Miscelánea Matemática **29** (2000) 1–31 SMM

Una Breve Historia de la Computación en el Siglo XX: Las Grandes Contribuciones de los Matemáticos

Carlos A. Coello Coello
Laboratorio Nacional de Informática Avanzada
Rébsamen 80
Jalapa, Veracruz 91090
ccoello@xalapa.lania.mx

1 Introducción

Tener que escribir sobre las aportaciones más significativas de los matemáticos a la computación durante el siglo XX es una tarea por demás difícil. Después de todo, la computación se erigió como una disciplina independiente gracias al trabajo de un puñado de matemáticos. De tal forma, y a fin de no pecar de ambicioso en lo referente a los alcances de este artículo, me limitaré a hablar de un pequeño conjunto de matemáticos que han sido los que, a mi manera de ver, han realizado las contribuciones que han tenido mayor impacto en el desarrollo de la computación en el mundo. Inevitablemente, muchos nombres se omitirán, y muchas otras contribuciones indudablemente importantes serán ignoradas, pero las limitaciones físicas obvias de un documento de esta naturaleza hacen necesarios estos sacrificios. Espero que el lector no objete de manera significativa mi modesto (y tal vez injustificado) criterio de selección.

2 La Teoría de la Información

La primera historia que narraré es la de un joven que deseaba ser ingeniero eléctrico, pero se sentía tan atraído por las matemáticas, que acabó graduándose en ambas disciplinas. Me refiero a Claude Elwood Shannon, nacido el 30 de abril de 1916 en un pequeño pueblo de Michigan llamado Gaylord.

Shannon asistió a la Universidad de Michigan, y luego ingresó al Instituto Tecnológico de Massachusetts (MIT por sus siglas en inglés), atraído por el analizador diferencial[1] que Vannevar Bush diseñó y que tenía en operación en aquella institución [46]. Shannon pronto se sintió fascinado con el funcionamiento de los relevadores del analizador diferencial y comenzó a interesarse en encontrar la mejor forma de diseñar un circuito eléctrico. Recordando un curso sobre lógica simbólica que había tomado en la Universidad de Michigan, Shannon decidió utilizar el álgebra Booleana para simplificar el diseño de circuitos. Su tesis de maestría, titulada "*A Symbolic Analysis of Relay and Switching Circuits*" fue terminada en 1937 y constituye una de las publicaciones más importantes en la historia de la ingeniería eléctrica y la computación. El trabajo de Shannon mostró la forma en que una computadora podía realizar no sólo operaciones aritméticas, sino también lógicas usando un alfabeto binario (de unos y ceros).[2]

A instancias de Bush, Shannon acabó inscribiéndose al doctorado en matemáticas del MIT, obteniendo el grado en 1940 (simultáneamente, se le otorgó el grado de maestro en ingeniería eléctrica) con una tesis sobre la aplicación de teoría algebraica a la genética [33]. Impresionado por el trabajo de este joven, el entonces director del Departamento de Matemáticas de Laboratorios Bell lo invitó a trabajar en aquella institución. Fue ahí donde Shannon publicó (en 1948) su "Teoría Matemática de Comunicaciones"[3], que lo inmortalizaría como el "padre de la teoría de la información". En este trabajo, Shannon proporcionó una serie de teoremas que sugerían la manera de efectuar de forma eficiente la transmisión de mensajes ante la presencia de ruido en el canal de comunicación. El impacto de la teoría de la información abarca las comunicaciones vía satélite, los sistemas de detección y corrección de errores de las computadoras, y la codificación de las señales de televisión, entre otras muchas cosas.

Shannon fue también pionero en la disciplina que después sería bautizada como "inteligencia artificial" (IA) en donde incursionó original-

[1] Una máquina diseñada para resolver sistemas de ecuaciones diferenciales.

[2] Algunas fuentes indican que realmente el primero en advertir el potencial del álgebra Booleana para representar circuitos eléctricos fue el físico Paul Ehrenfest (en 1910) [3]. Sin embargo, debido a que el costo de los materiales y la mano de obra requeridos para construir circuitos de gran tamaño eran muy elevados en esa época, su trabajo permaneció ignorado hasta los 1930s, en que dichos costos disminuyeron. De ahí que sea la propuesta de Shannon la que pasara a la historia.

[3] Gracias a este trabajo se popularizó el término "bit" (contracción de *binary digit*, o dígito binario). Este término, sin embargo, no lo acuñó Shannon, sino John Wiley Tukey.

mente con una monografía publicada en 1948 bajo el título "*Programming a Computer for Playing Chess*". Su trabajo en esta área daría después origen a toda una rama de investigación dentro de la IA. Pero lo que más llegó a intrigar a Shannon fue el proceso de aprendizaje mismo y la forma en que éste podría automatizarse. Convencido de que sería posible construir una máquina capaz de aprender, Shannon diseñó un ratón mecánico llamado Teseo, el cual era controlado por un circuito relevador. Teseo era puesto dentro de un laberinto, del cual intentaría salir. A pesar de poder almacenar rutas previas, Teseo realmente operaba mediante un algoritmo de fuerza bruta (es decir, enumeraba todas las rutas posibles), pero puede considerarse como la primera máquina diseñada para "aprender" una tarea.

De 1956 a 1957, Shannon fungió como profesor visitante en el MIT, y fue una pieza fundamental en la organización de la memorable conferencia del *Dartmouth College* que dio origen a la Inteligencia Artificial como una disciplina independiente de estudio. Tras un año en Palo Alto, California, Shannon regresó al MIT como Donner Professor of Science, permaneciendo en ese puesto hasta su retiro, en 1978. Actualmente vive en Winchester, Massachusetts, donde suele encontrársele en un sótano gigantesco trabajando en su pasatiempo favorito: el malabarismo (ha construido varias máquinas para hacer malabares).

3 El Modelo Matemático de una Computadora

La idea de usar el sistema binario para procesar información en una máquina de cálculo (o sea, en una computadora) parece haberse originado en los 1930s. Los franceses R. Valtat [51] y Louis Couffignal [9] propusieron máquinas basadas en un sistema binario. De hecho, Couffignal intentó construir la máquina que propusiera en su tesis doctoral (un dispositivo electromecánico), para lo cual se asoció con la empresa *Logabax*. Sin embargo, debido al advenimiento de la Segunda Guerra Mundial, el proyecto nunca se pudo llevar a cabo.[4]

Aunque el trabajo de Valtat y Couffignal era un primer paso hacia el establecimiento de un modelo teórico de una computadora, el logro más importante a este respecto lo produjo un excéntrico matemático

[4]Couffignal construyó a fines de los 1940s una computadora electrónica digital en el Instituto Blaise Pascal, tras entrevistarse con Herman Goldstine en Princeton, y con el equipo de la Escuela Moore, en la Universidad de Pennsylvania.

inglés (educado en Cambridge) llamado Alan Mathison Turing.

Antes de hablar del trabajo de Turing, conviene hacer un pequeño paréntesis para explicar el contexto dentro del cual se llevó a cabo este importante descubrimiento.

En 1900 se celebró en París el "Segundo Congreso Internacional de Matemáticos". En él, el legendario matemático alemán David Hilbert presentó una ponencia titulada "Problemas Matemáticos", la cual estaba llena de optimismo con respecto al futuro de las matemáticas. En ella planteó 23 problemas no resueltos que él consideraba de gran importancia[5]. Esta plática no consistió simplemente en leer un conjunto de problemas matemáticos interesantes, sino que ha constituido prácticamente un mapa de las matemáticas del siglo XX. Muchos de estos problemas se han resuelto, y se ha demostrado que otros carecen de solución. Algunos problemas, como el sexto ("axiomatizar aquellas ciencias físicas en las cuales las matemáticas desempeñan una función importante") son demasiado vagos para que sean solucionados por completo. Otros, tales como el problema tres, eran mucho más específicos y fueron resueltos rápidamente. Pero lo que más preocupaba a Hilbert era la integridad de las matemáticas y consideraba que ese tema era uno de los más importantes para los investigadores. Por ello, él mismo se dio a la tarea de demostrar la consistencia de las matemáticas. Según Hilbert, no debía haber inconsistencias ni contradicciones en las matemáticas e intuía que era factible demostrarlo.

Sin lugar a dudas, Hilbert es uno de los matemáticos más importantes de su época, pero eso no significa que todos estuvieran de acuerdo con él, ni tampoco que nunca se equivocara. Bertrand Russell y Alfred North Whitehead se contaban entre los que se atrevían a contradecir a Hilbert. En su libro de 1903, Russell se dio a la tarea de [38]: "demostrar que las matemáticas puras lidian exclusivamente con conceptos definibles en términos de un número muy pequeño de conceptos lógicos fundamentales. [Es decir...] todas sus proposiciones son deducibles a partir de un número muy pequeño de principios lógicos fundamentales...". Esto estaba en contraposición con las ideas de Hilbert, quien intentaba rescatar a las matemáticas "de los pegajosos cenagales de la metafísica clásica" [5] mediante la separación de las ideas matemáticas de su significado. Bajo el esquema de Hilbert, conceptos tales como el de los enteros positivos no tendrían más un significado intuitivo derivado de la experiencia, sino que serían considerados como entidades ab-

[5]El texto original de la ponencia de Hilbert se encuentra disponible en `http://aleph0.clarku.edu/~djoyce/hilbert/problems.html`

stractas que obedecían solamente ciertas leyes formales. Hilbert no estaba solo, y algunas otras grandes mentes matemáticas de la época, como John von Neumann, se sumaron a su esfuerzo.[6]

Pero las cosas cambiaron cuando en 1930 un joven lógico austriaco llamado Kurt Gödel demostró que ciertas estructuras—si se suponen consistentes—contienen proposiciones cuya verdad es indecidible dentro del sistema [20]. Su demostración fue muy directa. De hecho, Gödel produjo una fórmula aritmética cierta tal que la suposición de la existencia de una demostración formal de ella conducía a una contradicción [5]. Las estructuras en las que este comportamiento anómalo puede ocurrir no son de carácter extraño o patológico, pertenecen a la aritmética. Además, probó que no puede demostrarse siquiera que la aritmética está libre de contradicciones en el sentido concebido por Hilbert.

