene la computación cuántica alguna implicación para la *filosofía* – y específicamente, para la interpretación de la mecánica quántica?

Desde una perspectiva, la respuesta parece ser un obvio "no". Cualquier cosa que ella sea, la computación cuántica es "meramente" una aplicación de la mecánica cuántica, como esa teoría ha existido en los libros de texto de física durante 80 años. De hecho, si admites que la mecánica cuántica (como se entiende actualmente) es verdad, entonces presumiblemente también debieras aceptar la posibilidad de computadoras cuánticas, y hacer las mismas predicciones acerca de su funcionamiento como todos los demás. Parece irrelevante que describas la "realidad" detrás de los procesos cuánticos vía la Interpretación de Muchos-Mundos, la mecánica bohmiana, o alguna otra visión (o, siguiendo la Interpretación de Copenhague de Bohr, rechaces discutir la "realidad" en absoluto).

Sin embargo, desde una perspectiva diferente, una computadora cuántica escalable *probaría* la mecánica cuántica en un régimen extremadamente nuevo – y de hecho, por esa razón, podría promover nuevos problemas filosóficos. El "régimen" que las computadoras cuánticas probarían se caracteriza, no por una escala de energía o una temperatura, sino por la complejidad computacional. Uno de los hechos más llamativos sobre la mecánica

---

47. La referencia autoritaria para la computación cuántica es el libro de Nielsen y Chuang [100]. Para introducciones más ligeras, prueba Mermin [93, 94] o los artículos de estudio de Aharonov [10], Fortnow [58], o Watrous [136]. Para una discusión general de computación en tiempo polinomial y las leyes de física (incluyendo modelos especulativos más allá de la computación cuántica), consulta mi artículo de estudios "Problemas NP-completos y Realidad Física" [4].

48. Por contraste, si sólo queremos saber lo que es *computable* en el universo físico, sin ningún requerimiento de eficiencia, entonces permanece completamente consistente con el conocimiento actual la respuesta correcta que dieron Church y Turing en la década de 1930 – y la que hicieron sin incorporar ninguna física más allá de la que es "accesible a la intuición".

cuántica es que para representar el estado de *n* partículas enredadas, se necesita un vector de tamaño *exponencial* en *n*. Por ejemplo, para especificar el estado de mil partículas de espín-1/2 se necesita $2^{1000}$ números complejos llamados "amplitudes", uno por cada posible resultado de medir los espíns en la base {arriba, abajo}. El estado cuántico, denotado $|\psi\rangle$, es entonces una combinación lineal o "superposición" de los posibles resultados, con cada resultado $|x\rangle$ ponderado por su amplitud $\alpha_x$:

$$|\psi\rangle = \sum_{x \in \{\text{arriba,abajo}\}^{1000}} \alpha_x |x\rangle$$

Dado $|\psi\rangle$ se puede calcular la probabilidad $p_x$ de que se observará algún resultado particular, vía la regla $p_x = |\alpha_x|^2$.[49]

Esto significa, en particular, que las amplitudes satisfacen la condición de normalización $\sum_x |\alpha_x|^2 = 1$.

Ahora, sólo hay alrededor $10^{80}$ átomos en el universo visible, que es un número muy menor $2^{1000}$. Así que suponiendo que la mecánica cuántica sea verdadera, parece que la Naturaleza tiene que invertir asombrosos montos de "esfuerzo computacional" para seguir la pista de pequeñas colecciones – ¡ciertamente más de lo que cualquier física clásica requiere![50,51] En la década de 1980 temprana, Richard Feynman [56] y otros llamaron la atención sobre este punto, notando que subyacía a algo que, en la práctica, había sido aparente desde hacía mucho tiempo: la extraordinaria dificultad de simular mecánica cuántica usando computadoras convencionales.

Pero Feynman también asomó la posibilidad de revertir esa dificultad, construyendo nuestras computadoras con componentes cuánticos. Tales computadoras podrían resolver

---

**49.** Esto significa, en particular, que las amplitudes satisfacen la condición de normalización $\sum_x |\alpha_x|^2 = 1$.

**50.** Se podría objetar que incluso en el mundo clásico, si simplemente no conocemos el valor de (digamos) una cuerda del *n* bits, entonces también describimos nuestra ignorancia usando exponencialmente muchos números: a saber, ¡la *probabilidad* $p_x$ de cada posible cuerda $x \in \{0, 1\}^n$! Y hay, de hecho, una conexión sumamente íntima entre la mecánica cuántica y la teoría de probabilidades clásica; describo frecuentemente la mecánica cuántica como sólo "teoría de probabilidades con números complejos en lugar de reales no negativos". Sin embargo, una diferencia crucial es que siempre podemos describir una cuerda clásica como que tiene un valor *x* "realmente" definido; el vector de $2^n$ probabilidades $p_x$ es entonces simplemente una representación mental de nuestra propia ignorancia. Con un estado cuántico no podemos darnos el mismo lujo, debido al fenómeno de interferencia entre las amplitudes positivas y negativas.

**51.** Se podría también objetar que, incluso en la física clásica, toma *infinitamente* muchos bits registrar el estado de incluso una sola partícula, si su posición y momentum pueden ser números reales arbitrarios. Y de hecho, Copeland [43], Hogarth [74], Siegelmann [120], y otros escritores han especulado que la continuidad de las cantidades físicas realmente podría permitir "híper computaciones" – incluyendo resolver el problema de la detención en una cantidad finita de tiempo. Desde una perspectiva moderna, sin embargo, la mecánica y la gravedad cuánticas sugieren fuertemente que *la "continuidad" de cantidades mensurables tales como posiciones y momenta es un artefacto teórico*. En otras palabras, debe ser suficiente para propósitos de simulación aproximar estas cantidades a alguna precisión finita, probablemente relacionada con la escala de Planck de $10^{-33}$ centímetros o $10^{-43}$ segundos.
Pero la exponencialidad de los estados cuánticos es diferente, por al menos dos razones. Primero, no lleva a altas aceleraciones computacionales que sean casi tan "irrazonables" como las altas aceleraciones de la híper computabilidad. Segundo, nadie tiene alguna idea dónde la teoría en cuestión (mecánica cuántica) podría descomponerse, de una manera consistente con los experimentos actuales. En otras palabras, no hay ningún "obstáculo asesino" conocido para la computación cuántica, análogo a la escala de Planck para la híper computación. Ver Aaronson [2] para la discusión extensa de este punto, así como una estructura propuesta de complejidad teórica (llamada los "separadores de Sure/Shor") con la cual estudiar tales obstáculos.

ciertos problemas concebiblemente más rápido que las computadoras convencionales: ¡si nada más, entonces al menos el problema de simular mecánica cuántica!

Así, la computación cuántica no sólo es interesante debido a sus aplicaciones, sino (más aun, en mi opinión) porque es *la primera tecnología que directamente "sondearía" la exponencialidad inherente en la descripción cuántica de la Naturaleza*. Se puede hacer aquí una analogía con los experimentos en los años de la década de 1980 que primero violaron convincentemente la Desigualdad de Bell. Como los algoritmos cuánticos hoy, la refutación de Bell del realismo local era "meramente" una consecuencia matemática de la mecánica cuántica. Pero esa refutación (y los experimentos que inspiró) hizo que no fuera posible ignorar por más tiempo *aspectos* conceptualmente importantes de la mecánica cuántica – y por esa razón, cambió el paisaje filosófico. Me parece abrumadoramente probable que la computación cuántica hará lo mismo.

De hecho, podemos extender la analogía más allá: así como había "realistas locales duros de matar" que negaron que sería posible la violación de la Desigualdad de Bell (e intentaron explicarla mucho después que fue lograda), así hoy una minoría vocal de científicos de computación y físicos (incluyendo Leonid Levin [89], Oded Goldreich [62] y Gerard 't Hooft [76]) niega la posibilidad de computadoras cuánticas escalables, incluso en principio. Mientras ellos admiten que la mecánica cuántica ha pasado toda prueba experimental por un siglo, estos escépticos están *confiados* en que la mecánica cuántica fallará en el régimen probado por computación cuántica – y que cualquier nueva teoría que la reemplace, esa teoría permitirá sólo computación clásica.

Como la mayoría de los investigadores de computación cuántica son rápidos en señalar en respuesta, ¡se *estremecerían* si el esfuerzo por construir computadoras cuánticas escalables condujera en cambio a una revisión de la mecánica cuántica! Tal resultado constituiría probablemente la revolución más grande en la física desde la década de 1920, y últimamente sería mucho *más* interesante que construir una computadora cuántica. Por supuesto, también es posible que la computación cuántica escalable sea abandonada como muy difícil por razones tecnológicas "mundanas", en lugar de razones físicas fundamentales. Pero esa posibilidad "mundana" no es de lo que escépticos como Levin, Goldreich y 't Hooft están hablando.

### 8.1 Computación Cuántica y la Interpretación de Muchos-Mundos

Pero retornemos a la pregunta original: supón que los escépticos están equivocados, y es posible construir computadoras cuánticas escalables. ¿Eso tendría alguna relevancia para la interpretación de la mecánica cuántica? El argumento mejor conocido cuya respuesta es "sí" fue hecho por David Deutsch, un pionero de la computación cuántica y firme defensor de la Interpretación de Muchos-Mundos. Para ser precisos, Deutsch piensa que la mecánica cuántica implica *francamente* la existencia de universos paralelos, y que eso es así independientemente de la computación cuántica: en su visión, incluso el experimento de la doble hendidura sólo puede explicarse en término de dos universos paralelos que interfieren. Sin embargo, Deutsch también piensa que la computación cuántica agrega un impacto emocional al argumento. Así es cómo lo puso en su libro de 1997 *The Fabric of Reality*, El Tejido de la Realidad [48, pág. 217]:

> Lógicamente, la posibilidad de computaciones complejas cuánticas no agrega nada a un caso [para la Interpretación de Muchos-Mundos] eso ya es incontrovertible. Pero agrega impacto psicológico. Con el algoritmo de Shor, el argumento ha sido escrito muy grande. A aquéllos que todavía se aferran a una visión del mundo de un solo universo, publico este desafío: *explique cómo funciona el algoritmo de Shor*. No quiero decir meramente predecir que funcionará, lo cual es meramente una cuestión de resolver unas pocas ecuaciones no controversiales. Quiero decir proporcione una

> explicación. Cuando el algoritmo de Shor tiene que factorizar un número, usando $10^{500}$ o así veces recursos computacionales que se ve que están presentes, ¿dónde se factorizó el número? Hay solamente alrededor de $10^{80}$ átomos en el universo entero visible, un número absolutamente minúsculo comparado con $10^{500}$. Así si el universo visible fuera de la magnitud de la realidad física, ella no contendría ni remotamente los recursos requeridos para factorizar un número tan grande. ¿Quién lo factorizó, entonces? ¿Cómo y dónde se realizó el cómputo?

Hay bastante en el párrafo anterior para aprovechar por un filósofo emprendedor. En particular, ¿cómo *debería* responder un no creyente en Muchos-Mundos al desafío de Deutsch? En el resto de esta sección, me enfocaré en dos posibles respuestas.

La primera respuesta es negar que, si el algoritmo de Shor funciona como se predijo, eso sólo puede ser explicado postulando "vastos recursos computacionales". Al más obvio nivel, los teóricos de la complejidad todavía no han descartado la posibilidad de un algoritmo *clásico* de factorización rápido.[52] Más generalmente, que las computadoras cuánticas puedan resolver ciertos problemas súper polinomialmente más rápido que las computadoras clásicas no es un teorema, pero una (profunda, plausible) *conjetura*.[53,54] Si la conjetura fallara, entonces la puerta parecería abierta a lo que podríamos llamar "teorías de variable oculta en tiempo polinomial": teorías que reproducen las predicciones de la mecánica cuántica sin invocar cualquier cómputo fuera de P.[55] Éstas serían análogas a las teorías de variable *local* oculta en las que Einstein y otros habían confiado, antes que Bell las descartara.

