HIPERCOMPUTACIÓN. UNA ENTREVISTA CON ANDRÉS SICARD (VERSIÓN PRELIMINAR)

SERGIO PINEDA

1. Introducción

E.W. Dijkstra afirmó que "la ciencia de la computación trata sobre computadores tanto como la astronomía trata de telescopios" [1]. Ello se me hizo claro leyendo artículos de Andrés Sicard: máquina de Turing cuántica autorreferencial: ¿una posibilidad? [2], prototipo de un modelo de computación cuántica continua [3], computación inconsistente [**?**], etc.

Decidí contactarlo. Aceptó concederme una entrevista. Para ese entonces, se encontraba en Nueva Zelandia realizando una investigación durante su periodo sabático. Había ido a trabajar con quien acuñara el término "Hipercomputación" en 1999 (Jack Copeland, filósofo). Académico de la Universidad EAFIT, director del grupo de investigación "Lógica y Computación", Sicard lleva varios años realizando investigaciones en computación teórica, con una amplia gama de temas, desde lógicas paraconsistentes hasta computación cuántica. En un correo electrónico me dijo que en vez de una entrevista, escribiéramos un artículo juntos. Pasaron algunos meses, volvió, y nos encontramos en Medellín por pura casualidad. Luego de algunas charlas, el esquema de co-autoría perdió toda validez: yo era el de todas las preguntas, y él era el de todas las respuestas. Decidimos volver al esquema de entrevista.

En principio, trataríamos el tema de la hipercomputación.

> "Un hipercomputador es un dispositivo que computa todo lo que computa una máquina de Turing, y más. La hipercomputación es la disciplina que estudia los hipercomputadores" [4, p. 4].

Date: Junio-2001.

Quisiera agradecer a Andrés Sicard por su enorme generosidad con el conocimiento (generosidad que da forma final a esta entrevista).

Para entender la hipercomputación, pues, hay que entender qué es lo que computa una máquina de Turing.

2. Lo Turing-computable

David Hilbert, en el Congreso Internacional de Matemáticos de 1900, señaló 23 problemas que, a su parecer, marcarían el desarrollo de las matemáticas del siglo XX. El décimo de estos problemas era una pre-enunciación de lo que el mismo Hilbert, en 1928, denominaría el *Entscheidungsproblem*, o problema de la decisión. Éste consistía en encontrar un método efectivo general para determinar si una fórmula era o no verdadera en un sistema formal dado [5, p. 30].

Kurt Gödel dio un primer paso en la solución, al demostrar en 1931, que dentro de todo sistema formal con capacidad para expresar la aritmética puede construirse un enunciado indecidible. Es decir, en los sistemas formales con cierto poder expresivo es posible construir un enunciado gramaticalmente correcto del cual no pueda afirmarse que es verdadero, y tampoco pueda afirmarse que es falso. Gödel demostró que no existe un método efectivo universal para determinar si una fórmula es verdadera o no, pues hay fórmulas indecidibles de las cuales no puede afirmarse lo uno ni lo contrario [6]. Claro, quedaba sin resolver el problema para el conjunto de enunciados demostrables.

Alan Mathison Turing, muy pocos años después (1936), dio forma más precisa al concepto de "procedimiento efectivo de decisión", acuñando la noción de "computabilidad". Turing concibió un proceso mecánico, que se conoce actualmente como la máquina de Turing (MT); replanteó el *Entscheidungsproblem* en términos de sus máquinas, y demostró que no tenía solución, ni siquiera para enunciados demostrables [7].

Turing tenía la intención de describir una máquina que fuera capaz de llevar a cabo cualquier tarea que un matemático, con disposición ilimitada de tiempo, energía, lápices y papel, y sin ningún tipo de introversión o creatividad, pudiera llevar a cabo. Es decir, una máquina que pudiera realizar cualquier conjunto de instrucciones. Este modelo fue la base teórica para la construcción de los computadores que hoy día conocemos. Todo lo computable por cualquier computador actual, es Turing-computable, en el sentido de que una máquina como la descrita por Turing en 1936 lo habría podido

realizar. Claro: más lentamente.

La Hipercomputación nació, pues, como el estudio de ese borde entre lo computable y lo no computable. Su intención principal es encontrar nuevos modelos que puedan computar más allá de lo Turing-computable.

3. Entrevista

El formato de la entrevista será el siguiente: en los numerales (3.1) a (3.6) se abordarán temas diferentes, cada tema será introducido por un **comentario inicial** realizado por parte del entrevistador y después se continuará con una sección de **pregunta(s)** y **respuesta(s)**.

3.1. **Definiciones etimológicas.**
Comentario inicial:

www.dictionary.com
hyper-
pref.
1. Over; above; beyond: hypercharge.
2. Excessive; excessively: hypercritical.

Diccionario de la Lengua Española, Real Academia Española:
hiper-. elem. compos. que significa "exceso".

Pregunta: Si tomamos sus raíces etimológicas, hipercomputación significa "exceso de computación". ¿No parece excesivo ese nombre? ¿No sería más preciso el nombre "metacomputación" (más allá de la computación)?

Respuesta: Antes de presentar algunas observaciones acerca del uso del prefijo 'hiper', deseo introducir una definición del término 'hipercomputador' un poco diferente a la que fue presentada en la sección anterior:

> *"A hypercomputer is any information-processing device able to carry out tasks that cannot be carried out by a standard universal Turing machine"* [8].

Aunque la diferencia entre ambas definiciones es sútil, esta nueva definición admite la posibilidad de que un hipercomputador no compute todo

lo Turing-computable, pero exige que compute por lo menos un objeto —función, número, proceso, etc.— no Turing-computable. Debido a la posibilidad de existencia de modelos de hipercomputación con esta característica, me parece más adecuada la nueva definición presentada.

En relación con el uso del prefijo 'hiper' en el término 'hipercomputación', mi opinión es que se puede presentar una discusión etimológica similar a discusiones anteriores cuando se emplea un neologismo, es decir, quizas prefijos tales como 'para', 'meta', 'super' o 'ultra', puedan ser más adecuados. Sin embargo, desde mi punto de vista, lo importante en este contexto es que desde la acuñacion del término en 1999 por el profesor Copeland [9], contamos con un término adecuado —quizás no el más adecuado— para una área del saber que anteriormente era designada erróneamente.