La demostración de Gödel se considera una de las más importantes del siglo XX, y sus implicaciones tienen un fuerte efecto en la computación. Su sistema de asignar a cada teorema una secuencia de enteros sigue siendo utilizado hoy en día en teoría de la computación.

Aquí es donde aparece en escena el artículo publicado por Turing en 1936 [48] y que muchos consideran como el más importante en la historia de la computación: "*On Computable Numbers with an Application to the Entscheidungsproblem*" (Sobre Números Computables con una Aplicación al Problema de la Decidibilidad). El trabajo de Turing respondía a otra pregunta de Hilbert muy relacionada con la que respondiera Gödel: ¿existe un método definido que pueda ser aplicado a cualquier aseveración, de manera que garantice la generación de una decisión correcta sobre la veracidad de la aseveración? Hilbert intuía que la respuesta a esta pregunta era "sí". Turing demostró que Hilbert estaba equivocado.

Para construir su demostración, Turing usó un modelo teórico sumamente simple de una computadora (ahora denominado "máquina de Turing"). Este modelo consistía de una cinta de longitud infinita dividida en cuadros en cada uno de los cuales podía colocarse un solo símbolo de un cierto alfabeto. Las funciones posibles de esta primitiva computadora eran sólo: leer, escribir o borrar un símbolo en la cinta,

[6]La contribución más importante de von Neumann en esta área fue un artículo titulado "*Zur Hilbertschen Beweistheorie*", publicado en 1927. En este artículo, von Neumann definió un subsistema de análisis clásico y demostró rigurosamente, por medios finitos, que estaba libre de contradicción. A partir de este resultado, von Neumann conjeturó incorrectamente que mediante sus métodos podría demostrarse que era consistente todo el análisis [21].

por medio del movimiento de la cinta a razón de un cuadro a la vez (hacia la izquierda o hacia la derecha). A pesar de su simplicidad, esta máquina realmente modelaba un algoritmo y constituyó la primera herramienta teórica para explorar los límites de las aún inexistentes computadoras. Turing incluso demostró la existencia de una máquina que podría emular la operación de cualquier otra de sus máquinas y la denominó (apropiadamente) "máquina universal".

Turing también demostró que hay una cantidad infinita de problemas que una computadora nunca podrá resolver. Uno de estos problemas, denominado de la "detención" ("*halting*", en inglés) es de particular importancia en computación: La pregunta es si la Computadora X puede programarse para decidir en tiempo finito si la Computadora Y, cargada con otro programa y algunos datos iniciales, se llegará a detener alguna vez. Turing demostró que es imposible escribir un programa para la Computadora X que resuelva este problema aparentemente tan simple.

Otra solución al problema de la decidibilidad desde un enfoque radicalmente diferente provino del lógico norteamericano Alonzo Church [7]. Church y Stephen Kleene desarrollaron un formalismo llamado "cálculo lambda". Tras descubrir que todas las fórmulas aritméticas podían convertirse a una forma estándar con esta nueva herramienta matemática, Church procedió a construir un ejemplo para el cual no había ninguna fórmula del cálculo lambda que pudiera realizar la conversión requerida y que, por tanto, no podía resolverse. Aunque la demostración de Turing era mucho más directa y simple,[7] la de Church se publicó primero y uno de los revisores de su artículo le pidió que la mencionara en su trabajo. Turing trabajaría después con Church en Princeton, extendiendo las ideas de su artículo original hacia la "lógica ordinal". Esto sirvió como su tesis doctoral en Princeton, de donde se graduó en 1938.

Es interesante hacer notar que si Turing hubiese retrasado un poco más el envío de su artículo, su autómata bien podría llamarse la "máquina de Post", pues un matemático polaco-americano con este apellido envió un artículo con una propuesta muy similar a la de Turing, en octubre de 1936 (sólo cinco meses después). Emil Leon Post había leído el artículo de Church, y se percató de que su propuesta requería que cualquier método definido se describiera como una fórmula en cálculo lambda. Post propuso la noción de que un método definido sería aquel que pud-

[7]Era tan simple que el asesor de Turing en Cambridge (el topólogo Maxwell Herman Alexander Newman) dudó en un principio que la demostración estuviera correcta.

iese ser escrito en la forma de instrucciones para un 'trabajador' no pensante operando en una línea infinita de 'cajas'. Las operaciones que Post propuso para este operador son sorprendentemente similares a las del autómata de Turing [34]. Sin embargo, Post no propuso un 'trabajador universal' ni se involucró directamente con el problema de decidibilidad de Hilbert, sino que se limitó a relajar una restricción de la propuesta de Church. No obstante, resulta asombrosa la similitud de ambas propuestas las cuales, obviamente, fueron hechas de manera totalmente independiente. Sorprendentemente, Post también anticipó la demostración de Gödel antes mencionada desde principios de los 1920s, pero nunca publicó sus ideas al respecto [10]. Post sería después recordado por su definición de "computación" en términos de reglas de reescritura (reglas gramaticales generalizadas), que data de 1943. Estas reglas de re-escritura se volverían más tarde la base para las gramáticas transformacionales en lingüística y para los sistemas de conocimiento basados en reglas usados en inteligencia artificial.

Alan Turing jugó posteriormente un papel primordial en el esfuerzo inglés por descifrar el código Enigma de los alemanes durante la Segunda Guerra Mundial, y diseñó también una computadora electrónica inspirada en su modelo matemático (la Pilot ACE [49]). En 1950, publicó un artículo considerado hoy como clásico en inteligencia artificial [50]. En él, Turing plasma sus ideas sobre la filosofía de las máquinas y la mente y propone una prueba para poder determinar la manifestación de "inteligencia" por parte de una computadora. Un aparente suicidio con cianuro interrumpió prematuramente la carrera de este brillante matemático inglés en 1954.

4 La Guerra de los Matemáticos

Como Goldstine afirma [22], si la Primera Guerra Mundial puede considerarse como la "guerra de los químicos" debido al enorme apoyo que recibieron estos científicos durante el conflicto, la Segunda Guerra Mundial puede denominarse, sin lugar a dudas, la "guerra de los matemáticos". Esto se debió principalmente a la enorme importancia que adquirió la balística en los Estados Unidos durante la Segunda Guerra Mundial. Este interés tuvo como consecuencia el desarrollo de la primera computadora electrónica de la historia (al menos de acuerdo a los norteamericanos): la *Electronic Numerical Integrator and Computer* (ENIAC), que se destinó originalmente a la realización de complejos cálculos de balística [22].

De entre los diversos científicos importantes que contribuyeron al estudio científico de la balística en los Estados Unidos, destaca el matemático Oswald Veblen, quien no sólo fue uno de los líderes principales de la balística en su país, sino que también fue un importante impulsor de la investigación en matemáticas. Veblen usó sus fuertes influencias en el gobierno norteamericano para otorgar becas y estancias posdoctorales a un sinnúmero de jóvenes talentos para que realizaran investigación en matemáticas. Esto se oponía a la idea, entonces en voga en Estados Unidos, de que los doctores en matemáticas debían dedicarse exclusivamente a la enseñanza.

Veblen fue el que envió a Herman H. Goldstine a la Escuela Moore de la Universidad de Pennsylvania, y fue él también el que autorizó el dinero necesario para construir la legendaria ENIAC [22]. También fue el primer profesor del Instituto de Estudios Avanzados (IEA) de Princeton y fue quien invitó al matemático húngaro John von Neumann a unirse al IEA, que se convertiría, en muy poco tiempo, en una de las más selectas élites científicas de todos los tiempos.

De quien nos ocuparemos ahora será precisamente de John von Neumann, el prodigio matemático nacido en Budapest, Hungría el 28 de diciembre de 1903 en el seno de una familia judía con una posición económica envidiable. Dueño de una memoria prodigiosa y de una inteligencia suprema, von Neumann cursó la licenciatura en ingeniería química en Suiza (para complacer a su padre) mientras simultáneamente cursaba el doctorado en matemáticas en la Universidad de Budapest. Discípulo de Hilbert en Göttingen, von Neumann fue capaz de destacar dentro de la mejor escuela de matemáticas de su época.