Una segunda respuesta al desafío de Deutsch es que, aun cuando estamos de acuerdo que el algoritmo de Shor demuestra la realidad de vastos recursos computacionales en la Naturaleza, no es obvio que debamos pensar de esos recursos como "universos paralelos". ¿Por qué no decir simplemente que hay un universo y que es mecánico cuántico? ¿No

---

52. De hecho, no se puede descartar esa posibilidad, sin demostrar primero $\mathsf{P} \neq \mathsf{NP}$. Pero aun cuando el $\mathsf{P} \neq \mathsf{NP}$, un algoritmo rápido de factorización clásico podría *existir todavía*, de nuevo porque no se piensa que factorizar sea NP-completo.

53. Una versión formal de esta conjetura es $\mathsf{BPP} \neq \mathsf{BQP}$ dónde BPP (Bounded-Error Probabilistic Polynomial-Time, tiempo polinomial probabilístico de error limitado) y BQP (Bounded-Error Quantum Polynomial-Time, tiempo polinomial cuántico de error limitado) es respectivamente las clases de problemas eficientemente resoluble por algoritmos aleatorizados clásicos y cuánticos, respectivamente. Bernstein y Vazirani [29] mostró que $\subseteq \mathsf{BPP} \subseteq \mathsf{BQP} \subseteq \mathsf{PSPACE}$ dónde PSPACE es la clase de problemas resoluble por una máquina determinística de Turing usando una cantidad polinomial de *memoria* (pero posiblemente tiempo exponencial). Por esta razón, cualquier prueba de la conjetura $\mathsf{BPP} \neq \mathsf{BQP}$ implicaría también inmediatamente $\mathsf{P} \neq \mathsf{PSPACE}$. El último casi sería considerado un descubrimiento tan grande como $\mathsf{P} \neq \mathsf{NP}$.

54. Complicando asuntos, hay algoritmos cuánticos que demostrablemente logran aceleraciones exponenciales sobre cualquier algoritmo clásico: un ejemplo es el algoritmo de Simon [121], un importante predecesor del algoritmo de Shor. Sin embargo, la totalidad de tales algoritmos se formula en el "modelo de caja negra" (vea Beals et al. [23]), dónde el recurso a ser minimizado es el número de preguntas que un algoritmo hace a una caja negra hipotética. Porque es relativamente fácil de analizar, el modelo de caja negra es una fuente crucial de intuiciones sobre lo que *podría* ser verdad en el modelo de máquina de Turing convencional. Sin embargo, también se sabe que el modelo caja negra a veces nos desencamina acerca de la situación "real". Como ejemplo famoso, las clases de complejidad IP y PSPACE son iguales [117], a pesar de la existencia de una caja negra que las separa (vea Fortnow [57] para la discusión).
Además del modelo de caja negra, las separaciones exponenciales *incondicionales* entre las complejidades cuánticas y clásicas son conocidas en varios otros modelos restringidos, incluyendo la complejidad de comunicación [109].

55. Técnicamente, si la teoría de la variable oculta involucrara aleatoriedad clásica, entonces correspondería más estrechamente a la clase de complejidad BPP (Bounded-Error Probabilistic Polynomial-Time). Sin embargo, hoy hay fuerte evidencia que $\mathsf{P} = \mathsf{BPP}$ (ver Impagliazzo y Wigderson [80]).

refleja el lenguaje de los universos paralelos un *parroquialismo* irónico: un deseo de imponer una imagen familiar de ciencia-ficción sobre una teoría matemática que es más *extraña* que la ficción, que no coincide particularmente bien con *ninguna* de nuestras intuiciones pre-cuánticas (incluyendo intuiciones computacionales)?

Se puede aguzar el punto como sigue: si se tomara demasiado seriamente la explicación de cómo funciona una computadora cuántica (¡tanto como lo hacen muchos escritores populares!), entonces sería natural hacer inferencias adicionales sobre la computación cuántica que son llanamente erróneas. Por ejemplo:

> "*Usando sólo mil bits cuánticos (o qubits), una computadora cuántica podría almacenar* $2^{1000}$ *bits clásicos*".

Esto sólo es verdad para una definición bizarra de la palabra "almacenar". El problema fundamental es que, cuando mides el estado de una computadora cuántica, solamente ves uno de los posibles resultados; el resto desaparece. De hecho, un resultado célebre llamado *Teorema de Holevo* [75] dice que, usando *n* qubits, no hay ningún modo de almacenar más de *n* bits clásicos de modo que los bits puedan luego recuperarse confiablemente. En otras palabras: para al menos una definición natural de "capacidad de acarrear información", los qubits tienen exactamente la misma capacidad que los bits.

Para tomar otro ejemplo:

> "*A diferencia de una computadora clásica que sólo puede factorizar números probando divisores uno por uno, una computadora cuántica podría probar todos los posibles divisores en paralelo*."

Si las computadoras cuánticas pueden aprovechar vastos números de mundos paralelos, entonces lo anterior parece una suposición razonable acerca de cómo funciona el algoritmo de Shor. Pero *no es cómo funciona en absoluto*. Nótese que, si el algoritmo de Shor *funcionara* de ese modo, entonces no sólo podría usarse para factorizar enteros, sino también para la tarea mucho más grande de resolver problemas NP-completos en tiempo polinomial. (Como mencioné en nota a pie de página 12, se cree fuertemente que el problema de la factorización no es NP-completo). Pero contrariamente a un concepto erróneo común, no se sabe ni se cree que las computadoras cuánticas puedan resolver eficientemente problemas NP-completos.[56] Como de costumbre, el problema fundamental es que medir revela simplemente un solo resultado aleatorio $|x|$. Para ir alrededor de ese problema, y asegurar que el resultado *correcto* se observa con alta probabilidad, un algoritmo cuántico necesita generar un *patrón de interferencia*, en el cual los caminos computacionales que llevan a un resultado equívoco dado se cancelan unos a otros, mientras los caminos que llevan a un resultado correcto dado se refuerzan unos a otros. Éste es un requisito delicado, y hasta donde cualquiera sabe, sólo puede lograrse para unos pocos problemas, la mayoría los cuales (como el problema de la factorización) tiene estructura especial surgiendo de álgebra o teoría de los números.[57]

Un MultiMundista podría replicar: "seguro, estoy de acuerdo que la computación cuántica implica *emplear universos paralelos* en modos sutiles y no obvios, ¡mas todavía es emplear

---

**56.** Hay un algoritmo cuántico notable llamado *algoritmo de Grover* [70] que puede investigar cualquier espacio de $2^N$ posibles soluciones en sólo $\sim 2^{N/2}$ pasos. Sin embargo, el algoritmo de Grover representa una mejora *cuadrática* (raíz cuadrada) sobre la búsqueda de fuerza bruta clásica, en lugar de una mejora exponencial. Y sin suposiciones adicionales acerca de la estructura del espacio de búsqueda, el algoritmo de Grover es óptimo, como fue demostrado por Bennett et al. [27].

**57.** Aquéllos interesados en detalles adicionales de cómo funciona el algoritmo de Shor, pero aún no listos para una exposición matemática, podría querer tratar mi ensayo popular "Shor, I'll Do it" (Shor, Yo lo Haré) [1].

universos paralelos!". Pero incluso aquí hay una fascinante ironía. Supón que escogemos pensar de un algoritmo cuántico en términos de universos paralelos. Entonces para ponerlo crudamente, no sólo se deben interferir muchos universos para dar una gran amplitud final a la respuesta correcta; ¡ellos también deben, por interferir, *perder sus identidades como universos paralelos*! En otras palabras, en qué magnitud una colección de universos es útil para el cómputo cuántico, hasta qué punto es argumentable si debemos llamarlos "universos paralelos" en absoluto (como opuesto a partes de una auto interfiriente mancha mecánico-cuántica exponencialmente grande). Recíprocamente, en qué magnitud los universos tienen inambiguamente identidades separadas, hasta qué punto ellos son ahora "desadheridos" y fuera de contacto causal con todos los otros. Así podemos explicar las salidas de cualquier computación futura invocando sólo uno de los universos, y tratando los otros como hipotéticos no realizados.

Para clarificar, no considero ambas objeciones anteriores al argumento de Deutsch como decisivas, y estoy inseguro de qué pensar acerca del asunto. Mi propósito, estableciendo objeciones, fue simplemente ilustrar el potencial de la teoría de la computación cuántica para informar acerca de la Interpretación de Muchos-Mundos.

## 9 Nuevas Nociones Computacionales de Demostración

Desde la época de Euclides, ha habido dos nociones principales de demostración matemática:

(1) una "demostración" es una explicación verbal que induce un sentido de certeza (e idealmente, entendimiento) acerca de la declaración a ser demostrado, en cualquier matemático humano dispuesto y competente para seguirla.

(2) una "demostración" es una sucesión finita de símbolos que codifican deducciones sintácticas en algún sistema formal, la cual comienza con axiomas y finaliza con la declaración a ser demostrada.

La tensión entre estas dos nociones es un tema recurrente en la filosofía de la matemática.

Pero la ciencia teórica de la computadora trata regularmente con una tercera noción de demostración – una que parece haber recibido mucho menos análisis filosófico que cualquiera de las dos de arriba. Esta noción es la siguiente:

(3) Una "demostración" es cualquier proceso computacional o protocolo (real o imaginado) que puede terminar de una cierta manera si y sólo si la declaración a ser demostrada es verdadera.

### 9.1 Demostraciones de Conocimiento Cero

Como un ejemplo de esta tercera noción, considere *demostraciones de conocimiento cero*, introducidas por Goldwasser, Micali, y Racko [67]. Dados dos grafos $G$ y $H$, cada uno con $n \approx 10000$ vértices, supón que Merlín, un mago todo poderoso pero poco fiable, desea convencer a un escéptico rey Arturo que $G$ y $H$ no son isomorfos. Por supuesto, un modo en que Merlín podría hacer esto sería listar todos los $n!$ grafos que se obtienen permutando los vértices de $G$, entonces notar que ninguno de estos igualan a $H$. Sin embargo, una tal demostración agotaría claramente la paciencia de Arturo (de hecho, podría ni siquiera ser escrita dentro del universo observable). Alternativamente, Merlín podría señalar a Arturo alguna *propiedad* de $G$ y $H$ que los diferenciara: por ejemplo, quizá sus matrices de adyacencia tengan diferentes espectros de eigenvalores. Desafortunadamente, no está demostrado aún que, si $G$ y $H$ no son isomorfos, hay siempre una propiedad diferenciante que Arturo pueda verificar en tiempo polinomial en $n$.

Pero como fue notado por Goldreich, Micali y Wigderson [66], hay algo, en cambio, que Merlín puede hacer: dejar que Arturo lo *desafíe*. Merlín puede decir:

> Arturo, envíeme un nuevo grafo *K*, que usted obtenga *ya sea* permutando aleatoriamente los vértices de *G*, *o* permutando aleatoriamente los vértices de *H*. Entonces le garantizo que le diré, sin falla $K \cong G$ o $K \cong H$.

Está claro que, si *G* y *H* realmente no son isomorfos, entonces Merlín siempre puede responder a tales desafíos correctamente, por la suposición que él (Merlín) tiene poder computacional ilimitado. Pero está igualmente claro que, si *G* y *H* son isomorfos, entonces Merlín debe responder incorrectamente a algunos desafíos, sin tener en cuenta su poder computacional – puesto que una permutación aleatoria de *G* es estadísticamente indistinguible de una permutación aleatoria de *H*.

Este protocolo tiene cuatro características que ameritan reflexión por cualquiera interesado en la naturaleza de la demostración matemática.

Primero, el protocolo es *probabilístico*. Merlín no puede convencer a Arturo con certeza que G y H no son isomorfos, puesto que aun si fueran isomorfos, hay una probabilidad de 1/2 de que Merlín fuera afortunado y respondiera correctamente a un desafío dado (y, una probabilidad de $1/2^k$ de que respondiera *k* desafíos correctamente). Todo lo que Merlín puede hacer es ofrecer repetir el protocolo (digamos) 100 o 1000 veces, y por eso hacer menos probable que su demostración sea defectuosa que la que un asteroide cayera en Camelot, matándolo a él y a Arturo. ¡Además, hasta esta garantía estadística es sólo tan segura como la certeza de Arturo que sus elecciones durante el protocolo fueran "suficientemente aleatorias"! En otras palabras, no obstante Arturo generara sus desafíos "aleatorios" para Merlín – arrojando una moneda, rodando un dado, usando una fuente de números aleatorios de mecánica cuántica, etc. – tenemos que suponer que el método estuvo libre de "correlaciones desviadas" que permitirían que Merlín defraudara.