Designadores tales como *'super-Turing computation, 'computing beyond the Turing limit, 'breaking the Turing barrier, etc.* eran comúnmente utilizados para referirse a modelos de hipercomputación, sin embargo está designación es errónea por dos razones. En primer lugar, los designadores anteriores reflejan un desconomiento del papel jugado por Turing en la creación de modelos de hipercomputación, de hecho, fue él quien propuso unos de los primeros modelos de hipercomputación denominado *oracle machines* [10]. En segundo lugar, estos designadores reflejan además un mala interpretación de la tesis de Church-Turing, esta mala interpretación es llamada por Copeland *"the Church-Turing fallacy"* [11, p. 133] y ha sido señalada por él en varias ocasiones [12, 13, 14].

Es posible pensar que la imposibilidad de hipercomputación está sustentada en la tesis de Church-Turing. Usualmente está es entendida como la tesis que afirma que cualquier tarea efectiva de manipulación de información realizada por una máquina puede ser realizada por una máquina de Turing, sin embargo, lo que afirma la tesis de Church-Turing es lo siguiente:

> *"Tesis Church-Turing: Any procedure than can be carried out by an idealised human clerk working mechanically with paper and pencil can also be carried out by a Turing machine"* [14, p. 50].

Existe entonces una diferencia entre la tesis cuyos objetos son las máquinas y la tesis de Church-Turing cuyos objetos son los *computors*, es decir, *"idealized human calculating in a purely mechanical fashions"* [15, p. 292]. La primera tesis es más amplia y fue presentada inicialmente por Robin Gandy

—quien fue alumno de Turing— bajo el nombre de tesis M [16]. Esta tesis es presentada como sigue:

> *"Tesis M: Whatever can be calculated by a machine (working on finite data in accordance with a finite program of instructions) is Turing machine computable"* [12, p. 5].

Como es señalado por Copeland, la tesis M admite dos posibles interpretaciones:

> *"Thesis M itself admits of two interpretations, according to whether the phrase 'can be calculated by a machine' is taken in the narrow sense of 'can be calculated by a machine that conforms to the physical laws (if not to the resource constrains) of the actual world', or in a wide sense that abstracts from the issue of whether or not the notional machine in question could exist in the actual world. The narrow version of the thesis M is an empirical proposition whose truth-value is unknown. The wide version of the thesis M is known to be false. Various notional machines have been described which can calculate functions that are not Turing machine computable ... "* [12, p. 6].

Entonces, los modelos de computación usados para demostrar que la versión amplia de la tesis M es falsa son los modelos que hoy denominamos "hipermáquinas" o "hipercomputadores".

3.2. **MT on oráculo.**

Comentario inicial: Uno de los primeros ejercicios de hipercomputación fue la máquina de Turing con oráculo (MTO), propuesta por el mismo Turing en 1938 para sus tesis de doctorado, seis décadas antes de que el término hipercomputación fuera acuñado.

or·a·cle (ôr-kl, r-)
a.A shrine consecrated to the worship and consultation of a prophetic deity, as that of Apollo at Delphi.
b.A person, such as a priestess, through whom a deity is held to respond when consulted.
c.The response given through such a medium, often in the form of an enigmatic statement or allegory.
(www.dictionary.com)

Los sinónimos de oráculo son: (4)
** adivinación*
** predicción*
** profecía*
** vaticinio*
(http://tradu.scig.uniovi.es/sinon.cgi)

Pregunta: ¿Qué representa el oráculo planteado por Turing, y porqué produce una nueva computabilidad?

Respuesta: El oráculo puede ser intuido como un conjunto O y un procedimiento de decisión sobre su elementos, de forma tal que la máquina de Turing puede preguntar al oráculo si un determinado elemento sobre su cinta pertenece o no al conjunto O, y con base en la respuesta, la máquina realiza una posible acción. Sin presentar una definición formal de una máquina de Turing determinista, podemos pensar en un programa para dicha máquina como un conjunto finito de instrucciones $I = \{i_0, i_1, i_2, \ldots, i_n\}$, donde cada i_j es una quíntupla de la forma: $q_a\ s_b\ s_c\ m\ q_d$, tal que ningún par de instrucciones contiene sus dos primeros elementos iguales —esta condición dota a la máquina de su cáracter determinista—. Los elementos q_a, s_b, s_c, m, q_d son interpretados como: q_a es el estado actual, s_b es el símbolo actual, s_c es el símbolo a escribir, m es el movimiento a realizar y q_d es el estado siguiente.

Para construir una máquina de Turing determinista con un oráculo O es necesario modificar el formato de instrucciones por: $q_a\ s_b\ s_c\ m\ \{q_{d_1}, q_{d_2}\}$, donde el estado siguiente está determinado por la pertencencia del símbolo s_b al conjunto O, es decir, el estado siguiente es q_{d_1} si $s_b \in O$, de lo contrario el estado siguiente es q_{d_2} [17].

Sea $\mathcal{MT}_O$ una máquina de Turing con oráculo O. La característica de hipercomputación en la máquina $\mathcal{MT}_O$ es establecida por el oráculo O. Si el oráculo O es un conjunto recursivo (decidible) —su función característica es Turing-computable—; la máquina $\mathcal{MT}_O$ no es un modelo de hipercomputación. Por el contrario, si el oráculo O es un conjunto no recursivo, la máquina $\mathcal{MT}_O$ es un modelo de hipercomputación.

Pregunta: Es posible argumentar que el mismo nombre "oráculo" implica conciencia de su no implementabilidad, pues lo profético y lo adivinatorio son antónimos de lo programable (i.e., lo computable). ¿No estaba Turing planteando la no implementabilidad del modelo MTO al bautizarlo de tal

manera?

Respuesta: Inicialmente es necesario mencionar que tanto para las MTs como para las MTOs no se realizó ninguna consideración con respecto a su implementación [7, 10]. De hecho, con respecto a la naturaleza del óraculo Turing sólo se limitó a mencioar que éste trabaja por *"unspecified means"* y que *"we shall not go any further into the nature of this oracle"* [10, p. 172-3]. Si bien es cierto que en la actualidad existen implementaciones para las MTs (con las restricciones finitistas necesarias), la situación es diferente para las MTOs.

Sea τ un número real definido por $\tau = 0.a_1a_2a_3\ldots$ donde $a_i \in \{0,1\}$ tal que, $a_i = 1$ si la i-ésima máquina de Turing (bajo alguna codificación) se detiene y $a_i = 0$ si la i-ésima máquina de Turing no se detiene —este número es un poco diferente al número Ω de Chaitin [18, p. 11]—. El número τ así definido es un número real no computable [19] y puede constituirse en un orácuo para una $\mathcal{MT}_\tau$ en donde la pregunta al oráculo se haría en términos del valor de la i-ésima cifra decimal τ. La MT_τ así definida es un modelo de hipercomputación. Una alternativa de implementar el número τ sería contar con alguna constante de la naturaleza que lo representara. No obstante, en la actualidad es desconocida cualquier constante con estas características, más aún, la posibilidad de "leer" algún número real no computable a partir de alguna situación física, es considerada como un problema abierto [20, p. 3].