Aunque von Neumann se interesaba sobre todo en problemas de matemáticas puras, con el advenimiento de la Segunda Guerra Mundial, hubo de concentrarse en problemas más prácticos para servir al gobierno de Estados Unidos, país del que se hizo nacional. Fue consultor en proyectos de balística, en ondas de detonación y, más tarde, se involucró en el Proyecto Manhattan,[8] en donde demostró la factibilidad de la técnica de implosión que más tarde se usaría en la bomba atómica que detonó en Nagasaki. Sin embargo, debido a su valía como consultor en otras agencias gubernamentales ligadas a la guerra, von Neumann fue uno de los pocos científicos a quien no se le requirió permanecer de tiempo completo en Los Alamos [4].

[8]Este fue el proyecto (ultra secreto durante la Segunda Guerra Mundial) llevado a cabo en el desierto de Los Alamos, en Nuevo México, y que dio origen a la primera bomba atómica del mundo.

Fue precisamente durante la primera mitad de 1943, en plena guerra, que von Neumann se interesó por primera vez en la computación. Tras un viaje a Inglaterra, le dijo a Oswald Veblen que creía sumamente importante que se utilizaran máquinas para acelerar los complejos cálculos involucrados con su trabajo. Aunque comenzaron a utilizar equipo de IBM, éste no satisfizo las necesidades del Proyecto Manhattan y von Neumann empezó pronto a buscar opciones en otros lados. En 1944 sólo había unos pocos proyectos para desarrollar computadoras en los Estados Unidos: Howard Aiken en Harvard, George Stibitz en Laboratorios Bell, Jan Schilt en la Universidad Columbia, y John Presper Eckert y John W. Mauchly en la Universidad de Pennsylvania [28]. Aunque von Neumann contactó a los tres primeros científicos y conoció sus máquinas, la única computadora con la que realmente se involucró a fondo fue la última (la ENIAC), gracias a un fortuito encuentro en una estación de trenes con el matemático Herman Heine Goldstine. Durante mucho tiempo, la ENIAC fue ignorada por la comunidad científica, pero con el apoyo de von Neumann fue finalmente tomada en serio hasta convertirse en un proyecto de primera línea.

Curiosamente, la ENIAC tenía una arquitectura en paralelo, aunque casi carecía de memoria (sólo podía almacenar 20 palabras), y otra máquina más ambiciosa, llamada EDVAC (*Electronic Discrete Variable Arithmetic Computer*) nació del deseo de sus diseñadores de construir una computadora "más útil" que operara en serie [22].

La relación entre von Neumann y los (de por sí controversiales) inventores de la ENIAC (Eckert y Mauchly) ha sido motivo de constantes debates en los libros de historia de la computación. Shurkin [43] hace ver que a von Neumann no le agradaba que Eckert y Mauchly trataran de beneficiarse económicamente de lo que había sido un proyecto militar secreto.[9] Goldstine [22], por su parte, se deshace en elogios a von Neumann, sobre todo en lo que respecta al ahora famoso "*First Draft of a Report on the EDVAC*",[10] que fue escrito por von Neumann en marzo de 1945, poniendo en lenguaje formal las ideas de Eckert y Mauchly,[11] pero olvidando mencionar sus nombres en la portada del documento, causando un obvio disgusto en los inventores de la ENIAC. Macrae [29]

[9] John von Neumann se enteró de que Eckert y Mauchly planeaban comercializar las computadoras electrónicas.

[10] *Primer Borrador de un Reporte sobre la EDVAC.* El documento puede leerse en el libro de Randell [36] (páginas 355-364).

[11] La notación utilizada por von Neumann en este artículo, así como la analogía que presenta entre los circuitos de una computadora y el sistema nervioso, fueron influenciadas por el artículo de McCulloch y Pitts [31] en el que se bosquejaron las primeras ideas de las denominadas "redes neuronales" [2].

da una versión más balanceada de los hechos, diciendo que realmente von Neumann no tenía la intención de aparecer como autor único de la monografía de 101 páginas que se considera uno de los documentos más importantes de la historia de la computación, porque describe la arquitectura de las computadoras que todavía se usan en buena medida en nuestros días. Según Macrae, lo que pasó fue que el reporte pretendía ser un documento interno de la Universidad de Pennsylvania, pero Goldstine se emocionó tanto al verlo que inmediatamente lo empezó a copiar y distribuir a todo aquel que se lo pidiera. Claro que la actitud conciliadora de Macrae es difícilmente aceptable ante la frialdad de los hechos que realmente ocurrieron en aquellos días. Se sabe que von Neumann cometió la indiscreción (según algunos, intencional) de decir a un reportero del famoso periódico norteamericano *The New York Times* que la Marina, el Ejército y la Oficina Meteorológica Nacional estaban patrocinando el desarrollo de una "nueva calculadora electrónica" cuyo potencial de cálculo tal vez permitiría realizar predicciones más acertadas del clima. El artículo fue publicado el 11 de enero de 1946, un mes antes de que el gobierno norteamericano decidiera hacer del conocimiento público la ahora famosa ENIAC, justo en el momento propicio para arruinar los planes de Mauchly y Eckert de comercializar su nueva invención.

Mauchly y Eckert además de disgustarse mucho por no haber recibido el crédito debido, acabaron por ser despedidos de la Universidad de Pennsylvania ante su negativa de ceder a la institución los derechos de la ENIAC. Este error administrativo le costó a la Universidad de Pennsylvania el perder el liderazgo tecnológico (en lo que a computación se refiere) en los Estados Unidos y todavía hoy se recuerda a éste como uno de sus peores momentos históricos.[12] Con el paso del tiempo, la guerra terminó, la EDVAC se volvió del dominio público, Mauchly y Eckert fundaron su propia empresa y von Neumann regresó a Princeton con el sueño de construir su propia computadora.

Debido a los tropiezos que tuvo inicialmente para conseguir dinero para construir su computadora, varias universidades le ofrecieron trabajo a von Neumann después de la guerra,[13] y aunque estuvo cerca de aceptar al menos una de estas ofertas,[14] fue leal al IEA, y finalmente

[12]Irónicamente, Irven Travis, que fue el que trató de arrebatar las patentes de la ENIAC a Mauchly y Eckert, acabó por renunciar a la Universidad de Pennsylvania tres años después del escándalo, para aceptar un empleo en la industria [43].

[13]Se sabe que recibió ofertas formales de la Universidad de Chicago y del Instituto Tecnológico de Massachusetts (MIT), y tuvo también ofrecimientos informales de Harvard y la Universidad de Columbia [4, 29].

[14]La del MIT, ya que su amigo Norbert Wiener casi lo convenció de que el IEA no

logró conseguir los fondos que necesitaba para su proyecto con la ayuda de Princeton y la RCA. Su idea era construir una máquina similar a la EDVAC pero más poderosa y más rápida. La computadora IAS (*Institute for Advanced Studies*) fue finalmente construida en los 1950 y su diseño ha servido como inspiración para la mayoría de las computadoras que la sucedieron, si bien la arquitectura que hoy recibe su nombre no fue realmente producto de su inventiva.

Las principales contribuciones de von Neumann a la computación moderna fueron [4, 22, 29]: la noción del uso de monitores para visualizar datos, la invención del diagrama de flujo, la teoría de los autómatas celulares, incontables técnicas de cómputo matemático y la co-autoría de la teoría de juegos que dio pie al famoso método de Montecarlo.[15]

Después de una prolongada agonía de más de un año, John von Neumann falleció víctima de cáncer el 8 de febrero de 1957.

Estados Unidos no fue el único país que se apoyó en sus matemáticos para ganar la Segunda Guerra Mundial. Inglaterra hizo lo mismo, aunque los esfuerzos de los matemáticos británicos se concentraron en un área distinta: la criptografía.

Alan Turing y su mentor de Cambridge, Maxwell H. A. Newman, fueron tan sólo dos de los brillantes matemáticos que trabajaron en *Bletchley Park*, en Inglaterra, descrifrando mensajes secretos de los alemanes.

Pero la mayor contribución inglesa a la computación mundial durante esa época fue la construcción de una computadora electrónica llamada *Colossus*, diseñada para facilitar las tareas de desciframiento de mensajes secretos [8]. Sin embargo, el proyecto fue tan secreto que no ha sido sino hasta hace relativamente poco tiempo que se han podido averiguar más detalles sobre esta máquina, y algunas preguntas respecto a ella quedarán para siempre sin respuesta, pues 8 de las 10 unidades que estaban en operación hacia el final de la guerra fueron destruidas por órdenes directas de Winston Churchill.

---

le permitiría trabajar en un proyecto técnico, dado que era una torre de marfil donde vivían sólo científicos teóricos (en tono sarcástico, Wiener llamaba *Princetitute* al IEA).

[15] Inventado junto con Stanislaw Ulam mientras trabajaban en Los Alamos.

5 Funciones Recursivas

A principios de los 1930s, un joven matemático norteamericano que dejaría profunda huella en la teoría de la computación ingresó a Princeton para obtener un doctorado bajo la tutela de Alonzo Church. Se trataba de Stephen Cole Kleene, quien nació el 5 de enero de 1909 en Hartford, Connecticut. Kleene fue el principal creador de la teoría de las funciones recursivas (es decir, funciones definidas en una secuencia finita de pasos combinatorios), y proporcionó métodos para determinar qué tipos de problemas pueden ser resueltos o no en una computadora. Muchos de los teoremas fundamentales en el área de funciones recursivas se deben a Kleene: forma normal, usos de minimización, jerarquías aritmética y analítica, teoría de grados de irresolubilidad, y recursión en órdenes superiores.