Segundo, el protocolo es *interactivo*. A diferencia con las nociones de demostración (1) y (2), Arturo no es más un destinatario pasivo de conocimiento, sino un jugador activo que desafía al demostrador. Sabemos por experiencia que la habilidad para *interrogar* a un participante de seminario – hacer preguntas que no haya podido anticipar, evaluar las respuestas, y entonces posiblemente hacer preguntas de seguimiento – frecuentemente acelera el proceso de figurarse si el portavoz sabe de lo que está hablando. La teoría de la complejidad fortalece nuestra intuición aquí, por medio de su descubrimiento de demostraciones interactivas para declaraciones (como "*G* y *H* no son isomorfos") cuyas demostraciones convencionales conocidas más cortas son exponencialmente más largas.

La tercera característica interesante del protocolo del no isomorfismo del grafo – una característica raramente mencionada – es que su solidez descansa implícitamente en una suposición *física*. A saber, si Merlín tuviera el poder (por medio de magia o de espionaje ordinario) "de asomarse al estudio de Arturo" y observar directamente si Arturo comenzó con *G* o *H*, entonces claramente podría responder correctamente a cada desafío incluso si $G \cong H$. Se sigue que la persuasión de la "demostración" de Merlín sólo puede ser tan fuerte como la creencia extramatemática de Arturo de que Merlín no tiene tales poderes. Por ahora, hay muchos otros ejemplos en la teoría de la complejidad de "demostraciones" cuya validez descansa en las limitaciones supuestas de los demostradores.

Como señala Shieber [118], las tres propiedades anteriores de protocolos interactivos también se mantienen para el Test de Turing discutido en la Sección 4. El Test de Turing es interactivo por definición, es probabilístico porque incluso un programa que imprimiera jerigonza aleatoria tendría alguna probabilidad no nula de pasar el test por casualidad, y depende de la suposición física de que el programa de inteligencia artificial no decepcione

(por ejemplo) consultando secretamente con un humano. Por estas razones, Shieber argumenta que podemos ver al *propio* Test de Turing como un protocolo interactivo temprano – uno que convence al verificador, no de un teorema matemático, sino de la capacidad del demostrador para conducta verbal inteligente.[58]

Sin embargo, quizás la característica más llamativa del protocolo de no isomorfismo del grafo es que es *conocimiento-cero*: un término técnico que formaliza nuestra intuición que "Arturo no aprende nada del protocolo, más allá de la verdad de la declaración que se está demostrando".[59] Porque todo lo que Merlín alguna vez le dice a Arturo es con cual grafo, *G* o *H*, comenzó (Arturo). ¡Pero Arturo *ya sabía* con qué gráfico comenzó! Esto significa que no sólo Arturo no gana ningún "entendimiento" de lo que hace a *G* y *H* no isomorfos, él ni siquiera gana la habilidad de demostrar a una tercera parte lo que Merlín le demostró a él. Este es otro aspecto de las demostraciones computacionales que no tiene análogo con las nociones de demostración (1) o (2).

Uno podría quejarse que, tan interesante como es la propiedad de conocimiento-cero, hasta ahora sólo hemos mostrado que es lograble para un problema extremadamente especializado. Y de hecho, exactamente como con la factorización de enteros, hay hoy fuerte evidencia de que el problema del isomorfismo del grafo *no* es NP-completo [33].[60,61] Sin embargo, en el mismo trabajo en que proporcionaron el protocolo de no isomorfismo del grafo, Goldreich, Micali y Wigderson [66] también presentaron un famoso protocolo de conocimiento-cero (llamado ahora *protocolo GMW*) para los problemas NP-completos. Por la definición de NP-completo (ver la Sección 3.1), el protocolo GMW significó que *cada declaración matemática que tiene una demostración convencional (digamos, en la teoría de conjuntos de Zermelo-Fraenkel) también tiene una demostración de conocimiento-cero de tamaño comparable*. Como una aplicación de ejemplo, supongan que ustedes han demostrado la Hipótesis de Riemann. Quieren convencer a los expertos del triunfo suyo, pero son paranoicos acerca de que ellos les roben el crédito. En ese caso, "todo" lo que necesitan hacer es

(1) reescribir su demostración en un lenguaje formal,

(2) codificar el resultado como la solución a un problema NP-completo, y entonces

---

58. Incidentalmente, esto proporciona un buen ejemplo de cómo las nociones de la teoría de la complejidad computacional pueden influir en la filosofía incluso sólo al nivel de metáfora, olvidándose de los resultados reales. En este ensayo no intenté coleccionar tales aplicaciones "metafóricas" de la teoría de la complejidad, simplemente porque había demasiadas de ellas.

59. Técnicamente, el protocolo es el "conocimiento cero de verificador honrado", significando que Arturo no aprende nada de su conversación con Merlín además de la verdad de la declaración demostrándose, suponiendo que Arturo sigue el protocolo correctamente. Si Arturo decepciona – por ejemplo, enviando un grafo *K* para el cual todavía no conoce un isomorfismo a *G* o a *H* – entonces la respuesta de Merlín podría de hecho decirle a Arturo algo nuevo. Sin embargo, Goldreich, Mical y Wigderson [66] también dieron un protocolo de una demostración más sofisticada para no isomorfismo del grafo que sigue siendo conocimiento cero incluso en el caso en que Arturo decepciona.

60. De hecho, ¡no hay incluso una creencia de consenso que el isomorfismo del grafo está fuera de P! La razón principal es que, en contraste con la factorizar enteros, el isomorfismo del gráfico resulta ser extremadamente fácil en la práctica. ¡De hecho, encontrar grafos no isomorfos que no puedan ser distinguido por simples invariantes, es en sí mismo un problema difícil! Y en el pasado, para varios problemas (como la programación lineal y la verificación de primalidad) que fueron largamente conocidos "eficazmente resolubles para propósitos prácticos", fue eventualmente demostrado que estaban también en P en sentido matemático estricto.

61. Hay también fuerte evidencia que existen demostraciones convencionales cortas para no isomorfismo del grafo – en otras palabras, que resultarán estar últimamente en NP no sólo isomorfismos del grafo sino también no isomorfismos del grafo [85].

(3) como matemáticos en una corte del siglo XVI que desafían a duelo a sus competidores, ¡inviten a los expertos a que ejecuten con ustedes el protocolo GMW en la Internet!

Suponiendo que ustedes respondan correctamente a todos sus desafíos, los expertos pueden llegar a tener *certeza estadística* de que ustedes poseen una demostración de la Hipótesis de Riemann, sin aprender nada *acerca de* esa demostración más allá de un límite superior en su longitud.

Mejor todavía, diferente al protocolo de no isomorfismo del grafo, el protocolo GMW no supone un mago súper poderoso – solamente un ente ordinario de tiempo polinomial, que sucede que conoce una demostración del relevante teorema. Como resultado, hoy el protocolo GMW es mucho más que una curiosidad teórica: él y sus variantes han encontrado aplicaciones mayores en criptografía de Internet, donde clientes y servidores necesitan frecuentemente demostrarse mutuamente que están siguiendo correctamente un protocolo sin revelar información secreta mientras lo hacen.

Hay, sin embargo, una importante advertencia: a diferencia del protocolo de isomorfismo de grafo, el protocolo GMW descansa esencialmente en una *hipótesis criptográfica*. Porque así es cómo funciona el protocolo GMW: tú (el demostrador) publicas miles de mensajes encriptados, "comprometiéndote" en cada uno con una pieza aleatoriamente alterada de tu prueba propuesta. Entonces te ofreces a descifrar una diminuta fracción de esos mensajes, para que observadores escépticos "hagan un chequeo muestral" de tu prueba, sin aprender nada de su estructura, más allá del hecho inútil que, digamos, el paso 1729 es válido (¿pero cómo pudiera *no* serlo?). Si los escépticos quieren aumentar su confianza en que tu prueba es sólida, entonces simplemente ejecutas el protocolo una y otra vez con ellos, usando cada vez un lote fresco de mensajes encriptados. Si los escépticos pudieran descifrar todos los mensajes en un solo lote, entonces podrían juntar las piezas de tu prueba – pero para hacer eso, necesitarían *quebrar* el código criptográfico subyacente.

### 9.2 Otras Nuevas Nociones

Permítanme mencionar otras cuatro nociones de prueba que los teóricos de la complejidad han explorado a fondo durante los últimos veinte años, y que podrían merecer atención filosófica.

*Pruebas interactivas de demostrador múltiple* [26, 20], en que Arturo intercambia mensajes con *dos* (o más) magos computacionalmente poderosos pero poco fiables. Aquí, Arturo podría convencerse de alguna afirmación matemática, pero sólo bajo la suposición de que los magos no pueden comunicarse *entre sí* durante el protocolo (la analogía usual es la de un detective que pone a dos sospechosos en celdas separadas para impedirles coordinar sus respuestas). Interesantemente, incluso magos no comunicantes podrían coordinar con éxito sus respuestas a los desafíos de Arturo en algunos protocolos de demostrador múltiple (y por eso convencer a Arturo de una falsedad) a través del uso de enredo cuántico (*quantum entanglement*) [41]. Sin embargo, se conjetura que otros protocolos son sólidos incluso contra magos enredados (*entangled*) [84].

*Demostraciones controlables probabilísticamente* [55, 18], que son demostraciones matemáticas codificadas en un formato especial de corrección de errores, para que se pueda tener confianza en su validez verificando sólo 10 o 20 bits escogidos aleatoriamente en un modo correlativo. El *Teorema PCP* (*Probabilistically Checkable Proofs*, Demostraciones Verificables Probabilísticamente) [17, 50], uno de los logros de coronación de la teoría de la complejidad, dice que cualquier teorema matemático, en cualquier sistema formal normal como la teoría de conjuntos de Zermelo-Fraenkel, puede convertirse en tiempo polinomial en un formato controlable probabilísticamente.

*Demostraciones cuánticas* (*quantum proofs*) [135, 6], qué son demostraciones que dependen para su validez de la salida de una computación cuántica – posiblemente, incluso una computación cuántica que requiere un "estado de demostración (*proof*)" enredado alimentado a ella como su entrada. A causa de que *n* bits cuánticos podrían requerir $2^n$ bits clásicos para simular, las demostraciones cuánticas tienen la propiedad de que nunca podría ser posible listar todos los "pasos" que entraron en la prueba, dentro de las restricciones del universo visible. Por esta razón, la creencia en la declaración matemática siendo demostrada podría depender de la creencia en la exactitud de la mecánica cuántica como una teoría física.

*Demostraciones y argumentos computacionalmente sólidos* [35, 95], qué descansan para su validez en la suposición que el demostrador estuvo limitado a computaciones de tiempo polinomial – así como la conjetura matemática de que elaborar un argumento convincente para una falsedad le hubiera tomado al demostrador más que tiempo polinomial.

¿Qué implicaciones tienen estos nuevos tipos de demostración para los fundamentos de la matemática? ¿Hacen ellos más dramático lo que debe "haber sido obvio todo el tiempo": que, como argumenta David Deutsch en *The Beginning of Infinity* (El Comienzo de la Infinidad) [49], las demostraciones son procesos físicos que tienen lugar en cerebros o computadoras, las cuales, por consiguiente, no tienen validez independiente de nuestras creencias sobre física? ¿Son los problemas surgidos esencialmente los mismos que aquéllos que surgen por demostraciones "convencionales" que requieren cómputos extensos, como la demostración del Teorema de los cuatro colores de Appel y Haken[13]? O, ¿representa algo cualitativamente nuevo apelar, en el curso de una "demostración matemática", a (digamos) la validez de la mecánica cuántica, la aleatoriedad de los números aparentemente aleatorios, o la falta de ciertos súper poderes por parte del demostrador? Se busca análisis filosófico.

## 10 Complejidad, Espacio y Tiempo

¿Qué puede decirnos la complejidad computacional sobre la naturaleza de espacio y tiempo? Una primera respuesta podría ser "no mucho": después de todo, pueden mostrarse las definiciones de clases de complejidad estándar como P como insensibles a tales detalles como el número de dimensiones espaciales, e incluso si la velocidad de la luz es finita o infinita.[62] Por otro lado, pienso que la teoría de la complejidad ofrece una intuición sobre las *diferencias* entre espacio y tiempo.

