Elogio de la pereza

Una perspectiva histórica de la computación

José Galaviz Casas

Dedicatoria

Para:

Marcela (mi madre),
Mónica (mi esposa),
Carlos Arturo y Claudia Eréndira (mis hijos).

Índice general

Prefacio

En la edición de 1863 del *Dictionnaire Infernal* de Collin de Plancy se define a Belphegor como el demonio asociado a los descubrimientos y a las invenciones ingeniosas, pero también es el demonio encargado de apartar a los hombres del buen camino invitándolos a la pereza, uno de los siete pecados capitales. La ilustración creada por Luis Breton para la obra de Plancy, muestra a Belphegor como un demonio cornudo y holgazán: plácidamente sentado con un brazo completamente relajado y con la mano del otro cogiéndose la cola, en una actitud que nos evoca, salvo por el tono muscular, a *El Pensador* de Auguste Rodin (véase la figura 0.1).

Nos da pereza hacer labores de las que no obtenemos retribución espiritual alguna, no nos divierten, no descubrimos nada nuevo, no constituyen un reto, al terminarlas seguimos siendo los mismos que éramos al comenzarlas, sólo que más aburridos. Muchas de estas tareas se caracterizan por ser repetitivas, en las que hay que hacer lo mismo una y otra vez y en las que una pequeña distracción hace que tengamos que repetir el proceso completo. Cuando hay que hacer una tarea así nos gustaría dejar que alguien más la hiciera: siempre es mejor que alguien más lave los trastes de la comida, o limpie el piso, o lave la ropa. Nuestra pereza para hacer estas labores ha exacerbado nuestro ingenio; en el mejor de los casos las tareas aburridas pueden ser hechas por algo, ya no por alguien, así hemos inventado artefactos que trabajan en nuestro lugar: la lavadora de trastes, la aspiradora, la lavadora automática de ropa, etc. El demonio de la pereza es también el de la invención y del descubrimiento.

Una labor que cumple con todos los requisitos para dar pereza a cualquier persona saludable, es la de hacer cálculos aritméticos. Hacer "cuentas" es siempre aburrido y en nuestra pereza por hacerlas hemos inventado, a lo largo de la historia, métodos y auxiliares diversos que nos permiten hacerlas con facilidad y finalmente hemos creado máquinas capaces de hacer la tediosa labor por nosotros. Pero lo más importante es que en el camino hemos tenido que hacernos preguntas y buscar respuestas, hemos tenido que definir

Figura 0.1: Ilustración de Belphegor en el *Dictionnaire Infernal* de 1863.

lo que significa calcular y que algo sea calculable; clasificar los problemas que pretendemos resolver de acuerdo a su grado de complejidad y hemos descubierto algunos que ni siquiera tienen solución. En resumen, el engañoso camino de Belphegor nos llevó a crear una ciencia.

Generalmente los libros que versan sobre la historia de la computación se enfocan exclusivamente, en su aspecto tecnológico y hacen un recuento de los artefactos que la humanidad ha creado para calcular, desde el ábaco o la máquina de Antikythera, hasta las modernas computadoras electrónicas. Este enfoque deja de lado el aspecto que podemos llamar científico: las ideas que dieron origen, tanto a los artefactos, como a los desarrollos puramente abstractos vinculados con el cálculo y lo calculable. Es precisamente a esto último a lo que se dedica este trabajo.

Se ha elegido este enfoque en un intento por desterrar de la mente del lector la difundida idea de que el objeto de estudio de la computación, como disciplina científica, son las computadoras mismas[1], cuando éstas son realmente una de las consecuencias y no la causa de la computación.

[1]Véase por ejemplo la errónea definición contenida en *Webopedia* (`http://www.webopedia.com/TERM/c/computer_science.html`): "El estudio de las computadoras, incluyendo el diseño de *hardware* y *software*".

Sin embargo, a pesar del énfasis en lo abstracto, no se ha excluido de este trabajo el desarrollo de las tecnologías de cómputo (imposible no mencionar las máquinas de Pascal, Leibniz o la ENIAC), pero procurando siempre justificarlas en el contexto de las ideas que les dieron origen.

Debo agradecerle a muchas personas su participación en la realización de este trabajo: a Ángel Kuri y Elisa Viso por su lectura de las versiones preliminares, sus sugerencias y correcciones y por ser siempre un invaluable apoyo moral y un ejemplo; a Mario Magidin por sus valiosas sugerencias bibliográficas; a Guadalupe Ibargüengoitia por su apoyo y "marketing", a Ana Irene Ramírez, César Guevara y Mercedes Perelló por su excelente y ardua labor corrigiendo y haciendo legible la obra y por su apoyo entusiasta; a Silvia Torres por su asesoría técnica en cuanto a la calidad de las imágenes.

De manera especial, deseo agradecer a mi esposa Mónica Leñero sus correcciones, comentarios, la paciencia con la que escuchó muchos de mis monólogos acerca de lo que escribía, la abnegación con la que soportó ser viuda virtual durante muchas noches en las que me dediqué a leer y escribir a propósito de este trabajo y su estoica y frecuente labor de hacerse cargo sola de nuestros hijos. Agradezco también a estos últimos: Claudia Eréndira y Carlos Arturo, por dejar en paz (a veces) el teclado y el botón de reinicio de la computadora y haber sido pacientes (a veces) esperando su turno para usarla.

Quiero también agradecer a muchas otras personas su participación indirecta, pero también indispensable. A las premisas fundamentales a quienes nunca pude agradecerles debidamente: Marcela Casas, Carlos Casas López y Fabiana Herviou. A mi familia natural, mi familia adoptiva y buenos amigos: Carlos, Estela y Alberto Casas, Martha y Rosendo Cacho, Estela Padierna, José G. Villamil, Fernando Leñero, Susana Barrera, Claudia Escobar, Arturo Pi, Carlos Rivera, Jesús Gutierrez, Karla Ramírez, Francisco Solsona, Iván Hernández, Arturo Vázquez, Manuel Sugawara, Mauricio Aguilar, Mauricio Aldazosa, Fernanda Sánchez, Gildardo Bautista, Patricia Valle, Adriana García de la Cadena y Antonio Arrollave; por construir cada uno la parte del mundo que le tocó y en cuya intersección tengo la fortuna de vivir.

José Galaviz
Ciudad Universitaria, México D.F.
Octubre de 2003.

1

¿Qué es la ciencia de la computación?

Estás en una fiesta familiar, de esas grandes, como cuando cumple quince años o se casa una prima. Hay cientos de personas que no conoces, jamás las has visto, sin embargo son miembros de la familia, o amigos, o amigos de amigos. Todos ríen, los niños corren y tiran la comida, los tíos están hablando de temas profundos al borde de la ebriedad, brindando con sus vasos de plástico. De pronto tu madre, para tu disgusto, te llama para que saludes a uno de tus tíos. Él es contador, cosa evidente, porque todo el mundo puede leer, a varios metros de distancia, la inscripción del anillo de oro que abarca la totalidad de la falange de su dedo anular. Desde hace algunos años trabaja con computadoras y se ha erigido como el experto en computación de la familia, cargo que ejerce frecuentemente, prodigando consejos no solicitados a diversos miembros de su coto de poder. Como de costumbre tu madre le ha hablado de ti, de lo que estudias y de lo buen hijo que eres, y cuando te aproximas te deja solo a merced de él. Es en momentos como ese que dudas de haber sido un hijo deseado. Tu tío, luego de confirmar que estudias computación, y de señalar repetidamente que "es muy bonita la computación", te pregunta, como para probarte, si sabes hacer algo con una popular hoja de cálculo. Por supuesto el término usado para especificar la tarea a que hace referencia, es una extraña combinación, de un verbo en inglés con la terminación en el infinitivo del español ("pastear", por ejemplo). Tu dices que no sabes, luego viene el escarnio. Tu tío expresa sus dudas acerca de tu calidad como estudiante, de la calidad de tus profesores y de la institución en la que estudias, sentencia sabiamente que no tendrás oportunidad en el mundo real (por definición el mundo real es en el que él vive), si no estudias

cosas prácticas. No tiene sentido tratar de refutarlo, te pasa por la mente la idea de hablarle de lo interesantes que son las máquinas de Turing y él te diría que esas son tonterías, que no hay nada más poderoso que su nueva máquina Pentium 4, tu sonríes, sabes que en el fondo, ambas máquinas son igualmente poderosas, asi que mejor decides hablar de deportes.

Cuando uno estudia computación, desde el punto de vista científico o de la ingeniería, suelen ocurrir situaciones similares a la descrita. En general no hay entre las personas un concepto claro de los objetivos e importancia de las ciencias. Es común encontrarse a personas que piensan que un matemático "es bueno para hacer cuentas", cuando en realidad la gran mayoría de los matemáticos, incluyendo algunos verdaderamente excepcionales, son bastante ineptos para dividir la cuenta del restaurante entre el número de comensales. Del mismo modo, el común de las personas cree que el objeto de estudio de la ciencia de la computación son las computadoras, lo que equivaldría a afirmar que el objeto de estudio de la astronomía son los telescopios. Sin embargo este error de concepto está tan difundido, que incluso se pueden encontrar libros cuyo título original en inglés contiene los términos *computer science* y que exhiben en español la frase *ciencia de las computadoras* y no *ciencia de la computación* como debería ser.

Algunos piensan que el error mencionado puede ser corregido eliminando, del nombre en inglés de la disciplina, la palabra *computer* (computadora), usada para designar al aparato, reemplazándola por alguna otra como "información", o cambiando todo por "informática". Esto evidentemente no elimina la verdadera enfermedad, a saber, el concepto erróneo, sólo palia los síntomas. Para determinar cuál es el objeto de estudio de la ciencia de la computación debemos definir primero que es *computar*.

Computar es sinónimo de *calcular* que es, sin duda, una palabra mejor entendida, todo mundo tiene una idea más o menos clara de qué es hacer un cálculo. Pero para no correr riesgos es bueno recurrir al diccionario: "Cálculo. *mat.* conjunto de procedimientos matemáticos que permiten operar con números o símbolos numéricos, así como resolver problemas relacionados con dichas operaciones. *med.* cuerpo más o menos sólido, compuesto generalmente de sales minerales que se forma anormalmente en el organismo".

Por extraño que parezca nos abocaremos primero a la última de las acepciones del cálculo. Ésta se refiere a esas piedrecillas que ocasionalmente se forman en los riñones o en la vesícula. Los conocidos como cálculos renales o biliares respectivamente. Se llaman cálculos porque, como suele ocurrir en medicina, hay una palabra latina relacionada: *calculus* que significa piedrecilla o guijarro, de allí el nombre dado a las piedrecillas de los riñones.

La relación entre los cálculos renales o biliares y los matemáticos es producto de una coincidencia. Ocurrió que para facilitar el hacer operaciones aritméticas, como sumas y restas, los romanos usaban un ábaco hecho de metal en el que las cosas que se cuentan se representaban mediante pequeñas piedrecillas, es decir, *calculus*. Aquí entramos en contacto con uno de los dispositivos más antiguos que utilizó el hombre para calcular: el ábaco. Normalmente cuando se habla de historia de la computación se hace un recuento de todos estos dispositivos físicos que se han usado para "hacer cuentas"; quizás sea esto lo que ocasiona, en parte, la confusión que tratamos de esclarecer. La historia de la computación no debería verse como la historia del desarrollo de estos dispositivos, en realidad es algo un poco más general y más abstracto.

Entonces consideremos que computar es, en un sentido numérico, calcular, hacer cuentas, operaciones con números. En un sentido un poco más general es pasar una colección de datos a través de un proceso que finalmente nos proporciona la solución de un problema. Por ejemplo, ordenar alfabéticamente una lista de personas es un cierto tipo de cómputo.

La historia de la computación, en términos generales, podríamos considerarla una historia de la pereza. Todos somos en buena medida perezosos. Si hay que hacer un trabajo tedioso, repetitivo y con tendencia a dejar frustrado al ejecutante, toda persona sensata tratará de evitarlo. Ese es y ha sido siempre el caso de los cálculos matemáticos, ¿quién no recuerda con horror las decenas interminables de sumas, restas, multiplicaciones y divisiones (de tres cifras con punto decimal dentro de la "casita"), que había que padecer en la primaria? No es vergonzoso admitirlo, hasta los grandes intelectuales han reconocido esa ominosa cualidad de los cálculos. Leibniz escribió: "No es admisible que los estudiosos y científicos, en lugar de elaborar y confrontar nuevas técnicas, pierdan su tiempo como esclavos de las fatigas del cálculo, cuando esto podría ser delegado confiablemente a cualquier otro".

Pero ¿podemos no hacer cuentas?, si tan horribles son ¿no podemos evitarlas y dejarlas sólo como un castigo del purgatorio? No, en buena medida la humanidad ha sobrevivido porque hubo personas que hace milenios, se sacrificaron haciendo los cálculos necesarios para determinar el periodo de tiempo óptimo para sembrar y cosechar. Eso sólo por mencionar un ejemplo sencillo.

Así que hacer cuentas es necesario aunque es horrible y por eso a lo largo de su historia los seres humanos han tratado, valiéndose de diversos medios, de hacer los indispensables cálculos matemáticos de la manera más rápida y fácil posible. Si además estos cálculos pudiesen delegarse a un subordinado lo suficientemente capaz, como hubiera querido Leibniz, mejor.

Cuando estaba aprendiendo las tablas de multiplicar iba a que mi madre me preguntara la tabla que estuviera memorizando en ese momento. Mi madre misma me había explicado que la multiplicación realmente es una suma abreviada, 4 por 5 significa la suma de 4 consigo mismo cinco veces. Así que, mañosamente, fui a que mi madre me preguntara la tabla del 6, mi trampa consistía en que, una vez que tenía el resultado de $6 \times i$, entonces a ese resultado le sumaba 6 para obtener $6 \times (i + 1)$. Esto me ahorraba el esfuerzo de tener que memorizar las tablas, pero resultaba demasido lento, sobre todo cuando la tabla no era preguntada en la secuencia natural. Así que mi truco fracasó, esencialmente porque tenía que pensar demasiado en vez de ejecutar un proceso automático. El objetivo de memorizar las tablas es no tener que pensar en la definición de multiplicación cada vez que se hace una, sino tener interiorizado algo que agiliza las operaciones, tal y como el conducir un auto se convierte en un proceso automático. El conductor experimentado no piensa en lo que está haciendo, de hecho puede estar pensando en otra cosa totalmente ajena al automóvil que conduce. "Es un tópico profundamente erróneo, repetido en todos los libros escolares y por personas eminentes en sus discursos, que deberíamos cultivar el hábito de pensar en lo que estamos haciendo. Lo cierto es precisamente lo contrario. La civilización avanza ampliando el número de operaciones importantes que podemos realizar sin pensar en ellas", escribió Whitehead.

Así, cada vez que hacemos una multiplicación, lejos de remitirnos a la definición, echamos mano de nuestras tablas ya interiorizadas y de algo más: *un método*. Desde nuestra más tierna infancia nos enseñaron un método para hacer multiplicaciones de números de varias cifras. Este método está garantizado: si lo aplicamos correctamente, es seguro que obtenemos el resultado correcto. Sólo hay que seguir ordenadamente una serie de pasos que siempre son los mismos sin importar qué multiplicación deseemos efectuar en particular. El método, al igual que las tablas, debe ser memorizado e interiorizado para hacer automática su aplicación, esa es, de hecho, la intención de los cientos de ejercicios que debíamos padecer en la enseñanza elemental. El método que nos enseñaron para multiplicar, es sólo uno entre muchos que se han inventado y utilizado a lo largo del tiempo.

Para hacer cuentas más rápido y mejor, inventamos cosas que nos hacen pensar menos, de allí las tablas y los métodos de multiplicar de la enseñanza elemental. Entre menos haya que pensar para hacer un cálculo, mejor. Llevando esto al extremo, llegamos a los "esclavos confiables", a los que Leibniz desearía haber encargado los cálculos: a nuestras computadoras. Ahora alguien o algo "piensa" por nosotros; para calcular ya no hay que pensar en absoluto.

Esto es la historia de la computación: el largo camino que ha recorrido el género humano buscando métodos que le permitan hacer cálculos de manera abreviada, casi automática, y como siguiente paso, de manera natural, los mecanismos físicos que hagan los cálculos por él. Pero en una aventura nada está garantizado y en esta aventura intelectual la humanidad ha arribado a la incertidumbre, que es la madre de la ciencia. El último trecho de este camino ha sido marcado por preguntas como: ¿hay cosas que no se puedan calcular? ¿qué se puede calcular y qué no? ¿hay cosas más difíciles de calcular que otras? ¿podemos clasificar problemas de acuerdo a su complejidad? No nos queda más que seguir caminando, buscando respuestas a las preguntas que van surgiendo, no faltarán los tropiezos, pero eso, como todo buen explorador sabe, es parte de la diversión implícita en toda aventura.

2

Contar y calcular

Antes de calcular hay que saber contar y ésta no es una tarea trivial. Hay pueblos que sólo pueden contar cantidades muy pequeñas (desde nuestro punto de vista). Los bosquimanos del Kalahari en África no pueden contar más allá de cinco y hay algunos grupos humanos primitivos que no pueden contar más allá de dos; cualquier cantidad por arriba de ésta se expresa usando una palabra con el sentido de "muchos", aunque normalmente utilizan los dedos de las manos para resolver ambigüedades hasta una decena. Los bosquimanos no poseen palabras más que para "uno" y "dos", cualquier cantidad mayor (hasta el cinco, por supuesto) se expresa como una yuxtaposición de estas: "dos-uno", "dos-dos" y "dos-dos-uno". Los aborígenes aranda de Australia hacen lo mismo, para ellos "ninta" es uno y "tara" es dos, así que "tara-ma-ninta" es tres y "tara-ma-tara" es cuatro, el número más grande expresable en palabras. Se alcanza a distinguir que en los orígenes de la civilización occidental ocurrió algo similar, la palabra latina *tres* es muy afín a *trans* ("más allá"); en francés, lengua romance, *très* es "muy" o "mucho".

Quizás tanto o más importante que el contar sea el desarrollar un sentido abstracto de número. En algunas lenguas (por ejemplo el japonés y algunas de los nativos de norteamérica) hay distintas palabras para contar diferentes cosas. Así que el cuatro que se utiliza para decir "cuatro peces", por ejemplo, no es el mismo que el que se utiliza para decir "cuatro pájaros". Esto significa que el número es más una etiqueta asignada al grupo de cosas que se desea contar que un concepto abstracto, un ente ideal que se puede poner en correspondencia con diferentes objetos del mundo real. Posiblemente el germen de este proceso se encuentra en las diferentes ayudas de que se

han valido los seres humanos para contar. Cuando el pastor saca a pastar su ganado, coloca en un saco una piedrecilla o un grano de maíz por cada cabeza de ganado, al regresar el hato a su corral comprueba que no se ha extraviado ningún animal poniendo nuevamente en correspondencia el conjunto de piedrecillas o granos con el conjunto de animales. Sólo hace falta un paso de abstracción para pasar de las piedrecillas al concepto ideal de número.