Kleene se volvió profesor en la Universidad de Wisconsin en Madison, en 1935, permaneciendo ahí hasta su retiro, acaecido en 1979. Durante ese tiempo, fue un importante impulsor de la lógica matemática en aquella universidad y en el mundo. Entre los muchos honores que recibió, se encuentra la Medalla Nacional de Ciencia, que es la condecoración científica más importante que otorga el gobierno de los Estados Unidos.

Kleene fue también el primero en estudiar las interrelaciones entre la lógica intuicionista y la teoría (clásica) de funciones recursivas. Usando herramientas de recursión, introdujo la realización recursiva, que es una importante técnica para interpretar enunciados intuicionistas dentro de la aritmética.

Su libro de 1952, *Introduction to Metamathematics* [26], ha guiado a varias generaciones de lógicos de todo el mundo, y sigue siendo hoy en día un texto de gran importancia y una fuente de enorme inspiración.

Stephen Kleene murió el 25 de enero de 1994, en Madison, Wisconsin.

6 No Determinismo

Una de las personas que resultó influenciada por *Introduction to Metamathematics* fue un joven estudiante de la Universidad Hebrea de Israel, quien se topó con este texto a principios de los 1950s. El libro de Kleene hablaba sobre las máquinas de Turing, y mostraba que la computación era algo más que una nueva tecnología, pues tenía una base matemática sólida.

El nombre de este joven era Michael Oser Rabin, y su enorme interés por la computación lo motivó a emigrar a los Estados Unidos.[16]

Rabin estudió matemáticas en la Universidad de Pennsylvania y posteriormente se inscribió al doctorado en lógica de la Universidad de Princeton, donde trabajó, al igual que Kleene, bajo la dirección de Alonzo Church. La tesis doctoral de Rabin consistió en la demostración de que varios de los problemas relacionados con grupos algebraicos no podían ser resueltos por una computadora.

En 1957, mientras Rabin escribía los resultados de su tesis, se le ofreció un trabajo de verano en IBM, donde conocería a otro joven lógico de Princeton llamado Dana Stewart Scott.

En vez de asociarlos a un proyecto en particular, se le dijo a la pareja que podía trabajar en lo que quisieran. El resultado de esta colaboración resultó ser de gran valía para la teoría de la computación.

Rabin y Scott comenzaron por considerar una forma limitada de la máquina de Turing en la que estaba prohibido escribir en la cinta. A este modelo se le denomina "máquina de estados finitos" y registra todo lo que aprende en una memoria cuyo número de estados se fija por anticipado y nunca se cambia.

Rabin y Scott estaban intrigados por una limitación implícita del modelo de Turing: una máquina con un conjunto dado de instrucciones y una entrada en particular, siempre se comportará de la misma manera. Su comportamiento es "determinista".

La noción introducida por Rabin & Scott fue la de "no determinismo": dada una cierta máquina en un estado particular, ésta tendrá una serie de posibles nuevos estados a los cuales podría pasar, en vez de sólo uno. En este caso, a diferencia de una máquina determinista, no hay una ruta única dada una cierta entrada, sino que hay varias rutas posibles que podrían seguirse. Estas distintas opciones pueden llevarnos a la ambigüedad, por lo que Rabin y Scott definieron una máquina "no determinista" como aquella que "acepta" una cierta secuencia si *al menos una* de las rutas posibles producidas con la entrada dada alcanza un estado "aceptable" (o deseado). En otras palabras, una máquina no determinista "adivina" la solución a un problema.

Las máquinas no deterministas habrían sido una simple curiosidad científica de no ser porque Rabin & Scott demostraron dos cosas muy importantes acerca de ellas: (1) es posible transformar cualquier máquina no determinista en una determinista, si bien ésta puede re-

[16] Israel no tenía computadoras en aquel entonces.

querir muchos más estados que la primera, y (2) las máquinas no deterministas de estados finitos resultan ser un medio excelente para expresar patrones de búsqueda (por ejemplo, de cadenas de caracteres dentro de un procesador de textos).

El artículo de Rabin & Scott sobre el no determinismo fue publicado en 1959 [35] y su impacto en la computación fue enorme, atrayendo de inmediato la atención de un grupo considerable de teóricos que durante los 10 años siguientes extendieron el concepto hacia aplicaciones que ni sus mismos autores imaginaron.

Por este trabajo, Rabin y Scott recibieron el Premio Turing[17] en 1976.

Rabin haría ver posteriormente la gran importancia de introducir la aleatoreidad en la solución de problemas de alta complejidad. Sus métodos aleatorios han tenido un gran impacto en geometría computacional, cómputo distribuido, criptografía y comunicaciones, entre muchas otras áreas.

Rabin es actualmente profesor de Ciencia de la Computación en Harvard y en la Universidad Hebrea de Jerusalén, dividiendo equitativamente su tiempo entre las 2 instituciones durante el año.

Por su parte, Dana Scott tendría después otra fructífera alianza (esta vez en la Universidad de Oxford) con Christopher Strachey, de la que se derivaría un sistema teórico para el estudio de propiedades de los programas y definiciones de los lenguajes de programación. A este sistema se le conoce como "semántica denotacional". Aunque Strachey ya no vería el fruto de este trabajo debido a su muerte prematura, acaecida en 1975, Scott siguió trabajando en esta área, extendiendo después su investigación hacia la unificación entre la semántica y los formalismos lógicos constructivos, en un intento por encontrar métodos de demostración rigurosos (y que puedan implantarse en una computadora) para la construcción inferencial de programas correctos.

Scott es actualmente profesor de Filosofía en la Universidad Carnegie-Mellon, en Estados Unidos.

7 Problemas NP-Completos

En los 1950s y 1960s, varios problemas de diseño, investigación de operaciones e inteligencia artificial parecían compartir ciertas propiedades que los hacían sumamente difíciles de resolver. Algunos científicos sospechaban que pertenecían a una misma familia matemática, pero no fue sino hasta los

[17]El equivalente al premio Nobel para la computación, que es otorgado por la ACM cada año.

1970 que los matemáticos Stephen A. Cook (de Estados Unidos) y Leonid Levin (de Rusia) la pudieron describir. Stephen A. Cook nació en 1939 en Buffalo, Nueva York. Atraído por los circuitos, ingresó a la Universidad de Michigan para estudiar ingeniería eléctrica, aunque acabó graduándose en matemáticas, tras descubrir que tenía un talento innato para esta disciplina. Posteriormente, ingresó a la Universidad de Harvard, donde obtuvo un doctorado en matemáticas bajo la dirección de Hao Wang, quien trabajaba en demostración automática de teoremas. Tras su graduación, permaneció durante un corto tiempo en la Universidad de California, en Berkeley, y posteriormente se trasladó a la Universidad de Toronto. En 1971, Cook publicó un artículo donde discutía problemas para los cuales se podía verificar una solución posible (un candidato) en tiempo polinomial. Puesto que no siempre es posible decidir cuál es el mejor candidato sería necesario que un programa "adivinara" una solución. Por ello, Cook llamó a estos problemas *Nondeterministic Polynomial* (No determinísticos polinomiales, o NP). Cook demostró que el problema de la "satisfactibilidad"[18] es uno de los más difíciles dentro de la familia NP. Específicamente, demostró que si hubiese un algoritmo para resolver el problema de satisfactibilidad en tiempo polinomial, entonces todos los problemas de la clase NP tendrían un algoritmo similar. De tal forma, los problemas equivalentes con el de la satisfactibilidad, son llamados NP-completos. Poco después de la publicación del artículo de Cook, Richard Karp, de la Universidad de California, en Berkeley, demostró que otros 21 problemas eran NP-completos también. El método que usó, denominado *reductibilidad*, consistía en convertir (o reducir) un problema al de satisfacibilidad. Esta técnica se sigue usando hoy en día, y se han identificado ya miles de problemas NP-completos. Sin embargo, la pregunta que sigue abierta y que muchos consideran como el mayor reto en teoría de la computación para el siglo XXI es: ¿Requieren realmente de búsqueda exhaustiva los problemas NP-completos o no? Mientras Cook se involucraba en la teoría de la complejidad en Harvard, un joven estudiante de preparatoria en la Unión Soviética aprendía sobre ciertos problemas de optimización que requerían *perebor* (fuerza bruta). Su nombre era Leonid Levin.