La clase de problemas resolubles usando una cantidad polinomial de memoria (pero posiblemente una cantidad exponencial de tiempo[63] se llama PSPACE, por Polynomial Space (espacio polinomial). Ejemplos de problemas de PSPACE incluyen simular sistemas dinámicos, decidiendo si una gramática regular genera todas las posibles cuerdas,

---

62. Más precisamente, máquinas de Turing con cintas unidimensionales es polinomialmente equivalente máquinas de Turing con cintas *k*-dimensionales para cualquier *k*, y también son polinomialmente equivalentes a *máquinas de acceso aleatorio* (que pueden "saltar" a cualquier dirección de memoria en unidad de tiempo, sin restricción de localidad).
Por otra parte, si nos preocupamos de las diferencias polinomiales en velocidad, y especialmente si queremos estudiar modelos de computación paralela, detalles acerca de la disposición espacial de los elementos de computación y memoria (así como la velocidad de comunicación entre los elementos) puede llegar a ser vitalmente importante.

63. ¿Por qué "sólo" una cantidad exponencial? Porque una máquina de Turing con *B* bits de memoria puede correr por no más de $2^B$ pasos temporales. Después de eso, la máquina o debe detenerse o retornar a una configuración previamente visitada (entrando en un ciclo infinito por eso).

y ejecuta una estrategia óptima en juegos de dos jugadores como Reversi, Connect Four, y Hex.[64] No es difícil mostrar que PSPACE es al menos tan poderoso como NP

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}.$$

Aquí EXP representa la clase de problemas resolubles que usa una cantidad exponencial de tiempo, y también posiblemente una cantidad exponencial de memoria.[65] Se cree que cada una de las contenciones anteriores es estricta, aunque la única *demostrada* actualmente que lo es, es $\mathsf{P} \neq \mathsf{EXP}$, por un resultado importante de 1965 de Hartmanis y Stearns [71] llamado Teorema de la Jerarquía de Tiempo.[66, 67]

Nótese, en particular, que $\mathsf{P} \neq \mathsf{NP}$ implica $\mathsf{P} \neq \mathsf{PSPACE}$. Así mientras $\mathsf{P} \neq \mathsf{PSPACE}$ aún no se ha demostrado, es una conjetura sumamente segura por los estándares de la teoría de la complejidad. En forma de eslogan, los teóricos de la complejidad creen que *espacio es más poderoso que tiempo*.

---

64. ¡Nótese que para hablar sobre la complejidad computacional de tales juegos, necesitamos primero generalizarlos a un tablero de $n$ x $n$! Pero si hacemos eso, entonces para muchos juegos naturales, el problema de determinar qué jugador tiene la victoria de una posición dada no sólo está en PSPACE, sino PSPACE-*completo* (es decir, él captura la entera dificultad de la clase PSPACE). Por ejemplo, Reisch [111] mostró que esto es verdad para Hex.
¿Qué acerca de una generalización conveniente de ajedrez a un tablero de $n$ x $n$? Eso también está en PSPACE – pero hasta donde cualquiera sabe, sólo si imponemos un límite superior polinomial en el número de movimientos en un juego de ajedrez. Sin tal restricción, Fraenkel y Lichtenstein [60] mostraron que el ajedrez es EXP-completo; con tal restricción, Storer [129] mostró que el ajedrez es PSPACE-completo.

65. En este contexto, llamamos a una función $f(n)$ "exponencial" si puede ser limitada superiormente por $2^{p(n)}$, para algún $p$ polinomial. También, note que la memoria más que exponencial sería inútil aquí, puesto que una máquina de Turing que corre por $T$ pasos temporales puede visitar a lo sumo $T$ celdas de memoria.

66. Más generalmente, el Teorema de la Jerarquía del Tiempo muestra que, si $f$ y $g$ son dos funciones cualesquiera "que se comportan suficientemente bien" satisfaciendo $f(n) \langle\langle g(n)$ (por ejemplo: $f(n) = n^2$ y $g(n) = n^3$), entonces *hay problemas computacionales resolubles en tiempo $g(n)$ pero no en tiempo $f(n)$*. La demostración de este teorema usa diagonalización, y puede pensarse como una versión escalada hacia abajo de la irresolubilidad del problema de detenerse del Test de Turing. Es decir, argumentamos que, *si* siempre fuera posible simular una máquina de Turing de tiempo $g(n)$ por una máquina de Turing de tiempo $f(n)$, entonces podríamos construir una máquina de tiempo $g(n)$ que "predijera su propia salida por adelantado" y entonces la salida de algo más – causando una contradicción por eso.
Usando similares argumentos, podemos mostrar (por ejemplo) que allí existen problemas computacionales resolubles usando $n^3$ bits de memoria pero no usando $n^2$ bits de memoria, y así sucesivamente en la mayoría de los casos donde queremos comparar *más versus menos del mismo recurso computacional*. En la teoría de la complejidad, la parte difícil es comparar dos recursos *diferentes*: por ejemplo, determinismo versus indeterminismo (el problema $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$), tiempo versus espacio ($\mathsf{P} \stackrel{?}{=} \mathsf{PSPACE}$), o computación clásica versus computación cuántica ($\mathsf{BPP} \stackrel{?}{=} \mathsf{BQP}$). Porque en esos casos, la diagonalización ya no funciona.

67. El hecho que $\mathsf{P} \neq \mathsf{EXP}$ tiene una implicación entretenida, a menudo atribuida a Hartmanis: a saber, por lo menos una de las tres desigualdades
(i) $\mathsf{P} \neq \mathsf{NP}$
(ii) $\mathsf{NP} \neq \mathsf{PSPACE}$
(iii) $\mathsf{PSPACE} \neq \mathsf{EXP}$
¡debe ser verdad, aunque demostrar que cualquiera de ellas es verdad *individualmente* representaría un adelanto titánico en la matemática!
La observación anterior a veces se ofrece como evidencia circunstancial para $\mathsf{P} \neq \mathsf{NP}$. De todos nuestros centenares de creencias no demostradas sobre desigualdades entre pares de clases de complejidad, una gran fracción de ellas debe ser correcta, simplemente para evitar contradecir los teoremas de jerarquía. Así entonces, ¿por qué no $\mathsf{P} \neq \mathsf{NP}$ en particular (dado que nuestra intuición allí es más fuerte que nuestras intuiciones para la mayoría de las otras desigualdades)?

Ahora, algunas personas han preguntado cómo es posible que tal demanda pudiera ser consistente con la física moderna. ¿No nos enseñó Einstein que espacio y tiempo son meramente dos aspectos de la misma estructura? Una respuesta inmediata es que, incluso dentro de la teoría de la relatividad, espacio y tiempo no son intercambiables: el espacio tiene una signatura positiva mientras que el tiempo tiene una signatura negativa. En la teoría de la complejidad, la diferencia entre espacio y tiempo se manifiesta en el hecho franco de que se pueden reusar las mismas celdas de memoria una y otra vez, pero no se pueden reusar los mismos momentos de tiempo.[68]

Todavía, tan trivial como suena esa observación, ella conduce a un interesante pensamiento. Supón que las leyes de la física nos permitieran viajar *hacia atrás* en el tiempo. En tal caso, es natural imaginar que el tiempo se volvería un "recurso reusable" tal como es el espacio – y que, como resultado, los cómputos arbitrarios de PSPACE caerían dentro de nuestro alcance. ¿Pero es esa simplemente una especulación ociosa, o podemos justificarla rigurosamente?

### 10.1 Curvas Temporales Cerradas

Tanto Filósofos, como entusiastas de la ciencia-ficción, han estado desde hace tiempo interesados en la posibilidad de curvas temporales cerradas (*closed timelike curves*, CTCs), que surgen en ciertas soluciones a ecuaciones de campo de la relatividad general de Einstein.[69] En una comprensión tradicional, el problema filosófico central promovido por las CTCs es la paradoja del abuelo. Ésta es la situación donde regresas en el tiempo para matar a tu propio abuelo, por consiguiente, nunca naces, por consiguiente no matas a tu abuelo, por consiguiente naces, y así sucesivamente. ¿Esta contradicción implica inmediatamente que las CTCs son imposibles? No, no lo son: sólo podemos concluir que, *si* existen CTCs, entonces las leyes de físicas deben de algún modo impedir que se promueva la *paradoja del abuelo*. ¿Cómo podrían hacer eso? Una ilustración clásica es que "cuando regresas en el tiempo para tratar de matar a tu abuelo, el arma se atasca" – u ocurre inevitablemente algún otro evento "improbable" para mantener el estado del universo. Pero, ¿por qué debiéramos imaginar que siempre estará disponible tan conveniente "afuera", en cada experimento físico que involucre CTCs? Normalmente, nos gusta imaginar que tenemos la libertad para diseñar un experimento como quiera que lo deseemos, sin que la Naturaleza imponga condiciones al experimento (por ejemplo: "cada arma debe atascarse algunas veces"), cuyas razones sólo pueden entenderse en términos de eventos distantes o hipotéticos.

En su trabajo de 1991 "Quantum mechanics near closed timelike lines", Deutsch [47] dio una elegante propuesta para las paradojas de eliminar al abuelo. En particular, mostró que, con tal que supongamos que las leyes de la física son cuántico-mecánicas (o incluso sólo clásicamente probabilísticas), cada experimento que involucra una CTC admite por lo menos un punto fijo: es decir, una manera de satisfacer las condiciones del experimento que asegura evolución consistente. Formalmente, si $S$ es la asignación de estados cuánticos a ellos mismos inducida por "ir alrededor de la CTC una vez", entonces un punto fijo es cualquier estado mixto cuántico[70] $\rho$ tal que $S(\rho) = \rho$. La existencia de tal $\rho$ sigue

---

68. Ver mi blog www.scottaaronson.com/blog/?p=368 para más sobre este tema.

69. Aunque no se sabe si esas soluciones son "físicas": por ejemplo, si pueden sobrevivir o no en una teoría cuántica de la gravedad (vea [97]).

70. En mecánica cuántica, puede pensarse un *estado mixto* como una distribución de probabilidad clásica sobre estados cuánticos. Sin embargo, un giro importante es que el mismo estado mixto puede representarse por distribuciones de probabilidad *diferentes*: por ejemplo, una mezcla igual de estados $|0\rangle$

argumentos simples de álgebra lineal. Como ilustración, la "resolución de la paradoja del abuelo" es ahora que has nacido con probabilidad 1/2, y *si* has nacido, regresas en el tiempo para matar a tu abuelo – de lo cual se sigue que has nacido con probabilidad 1/2, y así sucesivamente. Meramente tratando los estados como probabilísticos (como, en algún sentido, ellos *tienen* que ser en mecánica cuántica[71]), hemos hecho consistente la evolución del universo.

Pero la cuenta de CTCs de Deutsch enfrenta por lo menos tres serias dificultades. La primera dificultad es que los puntos fijos no podrían ser únicos: podría haber muchos estados mixtos $\rho$ tales que $S(\rho) = \rho$, y entonces la pregunta que surge es cómo la Naturaleza escoge uno de ellos. Para ilustrar, considere la *anti-paradoja del abuelo*: un bit $b \in \{0, 1\}$ que viaja alrededor de una CTC sin cambiar. Podemos consistentemente suponer $b = 0$, o $b = 1$, o cualquier mezcla probabilística de los dos – y al contrario de la situación usual en física, aquí no hay ninguna posible condición de límite que pudiera resolver la ambigüedad.

La segunda dificultad, señalada por Bennett et al. [28], es la propuesta de que Deutsch viola la interpretación estadística de estados cuánticos mixtos. Así por ejemplo, si la mitad de un par enredado

$$\frac{|0\rangle_A |0\rangle_B + |1\rangle_A |1\rangle_B}{\sqrt{2}}$$

se coloca dentro de la CTC, mientras la otra media permanece fuera de la CTC, entonces el proceso de encontrar un punto fijo "romperá" el enredo entre las dos mitades. Como "remedio" para este problema, Bennett et al. sugieren requerir que el punto fijo de la CTC sea independiente del resto del universo entero. ¡Para mí, este remedio es tan drástico que básicamente equivale a definir las CTCs fuera de existencia!