3.3. **Los límites de lo computable.**

Comentario inicial: Los límites del universo de lo computable viven tocándose. El universo de lo computable (entendido como lo Turing-computable) es cada vez más estrecho.

La matemática diagnosticó el límite desde principios del siglo XX (Hilbert pre-enunció el *Entscheidungsproblem* en 1900).

La ciencia de la computación se encuentra con el límite, al plantear tareas que las MTs no pueden llevar a cabo. Una de tales tareas es el *"Terminating program task"* conocida como *"TP task"*. Tal límite consiste en que ningún programa de propósito general (en Pascal, BASIC, o C, por ejemplo) puede contener un *'crash debugger'*, o un detector de todos los errores de código que puedan conducir a un colapso del sistema, incluyendo errores que conduzcan a procesos de retro-alimentación infinitos [21].

La posibilidad de una inteligencia artificial es fuertemente criticada por algunos autores basados en que la inteligencia está más allá del límite de lo Turing-computable, puesto que las máquinas de Turing no son capaces de computar procesos autorrefenciales [22].

Pregunta: ¿Considera que son estrechos los límites de la computación? ¿Considera que la hipercomputación es pertinente (o ofrece respuestas pertinentes) a la estrechez de los límites de la computación?

Respuesta: En particular creo que los límites de la Turing-computación no son estrechos, puesto que el espectro de los objetos Turing-computables es de gran amplitud. Si pensamos por ejemplo en los números Turing-computables, nos encontramos conque todos los números racionales y un conjunto infinito enumerable de números irracionales —entre ellos π, e y $\sqrt{2}$— son números Turing-computables. No obtante lo anterior, si creo que los límites de la Turing-computación se pueden ampliar. De acuerdo a la definición de hipercomputación que estamos considerando, es posible contar con una hipermáquina que presente las características señaladas por la figura (1).

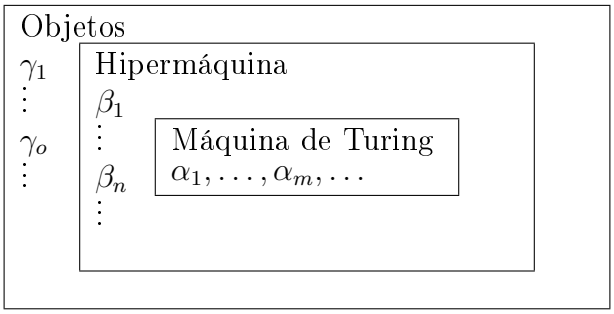

FIGURA 1. Objetos hipercomputables y Turing-computables.

La pregunta por los límites de la computación —desde una perspectiva general—, es la pregunta por los límites del conocimiento científico. Quizás un objeto hipercomputable pero no Turing-computable tal como el objeto β_1, sea simplemente un objeto formal con algunas características puntuales, pero existe también la posibilidad de que este objeto tenga la suficiente "potencia" que nos permita ampliar nuestras capacidades formales de razonamiento. Si bien es cierto que nuestros sistemas formales presentan limitaciones internas [23] —tal como el teorema de incompletitud Gödel—; no veo razones suficientes para la imposibilidad de sobreponernos de alguna forma a estas

limitaciones.

Una alternativa viable es cuestionar la lógica subyacente en la que fueron establecidos los resultados limitantes. Desde el punto de vista lógico, una teoría Σ se define como un subconjunto de fórmulas bien formadas (FBFs) —llamadas teoremas— obtenidas a partir de la aplicación de reglas de inferencia a un conjunto inicial de axiomas o postulados de la teoría. Una teoría Σ es trivial si el conjunto de sus teoremas es el conjunto de FBFs. Un teoría Σ es no trivial, si al menos una FBF no es un teorema de ella. Una teoría Σ es inconsistente si ella contiene al menos dos teoremas de la forma α y $\neg\alpha$, uno de los cuales es la negación del otro. Un teoría Σ es consistente si no es inconsistente. Si la lógica subyancente a una teoría Σ es la lógica clásica, la distinción entre teoría trivial y teoría inconsitente desaparece. Si por el contrario, la lógica subyancente es una lógica paraconsistente, la distinción entre teoría trivial y teoría inconsistente permanece. Se define entonces una lógica paraconsistente como una lógica subyacente a teorías inconsistentes pero no triviales [24, 25, 26].

En las teorías inconsistentes construidas sobre lógicas paraconsistentes, algunos resultados de limitación no aplican —en particular el teorema de incompletitud de Gödel— dado que estos resultados exigen por hipótesis que la teoría en cuestión sea consistente.

Aunque en la actualidad existen algunas propuestas de teorías inconsistentes para la teoría de conjuntos [27], el cálculo diferencial [28], y la aritmética [29, 30], entre otras. No conozco de ninguna propuesta de computación paraconsistente excepto por un artículo en donde se señalan algunos posibles caminos a seguir [31]. Una propuesta de computación paraconsistente podría ser un modelo de hipercomputación como el señalado por la figura (1) con la adición de una contradicción de la forma θ y $\neg\theta$ al conjunto de objetos hipercomputables.

Quizás sea el momento adecuado de mencionar que la modificación a la definición de hipercomputación realizada, está sustentada en la posibilidades que ofrecen las lógicas paraconsistentes en el contexto de la hipercomputación. En la figura (1), los objetos $\alpha_1, \ldots, \alpha_m, \ldots$ son hipercomputables y Turing-computables, los objetos $\beta_1, \ldots, \beta_n, \ldots$ son hipercomputables pero no Turing-computables y los objetos $\gamma_1, \ldots, \gamma_o, \ldots$ son ni hipercomputables ni Turing-computables. Un modelo de hipercomputación paraconsistente, en

lugar de tener las características representadas por la figura (1), puede tener las caracaterísticas representadas por la figura (2).

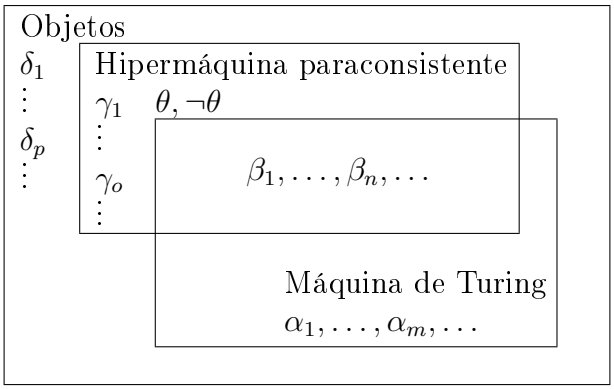

FIGURA 2. Posible modelo de hipercomputación paraconsistente.