Una vez que ya se puede contar, resulta necesario poder expresar las cantidades por escrito de manera que sea posible manipularlas, es necesario entonces crear un *sistema numérico*. A lo largo de la historia han surgido dos tipos de sistemas: los aditivos y los posicionales.

El ejemplo clásico de sistema aditivo es el que utilizaron los romanos originalmente. Es conveniente aclarar que las modificaciones sustractivas al sistema romano puramente aditivo y que fueron creadas para abreviar la expresión de ciertos números como el cuatro (IV), el nueve (IX) y el noventa (XC), nunca se usaron con propósitos operativos y no se popularizaron hasta principios del siglo XVII, cuando los números romanos ya no eran usados para hacer cálculos.

En el sistema romano original no es tan difícil hacer operaciones (como ocurre en el caso del sistema romano con modificaciones sustractivas), pero sí es lento. Para hacer una suma, por ejemplo, es necesario efectuar dos pasos para obtener el resultado final: si se quieren sumar 3769 y 347, primero se suman (mejor dicho, se acumulan) directamente los símbolos de cada tipo como se muestra en la figura 2.1 y luego se procede a simplificar la expresión obtenida usando el hecho de que IIIII=V, VV=X, XXXXX=L, LL=C, CCCCC=D y DD=M.

En los sistemas numéricos posicionales es un poco más fácil hacer operaciones. Que un sistema numérico sea posicional significa que, como aprendimos en la enseñanza elemental respecto a nuestro sistema numérico decimal usual, cada dígito de un número posee un valor dependiendo de su posición dentro del número. El valor de cada dígito es un múltiplo de una potencia de cierta base. En nuestro sistema decimal (de base diez), "345" significa "tres centenas, cuatro decenas y cinco unidades" o equivalentemente:

$$345 = 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$$

La elección de la base es arbitraria y determina la cantidad de símbolos necesarios para expresar cualquier dígito. En base diez, por ejemplo, se poseen diez símbolos para los dígitos: "0", "1", "2", "3", "4", "5", "6", "7", "8" y "9". En base dos en cambio sólo se requieren dos símbolos: "0"

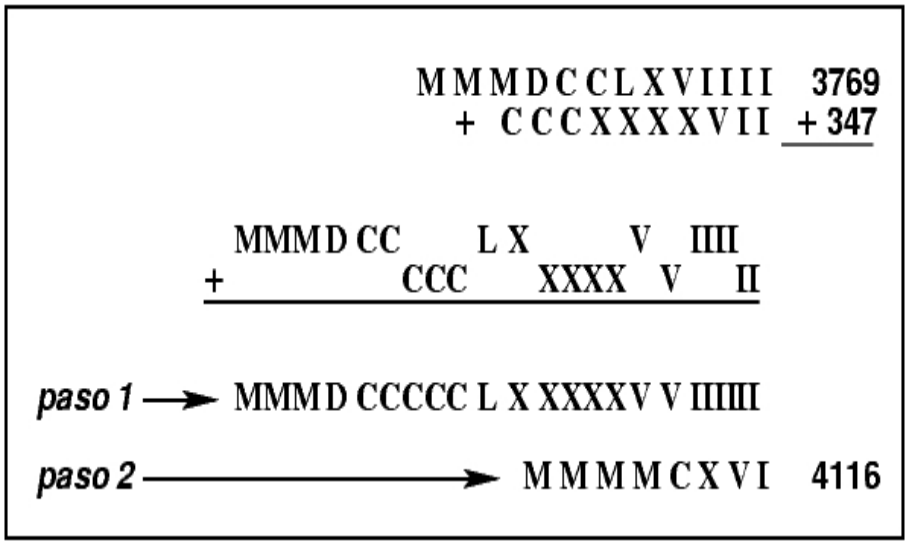

Figura 2.1: Suma de 3769 y 347 en el sistema romano antiguo.

y "1"[1]. Los babilonios extrañamente prefirieron el 60 como base para su sistema numérico, los chinos, igual que los indios, el 10; los mayas, como algunos pueblos celtas de Europa central prefirieron el 20, posiblemente porque para contar utilizaban todos los dedos disponibles en el cuerpo. Es posible rastrear aún el uso de la base 20 en el centro de Europa observando que en francés 80 se dice "quatre vingt" es decir "cuatro veinte" o cuatro veces veinte. El sistema que se utiliza actualmente en las computadoras es el *binario*, es decir con base 2. En este sistema los dígitos son solamente el cero y el uno. Por ejemplo el número 1110 en binario equivale a 14 en decimal:

$$1110_2 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 8 + 4 + 2 = 14_{10},$$

aquí hemos utilizado el subíndice para indicar la base en la que está expresado el número.

A mediados del siglo VIII los árabes estaban en posesión de un enorme imperio que abarcaba desde la parte media de la península ibérica hasta el

[1]Por supuesto los símbolos de los dígitos son también elegidos arbitrariamente. En el sistema hexadecimal, usado frecuentemente en computación, se ha optado por representar los dígitos con valores de 10, 11, 12, 13, 14 y 15 con las letras "A", "B", "C", "D", "E" y "F" por ejemplo, dado que se requieren dieciséis símbolos para los dígitos y los símbolos del sistema indo-arábigo sólo son diez.

valle del Indo, pasando por el norte de África. Como suele suceder en estos casos, luego del intercambio de violencia, se da un intercambio cultural. Durante el reinado de tres califas, Al-Mansur, Al-Raschid y Al-Mamun, la ciudad de Bagdad se hizo célebre, no por un ladrón, sino porque en ella se reunía la sabiduría importada por los árabes de muy diversas partes del imperio. Mientras Europa se sumía en la ignorancia y el subdesarrollo cultural olvidando la herencia griega y romana, los árabes se encargaron de rescatarla. La conjugaron con la propia, con la persa, la siria, la mesopotámica y la judía. Al-Mamun estableció en Bagdad la Bait al-hikma o "casa de la sabiduría", un símil de lo que fue la biblioteca de Alejandría: un centro de estudio y acervo de conocimientos. En la casa de la sabiduría fue donde Mohammed, el hijo de Moisés, originario del pueblo de Khowarizmi (Mohammed ibn Musa Al-Khowarizmi), conoció las matemáticas y la astronomía indias. Mohammed de Khowarizmi, al que para abreviar llamaremos Al-Khowarizmi, escribió algunos libros acerca de lo aprendido de los indios, en particular un libro llamado *Al-jabr wa'l muqabalah* que dio origen al álgebra y otro de aritmética donde se describía el sistema numérico indio y los métodos utilizados para hacer cálculos en ese sistema. Por desgracia la casa de la sabiduría, al igual que la biblioteca de Alejandría, fue destruida por Hulagu Khan (el nieto de Genghis Khan) en 1258 y no poseemos ningún ejemplar original de la aritmética de Al-Khowarizmi escrito alrededor del año 850.

En la Europa medieval, quien quisiera realmente tener una buena educación y acceder al conocimiento universal, debía estudiar bajo la tutela de "los infieles" árabes. Eso fue precisamente lo que hizo Gerbert, un aquitano de nacimiento que recibió su primera instrucción en la abadía de Aurillac y que más tarde se mudó a España donde entró en contacto con la ciencia árabe y por tanto con la aritmética y los procedimientos de cálculo usados por esa cultura. Gerbert fue uno de los primeros en estimular el uso de la notación indo-arábiga y perfeccionó el ábaco, de uso común en Europa como un mecanismo que permitía evadir las dificultades para hacer operaciones con la notación romana. Fue instructor en diversas instituciones educativas y más tarde fue electo Papa en 999 con el nombre de Silvestre II.

Un monje inglés del siglo XII, Adelardo de Bath, que recibió instrucción en la Córdoba ocupada por los árabes, tuvo acceso a la obra de Al-Khowarizmi y se dio a la tarea de traducirla del árabe a una lengua más difundida entre los hombres europeos "de razón". Así surgió un libro cuyas primeras palabras traducidas a nuestro idioma y terminología serían: "Ha dicho Al-Khowarizmi..." y que en el original en latín aparecen como: "Dixit Algorismi...", de donde proviene la palabra y el concepto de *algorismo* como

un método para hacer cálculos en el sistema indo-arábigo y *algoritmo*, un concepto actualmente más general al que nos referimos cuando hablamos de métodos que nos permiten resolver problemas en un tiempo finito.

Así fue como la cultura occidental entró en contacto con la notación indo-arábiga, que además posee la virtud de tener un símbolo específico para denotar la ausencia de ciertas cantidades: el cero. Viendo las cosas en retrospectiva, la invención del cero podría parecer una extensión obvia de cualquier sistema posicional, sin embargo muchas civilizaciones cultas pasaron de largo frente al cero sin percatarse de su conveniencia. En parte esto se debió a que muchos sistemas numéricos no fueron *estrictamente* posicionales, los babilonios, por ejemplo, sólo dejaban un hueco en vez de poner un símbolo explícito, esto hace que la expresión de un número sea ambigua. En el sistema babilonio con frecuencia no se puede distinguir un $1 \times 60^2 + 0 \times 60 + 1$ de un $1 \times 60 + 1$.

La civilización maya poseía, para fines astronómicos y cronológicos, un eficiente sistema numérico posicional en base 20 (salvo una pequeña violación que se hará evidente en seguida) con el símbolo y el concepto del cero. Los días, que en maya se denominan *kines*, eran la unidad más baja, 20 kines forman un *uinal*; 18 uinales constituyen un *tun* (esta es la violación mencionada al principio de sistema posicional, dado que se agrupan 18 y no 20 uinales); 20 tunes son un *katun* y 20 de estos hacen un *baktun*. La violación en la definición de tun se debe a que 360 es el múltiplo de 20 más próximo al número de días en un año. En la figura 2.2 se muestra el número 11529 escrito en maya, nótese que el número se escribe verticalmente.

Hacer cálculos con el sistema indo-arábigo era mucho más fácil que hacerlos en el sistema romano. Ya hemos dicho que para suplir las carencias operativas de la notación romana se utilizaban ábacos. Así que surgió el conflicto entre aquellos que utilizaban el ábaco y aquellos que promovían la utilización del sistema indo-arábigo para hacer cálculos. Sin embargo para hacer cuentas en el nuevo sistema se requería de papel para poder escribir las cantidades y obtener resultados parciales o totales, ahora casi cualquier persona dispone de un pequeño trozo de papel, pero no era el caso en la Europa medieval. Así que, a pesar de ser un método eficaz para calcular, el uso del sistema y métodos indo-arábigos no se popularizó hasta el siglo XVI cuando el papel se hizo suficientemente fácil de producir y de adquirir.

Sin embargo, durante los siglos XIV, XV y XVI, como parte del proceso de adopción del sistema indo-arábigo, se desarrollaron muchos algoritmos (métodos) diferentes para hacer cuentas. De esta época datan algunos de nuestros métodos actuales para multiplicar, restar y dividir; el de sumar es más antiguo, data de la época en que se inventó, en India, el sistema de

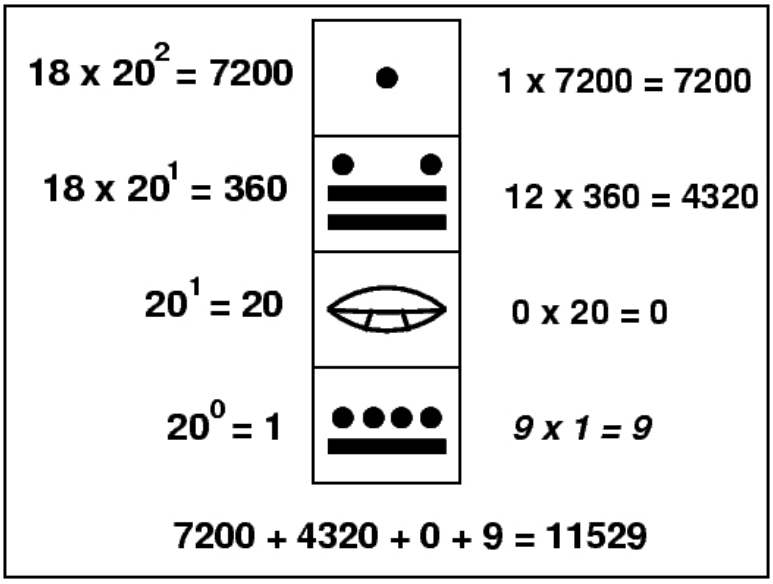

Figura 2.2: El número 11529 escrito en sistema maya. A la izquierda de la expresión maya aparece el valor de cada estrato, a la derecha el valor explícito en la expresión.

numeración indo-arábigo. Por supuesto, esto no quiere decir que no hubiera métodos anteriores. En Egipto se desarrolló, por ejemplo, un método para multiplicar tan útil que aún lo utilizan nuestras modernas computadoras, el de *duplicación y mediación*. Tomamos dos números, por ejemplo 24 y 38, el más grande lo partimos a la mitad tantas veces como sea necesario hasta obtener 1 y ese mismo número de veces duplicamos el más pequeño. Luego sumamos aquellas cantidades de la columna de duplicaciones que corresponden a cantidades impares en la columna de las mediaciones (ver fig. 2.3). Este método es el que usan nuestras computadoras porque, como es de dominio popular, nuestras computadoras trabajan en binario, es decir, todos los datos manipulables por la computadora y susceptibles de ser almacenados en su memoria son números en notación posicional binaria y en esta notación es particularmente fácil duplicar y "mediar" números.

Cuando en nuestro sistema decimal usual queremos multiplicar por 10 una cantidad entera simplemente le agregamos un cero a la derecha. Cuando queremos dividir por 10 (si sólo queremos la parte entera del cociente) sólo eliminamos el dígito de la extrema derecha. En binario ocurre lo mismo al multiplicar y dividir por 2. Estas operaciones de agregar ceros o eliminar

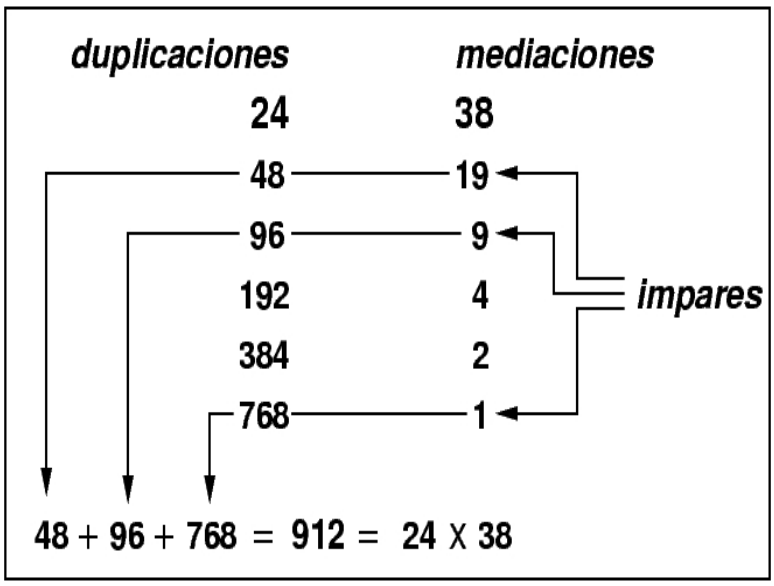

Figura 2.3: Método de multiplicación por *duplicación y mediación*.

dígitos por la derecha son fácilmente ejecutadas por nuestras computadoras a través de lo que se conoce como *desplazamientos*, cada vez que queremos agregar un cero a la derecha simplemente desplazamos el número en cuestión un lugar hacia la izquierda y el hueco lo rellenamos con cero, cada vez que deseamos eliminar el dígito de la derecha desplazamos el número en esa dirección y como el dígito del extremo se "sale", se pierde.

Otro de los primeros promotores del sistema indo-arábigo fue Leonardo de Pisa, mejor conocido como Leonardo Fibonacci[2]. Pisa era una de las ciudades más prósperas de Italia por ser uno de los principales centros comerciales del Mediterráneo, lo que la hacía una ciudad cosmopolita donde era posible encontrar productos y personas provenientes de Egipto, Grecia, Siria, el centro de Europa, el lejano oriente y, por supuesto, árabes. Según la leyenda, Fibonacci entró en contacto con la lengua y aritmética árabes a través de un tendero de la localidad. El hecho es que los comerciantes del siglo XII, como los de hoy en día, hacen cuentas y Fibonacci estaba en el lugar y la época correctos para familiarizarse con los procedimientos aritméticos usados en todo el mundo conocido. Además, Fibonacci incremento su acervo viajando por medio oriente y el centro de Europa, entró

[2]El nombre deriva de la frase "Leonardo el hijo de Bonaccio", lo que en latín se dice *Leonardo filius Bonacci* o, por brevedad, Leonardo Fibonacci.

Figura 2.4: Leonardo de Pisa (Fibonacci).

en contacto con diversos sistemas numéricos y métodos para calcular, los que consideró inferiores a los utilizados por los árabes. Leonardo Fibonacci escribió un libro llamado *Liber Abaci* o *El libro de las cuentas*[3], publicado en 1202 y que versa sobre la notación indo-arábiga y los diversos métodos que se utilizan en ella para hacer cálculos. En los últimos capítulos del libro Fibonacci muestra algunas aplicaciones del novedoso sistema y resuelve algunos problemas aritméticos, uno de ellos, el que le ha ganado fama, es el de los conejos. Supongamos que se coloca una pareja de conejos (macho y hembra) en un prado cercado y que se cumplen las siguientes condiciones:

- Los conejos nunca mueren.
- Los conejos alcanzan la madurez sexual a la edad de un mes.
- En cuanto alcanzan su madurez los conejos se aparean y a partir de ese momento lo hacen cada mes.
- Cada vez que se aparean la hembra queda preñada y da a luz una pareja (macho y hembra), luego de un mes de gestación.

[3]La palabra *abaci* no era usada para referirse al ábaco específicamente, sino al proceso general de hacer operaciones aritméticas.