Leonid Levin nació en 1948 en Dnepropetrovsk, una ciudad industrial localizada en el corazón de Ucrania. Aunque de niño se sintió fuertemente atraído por la Química, Levin pronto descubrió su enorme talento para las matemáticas. Cuando apenas contaba con 15 años de edad conoció a Andrei Kolmogorov, quien luego invitó a Levin a la Universidad de Moscú. Kolmogorov inició su carrera como historiador, pero su amplia gama de in-

[18] Consiste en preguntar si hay una asignación de valores ciertos y falsos que haga que una cierta fórmula sea verdadera. Si la hay, la fórmula es *satisfacible*; de los contrario, es *no satisfacible*.

tereses (entre los que se incluían la música y la poesía), lo llevaron a estudiar matemáticas. Mientras que la escuela norteamericana relacionaba los problemas de complejidad con la lógica (en la tradición de Alonzo Church), en Rusia la teoría de los problemas *perebor* estaba inspirada en la noción de las relaciones entre la aleatoriedad de una secuencia de caracteres y la dificultad para describirla. Esta noción fue establecida precisamente por Kolmogorov. Levin se sintió atraído por la noción de complejidad de Kolmogorov, y desarrolló su tesis doctoral en ese tema. Aunque Levin terminó su tesis y ésta fue aprobada por su comité de defensa, el gobierno ruso le negó el grado de doctor en represalia a su comportamiento rebelde e irrespetuoso hacia las autoridades. Esto orilló a Levin a emigrar a los Estados Unidos. Sin embargo, antes de su salida de la Unión Soviética, Levin publicó un artículo (en 1973) sobre "problemas universales de búsqueda secuencial" en una revista soviética llamada *Problemy Perdachi Informatsii*. En este arículo, Levin definió una relación formal entre los problemas *perebor* y la teoría de la información de Kolmogorov. Aunque Levin desconocía el trabajo de Cook, en su artículo re-descubrió los problemas NP-completos (llamados "universales" por él).

8 Lenguajes de Programación

En los 1950s, la programación de las escasas computadoras existentes era tortuosa y requería de un meticuloso uso de la memoria, que era muy escasa en aquellos días. Un matemático que trabajaba para IBM en 1953, tuvo entonces la osada idea de proponer el diseño de un "lenguaje de programación" que facilitaría el uso de la IBM 704. La idea fue fuertemente atacada por John von Neumann (entonces asesor de IBM), que no veía la necesidad de una herramienta como esa, que simplemente "desperdiciaría valiosos recursos de cómputo", según él. Sin embargo, contra viento y marea, Cuthbert Hurd (director del Departamento de Ciencias Aplicadas de IBM) autorizó el proyecto llamado "*FORmula TRANslation*". El encargado de llevarlo a cabo fue el mismo joven que propuso su diseño: John Backus. El resultado de este proyecto fue el FORTRAN, que resultó ser no sólo el primer lenguaje de programación de "alto nivel" de la historia, sino uno de los más famosos y longevos.

John Warner Backus nació el 3 de diciembre de 1924 en Filadelfia, en el seno de una familia que gozaba de una cómoda posición económica. Desorientado y poco motivado, Backus ingresó a la Universidad de Virginia en 1942 para estudiar química (según los deseos de su padre), pero pronto fue expulsado debido a su bajo desempeño académico [41]. En 1943 fue reclutado por

el ejército, pero sus elevados resultados en una prueba de aptitud lo hicieron regresar a la vida académica, esta vez a la Universidad de Pittsburgh. En 1945 comenzó a estudiar medicina en Nueva York, pero pronto advirtió su falta de vocación para esta profesión.

Motivado por su interés en los radios, se inscribió a una academia, y al aprender a calcular circuitos comenzó a sentir una fuerta inclinación hacia las matemáticas. Este interés lo incitó a tomar algunos cursos de matemáticas en la Universidad de Columbia. Para 1949 ya estaba terminando la licenciatura y para 1950, concluía la maestría. Ambos grados le fueron otorgados en matemáticas. Sin saber todavía qué hacer con su vida, Backus entró a trabajar a IBM como programador en 1950, a pesar de que nunca antes había tenido contacto alguno con una computadora [46].

Para 1953, Backus era ya un hábil programador, pero estaba consciente de que dos terceras partes del tiempo útil de una computadora se dedicaban a la programación y a la depuración, ya que no existía herramienta alguna que facilitara tal proceso.

Aunque otros brillantes programadores de la época[19] habían intentado facilitar la programación mediante la escritura de herramientas similares a lo que hoy llamamos "macro-ensambladores",[20] nadie había intentado escribir un verdadero "lenguaje de programación", y había mucho escepticismo en torno a la factibilidad de un proyecto de esa naturaleza.

Backus sabía mejor que nadie acerca de este escepticismo, y por ello concentró sus esfuerzos no en el diseño del lenguaje en sí, sino en el de su traductor de comandos a números binarios (lo que hoy se denomina "compilador"). Su énfasis era la eficiencia, y el resultado de este esfuerzo haría de FORTRAN uno de los lenguajes de programación con mejor desempeño hasta la fecha.

Aunque el FORTRAN fue diseñado para la IBM 704, y sólo había unos 80 usuarios de esta computadora en todo el mundo, pronto se corrió la voz sobre este nuevo lenguaje, y en poco tiempo se volvería tan exitoso que su mismo creador se sorprende de que siga siendo vigente hasta nuestros días.

Aunque Backus dejó de trabajar en el FORTRAN desde fines de los 1950s, sus contribuciones a los lenguajes de programación distan de haber terminado ahí.

[19] Por ejemplo, la matemática Grace Hopper, que escribió el A-0, que se considera uno de los primeros compiladores de la historia, aunque éste era realmente un mero macro-ensamblador.

[20] Un programa que traduce palabras y símbolos a números binarios. Estas palabras (o abreviaciones de ellas) realmente son una mera representación simplificada de los comandos internos de una computadora.

En mayo de 1958, un comité internacional formado por destacados expertos en computación de Europa y Estados Unidos se reunió en Zurich para diseñar un lenguaje de programación más flexible y poderoso que el FORTRAN (que fue diseñado específicamente para tareas matemáticas y científicas), el cual, según el comité, se adoptaría como estándar en todo el mundo. El resultado fue el *International Algorithmic Language*, mejor conocido como Algol.

A Backus le agradaba el Algol, pero se sentía frustrado por la dificultad de expresar los elegantes conceptos contenidos en el lenguaje de una manera formal. Para resolver este problema, Backus aplicó un formalismo llamado "lenguajes libres de contexto", que había sido inventado alrededor de esa época por el lingüista Noam Chomsky.[21]

Backus presentó su propuesta sobre el uso de gramáticas para representar la sintaxis de un lenguaje de programación en 1959, en un congreso celebrado en París. A pesar de que su ponencia no fue incluida en las memorias del evento, un matemático danés que asistió a su presentación, se interesó sobremanera en este trabajo y habló de inmediato con Backus para discutir con él algunas dudas al respecto. Se trataba de Peter Naur, quien extendió la propuesta de Backus y luego la usó para describir toda la sintaxis del Algol. Esta notación se sigue usando hoy en día y se le conoce como la "forma Backus-Naur", en honor a sus inventores.

Inspirado por los lenguajes diseñados por los matemáticos John MacCarthy (LISP) y Kenneth Iverson (APL), Backus propuso un lenguaje llamado FP, el cual se basa en la composición de funciones,[22] en vez de basarse en las modificaciones explícitas a la memoria de la computadora (como el FORTRAN).

Irónicamente, los lenguajes funcionales, a pesar de su elegancia y su alto grado de abstracción, no han tenido el éxito que Backus y otros tantos científicos esperaban, sobre todo debido a las dificultades para compilarlos eficientemente.[23]

---

[21] En 1957, Noam Avram Chomsky, del Instituto Tecnológico de Massachusetts, demostró que los autómatas de estados finitos son insuficientes para modelar la gramática inglesa, debido a que no permiten estructuras anidadas como las existentes en el idioma inglés. Esta limitación motivó a Chomsky a introducir las gramáticas libres de contexto, que permiten relaciones bidimensionales, en vez de las lineales de los autómatas de estados finitos. Chomsky propondría después toda una jerarquía de lenguajes (que hoy lleva su nombre) que permiten (además de las dos antes mencionadas) relaciones tridimensionales (lenguajes sensibles al contexto) y multidimensionales (lenguajes recursivamente enumerables). Por este importante trabajo, se considera a Chomsky como el "padre de los lenguajes formales" [41].

[22] Este tipo de lenguajes se denominan "funcionales".

[23] La eficiencia es precisamente la mayor virtud del FORTRAN.

Backus fue nombrado "*fellow*" de IBM en 1963 y en 1977 se le otorgó el *Premio Turing* por sus contribuciones a los lenguajes de programación. Se retiró de IBM en 1991, apartándose desde entonces de la computación casi por completo. Hoy en día practica la meditación y disfruta leyendo a Krishnamurti y Eva Pierrakos [41].

9 El Arte de la Programación

En enero de 1962, la editorial norteamericana *Addison-Wesley* solicitó a un joven estudiante de doctorado del *California Institute of Technology* (Caltech), que escribiera un libro sobre compiladores, que era un tema muy poco entendido en esa época. El nombre de este joven era Donald Knuth, y su trabajo se convertiría en una piedra angular en el desarrollo de la computación como una disciplina independiente.[24]

Donald Ervin Knuth nació el 10 de enero de 1938 en Milwaukee, Wisconsin. Tímido y un tanto inseguro de su capacidad intelectual, Knuth siempre se esforzó (según él) por no reprobar durante su época de estudiante, teniendo en consecuencia un desempeño excepcional que lo hizo acreedor de numerosos reconocimientos desde muy temprana edad. Knuth ingresó al *Case Institute of Technology* (hoy *Case Western Reserve*), en Ohio, con la intención de estudiar música, pero al ofrecérsele una beca para estudiar física decidió probar suerte en esta disciplina. Pronto descubriría que su verdadera vocación eran las matemáticas, y su desempeño académico fue tan excepcional que, al graduarse de la licenciatura en 1960, sus profesores decidieron otorgarle simultáneamente el grado de maestría (en matemáticas), en un hecho sin precedentes en la historia del *Case Institute of Technology* [46].