Motivados por estas dificultades, Lloyd et al. [91] propusieron recientemente una cuenta completamente diferente de las CTCs, basada en *tele-portación post-seleccionada*. La cuenta de Lloyd el al. evita los dos problemas anteriores – aunque quizás no sorprendentemente, introduce otros problemas propios.[72] Mi visión, para lo que pueda valer, es que Lloyd el al. están hablando menos sobre "verdaderas" CTCs como yo entendería el concepto, que sobre experimentos mecánico-cuánticos post-seleccionados que *simulan* CTCs en ciertos respetos interesantes. Si hay cualquier controversia en física que convoque atención experta filosófica, seguramente ésta es una de ellas.

### 10.2 El Principio Evolutivo

Hasta ahora, aún no hemos mencionado la que veo como la principal dificultad con la cuenta de Deutsch de CTCs. ¡Esto es lo que podría requerir la Naturaleza para encontrar un punto fijo para resolver un problema computacional astronómicamente difícil! Para ilustrar,

---

y $|1\rangle$ es físicamente indistinguible de una mezcla igual de $\frac{|0\rangle + |1\rangle}{\sqrt{2}}$ y $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$. Esto es porque los estados mixtos se representan matemáticamente usando el formalismo de la matriz de densidad de Heisenberg.

71. En más detalle, la propuesta de Deutsch funciona si el espacio de estados consiste en distribuciones clásicas de probabilidad $\mathcal{D}$ o estados cuánticos mixtos, pero no si ella consiste en estados puros $|\psi\rangle$. Así, *si* se creyera que los estados fundamentales en física fueran sólo los puros, y que las distribuciones de probabilidad y los estados mixtos siempre reflejaran ignorancia subjetiva, se podría rechazar la propuesta de Deutsch sobre ese fundamento.

72. En particular, en la propuesta de Lloyd el al, el único modo de tratar con la paradoja del abuelo es por alguna variante de "el arma se atasca": hay evoluciones con ninguna solución consistente, y necesita postularse que las leyes de física son tales que ellas nunca ocurren.

considera un guión de ciencia-ficción en el que regresas a tiempo y le dictas a Shakespeare sus obras. Shakespeare te agradece por ahorrarle el esfuerzo, publica literalmente las obras que le dictaste, y siglos después te llegan las obras, después de lo cual regresas a tiempo y se las dictas a Shakespeare, etc.

Nota que, aquí no hay ninguna contradicción lógica en contraste con la paradoja del abuelo: la historia como la contamos es completamente consistente. Pero la mayoría de las personas la encuentra "paradójica" en cualquier caso. ¡De algún modo después de todo, *Hamlet* se escribe, sin que nadie se tome jamás el trabajo de escribirlo! Como Deutsch [47] perceptivamente observó, si hay una "paradoja" aquí, entonces no es de lógica sino de *complejidad computacional*. Específicamente, el relato viola un principio de sentido común que podemos articular flojamente como sigue:

**El conocimiento requiere un proceso causal que lo traiga a la existencia.**

¡Como muchos otros principios importantes, este podría no reconocerse como un "principio" en absoluto antes de que contemplemos situaciones que lo violan! Deutsch [47] llama a este principio el Principio Evolucionario (EP). Nótese que alguna versión del EP fue invocada tanto por el argumento del relojero ciego de William Paley, e (irónicamente) por los argumentos de Richard Dawkins [45] y otros ateos contra la existencia de un diseñador inteligente.

En mi artículo "NP-Complete Problems and Physical Reality" [4], propuse y argumenté un análogo de complejidad teórica del EP que llamé la *Suposición de Dureza* NP:

**No hay medios físicos para resolver problemas NP-completos en tiempo polinomial.**

La declaración anterior implica el $\mathsf{P} \neq \mathsf{NP}$, pero es más fuerte en cuanto abarca computación probabilística, computación cuántica, y cualquier otro modelo computacional compatible con las leyes de la física. Vea [4] para un estudio de resultados recientes que afectan la Suposición de Dureza NP, análisis de contra ejemplos reclamados a la suposición, y posibles implicaciones de la suposición para la física.

### 10.3 Computación de Curva Temporal Cerrada

¿Pero podemos mostrar más rigurosamente que las curvas temporales cerradas violarían la Suposición de Dureza NP?

De hecho, permítasenos ahora mostrar que, en un universo dónde los cómputos arbitrarios podrían ser realizados dentro de una CTC, y donde la Naturaleza tendría que encontrar un punto fijo para la CTC, podríamos resolver problemas NP-completos que usan sólo recursos polinomiales.

Podemos modelar cualquier instancia del problema NP-completo por una función $f$: $\{0,..., 2^n - 1\} \rightarrow \{0, 1\}$ que asigne a cada posible solución $x$ el bit 1 si $x$ es válida, o el 0 si $x$ es inválida. (Aquí, para la conveniencia, identificamos cada cuerda de solución de $n$-bits $x$ con el entero no negativo que $x$ codifica en binario.) Nuestra tarea, entonces, es encontrar una $x \in \{0,..., 2^n - 1\}$ tal que $f(x) = 1$. Podemos resolver este problema exactamente con una sola evaluación a $f$, con tal de que podamos ejecutar el siguiente programa de computación $C$ dentro de una curva temporal cerrada [36, 4, 7]:

```
Dada entrada x ∈ {0,…, 2^n – 1}:
Si f (x) = 1, entonces salida x
De otro modo, salida (x + 1) mod 2^n
```

Suponiendo que existe por lo menos una tal $x$ que $f(x) = 1$, los *únicos puntos fijos* de $C$ – eso es, los únicos modos para que la salida de $C$ iguale a su entrada – son para C entrar, y

salir, tal solución válida *x*, la cual por consiguiente aparece en registro de salida de C "como por magia". (Si no hay ninguna solución válida, entonces los puntos fijos de *C* serán simplemente superposiciones uniformes o distribuciones de probabilidad sobre toda $x \in \{0,..., 2^n - 1\}$).

Extendiendo la idea anterior, John Watrous y yo [7] (siguiendo una sugerencia de Fortnow) mostramos recientemente que una computadora de CTC en el modelo de Deutsch pudiera resolver todos los problemas en PSPACE. (Recordar que se cree que PSPACE es aún más grande que NP). Más sorprendentemente, también mostramos que PSPACE constituye el límite de lo que puede hacerse con una computadora de CTC; y que esto es verdad si la computadora de CTC es clásica o cuántica. Una consecuencia de nuestros resultados es que la "intuición ingenua" acerca de computadoras CTC – cuyo efecto sería "hacer espacio y tiempo equivalentes como recursos computacionales" – es finalmente correcta, aunque no por razones ingenuas.[73] Una segunda y divertida consecuencia es que, una vez que las curvas temporales cerradas están disponibles, ¡cambiar de computadoras clásicas a cuánticas no proporciona ningún beneficio *adicional*!

Es importante darse cuenta de que nuestros algoritmos para resolver problemas duros con CTCs *no* se reducen simplemente a "usar grandes montos de tiempo para encontrar la respuesta, luego enviarla hacia atrás en el tiempo antes que la computadora comience". Porque hasta en el exótico escenario de una computadora de viaje en el tiempo, todavía requerimos que todos los recursos usados *dentro* de la CTC (tiempo, memoria, etc.) sean limitados polinomialmente. Así, la habilidad de resolver problemas duros llega solamente de la *consistencia causal*: el requerimiento de que la Naturaleza debe encontrar alguna evolución para la computadora CTC que evite las paradojas del abuelo.

En la cuenta alternativa de Lloyd et al. de CTCs basadas en postselección [91], *también* pueden resolverse los problemas duros, aunque por razones diferentes. En particular, edificando sobre un resultado más temprano mío [5], Lloyd et al. muestran que el poder de su modelo corresponde a una clase de complejidad llamada PP (Probabilistic Polynomial-Time, tiempo polinomial probabilístico) que se cree que es estrictamente menor que PSPACE pero estrictamente más grande que NP. Así, se podría decir que el modelo de Lloyd et al. "mejora" la situación computacional, ¡pero no por mucho!

Así que uno podría preguntarse: ¿hay algún modo de que las leyes de física pudieran permitir CTCs, sin abrir la puerta a implausibles poderes computacionales? Allí resta al menos una posibilidad interesante que me fue comunicada por el filósofo Tim Maudlin.[74] Quizá las leyes de la física tienen la propiedad que, no importando qué cómputos se realizan dentro de una CTC, la Naturaleza siempre tiene un "afuera" que evita la paradoja del abuelo, pero *también* evita resolver problemas computacionales duros – análogo al "arma que se atasca" en la paradoja original del abuelo. Tal afuera podría involucrar (por ejemplo) un asteroide golpeando la computadora de CTC, o la computadora fallando por otras misteriosas razones. Por supuesto, *cualquier* computadora en el mundo físico tiene alguna probabilidad no nula de fallar, pero ordinariamente imaginamos que la probabilidad de falla puede hacerse despreciablemente pequeña. Sin embargo, en situaciones donde la

---

73. Específicamente, *no* es verdad que en un universo CTC, una cabeza de cinta de máquina de Turing pudiera simplemente viajar de un lado a otro en el tiempo del mismo modo que viaja de un lado a otro en el espacio. Si se piensa de este modo, entonces realmente se tiene en mente un segundo, "meta-tiempo", mientras el tiempo "original" se ha vuelto meramente una dimensión más del espacio. Para poner el punto diferentemente: aunque una CTC hiciera tiempo *cíclico*, el tiempo todavía retendría su *direccionalidad*. Ésta es la razón por la qué, si queremos mostrar que computadoras CTC tienen el poder de PSPACE, necesitamos un argumento no trivial que involucra la consistencia causal.

74. Esta posibilidad también se discute en detalle en el trabajo de Deutsch [47].

Naturaleza está siendo "forzada" a encontrar un punto fijo, puede ser que las "misteriosas fallas de la computadora" llegarían a ser la norma en lugar de la excepción.

Para resumir, pienso que la teoría de la complejidad computacional *cambia* los problemas filosóficos por el viaje en el tiempo al pasado. Mientras la discusión tradicionalmente se enfocó en la paradoja del abuelo, hemos visto que no hay ninguna escasez de modos para la Naturaleza para evitar inconsistencias lógicas, incluso en un universo con CTCs. ¡El problema "real", entonces, es cómo escapar de *otras* paradojas que surgen en el curso de domar la paradoja del abuelo! Probablemente primera entre aquéllas está la "paradoja de la complejidad computacional", de problemas NP-completos y aun más duros siendo resueltos como por magia.

## 11 Economía

En la economía clásica, los agentes son modelados como racionales, agentes bayesianos que cualquier acción que tomen aumentará al máximo su utilidad esperada $\mathrm{E}_{\omega\in\Omega}[U(\omega)]$, dadas sus probabilidades subjetivas $\{p_\omega\}_{\omega\in\Omega}$ sobre todos los posibles estados $\omega$ del mundo.[75] Esto, por supuesto, es una caricatura que casi parece diseñada para ser atacada, y se la ha atacado desde casi todos los ángulos. Por ejemplo, los humanos no están ni siquiera cerca de los agentes bayesianos racionales, sino que sufren de sesgos cognoscitivos muy bien conocidos, como exploraron Kahneman y Tversky [81] entre otros. Además, la visión clásica parece no dejar ningún espacio para criticar creencias de la gente (es decir, sus probabilidades anteriores) o sus funciones de utilidad como irracionales – aún es fácil preparar probabilidades anteriores o funciones de utilidad que conducirían a una conducta que casi cualquiera consideraría insana. Un tercer problema es que, en juegos con varios agentes que cooperan o compiten actuando simultáneamente, la economía clásica garantiza la existencia de por lo menos un *equilibrio de Nash* entre las estrategias de los agentes. Pero la situación usual es que hay múltiples equilibrios, y no hay entonces ningún principio general para predecir qué equilibrio prevalecerá, aunque la elección pudiera significar la diferencia entre la guerra y la paz.