De acuerdo a la figura (2) existirían objetos que son ni Turing-computable ni hipercomputables $(\delta_1, \ldots, \delta_p, \ldots)$, objetos hipercomputables pero no Turing-computables $(\gamma_1, \ldots, \gamma_o, \ldots)$, objetos Turing-computables pero no hipercomputables $(\alpha_1, \ldots, \alpha_m, \ldots)$ y objetos hipercomputables y Turing-computables $(\beta_1, \ldots, \beta_n, \ldots)$. Todos los modelos de hipercomputación que conozco son de la clase representada por la figura (1), sin embargo, la existencia de teorías inconsistentes de la clase representada por la figura (2), me permite considerar la posible existencia de un modelo de hipercomputación paraconsistente con estas características.

3.4. **Las constantes de Planck y las MT aceleradas.**

Comentario inicial: Una máquina de Turing acelerada (MTA) se define como una MT que realiza cada instrucción en un tiempo menor que el que tomó realizando la instrucción anterior. Esto implica que, si por ejemplo cada instrucción se realiza en la mitad del tiempo de la instrucción anterior y si la primera instrucción se realiza en 1 segundo, la segunda se realiza en medio segundo y la instrucción 101 se demora 2^{-100} segundos en ejecutarse. Es decir, la MTA descrita podría ejecutar cualquier conjunto de instrucciones (inclusive conjuntos infinitos de instrucciones) en un tiempo máximo de 2 segundos.

La medida más pequeña de longitud con un significado físico es la longitud de Planck (c.a. $4 * 10^{-35}$m, aproximadamente 10^{-20} veces el tamaño del protón). La medida más pequeña de tiempo con un significado físico, es el

tiempo de Planck, el tiempo que se necesita para atravesar la longitud de Planck a la velocidad de la luz (ca. $1{,}4 * 10^{-}43$ s). El paso 101 de una máquina de Turing acelerada descrita, toma 2^{-100} segundos, que es menor que 10^{-43} segundos. Operaciones de esta velocidad no tienen ningún significado desde el punto de vista físico.

Pregunta: Las operaciones del tipo de una MTA son operaciones con un sentido absoluto de irrealidad. Su argumentación carece de sentido en términos físicos. ¿Qué interés tienen las MTAs por fuera del mero juego lógico?

Respuesta: Comienzo mi respuesta señalando una pequeña imprecisión en la definición de una MTA. No solamente es necesario que la siguiente instrucción se ejecute un tiempo menor que la anterior, sino que además, la serie que representa la suma del tiempo de ejecución sea convergente. Por ejemplo, supongamos una máquina de Turing $\mathcal{MT}$ cuya instrucción n-ésima se ejecute en $\frac{1}{n}$ segundos; puesto que $\frac{1}{n} > \frac{1}{m}$ para $n, m \in \mathbb{Z}^+$ y $n < m$, está maquina ejecuta la siguiente instrucción en un tiempo menor que la instrucción anterior. Sin embargo, la máquina $\mathcal{MT}$ no debe ser considerada una MTA puesto que la serie $\sum_{i=1}^{\infty} \frac{1}{n}$ no es convergente, lo que significa que la máquina $\mathcal{MT}$ necesitaría un tiempo infinito para ejecutar un número infinito de instrucciones. En general entonces, podemos definir una MTA como una MT donde la instrucción siguiente se ejecuta en la mitad del tiempo de la instrucción anterior [32, 33, 34].

Respecto a la posible existencia de un referente físico para las MTAs aceleradas —pues considero que la pregunta señala esta dirección— pienso que la legitimidad de un modelo formal no está sustentada en la (posible) existencia de su referente físico. Si bien es cierto que a partir de las propuestas de computación sobre teorías físicas no clásicas —tal como la computación cuántica— la pregunta por el referente físico para la computación ha cobrado mayor vigencia ([35, p. 175], [36, p. 268]), no debemos olvidar que históricamente para constructos formales para los cuales no existía un referente físico aparente, éste fue encontrado en años posteriores a los de su formulación —esta situación es perfectamente ilustrada en el caso de algunas geométrias no euclidianas y la física relativista general—.

Aunque no había escuchado mencionar los términos de 'longitud de Planck' o 'tiempo de Planck', supongo que se deducen de la constante de Planck e inducen una cierta cuantización tanto en el espacio como en el tiempo. No conozco ninguna situación física la cual tenga sentido por debajo de las

constantes de longitud y tiempo de Planck, no obstante, esto no me impide pensar de forma especulativa —quizás demasiado especulativa— en posibles referentes físicos para las MTAs, sustentado en la idea de que nuestro conocimiento de la realidad física está lejos de ser completo. Desafortunamente considero que mis conocimientos actuales no me permiten sostener un diálogo de cierto valor argumentativo sobre los referentes físicos en lo que estoy pensando (i.e. velocidades superiores a la velocidad de la luz, viajes en el tiempo, gravitación cuántica, etc.) y la computación bajo esas condiciones. En relación con la pregunta que nos preocupa se que los profesores Copeland y Hamkins —quien construyó un modelo de máquina de Turing con un tiempo infinito de ejecución [37]— estan preparando un artículo acerca la plausibilidad de las MTAs con relación a la física Newtoniana, la física relativista y la física cuántica.

No deseo terminar nuestro pequeño recorrido por la MTAs sin mencionar que éstas son un caso particular de situaciones más generales denominadas "supertareas", las cuales —en su formulación general— hacen referencia a un número infinito —enumerable o no enumerable— de acciones realizadas en una cantidad finita de tiempo [38]. Considero finalmente que el estudio de la posibilidad física de las supertareas, en particular de las MTAs —independiente de su resultado— generará un mayor comprensión de las teorías involucradas en su formulación.

3.5. **Redes neuronales.**
Comentario inicial: Hava Siegelmann ha demostrado las siguientes correspondencias entre las redes neuronales recurrentes análogas (ARNN's) y algunos tipos de máquinas abstractas [39, 40]:

$$\begin{aligned} \text{ARNN con pesos enteros} &\equiv \text{Autómata finito} \\ \text{ARNN con pesos racionales} &\equiv \text{Máquina de Turing} \\ \text{ARNN con pesos reales} &\equiv \text{Máquina de Turing con oráculo} \end{aligned}$$

Los modelos neurofisiológicos más adecuados hasta la fecha, de actividad cerebral, son las llamadas redes neuronales de "tercera generación" con neuronas *'spiking'*. Estas redes equivalen a las redes análogas recurrentes con pesos reales de Siegelmann [41].