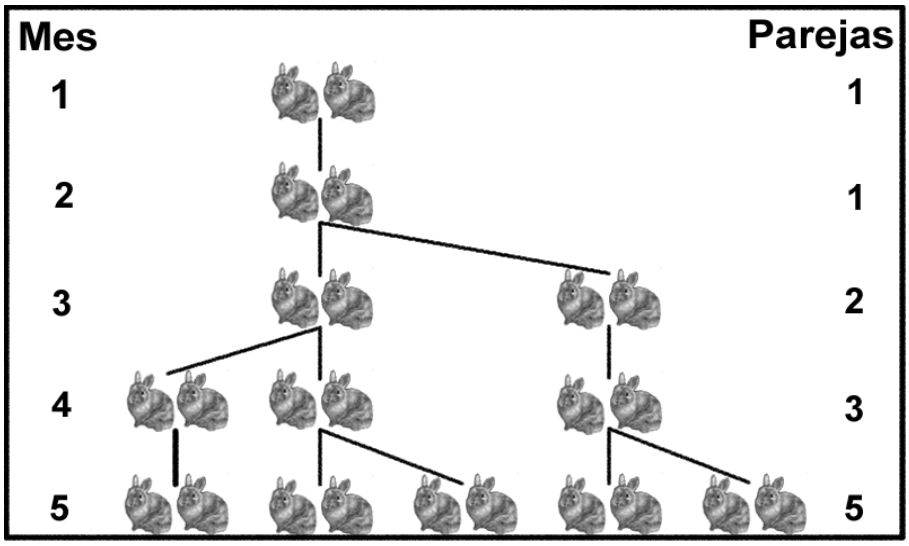

Figura 2.5: Análisis del problema de los conejos de Fibonacci.

La pregunta de Fibonacci era: ¿Cuántas parejas de conejos habrá en el prado luego de un año? Si se analiza el problema de cuántas parejas hay mes con mes lo que se obtiene es la conocida *sucesión de Fibonacci*: 1, 1, 2, 3, 5, etc., donde el i-ésimo término de la sucesión se obtiene sumando los dos términos previos; partiendo de que los dos primeros términos son 1 (véase fig. 2.5), la solución al problema está dada entonces por el doceavo término de la sucesión que resulta ser 144.

El *Liber Abaci* contenía 459 páginas, demasiadas para que llegara a ser un libro popular en 1202 (o 1228 cuando salió la segunda edición). Así que realmente la influencia de la obra fue muy escueta. Fueron dos los trabajos que realmente divulgaron el sistema indo-arábigo en Europa. El primero de ellos es un poema, *Carmen de Algorismo* de Alexander de Villa Dei, escrito alrededor de 1220. Este poema tuvo éxito por estar escrito haciendo uso de una impecable gramática latina y por ser breve (284 versos), por lo que se le usó como texto de estudio en los cursos de gramática[4]. El otro trabajo fue un libro de texto escrito en 1250 por un inglés, John de Halifax (mejor conocido como *Sacrobosco*), bajo el título de *Algorismus Vulgaris* y que fue ampliamente utilizado en los cursos de aritmética.

[4]Recordemos que la enseñanza medieval estaba dividida en dos secciones: el trivio, compuesto por gramática retórica y dialéctica y el quadrivio, compuesto por aritmética, geometría, astronomía y música.

Hemos estado hablando de diversos *métodos* para hacer cálculos eficientemente. La palabra clave aquí es *método*, es decir, un procedimiento fijo que nos permita, partiendo de ciertos datos y a través de una serie de pasos ordenados, obtener el resultado deseado luego de algún tiempo finito. Este es, básicamente, nuestro actual concepto de *algoritmo*, la piedra sobre la que se levanta toda la estructura de la actual ciencia de la computación, su esencia.

Nos hemos referido hasta ahora a métodos o algoritmos para hacer cuentas, cálculos numéricos, pero éste no es el único ámbito donde los algoritmos se utilizan y se han utilizado. Un buen ejemplo de computación sin números es la geometría euclidiana.

Los antiguos griegos desarrollaron toda una ciencia a partir de los postulados de la geometría dados por Euclides, una lista de cinco hechos naturalmente evidentes. Con estos postulados o axiomas y ciertas "reglas del juego" es posible construir una serie de hechos nuevos, teoremas que se deducen de los axiomas utilizando las reglas mencionadas. En la geometría euclidiana cualquier construcción ha de ser hecha con un número finito de operaciones elementales: aquellas que se pueden efectuar con una regla y un compás. Así que en el ámbito de la geometría euclidiana todas las construcciones son el resultado de un algoritmo, una serie de pasos ordenados que se efectúan con una regla y un compás.

Algunas personas se preguntaron ¿qué cosas se pueden construir siguiendo las reglas de la geometría euclidiana? ¿qué cosas se pueden hacer usando sólo una regla y un compás?. Surgieron así una serie de problemas que constituyeron un reto durante siglos. Usando sólo una regla y un compás:

- ¿Es posible dividir en tres partes iguales un ángulo? (trisecar un ángulo).
- ¿Es posible construir un cuadrado de área igual a la de un círculo dado? (cuadratura del círculo).
- Dado un cubo ¿es posible construir otro que tenga exactamente el doble de volumen del primero? (duplicar un cubo).

Más tarde se demostró que ninguno de estos problemas es soluble. Es decir, ninguna de las construcciones mencionadas puede hacerse utilizando solamente regla y compás. Así que hay construcciones geométricas que no pueden efectuarse con las reglas establecidas, no existe un algoritmo (euclidiano) que dé como resultado la cuadratura del círculo o la duplicación de un cubo. En el sistema formal de la geometría euclidiana hay cosas que no se pueden calcular. Más adelante regresaremos a este tipo de problemas imposibles.

3

Todo es aritmética

En el siglo XIII un catalán que sería llamado *Docto Illuminatus* y cuyo verdadero nombre era Ramón Lull o Raimundo Lulio, se preocupaba porque, para descubrir la verdad, el hombre tenía que pasar por las tribulaciones de la razón, haciendo que peligrara la empresa propensa al error humano. Así que inventó (según él mismo por inspiración divina) una máquina que permitiera hacer automático el razonamiento lógico, la llamó *Ars Magna*[1]. El dispositivo tenía una serie de círculos concéntricos con palabras escritas en ellos en cierto orden especial; cuando algunas de estas palabras se disponían de tal forma que formulasen una pregunta, la respuesta a ésta aparecía en otro lugar. El objetivo era automatizar el razonamiento, tratarlo como si fuese una operación aritmética donde dado un problema hay un procedimiento sistemático que lo resuelve y siempre se hace de la misma forma: un algoritmo. Probablemente el *Ars Magna* le inspiró a Swift el aparato que encontró Gulliver en uno de sus viajes. Dicho artefacto poseía unos cubos que en sus caras tenían palabras; los cubos estaban unidos por alambres de tal forma que un profesor podía moverlos y de manera automática surgían frases coherentes, "hasta la persona más ignorante podía escribir libros sobre filosofía, poesía, política, derecho, matemáticas o teología sin la menor asistencia del genio o el estudio". Años más tarde Leibniz reconsideraría el *Ars Magna* sugiriendo la creación de una "máquina de razonar".

Lull tenía una extraña preferencia por cortejar mujeres casadas, llegando incluso a persiguir a caballo a una dama hasta el interior de una iglesia. Hacia el final de su vida, luego de tener en dos ocasiones la visión de la cruz, se

[1] No confundirla con la obra de Cardano, parcialmente plagiada a Tartaglia, que ostenta el mismo nombre y que versa sobre álgebra. La obra de Lull se llamaba *Ars Compendiosa Inveniendi Veritatem seu Ars Magna et Maior*.

volvió un ferviente cristiano arrepentido y se obsesionó con tratar de convertir a los musulmanes al cristianismo, escribió el "Libro de la Contemplación" que en sus páginas finales demuestra a los infieles que el cristianismo es la verdadera fe. Se fue a África como misionero en dos ocasiones; en la última, en 1316, murió apedreado.

Algunos otros pensadores después de Lull opinaron también que la lógica puede ser sistematizada y mecanizada. El filósofo inglés Thomas Hobbes, declaró que "razón no es sino cómputo", es decir, el razonamiento es una especie de aritmética. En su obra *Leviatan* (1651) dice: "Cuando un hombre razona, no hace otra cosa que concebir una suma total, por adición de partes; o concebir un residuo, por sustracción de una suma respecto de otra [...] los lógicos enseñan lo mismo en cuanto a las consecuencias de las palabras: suman dos nombres, uno con otro, para componer una afirmación; dos afirmaciones, para hacer un silogismo, y varios silogismos, para hacer una demostración...". Un poco después (1655) en su obra *Elementarum Philosophiæ* (Elementos de Filosofía) incluyó una parte dedicada a la lógica titulada *Computatio sive Lógica* (Computación o Lógica).

Otro admirador de la obra de Lull fue Giordano Bruno quien escribió algunos tratados alrededor de la máquina de Lull[2] y de hecho construyó su propia versión: *revoluciones circulorum*, un artefacto de filosofía combinatoria. Por desgracia Bruno no tuvo mucho tiempo más para explorar nuevas posibilidades, fue encarcelado durante sus últimos siete años de vida y murió en la hoguera en 1600 condenado por el tribunal de la inquisición de Roma.

Por la misma época de Hobbes (siglo XVII), Descartes pensó que todos los problemas de la humanidad podían, de alguna manera, reducirse a problemas geométricos para los que él mismo inventó mecanismos de solución algebraicos infalibles, casi automáticos, muy lejos de las tribulaciones de la geometría sintética de los griegos, casi un arte sin métodos de solución fijos.

En medio del caos religioso y político que caracterizó a la isla de Gran Bretaña durante el siglo XVI, nació y vivió un noble escocés descendiente de una familia de notables guerreros: John Neper. Él se interesó por el estudio en general y por el cultivo de la aritmética y la teología en particular, cosa que no se podía hacer bien en su natal Escocia, así que realizó sus estudios en el extranjero y regresó después a Escocia a escribir muchos de sus pensamientos acerca de las materias de su interés. Estudió la notación árabe cuya historia reconstruyó hasta su origen indio, meditando acerca del principio posicional de la notación.

[2] *De Lampade Combinatoria Lulliana* (1587) y *De Specierum Scrutinio el Lampade Combinatoria Raymundi Lulli HeremitæOmniscij, Propemodunque Divini* (1588).

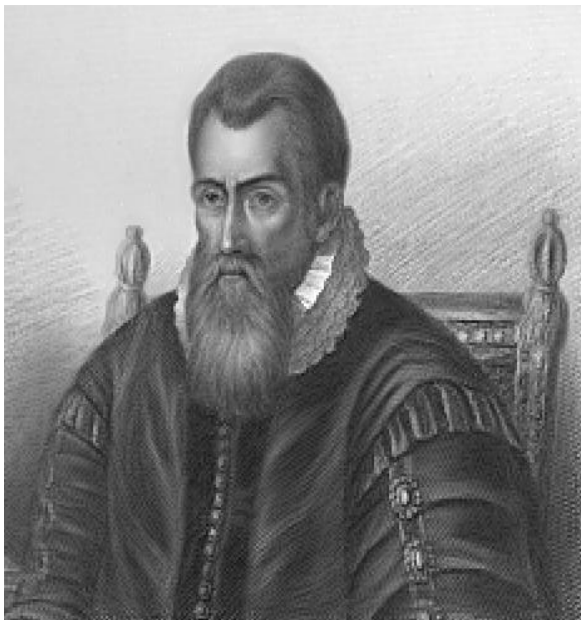

Figura 3.1: John Neper de Merchinson.

Para 1590 Neper había descubierto un artificio que transformaba multiplicaciones en sumas: los logaritmos. Es mucho más fácil sumar que multiplicar, así que este descubrimiento entusiasmó a muchos astrónomos ansiosos por facilitar los laboriosos cálculos a los que estaban acostumbrados. Si uno quiere multiplicar 532 por 248 basta obtener los logaritmos de ambas cantidades que son 6.2766 y 5.5134 respectivamente (en base e), sumar luego los números obtenidos, lo que nos da 11.79 y finalmente obtener al antilogaritmo de esta última cantidad: 131936, que resulta ser el producto de 532 por 248. La dificultad de este procedimiento radica en encontrar un método sencillo para obtener el logaritmo y el antilogaritmo (operación inversa del logaritmo) de un número. Lo más fácil es buscar en una tabla, elaborada con cierto nivel de precisión, cuál es el logaritmo de un número dado.

Por desgracia para algunos astrónomos, como Tycho Brahe, las tablas, elaboradas por el mismo Neper, no salieron a la luz sino hasta 1614 (Brahe murió en 1601 al excederse en contener su orina). De todos modos las tablas fueron muy útiles para otros como Johannes Kepler, que dedicó sus *Ephemerides* de 1620 a la memoria de Neper que murió en 1617.

En 1614 Neper también creo unas tablillas con las que se podía multiplicar casi sin esfuerzo (véase fig. 3.2).

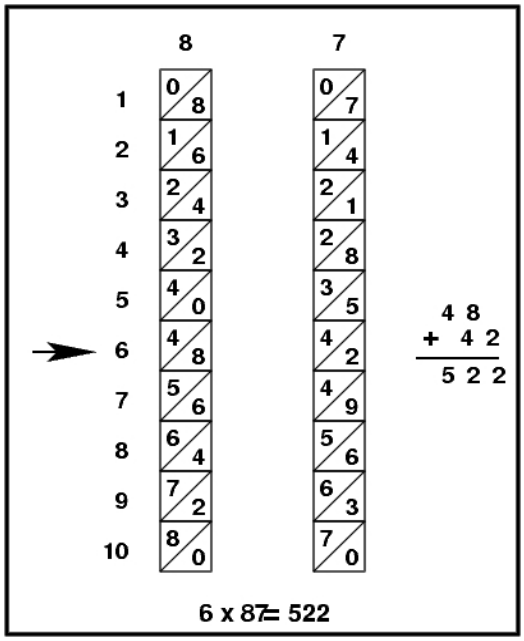

Figura 3.2: Las tablillas de Neper

En 1623 Wilhelm Schickard, profesor de astronomía y matemáticas en Tübingen, inventó un dispositivo para realizar algunos de sus cálculos astronómicos. La máquina de Schickard sumaba y restaba automáticamente y también podía usarse para multiplicar y dividir con ayuda del operario. Los conocimientos necesarios para la construcción de este ingenio mecánico se remontan hasta los antiguos relojeros chinos cuyos conocimientos pasaron luego a medio oriente y de allí, gracias a los árabes, al mundo occidental. Para la época de Schickard ya había en Europa una gran tradición en la construcción de relojes y juguetes que eran capaces de realizar complicadísimos movimientos, relojes que poseían multitud de personajes representando diferentes escenas de acuerdo a la hora del día que anunciaba el reloj.

La máquina de Schickard funcionaba mediante ruedas dentadas que giraban al dar vuelta a unas manivelas. Imaginemos que tenemos una rueda que representa las unidades y que, dada una cierta posición de esta rueda, que representa una cantidad, se le suma otra girando la rueda diente a diente hasta completar tantos dientes recorridos como indique la cantidad que se desea sumar. Cuando finalmente se han recorrido exactamente diez dientes esta rueda regresa a su posición inicial y en una suma normal esto significa que se debe incrementar ahora el número de decenas contabilizadas. La

Figura 3.3: Blaise Pascal.

máquina de Schickard recorría exactamente un diente la rueda de las decenas cada vez que se habían recorrido diez dientes en la de las unidades, de esta manera la máquina llevaba el acarreo automáticamente.

En 1633 un clérigo llamado William Oughtred inventó un dispositivo basado, justamente, en los logaritmos de Neper al que llamó *círculos de proporción* y que más tarde daría lugar a la regla de cálculo, usada hasta hace muy poco.

En 1642 el hijo de un abogado y cobrador de impuestos francés, se preocupaba por la gran susceptibilidad al error que había al hacer cuentas largas, como las que hacía su padre y decidió que era necesario crear un artefacto capaz de realizar el metódico y tedioso trabajo de hacer sumas y restas en particular; después de todo, el hecho de que fuera un trabajo metódico y repetitivo hacía pensar en ciclos a realizar una y otra vez, como en las ruedas de la máquina de Schickard. Así que este joven de 19 años (que tres años antes escribiera un tratado acerca de las secciones cónicas) llamado Blaise Pascal inventó, independientemente de Schickard, una máquina de sumar y restar. Pascal, siempre enfermizo, murió en 1662 tras cuatro años de padecimiento, desvariando acerca de cuestiones teológicas en las que se sumergió a raíz de un accidente de coche del que salió vivo milagrosamente.

Figura 3.4: Gottfried W. Leibniz.

En 1673 el mecánico del rey Carlos II, Sir Samuel Morland, inventó su propia máquina sumadora y publicó un libro que pone de manifiesto la pereza de que hemos hablado: *Un Nuevo y más útil Instrumento para la Adición y la Substracción de Libras, Chelines, Peniques y Cuartos de Penique, sin Recargar la Memoria, sin Perturbar la Mente y sin Exponer la Operación a Ninguna Incertidumbre.* La quintaesencia de la computación. Algunos otros inventos de Morland eran una bomba hidráulica, máquinas de vapor y un trompetista parlante.

Más ambicioso fue Leibniz quien en 1673, igual que Morland, inventó una máquina para calcular. Ya hemos citado a Leibniz quien deseaba poder encargar los complicados cálculos matemáticos a "esclavos" confiables y quien creía que se podría lograr una máquina de razonar. Leibniz, por cierto, conocía la máquina de Pascal. La máquina de multiplicar de Leibniz utilizaba también ruedas dentadas, pero con dientes de distintos tamaños representando distintas magnitudes. Leibniz, quien creía que había un álgebra que lo gobernaba todo, incluso los negocios y la política, envió su máquina a Pedro el Grande con el encargo de que éste la enviara al emperador de China con el fin de convencerlo para comerciar más intensamente con occidente.

Las máquinas con la idea original de Leibniz se siguieron produciendo hasta antes de 1970, época de la revolución electrónica[3].

La máquina de Leibniz fue una manera de concretar algo que giró todo el tiempo en su mente: todo es aritmética. Leibniz creía en la existencia de una especie de átomos espirituales: las *mónadas*, que gobernaban el comportamiento del universo entero. Las mónadas debían poseer, pensaba Leibniz, una especie de alfabeto, que no representara sonidos, como en los alfabetos usados por los seres humanos, sino conceptos; si poseyéramos el alfabeto y las reglas del lenguaje, sería posible, por medio de procedimientos de cálculo simbólico, llegar a conclusiones verdaderas y determinar las relaciones lógicas entre ellas. Algo muy cercano a las ideas de Lull.

[3]Al parecer un clérigo de la iglesia de Cristo, el doctor Hooke, ya había inventado en 1670 (tres años antes que Leibniz) una máquina de multiplicar y dividir.