Hacia el otoño de 1963, Knuth tenía listo el primer borrador de su libro, y comenzó a probarlo con sus estudiantes en el Caltech. Para 1966, ya tenía 3000 páginas escritas a mano, y pronto se dio cuenta de que se requeriría más de un libro para darles cabida. Tras varias discusiones con *Addison-Wesley*, se acordó que la obra, denominada *The Art of Computer Programming*, se publicaría en 7 volúmenes. Hasta ahora, sólo se han publicado 3 de ellos, los

[24] La computación se volvió una disciplina independiente en los 1960s. El término *Computer Science* (Ciencia de la Computación) fue acuñado por George Elmer Forsythe, un matemático especializado en análisis numérico que fundó uno de los primeros departamentos de Ciencia de la Computación en Estados Unidos (el de Stanford, en 1965) [28]. El primer departamento de Ciencia de la Computación de los Estados Unidos se fundó sólo 3 años antes, en 1962, en la Universidad de Purdue, en Indiana. La primera persona en obtener un doctorado en computación fue Richard Wexelbrat, quien se graduó en 1965 de la Universidad de Pennsylvania.

cuales han sido uno de los mayores éxitos comerciales de *Addison-Wesley*. Estos 3 libros, escritos con un increíble nivel de detalle, se volvieron lectura indispensable de todo estudiante de computación durante más de 20 años, y aún hoy en día siguen siendo de enorme importancia, a pesar de haberse publicado en los 1960s. Por este trabajo seminal, Knuth recibió el *Premio Turing* en 1974.

Aun si Knuth no hubiese publicado nada más en su vida, sus 3 libros serían suficientes para garantizarle un lugar de honor en la historia de la computación moderna. Sin embargo, su inagotable energía lo ha llevado a producir más de 150 publicaciones, que abarcan desde tres de los algoritmos más importantes jamás escritos, hasta meticulosos estudios históricos de los lenguajes de programación y los algoritmos, pasando por toda una gama de escritos diversos, que incluyen una novela de Ciencia Ficción [41].

Además, en un esfuerzo monumental de 9 años, Knuth inventó TEX, el primer lenguaje para tipografía computacional, junto con METAFONT, que es un sistema que hace uso de las matemáticas clásicas para diseñar alfabetos. En un acto característico de Knuth, decidió volver estos dos importantes programas de dominio público, a fin de promover su uso y motivar su perfeccionamiento. Pronto, TEX se volvió un importante estándar académico, sobre todo en las comunidades de matemáticas y computación.[25]

Donald Knuth es actualmente profesor emérito en Stanford, y continúa con su ritmo frenético de trabajo, preparando el volumen 4 de su *Art of Computer Programming* (sobre algoritmos combinatorios), escribiendo de paso algunos de los algoritmos más concisos y elegantes jamás creados.

10 Hacia el Futuro

En 1949, la ENIAC (la primera computadora electrónica de la historia) requería de 18000 bulbos para funcionar y pesaba unas 30 toneladas. Un artículo "visionario" de la revista *Popular Mechanics* predijo en aquel entonces que la computadora del futuro tal vez tendría sólo 100 bulbos y pesaría apenas 1.5 toneladas. La predicción, si bien optimista, se quedó extremadamente corta con respecto a lo que la tecnología (gracias en gran medida al invento del transistor a fines los 1940) nos ha proporcionado.

Pero aún con todos los adelantos de nuestra era y con la cada vez mayor dificultad de causar asombro en los semblantes de la gente, existen metas

[25]Este artículo, por ejemplo, fue escrito completamente en LATEX que es un dialecto de TEX.

y sueños más allá de nuestro alcance. Por ejemplo, el nivel de sofisticación que una computadora como HAL (de la novela "2001: Una Odisea Espacial") requiere, sigue estando fuera del alcance de nuestra tecnología actual, a pesar de las promesas que la inteligencia artificial hiciera en los 1950s. Hay también muchos problemas que, debido a su enorme complejidad, requieren de mucho mayor poder de cómputo del que la tecnología actual dispone (por ejemplo, la predicción del clima).

Por todo esto, resulta indudablemente difícil (y tal vez hasta inútil) intentar predecir hacia dónde se dirige la computación en el nuevo milenio. Por ello, en vez de atreverme a siquiera esbozar una respuesta, prefiero brindar a mis lectores un breve panorama de los proyectos y las ideas más novedosas (en lo que a computación refiere) que hoy son tema de investigación y/o discusión alrededor del mundo.

La primera inquietud de los investigadores es la miniaturización de los microprocesadores. Recientemente, un investigador de Intel [30] expresó su preocupación de que no parece haber manera de continuar reduciendo el tamaño de los componentes de un chip a menos que haya un cambio radical en la tecnología empleada para ello. Si la Ley de Moore acerca de la duplicación del número de transistores en un circuito integrado cada 18 meses sigue cumpliéndose (como ha ocurrido desde la invención de los chips en los 1970), se espera que para el año 2015, los diseñadores de microprocesadores se encuentren trabajando a nivel molecular [32]. Eso ha hecho de la "nanotecnología" una de las áreas de mayor interés en la actualidad para los diseñadores de hardware.

El concepto de "nanotecnología" no es nuevo, pues fue sugerido por el físico Richard Feynman desde 1959 [14] cuando indicó que la física no excluye el control de los átomos de manera independiente, por lo que planteó la posibilidad de construir máquinas diseñadas para construir máquinas más pequeñas que ellas mismas, las cuales construirían a su vez otras más pequeñas que ellas mismas, y así sucesivamente. A este enfoque se le conoce hoy en día como miniaturización de arriba hacia abajo.

En 1983, Eric Drexler [12] planteó un enfoque diferente: construir materiales y dispositivos de abajo hacia arriba, con cada átomo en una cierta posición deseada. Este enfoque, que se denominó nanotecnología molecular (o simplemente "nanotecnología"), causó gran controversia en un principio, pero ha ido ganando aceptación entre la comunidad científica tras los avances tecnológicos logrados en los últimos años [32]. Claro que la meta última de la nanotecnología, que consiste en construir máquinas moleculares que funcionen bien a pesar de los efectos del ambiente, permanece lejana todavía. Sin embargo, ya se han dado los primeros pasos en esa dirección. Por ejemplo,

el *Institute for Molecular Manufacturing*, en colaboración con el *Xerox Palo Alto Research Center*, han podido producir las tres máquinas moleculares más complejas de la actualidad: un engranaje, una bomba y un controlador de movimientos finos.

Aunque la investigación en nanotecnología todavía es incipiente, al menos dos premios Nobel se han interesado en ella, y hay un puñado de universidades y centros privados de investigación interesados en esta disciplina. De tal forma, no puede descartarse la posibilidad de que el nuevo milenio vea nacer las primeras nanomáquinas complejas (y verdaderamente útiles) del mundo.

Mientras los diseñadores de microprocesadores enfrentan los límites físicos de la miniaturización, los usuarios del cómputo intensivo enfrentan los límites del poder de las supercomputadoras. Durante más de 50 años, la arquitectura predominante en las computadoras ha sido la de von Neumann, según la cual existe un solo procesador muy complejo que ejecuta secuencialmente una sola tarea a la vez (independientemente de la complejidad de ésta).

Hace varios años, el paralelismo adquirió un enorme ímpetu, y se desarrollaron arquitecturas novedosas que permitían cómputo masivo en paralelo [24]. Estas arquitecturas se basan en el uso de unos cuantos procesadores poderosos que efectúan una serie de tareas relativamente complejas entre las que se incluyen el paso de mensajes, la coordinación de procesos, la sincronización y la concurrencia. Sin embargo, tal vez debido en gran parte a nuestra tendencia a diseñar algoritmos inherentemente secuenciales, resulta sumamente difícil encontrar grados significativos de paralelismo en un programa. Por ende, el paralelismo masivo no ha cumplido muchas de las promesas que hiciese en los 1980 en torno a la solución de problemas más complejos y ambiciosos [17].

Una alternativa interesante a las arquitecturas masivas en paralelo es la denominada "computación celular" [44]. Curiosamente, el mismo autor de la arquitectura más popular de hoy en día fue el que concibió el ejemplo clásico de computación celular. John von Neumann, junto con Stanislaw Ulam, concibieron a fines de los 1940s el modelo del autómata celular de un sistema dinámico en el cual el espacio y el tiempo son discretos. El modelo surgió cuando von Neumann estudiaba aspectos relacionados con la lógica de los seres vivos. Particularmente, el interés de von Neumann era averiguar si se podían usar consideraciones puramente matemáticas para establecer las propiedades específicas que les permiten a los seres vivos reproducirse.

Un autómata celular consiste de un arreglo de células, cada una de las cuales puede estar en uno de una cantidad finita de estados posibles. Las células se actualizan de manera síncrona a intervalos de tiempo discretos, de

acuerdo a una regla de interacción local idéntica para todas. El estado de una célula en el paso siguiente se determina mediante los estados actuales de un conjunto de células vecinas. Esta transición suele especificarse en la forma de una tabla de reglas, la cual marca el siguiente estado de la célula para cada posible configuración de sus células circundantes [45].