Pregunta: ¿Podrías dar una explicación básica de porqué los pesos reales en una ARNN corresponden a una situación de hipercomputación?

Respuesta: Para demostrar que un modelo de computación $\mathcal{X}$ es un modelo de hipercomputación usualmente se procede de dos maneras diferentes, o se demuestra que el modelo $\mathcal{X}$ computa un objeto no Turing-computable, o se demuestra que el modelo $\mathcal{X}$ es equivalente a otro modelo de hipercomputación ya establecido. En el caso de las ARNN's se ha se ha seguido la segunda alternativa estableciendo que una ARNN puede simular un modelo de hipercomputación denominado familia de circuitos booleanos no uniforme.

Un circuito booleano C_n es un modelo de computación representado por un árbol binario, donde cada nodo interior corresponde a una operación lógica —$\neg$: negación, $\vee$: disyunción y $\wedge$: conjunción— y las hojas del árbol corresponden a entradas boolenas tal como lo ejemplifica la figura (3).

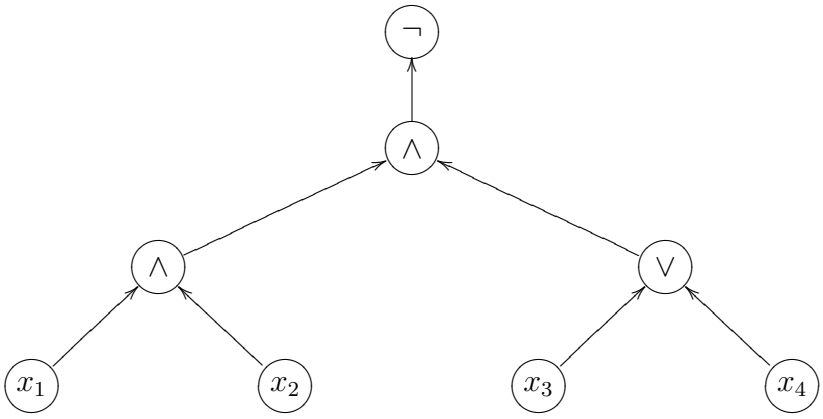

FIGURA 3. Circuito booleano C_4.

Una familia de circuitos booleanos $\mathcal{F}$ es un conjunto indexado $\{C_n \mid n \in \mathbb{N}\}$, donde C_n es el circuito que computa las entradas de longitud n [40]. Sea $\mathcal{L} \subseteq \{0,1\}^*$ un lenguage sobre el alfabeto $\Sigma = \{0,1\}$ y sea $\alpha \equiv a_1 a_2 \ldots a_n$ una palabra de Σ^*; se dice que la familia $\mathcal{F}$ —donde cada C_n tiene sólo una compuerta de salida— decide el lenguaje $\mathcal{L}$, si y sólo si, el valor de la compuerta de salida del circuito $C_n \in \mathcal{F}$ con las entradas $a_1 a_2 \ldots a_n$ es 1 si $\alpha \in \mathcal{L}$ y 0 si $\alpha \notin \mathcal{L}$.

Una familia de circuitos booleanos $\mathcal{F} = \{C_n \mid n \in \mathbb{N}\}$ se denomina uniforme (FUCB) si existe una máquina de Turing $\mathcal{M}$ que recibe como entrada 1^n y produce como salida C_n, de lo contrario, la familia de circuitos booleanos se

denomina no uniforme (FNUCB) [42]. A manera de ejemplo, sea $A \subset \mathbb{N}$ un conjunto no recursivo, una FNUCB está dada por $\mathcal{F} = \{C_n \mid n \in \mathbb{N}\}$ donde:

$$C_n = \begin{cases} \bigwedge_{i=1}^{n}(x_i \vee \neg x_i) & si\ n \in A,\ (\text{figura}\quad (4)), \\ \bigwedge_{i=1}^{n}(x_i \wedge \neg x_i) & si\ n \notin A. \end{cases}$$

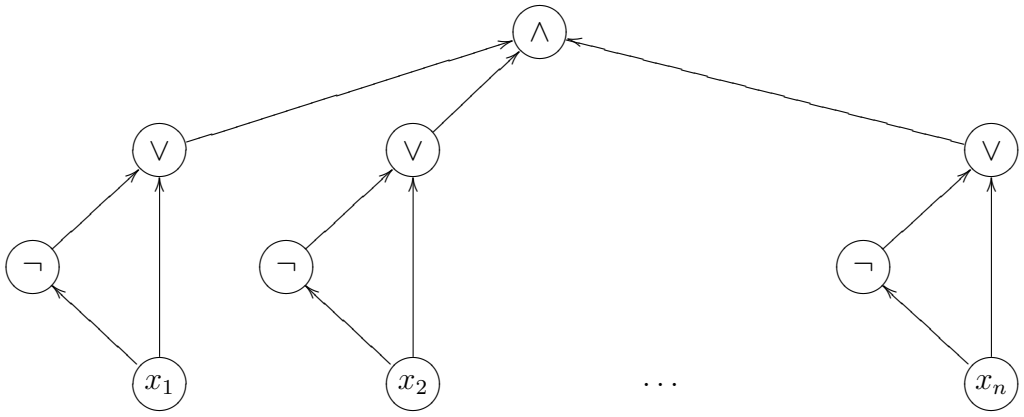

FIGURA 4. Circuitos $C_n = \bigwedge_{i=1}^{n}(x_i \vee \neg x_i)$.

Las familias no uniformes de circuitos boolenanos son un modelo de hipercomputación. La estrategia para demostrar que una ARNN puede simular una FNUCB consiste en: (i) codificar una familia de circuitos booelanos (uniforme o no uniforme) por un número real, (ii) parametrizar un peso de la ARNN con este número, (iii) construir la ARNN de tal manera que a partir de este número pueda obtener el circuito C_n para procesar una entrada de longitud n y (iv) simular el circuito C_n por medio de la ARNN. Para el caso de una FUCB su código es un número real computable, mientras que en el caso de una FNUCB su código es un número real no computable. Este número real (computable o no) puede ser manipulado por una ARNN puesto que éste se introduce como un parámetro de la red.

Pregunta: ¿Es el cerebro un hipercomputador?