4

Sobre hombros de gigantes

A fines del siglo XVIII el conde de Stanhope, un caballero inglés miembro de la cámara de los comunes y más tarde de la de los lores, quien conocía el trabajo de Sir Samuel Morland, inventó una máquina lógica que era capaz de ejecutar operaciones silogísticas y resolver problemas elementales de probabilidad; también creó una máquina calculadora al estilo de la de Leibniz, un bote de vapor y una imprenta manual.

En 1869 William Stanley Jevons, un economista inglés de buena posición (esposo de la hija del fundador del periódico *The Guardian*, aún en circulación), inventó una máquina lógica "con el suficiente poder para resolver un problema complicado más rápido que sin la asistencia de la máquina". El artefacto llamado *piano lógico* fue presentado en 1870 en la Royal Society y estaba basado en la obra de otro lógico inglés que mencionaremos adelante: George Boole, un profesor de matemáticas del Queens College en Cork (Irlanda).

La sentencia "más rápido que sin la asistencia de la máquina" hoy puede sonarnos familiar, estamos acostumbrados a que las máquinas extiendan nuestras capacidades, a que hagan algunas cosas más rápido y mejor que nosotros, pero en el siglo XIX sonaba a pesadilla de ciencia ficción. Las máquinas no sólo ganaban terreno en el campo de los cálculos rápidos, las operaciones lógicas y el razonamiento; también lo hacían en otros ámbitos, el movimiento mecanicista desarrollado a partir del éxito de la mecánica newtoniana había hecho posible crear máquinas que resolvían tareas arduas, difíciles y propensas al error de una manera rápida y libre de errores, la revolución industrial estaba en marcha y nada la detendría, ni siquiera las rebeliones y la destrucción de maquinaria del movimiento luddita en Inglate-

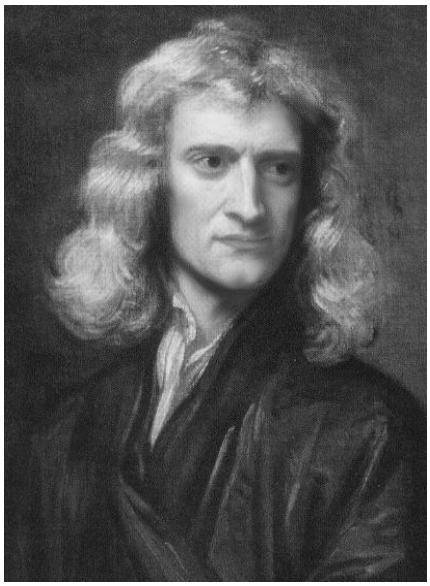

Figura 4.1: Issac Newton.

rra. Muchos trabajadores serían reemplazados por máquinas que producían más y mejor. De aquí en adelante la humanidad ya no dependería sólo del esfuerzo animal o del sudor de su frente, ni estaría a merced de los elementos, había creado sus propios y poderosos aliados, que hacían exactamente lo que se quería de ellos sin protestar. Newton le había regalado el fuego a los mortales, les había enseñado como funcionaba el mundo, ahora el ser humano podía adueñarse de él y nada podría frenarlo. La naturaleza había sido domada por los más pueriles de sus hijos.

Así como la naturaleza se regía por leyes mecánicas, también, ¿por qué no pensar que con el razonamiento podía suceder lo mismo?. De allí las máquinas lógicas de Morland y Jevons y de allí el trabajo de George Boole (en el que se basó el de Jevons) *Investigación sobre las leyes del pensamiento* (1854) donde sentó las bases de lo que conocemos como álgebra de Boole. El libro comienza diciendo *"El diseño de este tratado es para investigar las leyes fundamentales de aquellas operaciones de la mente mediante las que se lleva a cabo el razonamiento; para expresarlas en el lenguaje simbólico de un cálculo, ..."*. El libro es pues un intento por mecanizar el pensamiento, hacer cuentas con proposiciones, deducir automáticamente la verdad o falsedad de ciertos enunciados a partir de ciertas premisas, un retorno a las viejas ideas de Lull o Leibniz en un entorno más concreto.

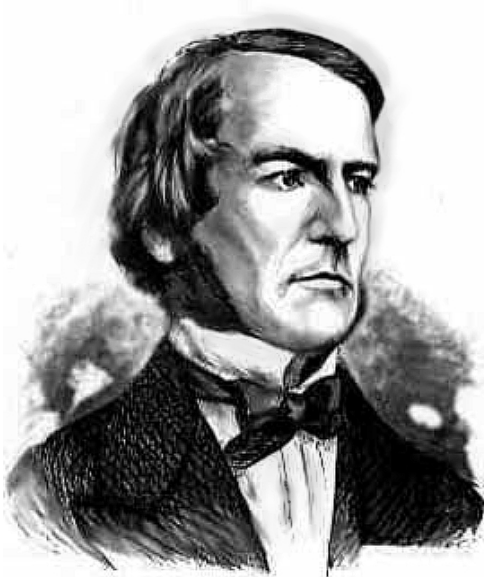

Figura 4.2: George Boole.

El objetivo último de Boole es entonces construir un sistema completo y estructurado, análogo a la mecánica de Newton, para describir la manera en que funciona la máquina de nuestro cerebro. Esto se vislumbra, por ejemplo, en el capítulo XXII (el último) *La constitución del intelecto*, donde escribe: *"De ser obedecidas uniformemente las leyes del razonamiento válido, existiría un paralelismo muy cercano entre las operaciones del intelecto y aquellas de naturaleza externa. La sujeción a leyes matemáticas en su forma y expresión, la sujeción de obediencia absoluta, señalaría un carácter común entre ambas. El reino de la necesidad sobre el mundo intelectual y el físico sería al mismo tiempo completo y universal."*. Luego señala (pensaría yo que con cierta desilusión) que las leyes del razonamiento que ha expuesto a lo largo del libro son sólo las del razonamiento correcto y que pueden ser y de hecho son transgredidas. Boole sintetiza el espíritu de su época, la búsqueda de las leyes universales que nos expliquen las cosas que desconocemos tal como lo ha hecho la mecánica newtoniana: el hombre es capaz de conocerlo todo, nada se sale del dominio de la mente, ni la mente misma.

Para esta época (primera mitad del siglo XIX) la mayoría de los cálculos en el contexto de la astronomía, la navegación, la ingeniería y la ciencia en general, eran hechos con base en tablas: tablas de logaritmos, como aquellas

que, por primera vez, elaborara Neper, tablas de funciones trigonométricas, de cuadrados, de cubos, de funciones estadísticas, en fin, una cantidad enorme de tablas. Los científicos de la época poseían en sus bibliotecas incontables volúmenes llenos de tablas plagadas de inevitables errores. Ya en el pasado se habían hecho esfuerzos considerables para evitar los errores en las tablas usadas para calcular. En Francia por ejemplo, luego de la revolución, un señor de apellido Prony estuvo a cargo de un proyecto para calcular tablas precisas. Prony reclutó un ejército de cien peluqueros[1] que se dieron a la tarea de calcular tablas matemáticas. Cada número en las tablas resultantes había sido calculado, al menos, dos veces y sin embargo las tablas tenían errores. Algunos de los errores en los libros de tablas eran producto del proceso de impresión y no de los calculistas al substituir, por ejemplo, un número por otro u omitir alguno al momento de formar la página en el taller de impresión.

Alguien que se preocupaba mucho por esta situación era un inglés, de actitud muy victoriana, llamado Charles Babbage. En 1826 Babbage publicó, en una primorosa edición, sus propias tablas de logaritmos. Las tablas, consideradas las más precisas de su época, fueron impresas usando tintas de varios colores y papel de color amarillo; según los propios experimentos de Babbage, esto mejoraba la legibilidad, lo que minimizaba la fuente última de error: copiar mal el número de la tabla.

Hacer una tabla consiste, esencialmente, en calcular una serie de valores sucesivos de una función. Eso puede hacerse fácilmente por el *método de las diferencias*. Imaginemos que tenemos un polinomio como: $3x^3+2x^2+5x+8$; le damos valores a la variable x, por ejemplo: 1, 2, 3, ..., 10 y evaluamos el polinomio en ellos; los resultados se muestran en la columna $f(x_i)$ en la tabla 4.1. Si nos fijamos en la diferencia entre los valores sucesivos del polinomio obtenemos lo mostrado en la columna d_i de la tabla, si aplicamos el mismo procedimiento, ahora obteniendo las diferencias (a las que podemos llamar segundas diferencias) entre las primeras diferencias, tenemos lo mostrado en la columna d_i^2; aplicando otra vez el procedimiento obtenemos las terceras diferencias que se muestran en la columna d_i^3. Nótese que las terceras diferencias son todas iguales; esto ocurre siempre en un polinomio de tercer grado, como el de nuestro ejemplo. Si hubiéramos usado un polinomio de grado 5 entonces habríamos tenido que esperar hasta las quintas diferencias para obtener una columna constante.

[1] La instalación de la república había puesto en desuso las tradicionales pelucas de la época de la monarquía, así que el proyecto de Prony contribuyó a resolver el problema de desempleo en el gremio de peluqueros.

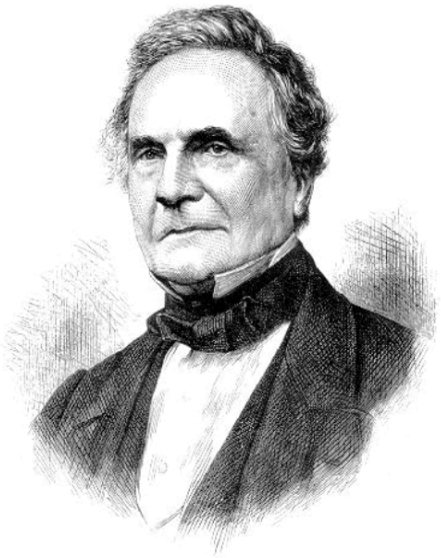

Figura 4.3: Charles Babbage.

Imaginemos que queremos obtener el valor de nuestro polinomio cuando $x = 11$, ese valor ya no lo tenemos en nuestra tabla, pero resulta ser 4298. Para obtener el valor podemos hacer lo siguiente: ya sabemos que la diferencia entre el último valor de la columna de las segundas diferencias (166) y el que le seguiría debe ser 18, así que el valor que se obtendría en las segundas diferencias para $x = 11$ sería $166 + 18 = 184$. Ese número que acabamos de obtener debe ser la diferencia entre el 856, último número de la columna de las primeras diferencias, y el que le seguiría en esa misma columna, que entonces sería $856 + 184 = 1040$. Por último, ese 1040 debe ser la diferencia entre el último valor del polinomio que aparece en la tabla, 3258, y el que le sigue, que es el que queremos saber; así que el valor deseado es: $3258 + 1040 = 4298$.

Ahora bien, como cualquier función de la que nos interesa tener tablas es expresable con polinomios, podemos aplicar nuestro procedimiento descrito arriba para obtener una serie arbitrariamente larga de valores consecutivos de las funciones.

A Babbage se le ocurrió la idea de hacer una máquina que calculara tablas de funciones por el método de las diferencias que acabamos de conocer. El objetivo era que la máquina no sólo produjera los valores consecutivos de la función deseada sino que además los imprimiera, así se evitaban tam-

x_i	$f(x_i)$	$d_i = f(x_i) - f(x_{i-1})$	$d_i^2 = d_i - d_{i-1}$	$d_i^3 = d_i^2 - d_{i-1}^2$
1	18			
2	50	32		
3	122	72	40	
4	252	130	58	18
5	458	206	76	18
6	758	300	94	18
7	1170	412	112	18
8	1712	542	130	18
9	2402	690	148	18
10	3258	856	166	18

Tabla 4.1: Tabla de valores para el polinomio $3x^3 + 2x^2 + 5x + 8$.

Figura 4.4: Ada Augusta Byron, condesa de Lovelance.

bién los indeseables errores de impresión. Así surgió la idea de la *máquina diferencial* de Babbage. Por desgracia elaborarla no era nada trivial, se requería de herramientas muy específicas y procedimientos de construcción nunca antes intentados. Babbage se asoció con el mejor mecánico de Inglaterra, un tal Samuel Clement que inició la construcción de la máquina diseñada por él. El proceso era sumamente lento: había que construir herramientas y maquinaria para construir las piezas de la máquina diferencial, además había que capacitar personal en técnicas nuevas y en el uso de las herramientas. Babbage solicitó ayuda al gobierno inglés en 1822 mostrando un prototipo muy modesto de la máquina que financió de su propio peculio; en 1823 le fue otorgado un primer subsidio de 1500 libras. Estos apoyos gubernamentales tardaban mucho en ser otorgados y ocasionaban indeseables pausas en el trabajo, algunas de hasta cuatro años. En una de esas pausas Babbage y Clement riñeron por algunas diferencias acerca del lugar donde debía instalarse el taller y esta discusión terminó con la ruptura de su sociedad. Clement era legalmente dueño de todas las herramientas, maquinaria y diseños de la máquina, así que Babbage se quedó sin sus diseños y sin posibilidades de continuar el trabajo ya hecho. En 1842 el gobierno inglés decidió retirar todo subsidio al proyecto, luego de haber gastado unas 17,000 libras.

Luego de perder sus diseños originales, Babbage se dió a la tarea de rehacerlos, y aprovecho para hacer algunas mejoras. Al hacer esto ideó una máquina diferente y mucho más ambiciosa, la llamó *máquina analítica*. Esta podía realizar diversas operaciones "elementales" y el usuario podía especificar cuáles, en qué orden y con qué datos de entrada. Es decir, era una máquina programable. Curiosamente la primera programadora de la historia era hija de Lord Byron, uno de los hombres que había pronunciado más discursos en el parlamento en contra de la industrialización. Su hija, Ada Augusta Byron, condesa de Lovelance, admiraba a Babbage y se dio a la tarea de hacer agujeros en unas placas donde se especificaba qué operaciones debía realizar la máquina analítica. Esta idea de los hoyos la tomó Babbage del método recientemente inventado (1805) por Jacquard, un mecánico de Lyon, para especificarle a una máquina de tejer cómo debía ser el entramado de la tela; de aquí la idea evolucionó sin muchos cambios, pasando por las bandas de las pianolas, hasta llegar a la tarjeta perforada en las que se guardaban los programas hasta hace unas décadas.

El diseño de la máquina analítica de Babbage constaba de tres partes fundamentales: una unidad de entrada, donde se leía el programa a ejecutar representado por hoyos en una tarjeta perforada; un "molino" (llamado así por Babbage) en el que se realizaban las operaciones requeridas; un

"almacén" donde se guardan los datos de entrada, los resultados intermedios y los resultados finales, es decir el almacén era también un dispositivo de salida; una unidad de control encargada de tomar decisiones en el flujo del programa (qué instrucción ejecutar como siguiente paso) dependiendo de los resultados en el almacén hasta el momento y del programa proporcionado en las tarjetas.

Por desgracia la máquina analítica, cuyo diseño fue concluido a mediados del siglo XIX, tampoco se hizo realidad, no durante la vida de Charles Babbage. En 1906 su hijo, Henry Babbage, terminó la construcción del molino con base en los diseños de su padre y la máquina realmente funcionó, fue capaz de calcular π con 29 decimales de precisión.

En 1899 la exposición universal de París otorgó un premio al invento de un mecánico francés dedicado a hacer autos de carreras, León Boleé. El invento premiado era una calculadora mecánica que efectuaba la multiplicación directa y no a través de sumas repetidas. La había inventado en 1887, a la edad de 18 años. No siempre los premios son expeditos.

En 1905, Percival Lowell, un astrónomo norteamericano, famoso por observar "canales de riego" en la superficie de Marte, se atrevió a proponer otra de sus descabelladas ideas. Según los cálculos que había hecho debía haber un planeta cerca de Urano, él le llamó planeta X. En 1930, catorce años después de la muerte de Lowell, X se rebautizó como Plutón. Los cálculos hechos por Lowell fueron auxiliados por un ingenio mecánico ideado y fabricado por el suizo Otto Steiger desde 1892. Al igual que la máquina de Boleé la de Steiger, llamada Millonaria, efectuaba la multiplicación directamente. De 1894 a 1935 se vendieron 4500 unidades de esta máquina, todo un éxito comercial para una calculadora en aquella época.

En 1887 la oficina de estadística de los Estados Unidos aun no poseía los resultados del censo que se había realizado en 1880; los resultados eran obtenidos por centenares de empleados que recopilaban y procesaban los datos a mano. Uno de esos empleados había sido un joven ingeniero de minas llamado Herman Hollerith que entró a trabajar a la oficina de estadística en 1879, justo un año antes del censo pero había dejado el puesto en 1882 para aceptar un trabajo de profesor en el MIT. En junio 8 de 1887 Hollerith obtuvo la patente de su calculadora de tarjeta perforada que sería usada poco después en el procesamiento de datos del censo de 1890, proceso que llevó sólo dos años y medio. La idea fundamental era similar a la del telar de Jacquard o la máquina de Babbage, en unas tarjetas se hacían orificios en ciertas posiciones dependiendo del dato que se estuviera almacenando: unos agujeros se hacían para denotar que un individuo de la población era mujer y otros en posiciones diferentes si era hombre. A través de los orificios

Figura 4.5: Herman Hollerith.

pasaban unos alambres (electrodos) que bajaban hasta unos pequeños tubos con mercurio, los alambres llevaban una corriente eléctrica, al entrar en contacto con el mercurio cerraban un circuito y se movía una manecilla en un contador.

Hollerith tuvo mucho éxito con su máquina, fue empleada también en los censos de Austria y en el primer censo ruso en 1896; fundó su propia compañía: la *Tabulating Machine Company*, que rápidamente se convirtió en una empresa internacional. En 1911 un adinerado traficante de armas, llamado Charles Flint, adquirió la empresa de Hollerith a cambio de 1.2 millones de dolares y un contrato que hacía a éste asesor de la compañía. Flint había comprado otras tres empresas dedicadas a hacer diversas máquinas: relojes checadores, cortadoras de carne y balanzas, así que fusionó las cuatro empresas en una sola a la que puso el nombre de *Computing-Tabulating-Recording Company* o CTR y a cargo de ella puso a un individuo de muy holgada moral en lo que a los negocios se refería, su nombre era Thomas J. Watson y podríamos decir que, si de hacer dinero se trataba, Watson era el mejor y el de menos escrúpulos. Su fama la ganó como agente de ventas de una compañía que fabricaba cajas registradoras, la NCR (*National Cash Register*). Durante su estancia en NCR se valió de cuanto truco sucio se pudo

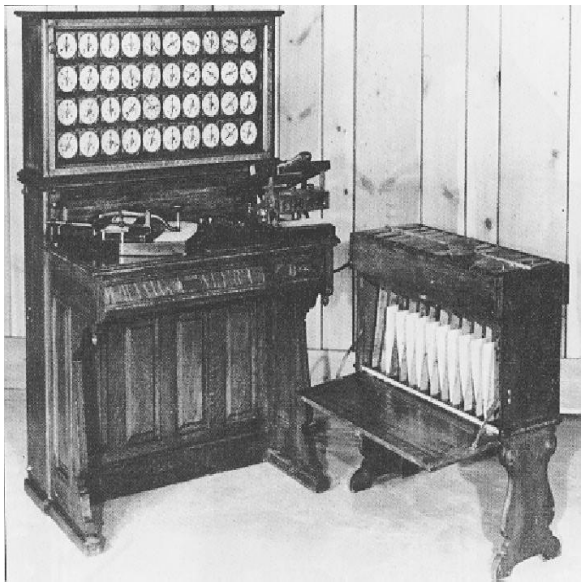

Figura 4.6: Máquina tabuladora de Hollerith.

para adueñarse por completo del mercado. Ahora Watson era, literalmente, *el líder* de CTR y Hollerith, el fundador, fue prácticamente ignorado.