La teoría de la complejidad computacional puede contribuir a los debates sobre los fundamentos de la economía mostrando que, incluso en la situación idealizada de agentes racionales que en su totalidad tienen información perfecta sobre el estado del mundo, será frecuentemente *computacionalmente intratable* para esos agentes actuar de acuerdo con la economía clásica. Por supuesto, alguna versión de esta observación ha sido reconocida durante mucho tiempo en la economía. Hay una literatura numerosa sobre *racionalidad limitada* (volviendo al trabajo de Herbert Simon [122]) que estudia la conducta de agentes económicos cuyas habilidades para tomar decisiones están limitadas de un modo u otro**.**

### 11.1 Racionalidad Limitada y el Dilema Iterado de los Prisioneros

Como un ejemplo de una visión que emerge de esta literatura, considere el Dilema del Prisionero Iterado Finito. Éste es un juego dónde dos jugadores se encuentran para algún número fijo de rondas $N$, que es finito y de conocimiento común entre los jugadores. En cada ronda, ambos jugadores pueden "Desertar" o "Cooperar" (no sabiendo la elección del otro jugador) después de lo cual reciben los siguientes pagos:

| | $\text{Desertar}_2$ | $\text{Cooperar}_2$ |
|---|---|---|
| $\text{Desertar}_1$ | 1, 1 | 4, 0 |
| $\text{Cooperar}_1$ | 0, 4 | 3, 3 |

75. Aquí suponemos por simplicidad que es conjunto de posibles estados $\Omega$ es contable; de otro modo podríamos por supuesto usar una medida de probabilidad continua.

Ambos jugadores recuerdan la entera historia previa de la interacción.

Es claro que los jugadores estarán mejor juntos (es decir, maximizan la suma de sus pagos) si ambos cooperan. Desafortunadamente, está igualmente claro que, *si* los jugadores se encuentran sólo para una ronda (N = 1), entonces la cooperación no es un *equilibrio*. Sin tener en cuenta lo que hace el Jugador 1, el Jugador 2 lo hará mejor desertando que cooperando, y viceversa.

Por otro lado, si *el número de rondas N fuera desconocido o infinito*, entonces los jugadores podrían decidir cooperar racionalmente, similarmente a cómo los humanos deciden cooperar en la vida real. Es decir, el Jugador 1 razona que si él deserta, entonces el Jugador 2 podría desquitarse desertando en rondas futuras, y el Jugador 2 razona igualmente. Ambos jugadores concluyen que, en una carrera larga, ellos harían mejor, *incluso para ellos mismos*, cooperando.

La "paradoja" es ahora la siguiente: ¡tan pronto *se conoce* el número de rondas *N*, el razonamiento anterior colapsa completamente! Porque claramente, ningún jugador puede tener algo que perder desertando *en la última ronda*, cuando el otro jugador ya no tiene ninguna oportunidad para desquitarse. Así si ambos jugadores son racionales, entonces eso es exactamente lo que harán. Pero si ambos jugadores saben que los dos desertarán en la ronda *N*, entonces *tampoco* ninguno tiene nada que perder desertando en la ronda $N-1$. Por supuesto, pueden continuar inductivamente hacia atrás de esta manera hasta la primera ronda. "Predecimos", por consiguiente, que ambos jugadores desertarán en cada ronda aunque eso no está en los propios intereses de los jugadores, ni en lo que los humanos reales hacen en los experimentos. Y todo porque *se conoció* el número de rondas *N*: ¡algo que parece intuitivamente como que no debiera representar ninguna diferencia!

En 1985, Neyman [99] propuso una solución ingeniosa de esta paradoja. ¡Específicamente, mostró que si los dos jugadores tienen *suficientemente pocos recuerdos* – técnicamente, si son autómatas finitos con $k$ estados, para $2 \leq k \leq N$ – entonces la cooperación se vuelve un equilibrio una vez más! La intuición básica es que, si a ambos jugadores les falta bastante memoria para contar hasta *N*, y ambos saben eso, y ambos saben que ambos saben eso, y así sucesivamente, entonces el argumento inductivo en el último párrafo falla, puesto que asume estrategias intermedias que ningún jugador puede llevar a cabo.

Mientras las consideraciones de la complejidad vencen *algunas* de las conclusiones contraintuitivas de la economía clásica, igualmente interesante para mí es que ellas no vencen otras. Como ejemplo, mostré en [3] el celebrado *teorema de acuerdo* de Robert Aumann [19] – los agentes bayesianos perfectos con probabilidades anteriores comunes nunca pueden "estar de acuerdo en discrepar" – persiste incluso en presencia de limitada comunicación entre los agentes.

Hay muchos otros resultados interesantes en la literatura de racionalidad limitada, demasiados para hacerles justicia aquí (pero ver Rubinstein [113] para un estudio). Por otro lado, la "racionalidad limitada" es algo de una frase para todo, abarcando casi toda desviación imaginable de la racionalidad – incluyendo los sesgos cognitivos humanos, límites en recolección de información y comunicación, y la restricción de estrategias para una forma específica (por ejemplo, funciones lineales de umbral). Muchas de estas desviaciones tienen poco que ver con la complejidad computacional per se. Así que la pregunta permanece de si la complejidad computacional puede proporcionar *específicamente* nuevas visiones sobre el comportamiento económico.

### 11.2 La Complejidad de los Equilibrios

Hay algunos adelantos muy recientes que sugieren que la respuesta es sí. Considere el problema de encontrar un equilibrio de un juego de dos jugadores, dada la matriz de pago $n \times n$ como entrada. En el caso especial de juegos de suma cero (que von Neumann estudió en 1928), se sabe desde hace mucho tiempo cómo resolver este problema en una cantidad de tiempo polinomial en $n$, por ejemplo por reducción a programación lineal. Pero en 2006, Daskalakis, Goldberg, y Papadimitriou [44] (con mejoras de Chen y Deng [39]) demostraron el espectacular resultado que, para un juego *general* (no necesariamente de suma cero) de dos jugadores, encontrar un equilibrio de Nash es PPAD-completo. Aquí PPAD ("Polynomial Parity Argument, Directed", argumento de paridad polinomial, dirigido) es, hablando gruesamente, la clase de *todos* los problemas de búsqueda para la cual se garantiza que existe una solución por la misma razón combinatoria que cada juego tiene al menos un equilibrio de Nash. Nótese que encontrar un equilibrio de Nash *no puede* ser NP-completo, por la razón técnica que NP es una clase de problemas de *decisión*, y la respuesta al problema de decisión "¿tiene este juego un equilibrio de Nash?" siempre es sí. Pero el resultado de Daskalakis et al. dice (informalmente) que el problema de la búsqueda para *encontrar* un problema de Nash está "tan cerca de NP-completo como posiblemente pueda estar", sujeto a su versión de decisión que es trivial. Los resultados similares de completitud de PPAD son ahora conocidos para otros problemas económicos fundamentales, tales como encontrar precios de compensación del mercado en mercados de Arrow-Debreu [38].

Por supuesto, se puede debatir la relevancia económica de estos resultados: ¿por ejemplo, qué tan a menudo levantan realmente su cabeza en la práctica ahora que sabemos[76] que la dureza computacional es inherente a los teoremas de equilibrio económico? ¡Pero se puede similarmente debatir la relevancia económica de los teoremas mismos de equilibrio! En mi opinión, si el teorema de que los equilibrios de Nash existen se considera relevante para los debates sobre (digamos) los mercados libres contra la intervención gubernamental, entonces el teorema que *encontrar* esos equilibrios es PPAD-completo también debe ser considerado relevante.

## 12 Conclusiones

El propósito de este ensayo fue ilustrar como pudiera enriquecerse la filosofía tomando en cuenta la teoría de la complejidad computacional, tanto como se enriqueció hace casi un siglo tomando en cuenta la teoría de la computabilidad. En particular, argumenté que la complejidad computacional proporciona nuevas intuiciones en el contenido explicativo del darwinismo, la naturaleza del conocimiento y demostración matemáticos, computacionalismo, sintaxis versus semántica, el problema de la omnisciencia lógica, debates que rodean al Test de Turing y al Salón Chino, el problema de la inducción, los fundamentos de la mecánica cuántica, las curvas temporales cerradas, y la racionalidad económica.

De hecho, podría decirse que la pregunta "real" es qué problemas filosóficos *no* tienen importantes aspectos de complejidad computacional. Mi propia opinión es que probablemente *hay* problemas tales (incluso dentro de la filosofía analítica), y que un buen candidato es el problema de qué debiéramos tomar como "lecho de roca de la realidad matemática": es decir, el conjunto de declaraciones matemáticas que son objetivamente verdaderas o falsas, sin tener en cuenta si pueden demostrarse o refutarse en un sistema formal dado. Para mí, si no estamos dispuestos a decir que una máquina de Turing $M$ dada acepta, rechaza, o corre por siempre (cuando comenzó en una cinta en blanco) – y que lo cual es un hecho objetivo, independiente de nuestras teorías axiomáticas formales, las leyes

76. Sujeto, como es usual, a suposiciones de complejidad ampliamente creídas.

de la física, la biología del cerebro humano, las convenciones culturales, etc. – *entonces no tenemos ninguna base para hablar sobre cualquiera de esas otras cosas* (teorías axiomáticas, las leyes de la física, y así sucesivamente). Además, los requerimientos de recursos de *M* son irrelevantes aquí: aun cuando *M* sólo detiene después de $2^{2^{10000}}$ pasos, su rendimiento es definido tan matemáticamente como si se hubiera detenido después de 10 pasos.[77]

¿Podemos decir algo general acerca de cuándo una perspectiva de complejidad computacional es útil en filosofía, y cuándo no lo es? Extrapolando de los ejemplos en este ensayo, yo diría que la complejidad computacional tiende a ser útil cuándo queremos saber si un hecho particular *realiza algún trabajo explicativo*: todas las secciones 3.2, 3.3, 4, 6, y 7 proporcionaron ejemplos de esto. Otras "aplicaciones filosóficas" de la teoría de la complejidad vienen del Principio Evolutivo y la Suposición de Dureza NP discutida en la Sección 10.2. *Si* creemos que ciertos problemas son computacionalmente intratables, entonces es factible poder deducir conclusiones interesantes de esa creencia sobre la racionalidad económica, la mecánica cuántica, la posibilidad de curvas temporales cerradas, y otros problemas. Por contraste, la complejidad computacional tiende a ser inútil cuando sólo queremos saber si un hecho particular "determina" otro hecho, y no nos importa la longitud de la cadena inferencial.

### 12.1 Críticas de la Teoría de la Complejidad

A pesar de su alcance explicativo, la teoría de la complejidad ha sido criticada en varios terrenos. Aquí están cuatro de las críticas más comunes:

(1) La teoría de la complejidad sólo hace declaraciones asintóticas (declaraciones acerca de cómo los recursos necesarios para resolver instancias de problemas de tamaño *n* escalan cuando *n* tiende a infinito). Pero como una materia de lógica, las declaraciones asintóticas en absoluto necesitan tener algunas implicaciones para los valores finitos de *n* (digamos, 10.000) acerca de los cuales los humanos realmente se preocupan, ni ninguna cantidad finita de datos experimentales puede confirmar o refutar una demanda asintótica.[78]

(2) Muchos de (lo que nos gustaría que fueran) los principios básicos de la teoría de la complejidad, como el $\mathsf{P} \neq \mathsf{NP}$, son conjeturas matemáticas actualmente no probadas, y probablemente permanecerán así durante mucho tiempo.

(3) La teoría de la complejidad se enfoca en un sólo tipo limitado de computadora – la serial, la máquina determinística de Turing – y no incorpora los fenómenos computacionales más "desarreglados" encontrados en la naturaleza.

(4) La teoría de la complejidad estudia sólo el comportamiento del peor-caso de los algoritmos, y no puntualiza si ese comportamiento es representativo, o si refleja unas pocas entradas "patológicas". Así por ejemplo, aun cuando $\mathsf{P} \neq \mathsf{NP}$, podría haber todavía heurísticas excelentes para resolver la mayoría de las instancias de problemas NP-completos que realmente surgen en la práctica; de un modo u otro, la teoría de la complejidad no nos dice nada sobre tales posibilidades.

---

77. La situación es muy diferente para las declaraciones matemáticas como la Hipótesis del Continuo que no *pueden* expresarse obviamente como predicciones acerca de los procesos computacionales idealizados (puesto que no son expresables por cuantificación sobre los enteros de primer orden o incluso de segundo orden). Para esas declaraciones, es realmente incierto para mí lo que uno significa por su verdad o falsedad aparte de su demostrabilidad en algún sistema formal.

78. Para un bonito ejemplo de límite inferior "concreto", no asintótico, en el tamaño de un circuito – ilustrando como la objeción (1) puede frecuentemente responderse con bastante trabajo duro – ver Stockmeyer y Meyer [128].