Respuesta: Pienso que una respuesta afirmativa o negativa, o incluso una respuesta parcialmente afirmativa o parcialmente negativa, debería explicitar las hipótesis —formales o informales— sobre las cuales está construida, y además debería presentar algunas conclusiones que se obtendrían de ella. Aunque he leído autores que sostienen cada una de las opciones mencionadas y me he formado una opinión al respecto, no he trabajado en ninguna

oportunidad en dirección hacia una —posible— respuesta. Quizás percibo cierta "nubosidad" en la pregunta que no me atrae, quizás espero contar con teorías más adecuadas para enfrentar la pregunta. Si aún así, me fuera solicitada una respuesta, me inclinaría por "posiblemente si", pues comparto la afirmación de que *"it would —or should— be one the greatest astonishments of science if the activity of Mother Nature were never to stray beyond the bounds of Turing-machine-computability"* [14, p. 64].

3.6. **Computación cuántica.**

Comentario inicial: En el sitio *web* del Centro de Computación Cuántica de Oxford (`http://www.qubit.org/`) se afirma que:

> *"The discovery that quantum physics allows fundamentally new modes of information processing has required the existing theories of computation, information and cryptography to be superseded by their quantum generalisations".*

Aparentemente, estos modos fundamentalmente nuevos de procesamiento de información abren nuevas perspectivas respecto al límite de lo computable.

Pregunta: ¿Cuáles son los modos fundamentalmente nuevos de procesamiento de información que se dan en la computación cuántica?

Respuesta: En el mismo Centro de Computación Cuántica en Oxford, David Deutsch quien en 1985 creó el modelo denominado máquina de Turing cuántica [43], a la pregunta ¿qué es la computación cuántica? responde de la siguiente forma:

> *"A fundamentally new mode of information processing that can be performed only by harnessing physical phenomena unique to quantum mechanics (especially quantum interference)"* [44].

Desafortunadamente en este momento de la entrevista no cuento con el suficiente espacio para hacer una presentación detallada de la computación cuántica —el lector interesado puede referirse ha alguno de los siguientes artículos introductorios o textos [45, 46, 47, 48, 49, 50, 51]—; no obstante, espero poder presentar algunas ideas al respecto.

Mientras que la unidad clásica de información, el *bit*, puede estar en uno de dos posibles valores, 0 ó 1, la unidad cuántica de información, el *qubit*, es decir, el *quantum bit*, puede estar en una superposición lineal de estos valores. La notación usual en computación cuántica es la notación de "brakets"

introducida por Dirac [52] en donde un *ket* $|\,\rangle$ representa un estado de un sistema cuántico y consiste en un vector de un espacio de Hilbert. Formalmente, un 1-qubit es un elemento de un espacio de Hilbert 2-dimensional cuya base está dada por los vectores $\{|\,0\rangle\,, |\,1\rangle\}$, y es representado por:

$$|\,x\rangle = a_0\,|\,0\rangle + a_1\,|\,1\rangle\,, \tag{1}$$

donde $a_0, a_1 \in \mathbb{C}$ y $|a_0|^2 + |a_1|^2 = 1$. Aunque un 1-qubit puede estar en cualquiera de los infinitos estados superpuestos que satisfacen la ecuación (1), una vez se realice una medida de él, es decir, una vez se observe en que estado se encuentra el 1-qubit $|\,x\rangle$, éste sólo estará en dos posibles estados, el estado $|\,0\rangle$ o el estado $|\,1\rangle$. La probabilidad de encontrar el 1-qubit $|\,x\rangle$ en el estado $|\,0\rangle$ es de $|a_0|^2$ y la probabilidad de encontrarlo en el estado $|\,1\rangle$ es de $|a_1|^2$.

Es posible generalizar la noción de un 1-qubit a un n-qubit, obteniendo la siguiente característica:

$$\begin{aligned}
\text{1-qubit:} &\quad |\,x\rangle \in \mathbb{H}_2 \\
\text{2-qubit:} &\quad |\,x_1, x_2\rangle \in \mathbb{H}_{2^2} \\
\text{3-qubit:} &\quad |\,x_1, x_2, x_3\rangle \in \mathbb{H}_{2^3} \\
&\quad \vdots \\
n\text{-qubit:} &\quad |\,x_1, x_2, \ldots, x_n\rangle \in \mathbb{H}_{2^n}
\end{aligned}$$

donde $\mathbb{H}_n$ representa un espacio de Hilbert n-dimensional. Es decir, mientras hay un crecimiento lineal en el número de qubits, hay un crecimiento exponencial en el espacio de computación sobre el cual se opera.

Por otra parte, la evolución o dinámica de un n-qubit es determinada por un operador de evolución unitario U, el cual actúa sobre el espacio de Hilbert 2^n-dimensional. El operador U actúa sobre el n-qubit, generando un nuevo n-qubit de la siguiente manera:

$$U\,|\,x_1, x_2, \ldots, x_n\rangle = |\,x'_1, x'_2, \ldots, x'_n\rangle\,. \tag{2}$$

Los operadores U son llamados "compuertas cuánticas" en el contexto de la computación cuántica y son los análogos cuánticos de las compuertas lógicas tales como la compuerta *and* o la compuerta *not*, de allí que se hable entonces de "circuitos cuánticos".

La ecuación (2) señala que en un sólo paso de computación se puede operar sobre todos los elementos que forman un n-qubit, es decir, un computador cuántico puede en un sólo paso computacional, realizar la misma operación sobre 2^n estados cuánticos, codificados en una superposición de un n-qubit. Por otro lado, un computador clásico debería realizar 2^n pasos computacionales para realizar la misma tarea. Esta capacidad de los computadores cuánticos de operar en paralelo sobre los 2^n estados es denominada "paralelismo cuántico" y es la característica principal de la computación cuántica.

Pregunta: ¿Existen implementaciones actuales de computación cuántica?

Respuesta: En relación con las propuestas de implementación para la computación cuántica, es necesario mencionar que no hay consenso sobre la posibilidad o imposibilidad de las mismas debido al fénomeno de la decoherencia para sistema físicos de determinado número de elementos [53]. Sin embargo, un alto porcentaje de la investigación actual en computación cuántica —apoyada por grandes inversiones en dinero tanto gubernamentales como privadas— tiene como objetivo la construcción de máquinas cuánticas. Se han realizado propuestas de implementación desde diferentes perspectivas tales como: óptica cuántica, resonancia nuclear magnética, el computador cuántico atómico e íones atrapados, entre otras. El logro más importante en está dirección —al momento de responder esta pregunta— es el computador cuántico de cinco qubits construido por IBM y la universidad Stanford [54].

Pregunta: ¿Amplía la computación cuántica el universo de lo computable?