En 1924 Thomas J. Watson cambió el nombre a la compañía fundada por Hollerith a *International Business Machines* (IBM). Para ese entonces el perfil de la compañía se había definido plenamente, ya no fabricaban cortadoras de carne ni relojes, sólo máquinas tabuladoras, lectoras de tarjetas perforadas, clasificadoras de tarjetas, etcétera. Con el tiempo las máquinas fueron evolucionando e incorporaron operaciones más complicadas, aunque su labor principal siguió siendo como mecanismos de registro, conteo y clasificación lo que las hacía ideales para las labores que les dieron origen: estadística poblacional y censos, algunos de ellos abominables.

Imaginemos la escena: un tranquilo pueblo alemán alrededor de 1935. A la plaza central arriba un vehículo del que desciende un oficial de la SS y pega un cartel en que aparecen los nombres de varias personas y se indica que deben presentarse al día siguiente, con sus pertenencias indispensables, en la estación de ferrocarril. Los del listado no lo sospechan aún, pero emprenderán un viaje del que la inmensa mayoría de ellos no regresará. En el listado, obtenido a partir de los datos del último censo, aparecen los nombres de todos los judíos de la aldea, hasta el nombre de algunos cuyas familias dejaron de serlo hace generaciones; aparecen perso-

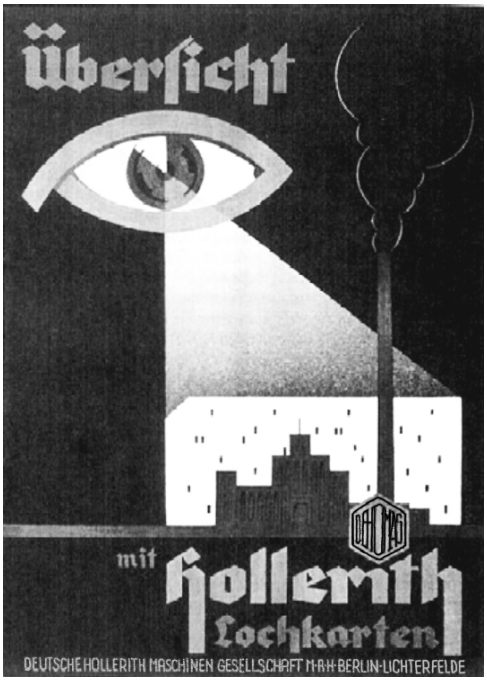

Figura 4.7: Poster publicitario de DEHOMAG.

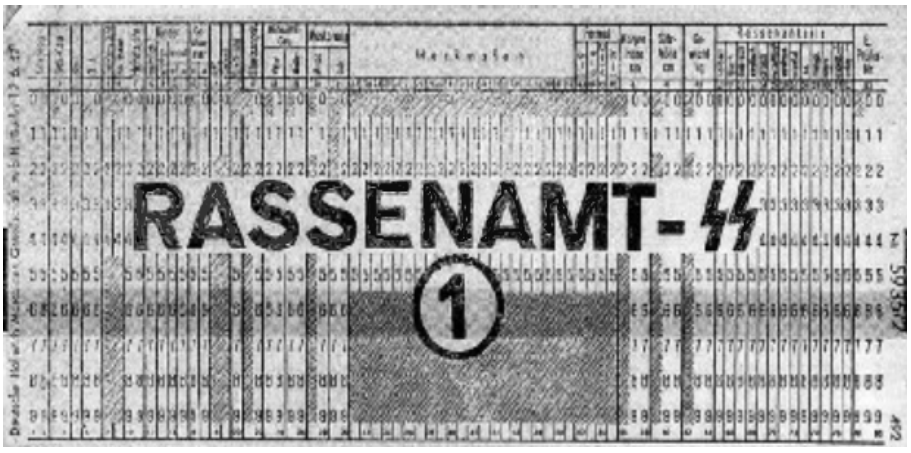

Figura 4.8: Tarjeta de clasificación racial usada por la SS.

nas que descienden de judíos en segundo o tercer grado. Todos irán al este; los "centros de detención preventiva" son muchos: Auschwitz, Buchenwald, Dachau, Gross-Rosen, Herzogenbusch, Mauthausen, etcétera. Una vez allí serían registrados y clasificados por la Arbeitseinsatz (oficina de asignación de trabajo), su nombre, identificador y demás datos se registraban en tarjetas perforadas que luego se clasificaban mecánicamente y eran contadas en repetidas ocasiones considerando diversas características registradas en ellas: por sexo, por edad, por lugar de origen, por "raza". Thomas Watson logró otro de sus lucrativos negocios, vendiendo al gobierno de la Alemania nazi las máquinas, capacitación de personal para usarlas, mantenimiento periódico de las mismas y el diseño y elaboración de las tarjetas; casi todo a través de una subsidiaria alemana: la *Deutsche Hollerith Maschinen Gesellschaft* o DEHOMAG; más de 2000 máquinas IBM se surtieron al Reich a través de esta compañía y miles más a través de otras subsidiarias en los países ocupados durante la guerra. Las máquinas que procesaron los datos del censo de 1933, las que clasificaron a la población y la contaron, las que estaban en los campos dando el soporte tecnológico para la "solución final al problema judío", eran IBM.

El antisemitismo alemán no surgió repentinamente durante el gobierno de Adolf Hitler, es algo que se venía incubando desde hacía mucho tiempo y permeaba hasta en los estratos cultos de la población, un ejemplo de ello fue Gottlob Frege. Nacido en Wismar en 1848, Frege fue un profundo admirador de Bismarck, del Kaiser Wilhelm y de la gloria del imperio alemán perdida luego de la primera guerra mundial. De este nacionalismo extremo surge su afinidad con el antisemitismo, patente en las páginas de su diario; más de un admirador se ha desilusionado al conocer este lado obscuro de un ser tan inteligente y de aportaciones tan valiosas para la lógica matemática.

Frege es, de cierta forma, un continuador del trabajo de Boole y de las ideas de Leibniz. En 1879 publicó un folleto cuyo título traducido al español sería aproximadamente: "El Modo de Escribir Conceptos" o "Lenguaje de Conceptos", lo que ciertamente recuerda a Leibniz. El título en alemán es una palabra compuesta: *Begriffsschrift*, el subtítulo es más claro: *Un lenguaje de fórmulas, como el de la aritmética, para el razonamiento puro.* Como se puede ver desde el subtítulo, la intención de Frege es elaborar un lenguaje que permita escribir y luego manipular, hacer operaciones, con conceptos. Boole tomó el álgebra como punto de partida para expresar relaciones lógicas entre símbolos que representan enunciados, proposiciones y conceptos. Esto no era muy del agrado de Frege, dado que a fin de cuentas el álgebra tiene su base estructural en la lógica, así que "el lenguaje" de Boole

debía ser cambiado por uno más puro. Introdujo símbolos para representar relaciones lógicas entre proposiciones, una sentencia como:

Todos los gatos maullan

puede escribirse en el lenguaje de la lógica simbólica como:

$$(\forall x)(g(x) \supset m(x))$$

donde $\forall$ se lee como *cualquier*, $g(x)$ como x *es un gato*, $m(x)$ como x *maulla* y el símbolo $\supset$ como "implica que". Si ahora se para frente a nosotros *Merlín*, el gato de mi tía, un gato en particular, es decir un ente x que satisface $g(x)$; como ya tenemos en nuestro acervo de conocimientos la fórmula que acabamos de escribir, podemos concluir $m(x)$, es decir *Merlín maulla*. Podemos ahora dar el siguiente paso, despojar de significado a $g(x)$ y $m(x)$, ahora podrían ser enunciados diferentes: por ejemplo $g(x)$ podría significar x *es un día lluvioso* y $m(x)$ podría ser x *es un día nublado*. Nuestra conclusión $m(x)$ sigue siendo válida si tenemos como premisas una x que satisface $g(x)$ y la fórmula $(\forall x)(g(x) \supset m(x))$; nuestra conclusión es válida por su forma, no por su significado, sólo hay que hacer una operación simple para deducir algo verdadero, sin importar lo que signifique, sin pensar. Partiendo de ciertas premisas es posible aplicar reglas que nos permiten llegar a conclusiones válidas sólo por la sintaxis del lenguaje. Pero hay que hacer evidente un pequeño hueco, que retomaremos más adelante: el trabajo de Frege no provee de un método que permita llegar a cierta conclusión dadas ciertas premisas; esto es, un método que, dadas algunas hipótesis y una conclusión deseada, nos indique qué reglas aplicar paso a paso para arribar a la conclusión a partir de las hipótesis.

La intención de Frege era dar un primer paso para establecer el fundamento lógico de la aritmética en particular y de la matemática en general. A lo largo del tiempo se había confiado demasiado en la intuición, se habían dado por sentadas ciertas cosas por ser "evidentes" y después se había llegado a la conclusión de que algunas de estas cosas evidentes eran falsas, un ejemplo recurrente de ello es que durante mucho tiempo se dió por sentado que toda función continua es diferenciable, lo cual, como hoy en día todo estudiante de cálculo elemental sabe, es falso. Frege, como muchos otros matemáticos de su época, deseaba, de una vez por todas, expresar la totalidad de la matemática como una rama de la lógica; hacer limpieza profunda, desechar todo lo falso y quedarse sólo con lo verdadero sin fiarse de la intuición.

Para lograr su objetivo inicial, expresar la aritmética como parte de la lógica, Frege debía definir los números naturales en términos puramente

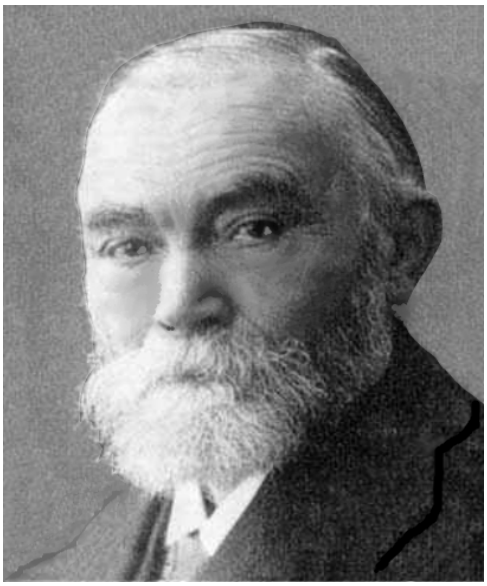

Figura 4.9: Gottlob Frege.

lógicos. En vez de decir el número "5", lo que equivale a decir el nombre del gato, este número debe definirse en términos de sus propiedades, por ejemplo: el número de elementos en el conjunto $A = \{a, b, c, d, e\}$ y de hecho el número de elementos en todos los conjuntos con el mismo número de elementos que A; una definición sin ambigüedades, pero suficientemente general. Así el número 5 queda definido como "el número de elementos en cualquier conjunto que pueda ponerse en correspondencia uno a uno con A". El siguiente paso es entonces poner en un mismo "costal" a todos los conjuntos en cuestión, es decir, a los que tienen cinco elementos; por supuesto este "costal" es un conjunto también. La aritmética de Frege se basaba en estos conjuntos de conjuntos.

En 1902 Frege recibió una carta de un joven británico llamado Bertrand Russell. En ella, luego de elogiar profundamente su trabajo, le hacía ver una debilidad que, de hecho, echaba por tierra el ambicioso proyecto de Frege. La carta en cuestión le señalaba que pensar en conjuntos de conjuntos trae consigo contradicciones, algo que por supuesto, es indeseable. Digamos que un conjunto es *especial* si alguno de sus elementos es él mismo y que los conjuntos que no son especiales son *comunes*. Supongamos ahora que tenemos un conjunto de conjuntos al que llamaremos Γ y que definimos como el conjunto de todos los conjuntos *comunes*. Preguntémonos ahora si

Γ es *especial* o *común*: si decimos que es *común* eso significa que ninguno de sus elementos (que son conjuntos) es él mismo, pero al decir que es común también estamos diciendo, dado que definimos a Γ como el conjunto de todos los conjuntos comunes, que Γ debe ser elemento de sí mismo, lo que es una contradicción. Si Γ no puede ser *común*, entonces debe ser *especial*: pero si decimos que Γ es especial estamos diciendo, por una parte, que Γ es elemento de sí mismo y como habíamos definido Γ como el conjunto de todos los conjuntos *comunes*, otra vez tenemos una contradicción, ya que por otra parte Γ es *común*. En síntesis, no importa si consideramos a Γ como *especial* o *común*, ambas alternativas llevan a una contradicción. Esta es la llamada *paradoja de Russell*.

La paradoja de Russell suele también mencionarse como *la paradoja del barbero*. Supongamos que en un pueblo el barbero afeita a todos aquellos que no se afeitan a sí mismos. Preguntémonos ahora ¿quién afeita al barbero? Si decimos que él mismo estamos en una contradicción y si decimos que lo afeita el barbero, también; no hay posibilidad de ganar (como cuando uno discute con su novia).

5

Echando a perder se aprende

En 1903, los hermanos Wright surcan los aires por primera vez, el descabellado sueño de Leonardo de Vinci se había hecho realidad. En 1909 Henry Ford hace más pequeño el mundo al producir en serie su modelo "T" y la noche nunca más sería obscura en el mundo civilizado gracias a la lámpara incandescente que Edison inventara a fines del siglo XIX, desligando para siempre la luz del fuego. En abril de 1912, cinco meses después que Amundsen pisara por vez primera el polo sur, un gigantesco transatlántico cruzaba el océano entre Europa y América. Por siglos el mar había amedrentado a los hombres, esa infinitud de agua que no era sino un obstáculo, que había cobrado las vidas de innumerables irreverentes, que formaba parte de las historias más aterradoras de naufragios y monstruos desconocidos, era ahora surcada plácidamente por una máquina del hombre, poderosa, indestructible, el fruto más acabado de la tecnología, una manifestación del grado de avance que habían logrado las ciencias. La naturaleza finalmente había entregado sus secretos y ahora era sojuzgada, el mundo le pertenecía al género humano, ya no había fronteras infranqueables ni sitios inalcanzables por el hombre y sus máquinas.

A alguien que quería estudiar física a principios del siglo XX o finales del XIX (como Planck) se le decía que esa rama del conocimiento estaba casi completa, todos los descubrimientos importantes ya se habían hecho, sólo faltaba por resolver un par de cosillas sin mucha importancia (la radiación del cuerpo negro entre ellas). Hilbert, en 1900, proponía su programa de lo que él consideraba que debía ser el desarrollo de las matemáticas futuras, lo que estaba por hacer. El segundo de los 23 problemas del programa de Hilbert era "demostrar la consistencia de los axiomas de la aritmética",

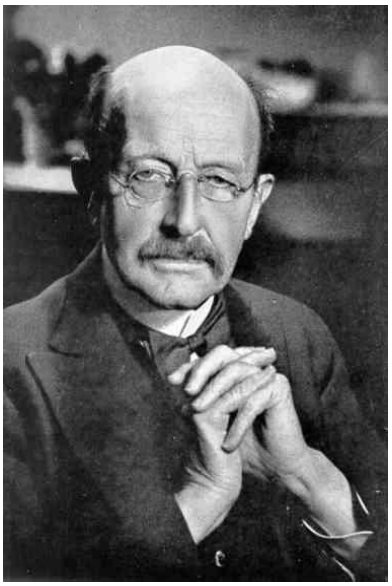

Figura 5.1: Max Planck.

esto es, probar que no es posible, partiendo de dichos axiomas, demostrar simultáneamente una proposición y su negación, asegurar que la maquinaria de la aritmética está bien construida, que es infalible, como la mecánica newtoniana. Casi al mismo tiempo otros (Whitehead y Russell) trataban de fundar las matemáticas sobre la lógica simbólica y demostrar la coherencia de ésta, es decir que el engranaje formal que se había construido a lo largo de la historia era perfecto. Una fría máquina perfecta por su forma, por su construcción, sin importar el significado, un sistema formal perfecto.

Existe una clara analogía entre sistema formal y máquina. En un sistema formal existen ciertos postulados a partir de los cuales todo el resto del sistema es construido, el sistema es todo aquello que puede deducirse a partir de los postulados, es decir, el conjunto de todos los teoremas que pueden ser demostrados a partir de los axiomas. Una máquina posee ciertos componentes que efectúan movimientos elementales regidos por las leyes de la física, que, como los postulados del sistema formal, al interactuar definen todo lo que la máquina puede hacer. Así pues, la actitud de principios de siglo fue en buena medida desencadenada por el éxito de la mecánica como "la explicación del mundo"; el programa de Hilbert, los *Principia Mathematica* de Russell y Whitehead y algunos otros intentos[1] por depurar

[1]Como los intentos de Poincaré por eliminar las paradojas del cuerpo de las ma-

Figura 5.2: David Hilbert.

y fundamentar las matemáticas en un conjunto de postulados a partir de los cuales todo el resto pudiera deducirse, son intentos por hacer con las matemáticas lo que ocurrió en la física. La actitud mecanicista invadió, como se ve, casi todos los ámbitos del conocimiento humano e hizo sentir que el hombre, finalmente, había logrado descifrar el mensaje secreto de la naturaleza.

De pronto, la naturaleza demostró, una vez más, lo insignificante y lo vanidoso que el género humano puede ser. El Titanic naufraga cerca de su destino, el barco "inhundible" se hunde; el "problemilla" de la radiación del cuerpo negro da al traste con la física, Planck nunca se arrepentirá suficiente de ello y Gödel en 1931 demuestra que el santo Grial buscado por Hilbert no existe. Un sistema formal que pueda contener a la aritmética es incompleto, (habrá verdades que no se puedan demostrar) o bien es inconsistente (se pueden demostrar simultáneamente una proposición y su negación). Russell declarará haber quedado asqueado de lógica.