Por lo que pueda valer, las críticas (3) y (4) se han vuelto mucho menos exactas desde la década de 1980. Como se discutió en este ensayo, la teoría de la complejidad ha echado ramas ahora lejos más allá de máquinas de Turing determinísticas, para incorporar (por ejemplo) mecánica cuántica, computación paralela y distribuída, y procesos estocásticos como la evolución darwiniana. Entretanto, aunque la complejidad del peor caso sigue siendo la clase de complejidad mejor comprendida, hay un cuerpo grande de trabajo hoy – mucho del cual propulsado por la criptografía – que estudia la dureza de *casos promedio* de problemas computacionales, para varias distribuciones de probabilidad sobre las entradas. Y así como casi todos los teóricos de la complejidad creen que $\mathsf{P} \neq \mathsf{NP}$, así casi todos subscriben la creencia que existe problemas NP *duros en promedio* – de hecho, esa creencia es uno de los apuntalamientos de la criptografía moderna. Unos pocos problemas, como calcular logaritmos discretos, son incluso conocidos como siendo *tan duros en entradas aleatorias como lo son en las entradas más duras posibles* (aunque si se mantiene tal "equivalencia caso peor/caso promedio" para cualquier problema NP-completo permanece como pregunta mayor abierta). Por estas razones, aunque hablar sobre el caso promedio en lugar de la complejidad del peor caso complicaría algunos de los argumentos en este ensayo, no pienso que las conclusiones cambiarían mucho.[79] Ver Bogdanov y Trevisan [32] para un excelente estudio reciente de complejidad de caso promedio, e Impagliazzo [79] para una discusión evocadora de teoría de la complejidad de "mundos posibles" (por ejemplo, el "mundo" dónde los problemas NP-completos resultan ser duros en el peor caso pero fáciles en promedio).

El punto más amplio es que, aun cuando admitimos que las críticas (1)-(4) tienen mérito eso no nos da licencia para desechar los argumento de la complejidad teórica cada vez que nos disgusten. En ciencia, siempre sólo tratamos con teorías aproximadas, imperfectas – y si rechazamos las conclusiones de la mejor teoría aproximada en algún área, entonces la carga está en nosotros para explicar por qué.

Para ilustrar, supón que crees que las computadoras cuánticas nunca tomarán ventaja sobre las computadoras clásicas para cualquier problema práctico. Entonces como una explicación para tu posición, podrías afirmar algo de lo siguiente:
(a) La mecánica cuántica es falsa o incompleta, y un intento para construir una computadora cuántica escalable llevaría en cambio a falsificar o extender la mecánica cuántica misma.
(b) Existen algoritmos clásicos en tiempo polinomial para factorizar enteros, y para todos los otros problemas que admiten algoritmos cuánticos en tiempo polinomial. (En términos de complejidad, las clases BPP y BQP son iguales).
(c) Los "sobrecostos (*overheads*) de factor constante" involucrados construyendo una computadora cuántica son tan grandes como para negar sus ventajas asintóticas, para cualquier problema de interés humano concebible.
(d) Mientras no sabemos todavía cuáles de (a)-(c) se mantienen, podemos conocer sobre algún fundamento *a priori* que al menos uno de ellos tiene que mantenerse.

El punto es que, aun cuando no podemos responder a cada posible limitación del análisis de complejidad teórica, todavía podemos usarlo para *clarificar las elecciones*: forzar a las personas a poner algunas cartas en la mesa, comprometiéndolas a una predicción que podría falsearse o a una conjetura matemática que podría refutarse. Por supuesto, esta es una característica común de todas las teorías científicas, no algo específico para la teoría de la complejidad. Si la teoría de la complejidad es inusual aquí, es *solamente* con el número de

---

**79.** Por otro lado, se presupondría que sabíamos definir las distribuciones de probabilidad razonables sobre las entradas. Pero como se discutió en la Sección 4.3, parece duro explicar que queremos decir por "instancias estructuradas" o "los tipos de instancias que normalmente surgen en la práctica".

"predicciones" que hace malabares que podrían confirmarse o refutarse por demostración matemática (y de hecho, sólo por demostración matemática).[80]

### 12.2 Direcciones Futuras

Aun cuando las diversas críticas a la teoría de la complejidad no nieguen su relevancia, sería excelente apuntar a esas críticas – y más generalmente, para obtener una comprensión más clara de la relación entre la teoría de la complejidad y los fenómenos del mundo real que intenta explicar. Hacia ese fin, pienso que las preguntas siguientes podrían beneficiar a todos desde un análisis filosófico cuidadoso:

- ¿Cuál es el estado empírico de las demandas asintóticas? ¿Qué sentido podemos dar a una declaración asintótica que "hace predicciones", o a ser soportado o rechazado por un número finito de observaciones?
- Cómo podemos explicar los hechos empíricos en los que la teoría de la complejidad descansa: por ejemplo, ¿qué raramente vemos algoritmos $n^{10000}$ o $1{,}0000001^n$, o que los problemas computacionales acerca de los cuales los humanos se preocupan tienden a organizarse en un número relativamente pequeño de clases de equivalencia?
- Cortas de demostraciones, ¿cómo hacen las personas para formar intuiciones sobre la verdad o falsedad de conjeturas matemáticas? ¿Cuáles son esas intuiciones, en casos como $\mathsf{P} \neq \mathsf{NP}$?
- Si pensamos en clases de complejidad ($\mathsf{P}$, $\mathsf{NP}$, etc.) como colecciones de problemas, entonces se hace más difícil entender resultados como lo de Baker, Gill y Solovay [21] que dicen que existe un "oráculo" $A$ tal que $\mathsf{P}^A \neq \mathsf{NP}^A$ (y además, esto permanece verdadero aun cuando en el mundo "real" $\mathsf{P} = \mathsf{NP}$). En términos filosóficos, ¿debemos pensar en una clase de complejidad como correspondiendo a su *definición* en lugar de a su *extensión*?
- Las conclusiones conceptuales que las personas a veces quieren extraer de conjeturas tales como $\mathsf{P} \neq \mathsf{NP}$ o $\mathsf{BPP} \neq \mathsf{BQP}$ – por ejemplo, sobre la naturaleza de la creatividad matemática o la interpretación de la mecánica cuántica – ¿dependen realmente de que esas conjeturas sean verdaderas? ¿Es más fácil demostrar declaraciones que discutiblemente podrían soportar las mismas conclusiones?

Si $\mathsf{P} \neq \mathsf{NP}$, entonces ¿cómo se las arreglan los humanos para hacer tan enorme progreso matemático, incluso ante la intratabilidad general de la demostración de teoremas? ¿Hay un "efecto de selección", por el cual los matemáticos favorecen problemas con estructura especial que les haga más fácil resolverlos que problemas arbitrarios? En ese caso, ¿en qué consiste entonces esta estructura? Para abreviar, veo alcance suficiente para el ensayo inverso a éste: "Por qué los Teóricos de la Complejidad Computacional Debieran Atender a la Filosofía".

## 13 Reconocimientos

Agradezco a Oron Shagrir por empujarme a terminar este ensayo, por comentarios útiles, y por sugerirme la Sección 7.2; a Alex Byrne por sugerir la Sección 6; a Agustín Rayo por sugerir la Sección 5; y a David Aaronson, Eric Allender, Seamus Bradley, Alessandro Chiesa, Terrence Cole, Michael Collins, Andy Drucker, Ron Fagin, Michael Forbes, Oded Goldreich, Bob Harper, Gil Kalai, Dana Moshkovitz, Jan Arne Telle, Dylan Thurston, Ronald de Wolf, Avi Wigderson, y Joshua Zelinsky por su retroalimentación.

---

80. Otro ejemplo que salta a la mente, de una teoría científica muchas de cuyas "predicciones" toman la forma de conjeturas matemáticas, es la teoría de las cuerdas.

**Referencias**


[1] S. Aaronson. Shor, I'll do it (weblog entry). www.scottaaronson.com/blog/?p=208.

[2] S. Aaronson. Multilinear formulas and skepticism of quantum computing. In *Proc. ACM STOC*, pages 118–127, 2004. quant-ph/0311039.

[3] S. Aaronson. The complexity of agreement. In *Proc. ACM STOC*, pages 634–643, 2005. ECCC TR04-061.

[4] S. Aaronson. NP-complete problems and physical reality. *SIGACT News*, March 2005. quantph/0502072.

[5] S. Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proc. Roy. Soc. London*, A461(2063):3473–3482, 2005. quant-ph/0412187.

[6] S. Aaronson and G. Kuperberg. Quantum versus classical proofs and advice. *Theory of Computing*, 3(7):129–157, 2007. Previous version in Proceedings of CCC 2007. quant-ph/0604056.

[7] S. Aaronson and J. Watrous. Closed timelike curves make quantum and classical computing equivalent. *Proc. Roy. Soc. London*, (A465):631–647, 2009. arXiv:0808.2669.

[8] S. Aaronson and A. Wigderson. Algebrization: a new barrier in complexity theory. *ACM Trans. on Computarion Theory*, 1(1), 2009. Conference version in *Proc. ACM STOC* 2008.

[9] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. www.cse.iitk.ac.in/users/manindra/primality.ps, 2002.

[10] D. Aharonov. Quantum computation - a review. In Dietrich Stauffer, editor, *Annual Review of Computational Physics*, volume VI. 1998. quant-ph/9812037.

[11] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87-106, 1987.

[12] D. Angluin, J. Aspnes, J. Chen, and Y.Wu. Learning a circuit by injecting values. *J. Comput. Sys. Sci*., 75(1):60–77, 2009. Earlier version in STOC'2006.

[13] K. Appel and W. Haken. *Every planar map is four-colorable*. American Mathematical Society, 1989.

[14] B. Applebaum, B. Barak, and D. Xiao. On basing lower-bounds for learning on worst-case assumptions. *In Proc. IEEE FOCS*, pages 211–220, 2008.

[15] S. Arora and B. Barak. *Complexity Theory: A Modern Approach*. Cambridge University Press, 2009. Online draft at ww.cs.princeton.edu/theory/complexity/.

[16] S. Arora, R. Impagliazzo, and U. Vazirani. Relativizing versus nonrelativizing techniques: the role of local checkability. Manuscript, 1992.

[17] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. J. ACM, 45(3):501–555, 1998.

[18] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[19] R. J. Aumann. Agreeing to disagree. *Annals of Statistics*, 4(6):1236–1239, 1976.

[20] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[21] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. SIAM *J. Comput*., 4:431–442, 1975.

[22] E. B. Baum. *What Is Thought?* Bradford Books, 2004.

[23] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. Earlier version in IEEE FOCS 1998, pp. 352-361. quant-ph/9802049.

[24] P. Beame and T. Pitassi. Propositional proof complexity: past, present, and

future. *Current Trends in Theoretical Computer Science*, pages 42–70, 2001.

[25] S. Bellantoni and S. A. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992. Earlier version in STOC 1992, p. 283-293.

[26] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: how to remove the intractability assumptions. In *Proc. ACM STOC*, pages 113–131, 1988.

[27] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997. quant-ph/9701001.

[28] C. H. Bennett, D. Leung, G. Smith, and J. A. Smolin. Can closed timelike curves or nonlinear quantum mechanics improve quantum state discrimination or help solve hard problems? *Phys. Rev. Lett.*, 103(170502), 2009. arXiv:0908.3023.

[29] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997. First appeared in ACM STOC 1993.

[30] N. Block. Searle's arguments against cognitive science. In J. Preston and M. Bishop, editors, *Views into the Chinese Room: New Essays on Searle and Artificial Intelligence*, pages 70–79. Oxford, 2002.

[31] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik- Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

[32] A. Bogdanov and L. Trevisan. Average-case complexity. Foundations and *Trends in Theoretical Computer Science*, 2(1), 2006. ECCC TR06-073.

[33] R. B. Boppana, J. Håstad, and S. Zachos. Does co-NP have short interactive proofs? *Inform. Proc. Lett.*, 25:127–132, 1987.

[34] R. Bousso. Positive vacuum energy and the N-bound. *J. High Energy Phys.*, 0011(038), 2000. hep-th/0010252.

[35] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput*. Sys. Sci., 37(2):156–189, 1988.

[36] T. Brun. Computers with closed timelike curves can solve hard problems. *Foundations of Physics Letters*, 16:245–253, 2003. gr-qc/0209061.