Respuesta: No, una máquina de Turing cuántica (MTC) y una máquina de Turing (MT) computan las mismas funciones Turing-computables ([43, p. 102], [55]). Sin embargo, si hablamos en un contexto diferente a las funciones computadas por una MTC y un MT, existe objetos que son generados por una MTC que no son generados por un MT, estos objetos son los números aleatorios "verdaderos":

> *"Although most modern programming languages contain some kind of command for generating a "random" number, in reality, they can only generate "pseudorandom" numbers. These are sequences of numbers that pass many of the test that a sequence of random numbers would also pass. But they are not true random numbers, because they are merely the completely predictable output from a definite function evaluation"* [51, p. 155].

La generación de números aleatorios es un proceso intrínseco para un computador cuántico. Sea $|x\rangle$ un 1-qubit y sea H la compuerta de Hadamard —operador de evolución— definida por:

$$H|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$
$$H|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

Inicialmente aplicamos la compuerta de Hadamard al qubit $|0\rangle$ y obtenemos la superposición de estados $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. Después realizamos una medida sobre esta superposición. "*The act of observation causes the superposition to collapse into either* $|0\rangle$ *or the* $|1\rangle$ *state with equal probability. Hence you can exploit quantum mechanical superposition and indeterminism to simulate a perfectly fair coin toss*" [51, p. 160].

Pregunta: ¿Cuáles son las diferencias entre la Turing-computabilidad y la computabilidad cuántica?

Respuesta: Quizás no sea exagerado afirmar que fue el algoritmo propuesto por Peter Shor [56] en 1994, quien situó a la computación cuántica en el lugar que hoy se encuentra. La complejidad algorítmica temporal asociada a este algoritmo, diseñado para factorizar un número es sus factores primos, establece un "ruptura" en términos de complejidad algorítmica con su contraparte clásico, es decir, mientras el mejor algoritmo clásico conocido para el problema en cuestión es de complejidad temporal exponencial, el algoritmo propuesto por Shor es de complejidad temporal polinómica.

Esta situación establece que —por lo menos desde la perspectiva teórica— mientras un computador clásico requiere alrededor de $4{,}5 \times 10^{25}$ años para factorizar un número de 1000 dígitos —la edad del universo está calculada en 2×10^{25} años aproximadamente—; un computador cuántico para realizar la misma tarea requiere alrededor de 3,07 días. La tabla (1) construida con base en las complejidades temporales asociadas a cada algoritmo, indica algunos tiempos necesarios para factorizar un número de n dígitos.

La complejidad algorítmica denomina problemas "intratables" aquellos problemas cuya solución tiene asociada una complejidad temporal exponencial y denomina problemas "tratables" aquellos cuya solución tiene asociada una complejidad temporal polinómica. Entonces, la computación cuántica ha

Número de dígitos	**Algoritmo clásico**	**Algoritmo de Shor**
129	1,85 años	45,9 minutos
250	$2{,}1 \times 10^6$ años	3,4 horas
1000	$4{,}5 \times 10^{25}$ años	3,07 días

CUADRO 1. Algoritmo de Shor vs algoritmo clásico.

demostrado su capacidad para trasladar algunos problemas de la categoría intratables a la categoría de tratables. Sin embargo en la actualidad, es un problema abierto el determinar si la computación cuántica puede o no trasladar la más importante categoría de problemas intratables denominada problemas NP-completos.

No deseo finalizar esta entrevista sin expresar mis agradecimientos al proyecto editorial "Hipercubo" por la cordial invitación realizada. Agradecezco además a mis colegas Daniel Velásquez, Juan Carlos Agudelo, Mario Elkin Vélez, Manuel Sierra y Raúl Gómez con quienes —de una u otra forma— he discutido algunas de las ideas presentadas.

REFERENCIAS

[1] Gary W. Flake. XXX Quotes, 2001. Eprint: `http://mitpress.mit.edu/books/FLAOH/cbnhtml/quotes.html` [03-Oct-2001].

[2] Andrés Sicard Ramírez and Mario Elkin Vélez Ruiz. ¿Máquina de Turing cuántica autorrefencial: una posibilidad?, 1999. U. EAFIT. Grant: 817407.

[3] Andrés Sicard Ramírez and Mario Elkin Vélez Ruiz. Prototipo de un modelo de computación cuántica continua, 2000. U. EAFIT. Grant: 817424.

[4] Gert-Jan C. Lokhorst. XXX Hypercomputation (slides). Eprint: `http://www.tbm.tudelft.nl/webstaf/gertjanl/hypercomputation.helsinki.html[02-Sept-2001]`, 2001.

[5] Andrés Sicard Ramírez. Máquinas de Turing. *Rev. U. EAFIT*, 103:29–45, 1996.

[6] Kurt Gödel. On formally undecidable propositions of Principia Mathematics and related systems. In Martin Davis, editor, *The undecidable*, pages 4–38. Hewlett, 1965.

[7] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.*, 42:230–265, 1936-7. A correction, ibid, vol. 43, no. 2198, p. 544–546, 1937.

[8] B. Jack Copeland and Gordon Aston. Hypercomputation. Eprint: `http://www.alanturing.net/pages/Reference\%20Articles/hypercomputation/`, [20-Jun-2000], 2001.

[9] B. Jack Copeland and Diane Proudfoot. Un Alan Turing desconocido. *Investigación y Ciencia*, pages 15–19, junio 1999.

[10] Alan M. Turing. Systems of logic based on ordinales. *Proc. London Math. Soc.*, 45(2239):161–228, 1939.

[11] B. Jack Copeland. Turing's o-machines, Penrose, Searle, and the brain. *Analysis*, 58:128–138, 1998. Preprint: `http://www.phil.canterbury.ac.nz/philsite/people/jack_copeland` [15-May-1999].

[12] B. Jack Copeland. The Church-Turing thesis. Stanford Encyclopedia of Philosophy, 1996.

[13] B. Jack Copeland. Narrow versus wide mechanism: including a re-examination of Turing's view on the mind-machine issue. *J. Philos.*, XCVI(1):5–32, january 2000. Preprint: `http://www.phil.canterbury.ac.nz/philsite/people/jack_copeland`.

[14] B. Jack Copeland and Richard Sylvan. Beyond the universal Turing machine. *Australasian Journal of Philosophy*, 77:44–66, 1999.

[15] Robert I. Soare. Computability and recursion. *The Bulletin of Symbolic Logic*, 2(3):284–321, sept. 1996.

[16] Robin Gandy. Church's thesis and principles for mechanisms. In Jon Barwise, H. Jerome Keisler, and Kenneth Kunen, editors, *The Kleene Symposium*, volume 101 of *Studies in Logic and the Foundations of Mathematics*, pages 123–148. Amsterdan: North-Holland, 1980.