¿Por qué es importante el trabajo de Gödel para las ciencias de la computación? Como bien se ha señalado en este texto, las ciencias de la computación, aún desde antes de existir, buscan calcular minimizando la cantidad

temáticas o el intento de Weyl por eliminar definiciones circulares.

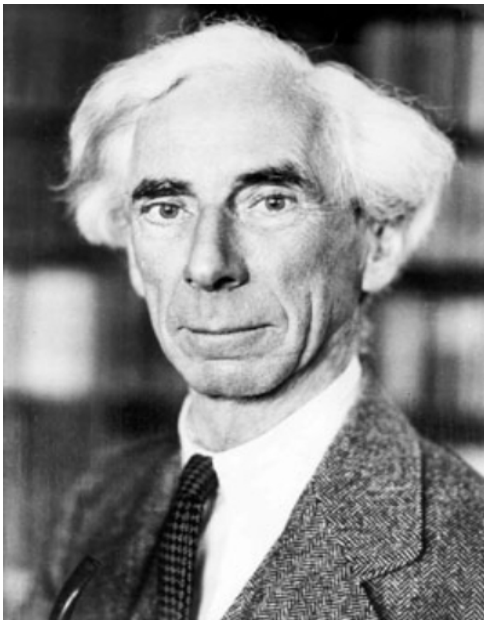

Figura 5.3: Bertrand Russell.

Figura 5.4: Kurt Gödel.

de cosas que hay que pensar para hacerlo, para esto se plantea el problema de cómo "automatizar" los procesos de cálculo, cómo hacerlos metódicos de tal forma que no se tenga que pensar en lo que se hace y en última instancia, dada la existencia de un método fijo que siempre funciona y que sólo hay que repetir en diferentes circunstancias, hacer máquinas que lo efectúen por nosotros. Ahora el espectro es un poco más amplio, Gödel demuestra que hay cosas que no se pueden deducir en un sistema formal, peor aún, demuestra que la aritmética es incompleta, esa aritmética que se utiliza para hacer cálculos. Entonces, si no todas las cosas se pueden calcular en la aritmética, o en un sistema formal en general, ¿qué cosas sí se pueden calcular? ¿cuántas cosas se pueden calcular y cuántas no? ¿cuáles son más? Gödel abrió el espectro de lo que las ciencias de la computación estudian. Ahora el problema no sólo es saber cómo calcular las cosas y cuál es la mejor manera de hacerlo, también hay que saber cuáles cosas se pueden calcular y cuáles no y luego, dando un pequeño paso más, cabe preguntarse si es posible clasificar las primeras de acuerdo al grado de dificultad para calcularlas, problema al que se enfocaron los esfuerzos de Church y Kleene.

Con frecuencia las mentes excesivamente brillantes bordean la locura. Gödel sufrió trastornos mentales al final de su vida que lo hacían un paranoico. Hacía citas con las personas para un lugar y hora precisas y jamás se presentaba, esto lo hacía, según él mismo declaró, para estar seguro de que a esa hora no se las encontraría accidentalmente. Decía también que una foto de McArthur publicada en New York Times era la de un impostor porque si se dividía el largo de la nariz por la distancia que había de la punta de la nariz hasta la barbilla el resultado no coincidía con el obtenido en otra foto que poseía el mismo Gödel.

Pero regresemos al programa de Hilbert. El décimo de los problemas planteados era el siguiente: dada una ecuación diofantina con coeficientes enteros, encontrar un método mediante el cual, con un número finito de operaciones, se pueda decidir si la ecuación tiene soluciones enteras o no.

Este problema trasladado al terreno de la lógica de primer orden (esa del "o", "y", "si, entonces", "para todo", y "existe") constituía uno de los problemas fundamentales, denominado *el problema de la decidibilidad* (Entscheidungsproblem). Dado un conjunto de axiomas y una proposición, ¿existe una serie finita de deducciones en el sistema formal que nos lleven de los axiomas a la proposición o a su negación? En términos más simples: si se dan un conjunto de premisas fundamentales y una proposición ¿hay un método, un algoritmo que permita decidir si la proposición es verdadera o no?

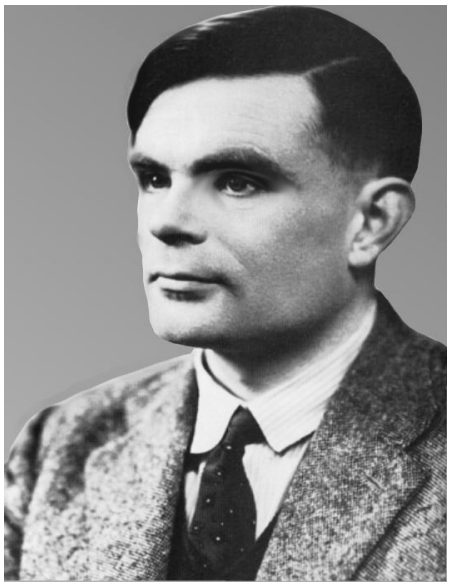

Figura 5.5: Alan Turing.

Hilbert y su grupo estaban trabajando también sobre este problema. Curiosamente Gödel demostró que el problema de la decidibilidad era soluble en ciertos sistemas formales. Por su parte, como hemos mencionado, Rusell y Whitehead trabajaban en una fundamentación de las matemáticas sobre la lógica. Resolver el problema de la decidibilidad era resolverlo para toda la matemática.

Con el fin de resolver el problema, o demostrar que no era posible resolverlo, y teniendo en mente el objetivo de clasificar problemas de acuerdo a su grado de dificultad (lo que habían empezado a hacer Alonzo Church y Stephen Kleene), un inglés llamado Alan Turing se puso a trabajar en un marco teórico que permitiera establecer qué se podía obtener tras una cadena finita de deducciones o cálculos elementales. A fin de cuentas un algoritmo es una función, recibe una entrada y produce una salida. Turing se puso a averiguar qué funciones eran calculables. Para hacer esto definió una abstracción de lo que es una máquina. Una computadora muy elemental con un conjunto mínimo de operaciones posibles. Evidentemente el objetivo de Turing no era obtener un dispositivo de cálculo, sino tener un modelo teórico, abstracto, de lo que es un dispositivo de cómputo muy sencillo y muy general al mismo tiempo. Con base en este modelo Turing afirmó que para cualquier función calculable existe una máquina de Turing

que la calcula. Es decir, la respuesta a la pregunta ¿qué se puede calcular? es *todo aquello para lo que exista una máquina de Turing que lo calcule*. Con este formalismo Turing pudo demostrar que el problema de la decidibilidad no es soluble, dados un conjunto de axiomas y una proposición, no existe un algoritmo que nos permita decidir si la proposición es verdadera en ese sistema formal.

En 1970 el décimo problema de Hilbert, tal y como él lo había planteado, fue finalmente resuelto por un estudiante de 22 años de la Universidad de Leningrado, Yuri Matiyasevich. Por supuesto Matiyasevich demostró que el procedimiento (algoritmo) buscado por Hilbert no existía.

Leibniz descubrió el cálculo infinitesimal al mismo tiempo que Newton y de manera independiente (aunque éste último no opinara igual). En 1936 ocurrió algo similar. En ese año se publicaron tres maneras diferentes de definir qué era calculable y qué no. Ya hemos hablado de Turing y sus máquinas, los otros dos fueron Alonzo Church (un argentino que por ese entonces andaba en Princeton) y Emile Post (norteamericano). La aproximación al problema por parte de Post fue muy similar a la de Turing. La aproximación de Church (que se convertiría poco después en asesor de Turing durante la estancia de éste en Princeton), fue diferente. Church definió un formalismo llamado cálculo λ. Más tarde se demostró que las tres maneras de definir computabilidad son equivalentes. De hecho existen algunos otros modelos que resultaron también equivalentes: las funciones de Herbrand, el modelo de funciones recursivas de Gödel y las funciones de Kleene.

Dado que todos los modelos de computabilidad resultan equivalentes, aún cuando estén basados en ideas muy diferentes, es plausible considerar que se ha logrado captar la esencia de lo que significa que algo es computable. Así que actualmente los científicos de la computación trabajan sobre la hipótesis de que es computable todo aquello que se puede expresar en alguno de los modelos mencionados (de hecho en todos). Podemos decir, por ejemplo, *que para todo aquello que es computable existe una máquina de Turing que lo calcula*. Esta hipótesis de trabajo es lo que se conoce como la *tesis de Church* y hasta antes de 1997 no había indicios de que fuera falsa.

Durante su estancia en Estados Unidos, Turing conoció e impresionó agradablemente a un notable matemático nacido en Hungría, consagrado por sus trabajos en geometría, álgebra y física atómica, quien por cierto había trabajado al lado de Hilbert. Su nombre era John Von Neumann y sería el coautor de la teoría de juegos junto con Morgenstern (premio Nobel de economía y buen amigo de Gödel al igual que Einstein).

Tanto Turing como Neumann se verían en el futuro diseñando, planeando y elaborando verdaderas computadoras, ya no sólo modelos teóricos, sino

Figura 5.6: John Von Neumann.

artefactos reales. Durante la segunda guerra Turing participó en el proyecto ULTRA, encargado de descifrar los códigos utilizados por los submarinos alemanes para así poder contrarrestar el efecto de éstos en la flota aliada, como resultado de este proyecto fue construida una computadora electrónica, COLOSSUS, en funcionamiento en 1943. Turing fue también pionero de algunas áreas de la computación actualmente redescubiertas y en boga, como la computación evolutiva o las redes neuronales artificiales. Al igual que Oscar Wilde años atrás, Alan Turing también fue merecedor del reconocimiento británico a sus aportaciones y fue arrestado en 1952 por ser homosexual, de lo que al parecer él mismo se sentía culpable. A diferencia de Wilde, que fue encarcelado, Turing fue condenado a recibir tratamientos hormonales. En junio de 1954, poco antes de cumplir 42 años, Turing murió envenenado, presumiblemente por su propia mano.

En 1937 un estudiante del MIT terminó su tesis de maestría. En ella sentaba las bases teóricas por las que, en buena medida, nuestras computadoras actuales utilizan el sistema binario para operar: es posible realizar operaciones lógicas (verdadero-falso, 0-1) con dispositivos eléctricos. El estudiante se llamaba Claude Shannon y más tarde fundaría lo que se conoce hoy como *teoría de la información*. Ya en 1910 al físico ruso P. Ehrenfest se le había ocurrido lo mismo al escribir una crítica sobre un libro recientemen-

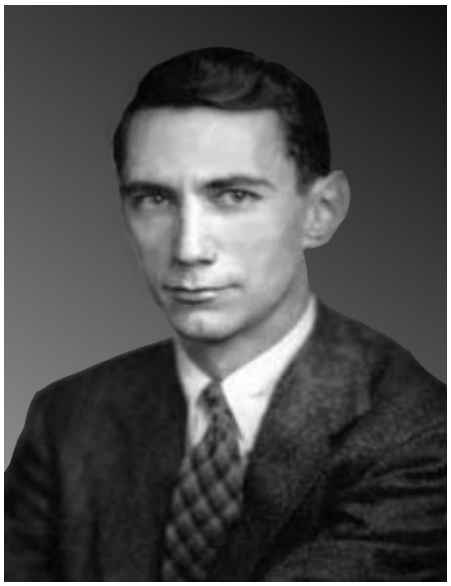

Figura 5.7: Claude Elwood Shannon.

te traducido al ruso. El libro en cuestión era *L'algèbre de la logique* escrito en 1905 por un historiador francés de la lógica, Louis Couturat, en el que se trataba de divulgar el trabajo de Boole. Ehrenfest escribió que el álgebra de la lógica puede ser realizada en redes eléctricas y aparentemente, nadie le hizo mucho caso. El trabajo de Shannon fue elaborado sin conocimiento del de Ehrenfest. En 1946 un lógico ruso, Szestankow, publicó el mismo resultado, obtenido también independientemente.

6

Modelos simples de cosas complejas

Un buen día Norbert Wiener, distinguido matemático y profesor del MIT, doctorado a los 18 años en la Universidad de Harvard, salió del trabajo rumbo a su casa, pero él y su familia se habían mudado recientemente y aún no sabía cómo llegar a su nueva casa distante unas cuadras de su anterior domicilio; tampoco encontró el papel donde su condescendiente esposa, Margaret, le había escrito las indicaciones para llegar, así que se fue a su casa anterior con la esperanza de encontrar a alguien que pudiera indicarle cómo llegar a la nueva. Luego de un rato apareció una niña a la que preguntó si sabía a dónde se habían mudado los Wiener, la niña respondió, "sí papá, mi madre me envió a buscarte pensando que probablemente estarías aquí para que te llevara a casa". Wiener era el prototipo del científico distraído, pero también el del científico comprometido moral y socialmente con el buen uso de la ciencia. A lo largo de su vida se interesó en muchas cosas, desde el movimiento browniano hasta la fisiología.

Precisamente de su interés por la fisiología, que compartía con su amigo el médico mexicano Arturo Rosenblueth, y por la ingeniería, surgió lo que hoy día se conoce como *cibernética*, término acuñado por el mismo Wiener en la segunda mitad de la década de los cuarentas y que guarda una relación muy estrecha con la computación y la teoría de la información. El objeto de estudio de la cibernética es, como lo dice el subtítulo del libro de Wiener (Cibernética, 1948) "el control y la comunicación en el animal y la máquina", es decir el estudio de los mecanismos que rigen la comunicación entre diversos sistemas en los organismos vivos y sus símiles en los ingenios construidos por los seres humanos. Así como existen mecanismos reguladores de la temperatura en los seres vivos de sangre caliente también existen termostatos

Figura 6.1: Norbert Wiener.

que regulan la temperatura de ciertas máquinas. Para lograr dicho control se deben establecer ligas entre diversos sensores de temperatura y la entidad encargada de regularla, así como un mecanismo de retroalimentación que permita decidir qué acciones tomar con base en los efectos producidos por las acciones previas sobre el estado del medio a controlar. Los ejemplos son innumerables y su estudio ha permitido el desarrollo de *servomecanismos*, dispositivos que interactúan con el medio recibiendo estímulos de él como entrada y provocando, como salida, efectos medibles que permiten decidir qué tan cercanamente la situación real se ajusta a la deseada.

En 1943 Warren S. McCulloch, un médico que trabajaba en el laboratorio de investigación en electrónica del MIT y Walter Pitts, un matemático que más tarde (1947) iría a trabajar al lado de Wiener, publicaron un artículo en el boletín de biofisiología matemática "Un Cálculo Lógico de las Ideas Inmanentes a la Actividad Nerviosa" (*A Logical Calculus of the Ideas Immanent in Nervous Activity*). El trabajo fue resultado de investigar cómo es que, dado el funcionamiento relativamente sencillo de una sola neurona y el modo en que estas se interconectan, estos elementos se combinan para darle al cerebro su complejidad y sus enormes habilidades, lo que hoy llamaríamos sus *propiedades emergentes*. El pensamiento, la memoria visual y auditiva, el lenguaje, todos son procesos cuya complejidad no está explícitamente es-

pecificada en cada neurona, sólo *surgen* o *emergen* de la interacción de estos simples elementos. La idea de McCulloch y Pitts fue elaborar un modelo lógico de las redes nerviosas, suponiendo que la actividad de cada neurona es un proceso de todo o nada (o se excita o no lo hace ante un estímulo) y por lo tanto es posible utilizar la lógica simbólica para modelar su comportamiento individual y el de conjuntos de neuronas interconectadas: redes neuronales.

El trabajo de McCulloch y Pitts entusiasmó a muchas personas, entre ellas a Von Neumann y a Wiener. Por una parte el trabajo de Turing modelando dispositivos de cómputo, por otra el de Gödel, que hacía de los conceptos de la lógica algo recursivo, que podía calcularse con una máquina de Turing y finalmente el trabajo sobre la relación entre redes neuronales y lógica estimularon a Von Neumann que había trabajado, hacía ya bastante tiempo, en lógica matemática. En 1947 Von Neumann comenzó a trabajar en la pregunta ¿qué tan complejo debe ser un dispositivo para reproducirse a sí mismo?, esto inspirado en la posibilidad de que una máquina de Turing puede recibir como entrada la especificación de otra máquina de Turing.

Otros interesados en el trabajo de McCulloch y Pitts fueron Huffman, Mealy, Kleene y Moore quienes, de diferentes maneras, se dieron a la tarea de formalizar el modelo y enmarcarlo en uno más general. De estos trabajos, de la segunda mitad de la década de los 1950's, surgió lo que hoy conocemos como la teoría de autómatas, a los autores mencionados les debemos en particular los autómatas finitos, a los que se les dió (Myhill, 1963) todo un tratamiento algebraico basado en teoría de semigrupos.

Casi al mismo tiempo, de 1956 a 1959, un lingüista del MIT, Noam Chomsky, quizás mejor conocido por ser un disidente político de izquierda anarquista, en protesta permanente por el intervencionismo norteamericano en los países en desarrollo, se dió a la tarea de clasificar lenguajes simbólicos caracterizándolos algebráicamente por el tipo de transformaciones que utilizan.

Un lenguaje está constituido, en términos simples, por el conjunto de posibles palabras en él, su diccionario. Así que definir un lenguaje consiste en definir el conjunto de todas las posibles palabras que lo constituyen. Supongamos que queremos definir un lenguaje muy simple, el que está constituido por todas las palabras de longitud arbitraria (incluyendo longitud cero, es decir la palabra vacía), en las que siempre aparece una letra "a", después de cualquier otra letra, es decir palabras de la forma: $s_1 a s_2 a \ldots s_k a$ donde cada uno de los símbolos s_i es una letra del conjunto $\{b, \ldots z\}$. Este es un lenguaje muy tonto, pero servirá de ejemplo.

Es posible definir las palabras del lenguaje mediante una *gramática*, es decir, una serie de reglas que permite construirlas. Para este ejemplo podemos hacer lo siguiente: mediante γ denotaremos cualquier letra en el conjunto $\Sigma = \{b, \ldots, z\}$. Con el símbolo A denotaremos una palabra cualquiera de nuestro lenguaje. Ahora bien, sabemos que A debe terminar con una "a", así que A la podemos construir poniendo algunas letras y terminando con una "a", esto se escribe

$$A \longrightarrow Ba$$

donde B es un símbolo que utilizamos para denotar toda la palabra salvo su última letra (la "a"). Como también queremos incluir como parte de nuestro lenguaje a la palabra vacía (sin letras) entonces tenemos que decir también que

$$A \longrightarrow \varepsilon$$

donde ε denota la palabra vacía. Ambas cosas las podemos resumir diciendo que A puede ser Ba o ε de la siguiente forma:

$$A \longrightarrow Ba|\varepsilon$$

donde el símbolo | denota esa disyuntiva.