[37] D. J. Chalmers. *The Conscious Mind: In Search of a Fundamental Theory*. Oxford, 1996.

[38] X. Chen, D. Dai, Y. Du, and S.-H. Teng. Settling the complexity of Arrow-Debreu equilibria in markets with additively separable utilities. In *Proc. IEEE FOCS*, pages 273–282, 2009.

[39] X. Chen and X. Deng. Settling the complexity of two-player Nash equilibrium. In Proc. *IEEE FOCS*, pages 261–271, 2006.

[40] C. Cherniak. Computational complexity and the universal acceptance of logic. *The Journal of Philosophy*, 81(12):739–758, 1984.

[41] R. Cleve, P. Høyer, B. Toner, and J. Watrous. Consequences and limits of nonlocal strategies. In *Proc. IEEE Conference on Computational Complexity*, pages 236–249, 2004. quantph/0404076.

[42] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of Logic, Methodology, and Philosophy of Science* II. North Holland, 1965.

[43] J. Copeland. Hypercomputation. *Minds and Machines*, 12:461–502, 2002.

[44] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibrium. *Commun. ACM*, 52(2):89–97, 2009. Earlier version in Proceedings of STOC'2006.

[45] R. Dawkins. *The God Delusion*. Houghton Mifflin Harcourt, 2006.

[46] D. C. Dennett. *Darwin's Dangerous Idea: Evolution and the Meanings of Life*. Simon & Schuster, 1995.

[47] D. Deutsch. Quantum mechanics near closed timelike lines. *Phys. Rev. D*, 44:3197–3217, 1991.
[48] D. Deutsch. *The Fabric of Reality*. Penguin, 1998.
[49] D. Deutsch. *The Beginning of Infinity: Explanations that Transform the World*. Allen Lane, 2011.
[50] I. Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.
[51] A. Drucker. Multiplying 10-digit numbers using Flickr: the power of recognition memory. people.csail.mit.edu/andyd/rec_method.pdf, 2011.
[52] R. Fagin. Finite model theory - a personal perspective. *Theoretical Comput. Sci.*, 116:3–31, 1993.
[53] R. Fagin and J. Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34(1):39–76, 1987.
[54] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge.* The MIT Press, 1995.
[55] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996.
[56] R. P. Feynman. Simulating physics with computers. *Int. J. Theoretical Physics*, 21(6-7):467–488, 1982.
[57] L. Fortnow. The role of relativization in complexity theory. *Bulletin of the EATCS*, 52:229–244, February 1994.
[58] L. Fortnow. One complexity theorist's view of quantum computing. *Theoretical Comput. Sci.*, 292(3):597–610, 2003.
[59] L. Fortnow and S. Homer. A short history of computational complexity. *Bulletin of the EATCS*, (80):95–133, 2003.
[60] A. Fraenkel and D. Lichtenstein. Computing a perfect strategy for nxn chess requires time exponential in n. *Journal of Combinatorial Theory A*, 31:199–214, 1981.
[61] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proc. ACM STOC*, pages169–178, 2009.
[62] O. Goldreich. On quantum computing. www.wisdom.weizmann.ac.il/~oded/on-qc.html, 2004.
[63] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. Earlier version at www.wisdom.weizmann.ac.il/~oded/cc-drafts.html.
[64] O. Goldreich. *A Primer on Pseudorandom Generators*. American Mathematical Society, 2010. www.wisdom.weizmann.ac.il/~oded/PDF/prg10.pdf.
[65] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1984.
[66] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(1):691–729, 1991.
[67] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186208, 1989.
[68] N. Goodman. *Fact, Fiction, and Forecast*. Harvard University Press, 1955.
[69] P. Graham. How to do philosophy. www.paulgraham.com/philosophy.html, 2007.
[70] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. ACM STOC*, pages 212–219, 1996. quant-ph/9605043.
[71] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.
[72] J. Haugeland. Syntax, semantics, physics. In J. Preston and M. Bishop, editors,

*Views into the Chinese Room: New Essays on Searle and Artificial Intelligence*, pages 379–392. Oxford, 2002.

[73] J. Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.

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

[75] A. S. Holevo. Some estimates of the information transmitted by quantum communication channels. *Problems of Information Transmission*, 9:177–183, 1973. English translation.

[76] G. 't Hooft. Quantum gravity as a dissipative deterministic system. *Classical and Quantum Gravity*, 16:3263–3279, 1999. gr-qc/9903084.

[77] D. Hume. *An Enquiry concerning Human Understanding*. 1748. 18th.eserver.org/humeenquiry. html.

[78] N. Immerman. *Descriptive Complexity*. Springer, 1998.

[79] R. Impagliazzo. A personal view of average-case complexity. In *Proc. IEEE Conference on Computational Complexity*, pages 134–147, 1995.

[80] R. Impagliazzo and A. Wigderson. P=BPP unless E has subexponential circuits: derandomizing the XOR Lemma. In *Proc. ACM STOC*, pages 220–229, 1997.

[81] D. Kahneman, P. Slovic, and A. Tversky. *Judgment Under Uncertainty: Heuristics and Biases*. Cambridge University Press, 1982.

[82] M. J. Kearns and L. G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994.

[83] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[84] J. Kempe, H. Kobayashi, K. Matsumoto, B. Toner, and T. Vidick. Entangled games are hard to approximate. *SIAM J. Comput*., 40(3):848–877, 2011. Earlier version in FOCS'2008. arXiv:0704.2903.

[85] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput*., 31:1501–1526, 2002. Earlier version in ACM STOC 1999.

[86] R. E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22:155–171, 1975.

[87] D. Leivant. A foundational delineation of poly-time. *Information and Computation*, 110(2):391–420, 1994. Earlier version in LICS (Logic In Computer Science) 1991, p. 2-11.

[88] H. J. Levesque. Is it enough to get the behavior right? In *Proceedings of IJCAI*, pages1439–1444, 2009.

[89] L. A. Levin. Polynomial time and extravagant models, in The tale of one-way functions. *Problems of Information Transmission*, 39(1):92–103, 2003. cs.CR/0012023.

[90] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications (3rd ed.)*. Springer, 2008.

[91] S. Lloyd, L. Maccone, R. Garcia-Patron, V. Giovannetti, and Y. Shikano. The quantum mechanics of time travel through post-selected teleportation. *Phys. Rev. D*, 84(025007), 2011. arXiv:1007.2615.

[92] J. R. Lucas. Minds, machines, and Gödel. *Philosophy*, 36:112–127, 1961.

[93] N. D. Mermin. From cbits to qbits: teaching computer scientists quantum mechanics. *American J. Phys*., 71(1):23–30, 2003. quant-ph/0207118.

[94] N. D. Mermin. *Quantum Computer Science: An Introduction*. Cambridge University Press, 2007.

[95] S. Micali. Computationally sound proofs. *SIAM J. Comput*., 30(4):1253–1298, 2000.

[96] C. Moore and S. Mertens. *The Nature of Computation*. Oxford University Press, 2011.
[97] M. S. Morris, K. S. Thorne, and U. Yurtsever. Wormholes, time machines, and the weak energy condition. *Phys. Rev. Lett.*, 61:1446–1449, 1988.
[98] A. Morton. Epistemic virtues, metavirtues, and computational complexity. *Noûs*, 38(3):481–502, 2004.
[99] A. Neyman. Bounded complexity justifies cooperation in the finitely repeated prisoners' dilemma. *Economics Letters*, 19(3):227–229, 1985.
[100] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
[101] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
[102] I. Parberry. Knowledge, understanding, and computational complexity. In D. S. Levine and W. R. Elsberry, editors, *Optimality in Biological and Artificial Networks?*, pages 125–144. Lawrence Erlbaum Associates, 1997.
[103] R. Penrose. *The Emperor's New Mind*. Oxford, 1989.
[104] R. Penrose. *Shadows of the Mind: A Search for the Missing Science of Consciousness.* Oxford, 1996.
[105] L. Pitt and L. Valiant. Computational limitations on learning from examples. *J. ACM*, 35(4):965–984, 1988.
[106] C. Pomerance. A tale of two sieves. *Notices of the American Mathematical Society*, 43(12):1473–1485, 1996.
[107] H. Putnam. *Representation and Reality*. Bradford Books, 1991.
[108] Y. Rabinovich and A. Wigderson. Techniques for bounding the convergence rate of genetic algorithms. *Random Structures and Algorithms*, 14(2):111–138, 1999.
[109] R. Raz. Exponential separation of quantum and classical communication complexity. In *Proc. ACM STOC*, pages 358–367, 1999.
[110] A. A. Razborov and S. Rudich. Natural proofs. *J. Comput. Sys. Sci*., 55(1):24–35, 1997.
[111] S. Reisch. Hex is PSPACE-complete. *Acta Informatica*, 15:167–191, 1981.
[112] H. E. Rose. *Subrecursion: Functions and Hierarchies*. Clarendon Press, 1984.
[113] A. Rubinstein. *Modeling Bounded Rationality*. MIT Press, 1998.
[114] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, (7):281–292, 1971.
[115] J. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(417-457), 1980.
[116] J. Searle. *The Rediscovery of the Mind*. MIT Press, 1992.
[117] A. Shamir. IP=PSPACE. *J. ACM*, 39(4):869–877, 1992.
[118] S. M. Shieber. The Turing test as interactive proof. No^us, 41(4):686–713, 2007.
[119] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput*., 26(5):1484–1509, 1997. Earlier version in IEEE FOCS 1994. quant-ph/9508027.
120] [H. T. Siegelmann. Neural and super-Turing computing. Minds and Machines, 13(1):103–114, 2003.
[121] D. Simon. On the power of quantum computation. In Proc. *IEEE FOCS*, pages 116–123, 1994.
[122] H. A. Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118, 1955.
[123] M. Sipser. The history and status of the P versus NP question. In *Proc. ACM STOC*, pages603–618, 1992.
[124] M. Sipser. *Introduction to the Theory of Computation (Second Edition)*. Course Technology, 2005.

[125] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. SIAM J. Comput., 28(4):1364–1396.

[126] R. Stalnaker. The problem of logical omniscience, I and II. In *Context and Content: Essays on Intentionality in Speech and Thought*, Oxford Cognitive Science Series, pages 241–273. Oxford University Press, 1999.

[127] L. J. Stockmeyer. Classifying the computational complexity of problems. *J. Symbolic Logic*, 52(1):1–43, 1987.

[128] L. J. Stockmeyer and A. R. Meyer. Cosmological lower bound on the circuit complexity of a small problem in logic. *J. ACM*, 49(6):753–784, 2002.

[129] J. A. Storer. On the complexity of chess. *J. Comput. Sys. Sci*., 27(1):77–100, 1983.

[130] A. M. Turing. *Computing machinery and intelligence*. Mind, 59:433–460, 1950.

[131] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[132] L. G. Valiant. Evolvability. *J. ACM*, 56(1), 2009. Conference version in MFCS 2007. ECCC TR06-120.

[133] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[134] H. Wang. *A Logical Journey: From Gödel to Philosophy*. MIT Press, 1997.

[135] J. Watrous. Succinct quantum proofs for properties of finite groups. In *Proc. IEEE FOCS*, pages 537–546, 2000. cs.CC/0009002.

[136] J. Watrous. Quantum computational complexity. In *Encyclopedia of Complexity and Systems Science*. Springer, 2008. arXiv:0804.3401.

[137] A. Wigderson. P, NP and mathematics - a computational complexity perspective. In *Proceedings of the International Congress of Mathematicians2006 (Madrid)*, pages 665–712. EMS Publishing House, 2007. www.math.ias.edu/~avi/PUBLICATIONS/MYPAPERS/W06/w06.pdf.

[138] A. Wigderson. Knowledge, creativity and P versus NP, 2009. www.math.ias.edu/~avi/PUBLICATIONS/MYPAPERS/AW09/AW09.pdf.

[139] E. Wigner. The unreasonable effectiveness of mathematics in the natural sciences. *Communications in Pure and Applied Mathematics*, 13(1), 1960.

[140] R. de Wolf. Philosophical applications of computational learning theory: Chomsky an innateness and occam's razor. Master's thesis, Erasmus University, 1997. homepages. cwi.nl/~rdewolf/publ/philosophy/phthesis.pdf.