[17] Martin Davis. *Computability and unsolvability*. New York: Dover Publications, Inc., 1982.

[18] Gregory Chaitin. *The limits of mathematics*. Singapur: Springer-Verlag, 1997.

[19] Andrés Sicard and Mario Vélez. Algunos comentarios sobre los números reales computables. Eprint: `http://sigma.eafit.edu.co:90/~asicard/archivos/mt-reales.ps.gz`, 2000.

[20] Cristian S. Calude. Real numbers: From computable to random. Technical report, CDMTCS, 2000. Eprint: `http://www.cs.auckland.ac.nz/CDMTCS/researchreports/141cris.pdf` [20-Nov-2000].

[21] B. Jack Copeland and Diane Proudfoot. Introduction to hypercomputation: computing the uncomputable. Eprint: `http://www.alanturing.net/pages/Reference\%\,20Articles/hypercomputation` [07-Nov-2000], 2000.

[22] Roger Penrose. *La nueva mente del emperador*. Colección: Libro de mano No. 38. Barcelona: Grijalbo Mondadori, 1991.

[23] Jean Ladrière. *Limitaciones internas de los formalismos*. Madrid: Editorial Tecnos, 1969.

[24] Newton C. A. da Costa. On the theory of inconsistent formal system. *Notre Dame Journal of Formal Logic*, XV(4):497–510, octuber 1974.

[25] Graham Priest, Richard Routley, and Jean Norman, editors. *Paraconsistent logic: essays on the inconsistent*. München, Viena: Philosophia Verlag, 1989.

[26] Jean-Yves Béziau. What is paraconsistent logic? In Diderik Batens, Chris Mortensen, Graham Priest, and Jean-Paul van Bendegem, editors, *Frontiers of paraconsistent logic*, pages 95–111. Hertfordshire, England: Research Studies Press LTD., 2000.

[27] Newton C. A. da Costa, Jean-Yves Béziau, and Otávio Bueno. *Elementos de teoria paraconsistente de conjuntos*. Centro de Lógica, Epistemologia e História da Ciência. U. Estadual de Campinas, Brasil, 1993.

[28] Chris Mortensen. Models for inconsistent and incomplete differential calculus. *Notre Dame Journal of Formal Logic*, 31(2):274–285, 1990.

[29] Graham Priest. Inconsistent models of arithmetic. Parte I: Finite models. *J. Philos. Logic*, 26(2):223–235, 1997.

[30] Graham Priest. Inconsistent models of arithmetic. Parte II: The general case. *The Journal of Symbolic Logic*, 65(4):1519–1529, 2000.
[31] Richard Sylvan and Jack Copeland. Computability is logic-relative. *Philosophy Research Papers, University of Canterbury*, 5:1–16, 1998.
[32] Cristian S. Calude and Michael J. Dinneen. Breaking the turing barrier. Technical report, CDMTCS, 1998.
[33] B. Jack Copeland. Even Turing machines can compute uncomputable functions. In Cristian C. Calude, J. Casti, and M. J. Dinneen, editors, *Unconventional models of computation*, pages 150–164. Singapure: Springer-Verlag, 1998.
[34] B. Jack Copeland. Super Turing-machines. *Complexity*, 4(1):30–32, 1998.
[35] Samuel Buss et al. The prospects for mathematical logic in twenty-first century. *The Bulletin of Symbolic Logic*, 7(2):169–196, 2001.
[36] David Deutsch, Artur Ekert, and Rossella Lupacchini. Machines, logic and quantum physics. *The Bulletin of Symbolic Logic*, 6(3):265–283, 2000.
[37] Joel Hamkins and Andy Lewis. Infinite time Turing machines. *The Journal of Symbolic Logic*, 65(2):567–604, june 2000.
[38] Jon Pérez Laraudogoitia. Supertasks. Stanford Encyclopedia of Philosophy. Eprint: `http://plato.stanford.edu/entries/spacetime-supertasks/` [07-Sept-2001], 2001.
[39] Hava T. Siegelmann and Eduardo D. Sontag. Turing computability with neural nets. Eprint: `http://citeseer.nj.nec.com/siegelmann91turing.html` [18-Ago-2001], 1991.
[40] Hava T. Siegelmann. *Neural networks and analog computation. Beyond the Turing limit*. Progress in Theorical Computer Science. Boston: Birkhäuser, 1999.
[41] Wolfgang Maass. XXX Networks of spiking neurons: the third generation of neural network models. In *Australian Conference on Neural Networks*, 1997. Eprint: `http://citeseer.nj.nec.com/maass97network.html` [18-Ago-2001].
[42] Howard Straubing. *Finite automata, formal logic, and circuit complexity*. Boston: Birkhäuser, 1994.
[43] David Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. R. Soc. Lond. A*, 400:97–117, 1985.
[44] David Deutsch. Frequently Asked Questions about Quantum Computation. `http://www.qubit.org/library/QuantumComputationFAQ.html`, Septiembre, 2001, 2001.
[45] Isaac L. Chuang and Michael A. Nielsen. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
[46] Jozef Gruska. *Quantum computing*. Cambridge: McGraw-Hill International (UK) Limited, 1999.
[47] Eleanor Rieffel and Wolfgang Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys*, 32(3):300–335, 2000.
[48] Andrés Sicard and Mario Vélez. Algunos elementos introductorios acerca de la computación cuántica, 1999. Memorias VII Encuentro ERM. U. de Antioquia, Medellín. Agosto 23-27.
[49] Peter W. Shor. Introduction to quantum algorithms. Eprint: `http://arXiv.org/abs/quant-ph/0005003`, 2000.
[50] Mario E. Vélez Ruiz and Andrés Sicard Ramírez. Cuántica y computación: una aproximación desde los postulados de la mecánica cuántica. (Preprint), 1999.

[51] Colin P. Williams and Scott H. Clearwater. *Explorations in quantum computing*. New York: Springer-Telos, 1997.
[52] P. A. M. Dirac. *Principios de mecánica cuántica*. Barcelona: Ediciones Arial, 1967.
[53] David P. DiVincenzo. The physical implementation of quantum computation. *Fortschr. Phys.*, 48(9–11):771–783, 2000.
[54] Mathias Steffen and Isaac L. Chuang. Toward quantum computation: A five-qubit quantum processor. *IEEE Micro*, pages 24–34, march-april 2001.
[55] Andrés Sicard and Mario Vélez. Some relations between quantum Turing machines and Turing machines. (Preprint), 1999.
[56] Peter W. Shor. Polinomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.