Pero a su vez esta "subpalabra" B debe ser construida siguiendo ciertas reglas, de hecho sabemos que debe terminar con alguna letra del conjunto Σ, nuestro alfabeto, es decir:

$$B \longrightarrow C\gamma$$

donde γ es, como ya dijimos, una letra del conjunto Σ. Nuevamente hemos usado un símbolo: C, para denotar la palabra completa salvo sus dos últimas letras[1], es decir la "subpalabra" de la que se excluyen la última y penúltima letras. Por supuesto C también debe construirse mediante alguna regla, sabemos que debe terminar con una "a". Pero esto ya lo hemos hecho, utilizamos el símbolo A para denotar a las palabras que terminaban con "a", así que lo podemos reutilizar y decir:

$$C \longrightarrow A$$

Tenemos entonces las siguientes reglas de construcción:

1. $A \longrightarrow Ba|\varepsilon$
2. $B \longrightarrow C\gamma$
3. $C \longrightarrow A$

[1] A B le faltaba la última y esta regla dice que B está hecha de C y una letra más, así que a C le faltan las dos últimas letras.

Con estas reglas es posible construir palabras como: *mama, papa, pata, sara, saca, mapa, haba, oaxaca, xalapa, patata, banana* y *palapa*, pero no palabras como *perro* o *sapo*. Para construir la palabra *pata* por ejemplo utilizamos primero la regla 1 y decimos que nuestra palabra termina con "a" y antes tiene una subpalabra B, luego decimos, usando la regla 2, que esa subpalabra B termina con alguna letra del alfabeto, de hecho una "t" y antes tiene una subpalabra C, luego usando la regla 3 que nos remite a la 1, decimos que C termina con una "a" y antes tiene una subcadena a la que nuevamente llamamos B, usando la regla 2 decimos que B termina con una letra del alfabeto que es "p" y que antes posee una subcadena llamada C, acudiendo a la definición de C en la regla 3, que nos remite nuevamente a la 1, decimos que esa subcadena es ε, es decir nada (palabra vacía), así que nuestra palabra es *pata*.

Las ecuaciones o reglas que escribimos arriba usando una flecha para indicar en qué se puede transformar un símbolo dado se llaman *producciones* o *transformaciones* y la labor de Chomsky fue clasificar los lenguajes en función del tipo de producciones que poseen sus gramáticas. En nuestro ejemplo utilizamos una gramática del tipo más sencillo, llamada gramática regular o de tipo 3 en la clasificación de Chomsky, pero existen otros tres tipos de gramáticas que definen lenguajes simbólicos más complejos conforme sus reglas son menos restrictivas: libres de contexto (tipo 2), sensibles al contexto (tipo 1) y finalmente las que definen lenguajes generales o *recursivamente numerables* (tipo 0).

Alrededor de 1960 algunos investigadores, Bar-Hillel y Shamir por una parte y Kleene por otra, se dieron cuenta de que existía una relación entre las gramáticas regulares y cierto tipo de autómatas de estados finitos (aceptores): un autómata de estado finito puede decidir si una palabra (o para hablar propiamente, una cadena), forma parte de un lenguaje regular o no. Para toda gramática regular existe un autómata finito que decide si una cadena arbitraria forma o no parte del lenguaje generado por la gramática y a la inversa, todo autómata finito acepta un conjunto de cadenas que constituye un lenguaje regular. Más tarde, en 1961, el trabajo de Bar-Hillel, Shamir y Perles acerca de lenguajes libres de contexto y los trabajos de Chomsky de 1962 y 1963 establecieron una equivalencia análoga entre este tipo de lenguajes y otro tipo de autómatas denominados *autómatas de pila* (o *stack* en inglés). En 1960 Myhill definió otro tipo de autómatas denominado *autómata linealmente acotado* que resultaron ser aquellos que deciden si una cadena es o no parte de un lenguaje sensible al contexto. Finalmente, muy en relación con el trabajo de Gödel y su definición de computabilidad,

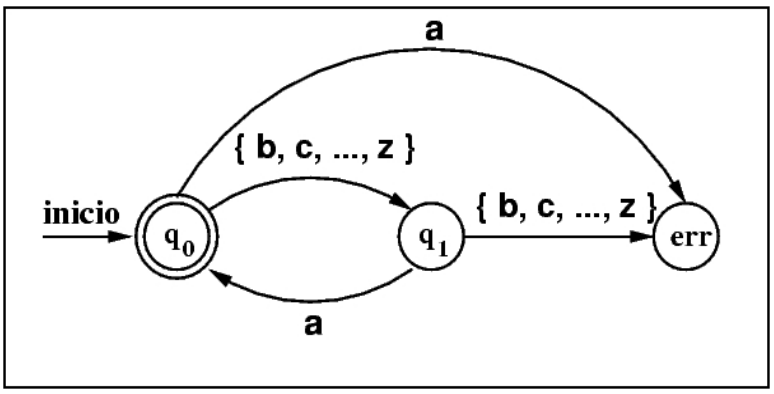

Figura 6.2: Un autómata finito para reconocer las cadenas del lenguaje utilizado como ejemplo. $\gamma \in \{b, \ldots, z\}$. Este autómata termina en estado final de aceptación (el de doble círculo) cuando recibe una cadena con la letra "a" en todas las posiciones impares de la palabra (contando de derecha a izquierda), como *mama, papa, patata* o *banana*.

se estableció una equivalencia entre las máquinas de Turing y los lenguajes recursivamente numerables de la clasificación de Chomsky.

En la figura 6.2 se muestra una representación gráfica del autómata finito que reconoce el lenguaje que utilizamos de ejemplo. Cada círculo (nodo) representa un posible estado de la máquina abstracta y está etiquetado con un nombre (q_0, q_1, o err). Cada flecha parte de un estado y llega a otro, esta transición de estado se efectúa cuando se recibe en la entrada los caracteres que etiquetan la transición. El autómata lee, letra por letra una cadena de entrada; si luego de leerla toda termina en el estado final (el de doble círculo) entonces la cadena forma parte del lenguaje que definimos en nuestro ejemplo.

Paralelamente a estos trabajos acerca de lenguajes y máquinas abstractas, se hacían esfuerzos para interconectar interruptores (*switches*) de manera confiable y estudiar las propiedades de las redes de *switcheo* (o redes de conmutación, si hablamos con propiedad) que resultan de dicha interconexión. A fin de cuentas las operaciones en las computadoras se realizan interconectando circuitos que calculan básicamente, funciones lógicas, funciones de *switcheo* que entregan un cero o un uno en función de los valores que reciben de entrada (que a su vez son conjuntos de ceros y/o unos). En este sentido se orientaron los trabajos de Shannon desde 1938 y más tarde de Moore (el mismo de los autómatas mencionado antes). En 1952 W. V.

Quine formuló un método para simplificar funciones de *switcheo* utilizando un modelo algebraico que fue retomado por E. J. McCluskey Jr. en 1956 dejándolo en la forma en que es utilizado por los diseñadores de circuitos de hoy en día.

Habíamos dicho que alrededor de 1947 comenzó a dar vueltas en la mente de Von Neumann una mezcla de ideas de Turing (máquinas de Turing, en particular que pueden recibir como entrada la especificación de otra máquina y simularla), Gödel (lógica matemática, conceptos recursivos) y McCulloch-Pitts (análisis lógico de modelos neuronales,). En 1948 Von Neumann ya poseía una primera aproximación a lo que el llamó *modelo cinemático* de una máquina autorreproducible, que pudiera construir copias de sí misma. Luego, en una conversación con el célebre Ulam, éste lo convenció de que era mayor el potencial de poseer una red de autómatas de estados interconectados para lograr sus propósitos, así que en 1952 elaboró un nuevo modelo para máquinas autorreproducibles al que llamó *modelo celular*, que consiste básicamente en una cuadrícula bidimensional en la que en cada cuadro se pone un autómata de estados (de hecho en su modelo poseía 29 estados), en cada instante de tiempo (que avanza en pasos discretos y no de manera continua) cada autómata en esa malla posee un estado que es determinado por su estado mismo y el de algunos de sus vecinos en el paso de tiempo inmediato anterior. Esto dio lugar a lo que ahora conocemos como *autómatas celulares*: un conjunto finito de celdas ordenadas en una malla regular donde cada celda puede estar, en todo instante de tiempo en uno de un conjunto de estados posibles y este estado es determinado por el estado mismo de la celda y el de sus celdas vecinas en el instante anterior. En el caso de Von Neumann los vecinos de una celda eran las cuatro celdas que se encuentran arriba, abajo, a la izquierda y a la derecha de la celda en cuestión; existen muchas otras definiciones de *vecindad* para una celda en autómatas celulares. En la definición de Moore, por ejemplo (el mismo Moore de los autómatas finitos) la vecindad está constituida, además de las celdas de la vecindad de von Neumann por las que están inmediatamente sobre las dos diagonales que pasan por la celda en cuestión.

Desde que Von Neumann concibió los autómatas celulares se le ocurrió analizar su comportamiento usando modelos matemáticos continuos, tratarlos con ecuaciones diferenciales; éstas constituyen el corazón de la teoría de sistemas dinámicos elaborada por George Birkhoff desde 1927. Von Neumann no estaba errado: actualmente los autómatas celulares son estudiados como sistemas dinámicos, aunque discretos (porque el tiempo avanza en pasos y los valores posibles en cada celda pertenecen a un conjunto finito).

A finales de los 50, continuando el trabajo de McCulloch y Pitts (que como se ve resulta ser de los que más impacto han tenido), Frank Rosenblatt, un psicólogo, definió el *perceptrón*, un modelo simple de neurona, con el fin de "ilustrar algunas de las propiedades fundamentales de los sistemas inteligentes en general, sin profundizar demasiado en las condiciones especiales y frecuentemente desconocidas que ocurren en los organismos vivos". El modelo de Rosenblatt es probabilístico a diferencia del de McCollock-Pitts que está basado en la lógica simbólica. Pero más tarde Marvin Minsky retomó el perceptrón de Rosenblatt e hizo dos cosas: regresó el modelo al terreno de la lógica simbólica e hizo que el interés en las redes neuronales decayera de 1969 a 1980. En 69 Minsky y Seymour Papert publicaron un libro "Perceptrons" en el MIT, en el demostraban que el perceptrón posee serias limitaciones como elemento de cómputo.

Las neuronas en los seres vivos son las encargadas de recibir estímulos del medio y en función de la magnitud del estímulo recibido, transmiten o no una señal al resto de neuronas conectadas a ellas. Cuando colocamos la mano en un recipiente que ha sido puesto al fuego recientemente, nuestras neuronas pueden percibir el calor, que al principio no es excesivo, pero que conforme pasa el tiempo se vuelve doloroso; si se sobrepasa cierto umbral de dolor nuestro cerebro decide que debemos retirar la mano para que ésta no sufra daños de consideración. Básicamente un perceptrón es una caja negra que recibe ciertos estímulos del medio a través de algunas entradas; cada uno de estos estímulos posee cierta importancia, los estímulos pueden no ser igual de relevantes. El perceptrón se encarga entonces de discernir si el estímulo total recibido del medio es suficientemente importante como para ponerlo en acción, de ser así produce una respuesta, en otro caso permanece en reposo y su respuesta es nula. Así pues un perceptrón posee un conjunto de n diferentes entradas binarias (cero o uno), a cada entrada (x_i) se le asocia un grado de importancia o peso (w_i). El perceptrón calcula el estímulo total:

$$T = \sum_{i=1}^{n} x_i w_i$$

si el valor de T rebasa cierto valor de umbral θ entonces el perceptrón produce un 1; en otro caso produce un 0.

En su trabajo Minsky y Papert demostraron que un perceptrón no puede diferenciar patrones que no sean *linealmente separables*, esto quiere decir que no puede calcular ciertas funciones sencillas, por ejemplo un O-exclusivo (*XOR* en la jerga de computación) como se puede ver en la tabla 6.1.

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 6.1: Tabla de verdad de la función O-exclusivo (XOR).

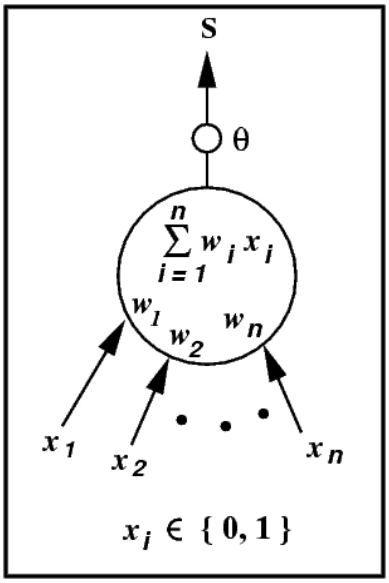

Figura 6.3: Modelo esquemático del perceptrón. Las entradas (x_i) son binarias, el perceptrón realiza la suma ponderada de sus entradas usando ciertos pesos (w_i) y si esta suma excede o iguala el valor del umbral (θ), entonces la salida del perceptrón (S) es 1, en caso contrario la salida es 0.

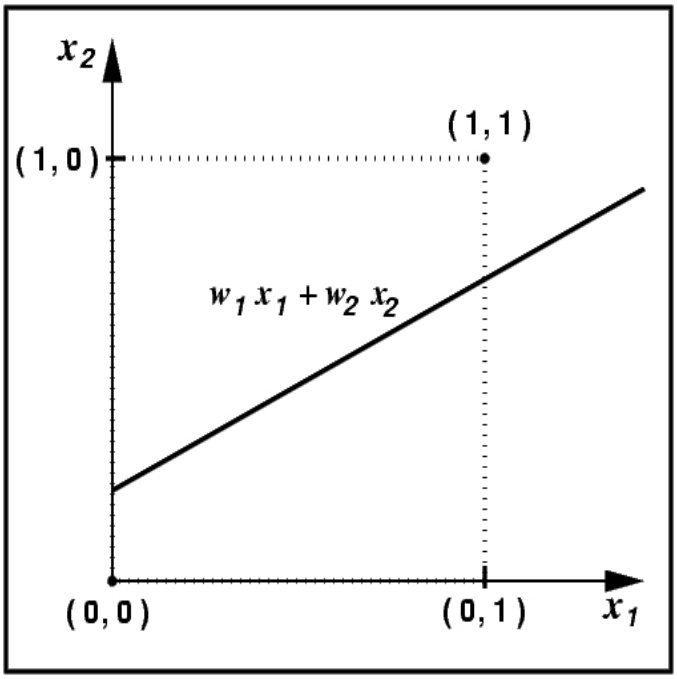

Figura 6.4: La disyunción exclusiva no es una función linealmente separable, por lo que no puede ser calculada por un solo perceptrón.

Es sencillo verificar que un perceptrón no puede calcular la función mencionada. Supongamos que un perceptrón, como el mostrado en la figura 6.3 recibe como entradas x_1 y x_2 y que deseamos que su salida sea $(x_1\ XOR\ x_2)$. Sabemos que la salida S es tal que

$$S = \begin{cases} 1 & si \quad w_1x_1 + w_2x_2 \geq \theta \\ 0 & si \quad w_1x_1 + w_2x_2 < \theta \end{cases}$$

así que hay que elegir correctamente w_1, w_2 y θ. Si hemos elegido θ de alguna manera ¿cuándo ocurre que $w_1x_1 + w_2x_2 = \theta$?. Esto pasa siempre a lo largo de una línea en el plano definido por x_1 y x_2 como se muestra en la figura 6.4. Pero sabemos que la salida del perceptrón debe ser igual (cero) en los casos en que la entrada es $(0, 0)$ y $(1, 1)$ y también debe ser igual (uno) en los casos en que la entrada es $(0, 1)$ o $(1, 0)$ y no es posible trazar ninguna línea tal que nos divida el plano en dos regiones (como todas las líneas) y deje al $(0, 0)$ del mismo lado que al $(1, 1)$ y por otra parte (en el otro lado) al $(1, 0)$ del mismo lado que al $(0, 1)$.

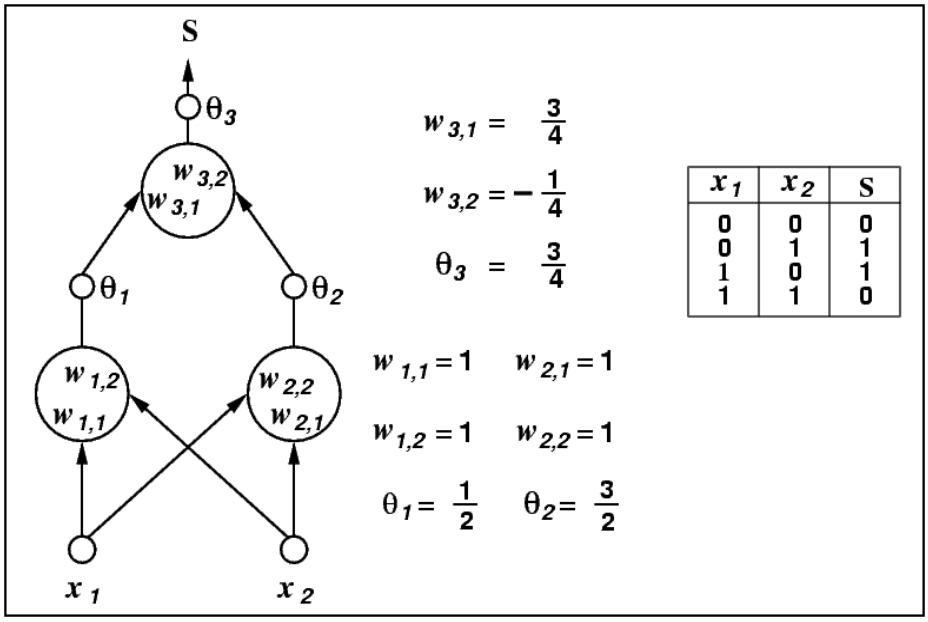

Figura 6.5: Cálculo del XOR con dos niveles de perceptrones.

Más tarde se recuperó el interés en las redes neuronales; se han elaborado algoritmos para entrenarlas y hacer que conjuntos de "neuronas", ordenados en capas, aprendan. Hoy en día son utilizadas para el reconocimiento de patrones. Por cierto; el cálculo del XOR puede ser llevado a cabo por dos capas de perceptrones como se muestra en la figura 6.5.