Desde su creación misma, los autómatas celulares han sido utilizados como modelos formales para estudiar fenómenos en diversas áreas, de entre las que destacan la física, la biología y la computación. Usando los principios básicos de este modelo matemático, algunos científicos como Sipper [44] han adoptado el término "computación celular" para referirse a una arquitectura basada en tres principios fundamentales: simplicidad, paralelismo vasto y localidad. El concepto es muy interesante, pues promete incrementar significativamente la capacidad de procesamiento de las computadoras.

Aunque se han realizado numerosos experimentos en software con la computación celular, su implantación en hardware sigue siendo objeto de investigación. Sin embargo, existen ya algunos avances interesantes tales como [45]: (a) el chip analógico para redes neuronales celulares; (b) los automátas celulares no uniformes evolutivos implantados usando procesadores configurables; (c) los autómatas celulares implantados como hardware digital de propósito específico, etc.

Al igual que el paralelismo masivo, la computación celular no resultará adecuada para todo tipo de problemas, pero aquellos que trabajan en esta área prometen al menos poder atacar más eficientemente algunos problemas NP completos, mejorar la generación de números aleatorios y mejorar el desempeño en el hardware para procesamiento de imágenes, entre otras cosas [45]. Curiosamente, la computación celular se intersecta con la nanotecnología, pues debido a su simplicidad, se cree que sería la arquitectura idónea para desarrollar dispositivos de cálculo en muy pequeña escala [45].

También relacionada con el concepto de cómputo celular, se encuentra otra idea intrigante que surgió a mediados de los 1990s: la computación basada en el ADN (ácido desoxirribonucleico). En 1994, Leonard M. Adleman (de la Universidad del Sur de California) publicó un artículo en la revista *Science* [1] en el cual mostraba cómo era posible usar moléculas de ADN para resolver el problema de las rutas hamiltonianas dirigidas.[26] Para sus experimentos, Adleman utilizó *oligonucleótidos* (cadenas cortas de hasta 20 nucleótidos) para codificar los vértices y las conexiones de un grafo. Posteriormente, colocó múltiples copias de los oligonucleótidos en un tubo de ensayo. En el tubo, los oligonucleótidos se enlazaron aleatoriamente unos con otros,

[26] Dado un grafo dirigido arbitario, debe encontrarse si existe una ruta entre dos vértices dados que visite cada vértice exactamente una sola vez.

formando moléculas que representaban rutas posibles del grafo. Adleman aplicó entonces técnicas de biología molecular para filtrar la enorme cantidad de soluciones de moléculas de ADN existentes, de manera que pudiese obtener la respuesta deseada. Este proceso de "filtrado" consiste en someter los segmentos de ADN a una serie de reacciones químicas similares a los cálculos matemáticos que efectúa una computadora. Por ejemplo, algunas operaciones matemáticas son efectuadas por enzimas cuya función depende de los valores específicos de la información en una o más partes del segmento de ADN. De esta manera, una enzima podría agregar un valor de uno al final de un segmento de ADN si cualquiera de dos valores específicos de dicho segmento tuviesen un valor de uno. Al organizar de manera coherente estas operaciones, es posible realizar complejos cálculos matemáticos.

Una vez que todas las reacciones químicas terminan, es posible examinar los segmentos de ADN resultantes para obtener la respuesta al problema planteado (por ejemplo, se puede localizar el segmento de ADN más corto y luego se decodifica para obtener la secuencia correspondiente a la respuesta). Una de las ventajas del uso de ADN es que los cálculos que se efectúan por medio de reacciones químicas se llevan a cabo en paralelo.

A raíz de la publicación del artículo de Adleman en 1994, tanto la *National Science Foundation* como DARPA decidieron apoyar un buen número de proyectos relacionados con el cómputo con ADN en los Estados Unidos. El interés en particular del Departamento de Defensa va más allá de la pura curiosidad científica. A mediados de los 1990s, un grupo de científicos norteamericanos plantearon un método mediante el cual podría usarse una computadora basada en ADN para descifrar mensajes codificados con el *Data Encryption Standard* (DES), que es un estándar de criptografía ampliamente utilizado en los Estados Unidos [6]. Si la idea de estos científicos se llevase a la práctica, se cree que las llaves del DES, que requieren unos 72 mil trillones de cálculos para descifrarse,[27] podrían obtenerse en sólo dos horas.

Claro que el estado actual de esta tecnología es también muy incipiente, y los cálculos que se han podido efectuar hasta ahora usando el ADN no rebasan aquellos que un humano puede realizar con lápiz y papel. Pero el potencial está allí, y ya hay varios proyectos norteamericanos que intentan desarrollar una computadora basada en estos principios (por ejemplo, el de la Universidad de Wisconsin, en Madison, que es dirigido por Robert M. Corn). De tener éxito, ésta es otra tecnología que podría cambiar nuestra forma de procesar información en el futuro.

Para terminar este breve asomo al futuro, me referiré a la que sea tal vez

[27]Sobra decir que estos requerimientos están mucho más allá de las capacidades de cálculo de cualquier computadora construida hasta ahora.

la más osada de las ideas revolucionarias sobre el diseño de computadoras: la computación cuántica.

Los principios de las computadoras de hoy en día se derivan de las leyes de la física clásica. Sin embargo, los físicos nos han demostrado que existen leyes más sutiles para describir el mundo real de las que abarca la física clásica. Por ende, estas leyes podrían tener un impacto significativo en la forma en que efectuamos cálculos con una computadora.

Algunos científicos consideran que, junto con la teoría de la información (de cuyos orígenes hablamos al principio de este artículo), la otra revolución conceptual más importante del siglo XX fue la teoría cuántica [47]. Indudablemente, la teoría de la información ha tenido un impacto significativo en nuestra vida diaria, debido a la globalización de las comunicaciones y al impacto de ésta en todas nuestras actividades cotidianas. Pero, ¿qué hay acerca de la física cuántica?

A pricipios de los 1980s, Richard Feynman mostró la forma en que un sistema cuántico podría realizar cálculos, e incluso especuló sobre la posibilidad de construir una computadora basada en estos principios [15, 16]. La promesa de la computación cuántica no radica en rebasar los límites teóricos de la computación clásica (es decir, la máquina universal de Turing sigue siendo el modelo teórico válido de las computadoras cuánticas), sino más bien estriba en aumentar de manera excepcional la eficiencia de procesamiento de las computadoras.

Las computadoras digitales de hoy en día utilizan como base para almacenar y procesar su información al bit, que puede tomar únicamente un valor de cero o uno. En contraste, la mecánica cuántica nos permite codificar la información en bits cuánticos (llamados *qubits*, en inglés). Un qubit puede representarse mediante un átomo en uno de dos estados posibles, los cuales también pueden denotarse como cero y uno. Dos qubits, al igual que dos bits clásicos, pueden alcanzar cuatro estados diferentes bien definidos: 00, 01, 10, o 11 [19]. Sin embargo, a diferencia de los bits convencionales, los qubits pueden almacenar al cero y al uno al mismo tiempo [47].

El poder de las computadoras cuánticas radica en su capacidad para estar en estados múltiples simultáneamente (a este fenómeno se le denomina "superposición" [19]) y a su capacidad de actuar sobre todos estos estados a la vez. Esto significa que una computadora cuántica podría efectuar una cantidad inmensa de operaciones simultáneamente, usando una sola unidad de procesamiento. Mientras que en las computadoras convencionales la capacidad de paralelismo se incrementa en proporción directa al tamaño del sistema, en los sistemas cuánticos, se incrementa exponencialmente con respecto a su tamaño. De tal forma, para emular el grado de paralelismo de

300 procesadores cuánticos, necesitaríamos 2^{300} procesadores convencionales (una cifra mayor al número de partículas en el universo) [47].

Desgraciadamente, extraer los resultados del procesamiento masivo efectuado por una computadora cuántica es un problema más difícil de lo que pudiera pensarse. Cualquier intento por extraer información de un estado requiere de mediciones. Desafortunadamente, en la computación cuántica, cualquier medición perturba el estado, con lo que se destruye el paralelismo cuántico. Esencialmente, se puede hacer una y solamente una pregunta acerca de los resultados generados por el paralelismo cuántico antes de que tengamos que rehacer los cálculos. Además, el tipo de pregunta que puede hacerse está restringido, y actualmente es un área activa de investigación [47].

Peter W. Shor [42], de AT&T, encontró una forma de hacer una sola pregunta para encontrar los factores primos de un número. Su descubrimiento tuvo un enorme impacto en criptografía, y desató un súbito interés en la computación cuántica alrededor del mundo.

En 1996, científicos de IBM, el MIT y la Universidad de California en Berkeley construyeron una computadora cuántica modesta de dos qubits hecha a partir de una pequeña cantidad de cloroformo (apenas suficiente para llenar un dedal). La preparación de los datos de entrada para esta pequeña máquina fue un proceso complejo y engorroso [19]. Sin embargo, los investigadores fueron capaces de probar su ingenioso dispositivo con un problema de búsqueda.

Usando un algoritmo de búsqueda diseñado por Lov K. Grover [23], de Laboratorios Bell, este grupo de investigadores se dio a la tarea de implantarlo en su computadora cuántica. Típicamente, la búsqueda de un elemento en una base de datos con n registros, requeriría, en promedio $\frac{n}{2}$ intentos antes de tener éxito. Sin embargo, el algoritmo de Grover puede encontrar el elemento deseado en $\sqrt{n}$ intentos [19]. Aunque resulta difícil escribir algoritmos para computadoras cuánticas, y no todos los problemas que nos interesa resolver (por ejemplo, muchos de los problemas NP Completos) resultarán adecuados para el poder de este tipo de computadoras,[28] es indudable que, de ser posible construir computadoras cuánticas a gran escala, se abren posibilidades sumamente interesantes para los científicos en los próximos años.

Todas estas tecnologías posibles son sumamente interesantes y plantean preguntas que mantendrán bastante ocupados a los científicos durante el nuevo milenio. No obstante, no se visualizan avances únicamente en lo referente a tecnología, sino también en cuanto a otros aspectos de la computación.

[28] Claro que una vez iniciado el camino especulativo, no hay razón para detenerse, y algunos hablan ya de *heurísticas cuánticas* para algunos de estos problemas.

Por ejemplo, el final del milenio ha visto el ascenso (en popularidad) de las heurísticas,[29] y la inspiración de muchas de ellas ha sido la naturaleza. En los últimos años se han popularizado algoritmos basados en las colonias de hormigas [11], los patrones de vuelo de las aves [13], el proceso de selección natural que rige la evolución de las especies [18, 25, 40, 27], e incluso las normas de comportamiento de los grupos sociales [37].

Pero sin importar si nuestra meta última es la búsqueda de máquinas inteligentes, o la obtención del algoritmo de encriptamiento de datos más poderoso sobre la Tierra, de una cosa no queda duda: la computación durante el nuevo milenio seguirá siendo una disciplina sumamente activa en cuanto a investigación y será, más que nunca, una parte importante de nuestras vidas.

Referencias

[1] L. M. Adleman. Molecular Computation of Solutions to Combinatorial Problems. *Science*, 266:1021–1024, November 1994.

[2] Rutherford Aris, H. Ted Davis, and Roger H. Stuewer, editors. *Springs of Scientific Creativity. Essays on Founders of Modern Science.* University of Minnesota Press, Minneapolis, USA, 1983.

[3] F. Gareth Ashurst. *Pioneers of Computing.* Frederick Muller Limited, London, 1983.

[4] William Aspray. *John von Neumann and the Origins of Modern Computing.* The MIT Press, 1990.

[5] E. T. Bell. *The Development of Mathematics.* McGraw-Hill, New York, 1945.

[6] Dan Boneh, Christopher Dunworth, and Richard J. Lipton. Breaking DES using a molecular computer. In *Proceedings of DIMACS workshop on DNA computing*, Princeton, New Jersey, April 1995.

[29] La definición del término "heurística" ha variado con el paso del tiempo, como bien lo indican algunos autores [39]. Actualmente, el término suele usarse para referirse a técnicas usadas para aproximar la solución de un problema (generalmente complejo), en menor tiempo que las técnicas convencionales. No puede garantizarse optimalidad de las soluciones obtenidas por una heurística, ni tampoco el que funcione todo el tiempo, pero en promedio, al menos, suelen producir mejores resultados que las técnicas tradicionales de búsqueda y optimización.

[7] Alonzo Church. An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics*, 58:345–363, 1936.

[8] Carlos A. Coello Coello. Colossus: El Secreto mejor guardado por los ingleses durante la Segunda Guerra Mundial. *Soluciones Avanzadas*, 7(69):3–4, Mayo 1999.

[9] Louis Couffignal. *Sur l'analyse mécanique: Application aux machines à calculer et aux calculs de la mécanique céleste.* PhD thesis, Faculté de Sciences de Paris, 1938.

[10] Martin Davis, editor. *The Undecidable.* Raven Press, Hewlett, New York, 1965.

[11] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B*, 26(1):29–41, 1996.

[12] K. E. Drexler. Molecular Engineering: An Approach to the Development of General Capabilities for Molecular Manipulation. In *Proceedings of the National Academy of Science*, pages 5275–5278, September 1981.

[13] R. C. Eberhart and J. Kennedy. Particle Swarm Optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Press, Perth, Australia, 1995.

[14] Richard P. Feynman. There's Plenty of Room at the Bottom. *Engineering and Science*, 23:22–36, 1960.

[15] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7):467–488, 1982.

[16] Richard P. Feynman. Quantum mechanical computers. *Foundations of physics*, 16:507–531, 1986.

[17] Michael J. Flynn. Parallel Processors Where the Future ... May Yet Be. *Computer*, 29(7):151–152, December 1996.

[18] Lawrence J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley, 1966.

[19] Neil Gershenfeld and Isaac L. Chuang. Quantum Computing with Molecules. *Scientific American*, pages 66–71, June 1998.

[20] Kurt Gödel. Über formal unentscheidbare sätze der *principia mathematica* und verwandter Systeme. *Monatschafte für Mathematik und Physik*, 38:173–198, 1931.

[21] Herman H. Goldstine. El Papel de John von Neumann en el Campo de las Computadoras. In Rutherford Aris, H. Ted Davis, and Roger H. Stuewer, editors, *Resortes de la Creatividad Científica. Ensayos sobre fundadores de la ciencia moderna*, chapter XII, pages 273–289. Fondo de Cultura Económica, México, 1989.

[22] Herman H. Goldstine. *The Computer from Pascal to von Neumann*. Princeton University Press, Princeton, New Jersey, 1993.

[23] Lov K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the 28th Annual Symposium on Theory of Computing*, pages 212–219, New York, 1996. ACM Press.

[24] W. D. Hillis. *The Connection Machine*. MIT Press, Cambridge, Massachusetts, 1985.

[25] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Harbor, 1975.

[26] Stephen Cole Kleene. *Introduction to Metamathematics*. D. Van Nostrand, Princeton, New Jersey, 1952.

[27] John R. Koza. *Genetic Programming. On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, Massachusetts, 1992.

[28] J.A.N. Lee. *Computer Pioneers*. IEEE Computer Society Press, Los Alamitos, California, 1995.

[29] Norman Macrae. *John von Neumann*. Cornelia & Michael Bessie Book, New York, 1992.

[30] J. Markoff. Has Size of Chips Reached Its Limits. San Jose Mercury News, 9 October 1999.

[31] Warren Sturgis McCulloch and Walter H. Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[32] Christine Peterson. Taking Technology to the Molecular Level. *Computer*, 33(1):46–52, 2000.

[33] John R. Pierce. Shannon, Claude. In Bryan Ralston and Edwin D. Reilly, editors, *Encyclopedia of Computer Science*, pages 1196–1197. Van Nostrand Reinhold, New York, third edition, 1993.

[34] Emil Leon Post. Finite Combinatory Processes. Formulation I. *Journal of Symbolic Logic*, 1:103–105, 1936.

[35] Michael O. Rabin and Dana S. Scott. Finite Automata and their Decision Problems. *IBM Journal of Research and Development*, 3:114–125, 1959.

[36] Brian Randell. *The Origins of Digital Computers. Selected Papers.* Springer-Verlag, Berlin, 1973.

[37] Robert G. Reynolds. An Introduction to Cultural Algorithms. In A. V. Sebald, , and L. J. Fogel, editors, *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 131–139. World Scientific, River Edge, New Jersey, 1994.

[38] Bertrand Russell. *The Principles of Mathematics.* Cambridge University Press, Cambridge, UK, 1903.

[39] Stuart Russell and Peter Norvig. *Artificial Intelligence. A Modern Approach.* Prentice Hall, 1995.

[40] Hans-Paul Schwefel. *Numerical Optimization of Computer Models.* John Wiley & Sons, Great Britain, 1981.

[41] Dennis Shasha and Cathy Lazere. *Out of their Minds. The Lives and Discoveries of 15 Great Scientists.* Copernicus, 1995.

[42] Peter W. Shor. Polynomial–Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[43] Joel Shurkin. *Engines of the Mind. The Evolution of the Computer from Mainframes to Microprocessors.* W. W. Norton & Company, New York, 1996.

[44] Moshe Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach.* Springer-Verlag, Heidelberg, 1997.

[45] Moshe Sipper. The Emergence of Cellular Computing. *Computer*, 32(7):18–26, 1999.

[46] Robert Slater. *Portraits in Silicon.* The MIT Press, Cambridge, Massachusetts, 1992.

[47] Andrew M. Steane and Elanor G. Rieffel. Beyond Bits: The Future of Quantum Information Processing. *Computer*, 33(1):38–45, 2000.

[48] Alan Mathison Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[49] Alan Mathison Turing. Proposals for the development in the Mathematics Division of an Automatic Computing Engine (ACE). Technical Report E882, Executive Committee, NPL, Teddington, Middlessex, 1946.

[50] Alan Mathison Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[51] R. L. A. Valtat. Calcul mécanique: Machine à calculer fondée sur l'emploi de la numération binaire. *Compt. Rend. Acad. Sci. Paris*, 202:1745–1748, 1936.