7

Desarrollo tecnológico

En el mismo año en que se publicaron los trabajos de Turing y Church, un alemán de 26 años, llamado Konrad Zuse, trabajaba asiduamente en la sala de la casa de sus padres con el propósito de construir una calculadora electromecánica, la Z1. Este modelo de prueba nunca funcionó debido a las imperfecciones mecánicas de los elementos que la constituían. Posteriormente el joven elaboró otros modelos basados en el primero: la Z2[1], que en 1940 se convirtió en la primera calculadora electromecánica funcional y la Z3 terminada en 1941. En 1943 el ministerio Alemán del Aire ordenó a Zuse la construcción de una computadora de propósito general, la Z4. Zuse trabajaba en y para Alemania, sus calculadoras estaban financiadas por el gobierno del Reich y de hecho algunas fueron destruidas durante los bombardeos aliados sobre Alemania, hacia el fin de la guerra en 1944. Si COLOSSUS no hubiera sido hecha un poco antes que las máquinas alemanas, si el gobierno alemán hubiera estado consciente del potencial de los trabajos de Zuse un poco antes y si éste hubiera utilizado válvulas de vacío (lo que conocemos como bulbos), probablemente viviríamos en un mundo muy diferente.

Entre 1937 y 1944, en Estados Unidos, el profesor Howard Aiken, de la Universidad de Harvard, construyó una computadora electromecánica (como las de Zuse) llamada Mark I o bien *IBM Automatic Secuence Controlled Calculator* (ASCC), donde se retoma el concepto de dar las instrucciones y datos a través de agujeros en una superficie, en esta ocasión una cinta

[1]Los nombres originales con los que Zuse bautizó a sus máquinas eran V1, V2, etc. pero después de la guerra decidió cambiarles el nombre para evitar confusiones con los cohetes que su amigo, Von Braun, diseñaba.

de papel. ASCC podía sumar o restar dos números de veintitrés dígitos en tres décimas de segundo, multiplicarlos en cuatro segundos y dividirlos en diez. Entregaba los resultados perforando unas tarjetas o bien escribiéndolos mediante un par de máquinas de escribir, un ancestro de las impresoras. La Mark I estaba hecha con unos 3,300 relevadores, una especie de interruptores operados eléctricamente.

En 1943, también en los Estados Unidos, John Von Neumann participaba como consultor en un proyecto formulado por la Universidad de Pennsylvania, con el fin de resolver rápidamente los problemas balísticos de la artillería. Presper Eckert, John Mauchly y Herman Goldstine son los principales involucrados en el proyecto. Por desgracia esta computadora no estuvo lista sino hasta febrero de 1946, demasiado tarde para ayudar en la guerra. La computadora de Eckert y Mauchly, llamada ENIAC (*Electronic Numerical Integrator and Computer*), hacía 300 multiplicaciones por segundo usando la representación numérica en base diez y no la binaria (en base dos), que utilizan nuestras computadoras actuales. ENIAC constaba de 18,000 válvulas de vacío, pesaba 30 toneladas y ocupaba 180 metros cuadrados.

Por supuesto ENIAC era una máquina programable, pero de una manera radicalmente diferente a la que estamos acostumbrados hoy en día. Para especificar el algoritmo a ejecutar, el operador debía ir a un panel de control donde había decenas de cables enchufados, desenchufar varios de ellos y enchufarlos en otro lugar. Esto significa que programar a ENIAC consistía, esencialmente, en reconfigurar sus circuitos para hacer específicamente los cálculos requeridos, había que hacer una computadora diferente por cada programa que se quisiera ejecutar. ENIAC, la primera computadora electrónica de propósito general, era en realidad una familia de computadoras de propósito específico. La ardua y delicada labor de programación era asignada típicamente a mujeres, que suelen ser mucho más cuidadosas que los hombres.

Eckert y Mauchly declararon haber trabajado las ideas fundamentales de ENIAC con un helado y una taza de café en un restaurante de Philadelphia.

También en 1943 Von Neumann comienza el proyecto de otra computadora más general y poderosa que ENIAC. La EDVAC (*Electronic Discrete Variable Automatic Computer*) es el primer prototipo verdadero de nuestras modernas computadoras. En un documento titulado *Primer borrador de reporte sobre* EDVAC (*First Draft of a Report on the* EDVAC), Von Neumann describe los mecanismos necesarios para almacenar el programa a ejecutar en la memoria de la computadora misma, lo que hace posible cambiarlo fácilmente ya que no hay que cablear el programa. De allí surge la esencia de lo que conocemos hoy en día como "arquitectura de Von Neumann": no

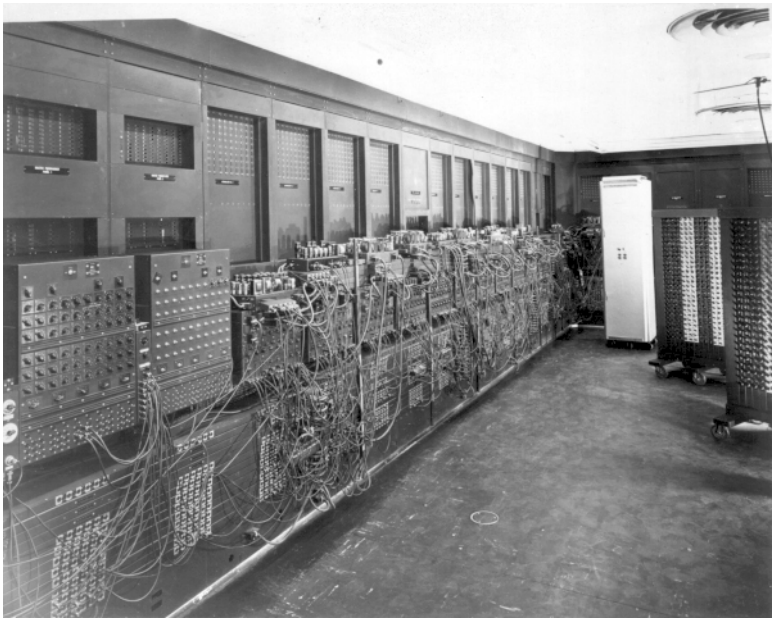

Figura 7.1: ENIAC.

sólo los datos y los resultados se guardan en la memoria de la computadora, sino también el programa, el conjunto de operaciones a realizar con los datos. La arquitectura de Von Neumann consiste de tres partes esenciales: una unidad central de proceso (comúnmente llamada CPU por sus siglas en inglés), que se encarga del cómputo propiamente dicho; una memoria para almacenar datos y programas en ejecución y los dispositivos de interfaz con el exterior, es decir aquellos que permiten la entrada de datos y programas y los que permiten la entrega de resultados, lo que suele llamarse el sistema de entrada/salida.

Vinieron después otras computadoras diseñadas con las ideas de Von Neumann y que aprovecharon bastante bien las experiencias anteriores. La EDSAC británica y la muy famosa UNIVAC (*Universal Automatic Computer*) diseñada por Eckert y Mauchly[2]. De hecho la UNIVAC era algo así como el arquetipo de la computadora en la década de los cincuentas; podemos verla hasta en algunos dibujos animados de la Warner Brothers de la época (analizando un caso policíaco difícil para Daffy). La UNIVAC también ganó fama al predecir correctamente los resultados de las elecciones presidencia-

[2]Luego de ENIAC, Eckert y Mauchly diseñaron la BINAC, que, a diferencia de aquella, operaba completamente en binario (base 2). La UNIVAC es la sucesora inmediata de BINAC.

Figura 7.2: UNIVAC. No es fortuita la aparición de mujeres en la fotografía, tampoco lo es en relación con la ENIAC. La programación de las primeras computadoras casi siempre estuvo a cargo de mujeres.

les de Estados Unidos en 1952, en las que Eisenhower obtuvo el cargo; los analistas políticos le daban la victoria a Adlai Stevenson, su oponente, por un amplio margen. La UNIVAC procesaba datos parciales de la elección y, con base en las tendencias predecía al ganador y los porcentajes por estado. La jornada electoral estaba siendo televisada y el conductor leía los resultados entregados por la UNIVAC, al notar que no concordaban con lo que predecían los analistas, los operadores comenzaron a falsear los resultados entregados por la computadora, hasta que resultó imposible hacerlo y acabaron confesando su impotencia ante el supuesto mal funcionamiento de la máquina, afortunadamente todo salió bien cuando el gobierno anunció sus propios resultados, que le daban la ventaja a Eisenhower justo en los mismos porcentajes predichos por la UNIVAC.

De hecho, el proyecto de construcción de la UNIVAC fue el que impulsó a Eckert y Mauchly a fundar la *Eckert-Mauchly Computer Corporation* (EMCC) a finales de 1947. Pero los costos de producción del aparato excedieron los pronósticos y EMCC, recién nacida, se vió al borde la bancarrota. La situación fue salvada por la compañía *Remington-Rand*, que adquirió

Figura 7.3: Presper Eckert (derecha) y John Mauchly (izquierda).

EMCC en febrero de 1950, lo que permitió entregar la primera UNIVAC en marzo de 1951 a aquél que fuera el primer cliente de IBM y su *alma mater*, por así decirlo: la oficina del censo de los Estados Unidos.

Remington-Rand cambiaría de nombre varias veces en el futuro. Al fusionarse con la compañía *Sperry* se convirtió en *Sperry-Rand* y más tarde regresaría a ser *Sperry*. Finalmente, luego de una fusión con Burroughs, se convirtió en la actual *Unisys*, a quien pertenecen hoy en día las patentes de ENIAC.

En 1973, siendo *Sperry-Rand*, la compañía se vió envuelta en un litigio legal por determinar cuál era la primera computadora electrónica de la historia. Hasta antes de ese año se había considerado a ENIAC la poseedora de tal calificativo y por tanto *Sperry-Rand* era la dueña de las patentes de tan significativo artefacto. La compañía *Honeywell* argumentaba que esto no era cierto y para demostrarlo sacó a la luz una historia hasta entonces desconocida.

Entre 1937 y 1942 un profesor de física de la Universidad Estatal de Iowa en Ames, John Vincent Atanasoff, construyó junto con uno de sus estudiantes, Clifford Berry, una computadora electrónica digital (esta vez binaria), con el fin de resolver sistemas de ecuaciones lineales simultáneas. La

máquina se llamó ABC (*Atanasoff-Berry Computer*) y fue capaz de resolver sistemas de hasta 29 ecuaciones con 29 incógnitas. Las ideas fundamentales de esta máquina se le ocurrieron a Atanasoff en un bar mientras bebía una copa de bourbon con agua.

En agosto de 1940 Atanasoff escribió un detallado reporte técnico del avance de la ABC y en diciembre de ese mismo año conoció a Mauchly en Philadelphia, en una reunión de la Asociación Americana para el Avance de la Ciencia. Allí por primera vez le habló de su computadora y lo invitó a Ames donde le mostró el reporte y le explicó los principios de operación y los detalles del diseño.

A pesar de que Mauchly dijo a Atanasoff que le interesaban más las computadoras analógicas que las digitales, al parecer, utilizó algunas de las ideas de Atanasoff en la elaboración de ENIAC, su propia computadora digital. A esto hay que añadir que Atanasoff fue llamado al principio de la segunda guerra mundial para hacer trabajos de física relacionados con minas y cargas de profundidad para la armada norteamericana. La Universidad de Iowa aseguró los derechos de su trabajo, pero no continuó el proceso de patente al que Atanasoff no pudo dar seguimiento por su trabajo militar.

Todo esto llevó al juez Larson de Minneapolis a declarar que ENIAC fue derivada de la ABC y por tanto de las ideas de Atanasoff y falló a favor de *Honeywell* en octubre de 1973, con lo que *Sperry-Rand* ya no puede presumir de poseer las patentes de la primera computadora electrónica de la historia e implícitamente, ENIAC ya no puede ser calificada como tal.

En 1990 el presidente Bush (padre) otorgó a Atanasoff la medalla nacional de tecnología en reconocimiento a su trabajo pionero. Cinco años después, a la edad de 91 años, Atanasoff murió.

Al parecer el error de Mauchly fue nunca haber dado a Atanasoff el crédito que merecía al haber diseñado la primera computadora electrónica digital. De hecho él y Mauchly cometieron un error similar con el resto del equipo de desarrollo de ENIAC. Por supuesto la máquina dista mucho de ser creación exclusiva de ambos; tras ellos estaba todo un grupo de ingenieros que participó con ideas y resolvió problemas no triviales ni siquiera sospechados por los coordinadores del proyecto. Por otra parte hay también un error de apreciación, la ABC no era del todo una máquina de propósito general por lo que no se podría afirmar que fue *la primera computadora electrónica digital de propósito general* de la historia.

Dos cosas vienen a la mente después de todo esto:

- A pesar de no ser un logro éticamente limpio, no hay que menospreciar la titánica labor intelectual de Eckert y Mauchly.
- El bourbon da mejores ideas que el helado.

Pero regresemos a la UNIVAC, que había sido adquirida por la oficina del censo de los Estados Unidos, invadiendo lo que había sido el santuario de IBM. La respuesta no se dejó esperar, en mayo de 1952 se anunció el lanzamiento de la IBM 701. Una computadora de "programa guardado" (la idea de Von Neumann) igual que la UNIVAC. A partir de ese momento despegó la competencia real en el ámbito de la computación electrónica, a pesar del restringido mercado. Para fortuna de las compañías fabricantes de este tipo de equipos, la guerra fría hacía necesario el cómputo científico preciso y rápido. Para 1955 unas nueve empresas compartían el mercado, algunas muy conocidas, aunque no en el rubro de la computación como: *General Electric*, *RCA*, *Philco*, *NCR* o *Western Electric* (división de manufactura de AT&T). Otras más famosas en el ramo eran: *Honeywell*, *Burroughs*, *Raytheon* y *Remington-Rand* (que para entonces ya se había fusionado con *Sperry Corporation*).

Las máquinas de Zuse y la Mark I operaban con relevadores, eran electromecánicas, la corriente eléctrica que circulaba por ellas tenía como propósito abrir o cerrar interruptores. La ENIAC, la EDVAC, la EDSAC y la UNIVAC trabajaban con válvulas de vacío, también llamados *tubos de vacío* o bulbos en lenguaje coloquial. En estas máquinas no había movimientos mecánicos involucrados su funcionamiento esencial, los bulbos hacían las veces de los relevadores. Ambos son, en esencia, interruptores.

En 1947 ocurrió un acontecimiento que revolucionó la electrónica. En los laboratorios *Bell* de la AT&T, tres físicos que experimentaban con materiales semiconductores, germanio para ser precisos, descubrieron el transistor, lo que les hizo acreedores al premio Nobel de física en 1956; sus nombres eran William Shockley, Walter Brattain y John Bardeen. Un semiconductor es un material que, de ser completamente puro, no conduce la electricidad, pero si se le añaden algunas impurezas de algún otro elemento y se le suministra un cierto voltaje entonces sí puede hacerlo. Esto es equivalente a un interruptor, así que es posible diseñar computadoras usando transistores en vez de válvulas de vacío. Las ventajas de usar transistores en vez de válvulas de vacío son las mismas que existen entre usar estas últimas y utilizar relevadores electromecánicos: se gana velocidad de operación, se reduce el consumo de energía y se reduce el tamaño.

Figura 7.4: William Shockley (izq), Walter Brattain y John Bardeen (der).

Podríamos preguntarnos ahora el por qué es tan importante tener interruptores, llámense estos relevadores, bulbos o transistores.

Ya hemos hablado de Boole y su algebraización de la lógica: en vez de manipular enunciados y aplicar las reglas de la lógica para determinar la veracidad o falsedad de las proposiciones, usamos símbolos y manipulaciones algebraicas. Una operación lógica como la conjunción de dos enunciados: *el pasto es verde Y el cielo es azul*, se traduce en algo como: $A \wedge B$ o mejor aún como AB. Una disyunción como: *el pasto es verde O el cielo es verde*, se representa como: $A \vee B$ o $A + B$. Sabemos que para que una conjunción de dos enunciados sea verdadera se requiere que ambos enunciados lo sean, por eso es falso el enunciado: *el pasto es verde Y el cielo es verde*, y también sabemos que para que una disyunción sea falsa ambos enunciados deben serlo, es verdadero el enunciado *el pasto es verde O el cielo es verde* porque el primer enunciado que lo compone es verdadero. Si representamos el valor de verdad *verdadero* como 1 y el *falso* como 0, podemos escribir todas las posibles combinaciones de valores de verdad de A y B y los resultados de la conjunción y disyunción de ellos en una tabla:

A	B	$A+B$	AB
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

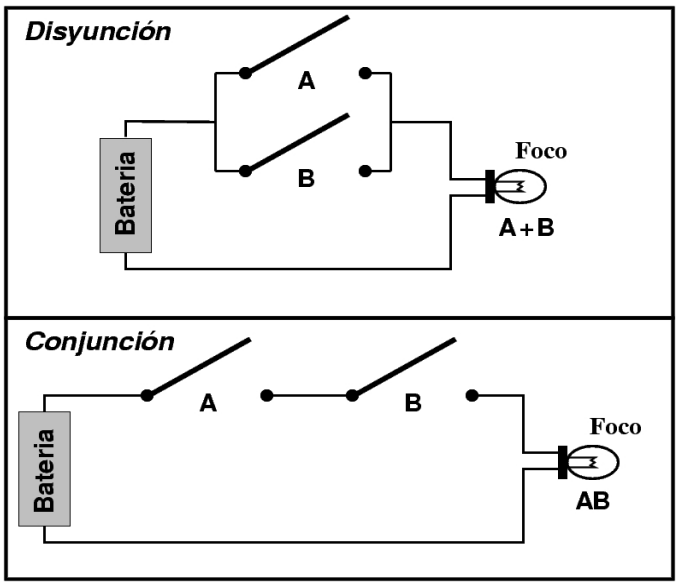

Figura 7.5: Circuitos eléctricos para calcular la conjunción y la disyunción lógicas. En la disyunción el foco se enciende si algún iterruptor está cerrado, en la conjunción sólo si ambos lo están.

Si ahora nos ponemos a pensar en la labor de Claude Shannon podemos idear un circuito que calcule la disyunción y otro que calcule la conjunción usando interruptores. Supongamos que el 1 se traduce ahora en *prendido* y el 0 en *apagado* refiriéndonos a un foco y que se traducen en *cerrado* y *abierto* respectivamente, refiriéndonos a un interruptor. Los circuitos que se muestran en la Figura 7.5 calculan las funciones lógicas de las que hablamos. En la disyunción basta con que alguno de los interruptores este cerrado para que el foco se prenda. En la conjunción ambos interruptores deben estar cerrados para que el foco se prenda.

Con base en funciones lógicas como la disyunción, la conjunción y la negación (la negación de *verdadero* es *falso* y viceversa), podemos construir funciones más complejas que calculen un valor de verdad, un 0 o un 1, en función de los valores de verdad de muchas variables. Luego podemos juntar varias de estas funciones, a las que llamaremos *funciones booleanas*, para construir números escritos en notación posicional en base 2 (binario). Podemos ahora hacer grandes circuitos con interruptores y muchos focos, que calculen funciones más complicadas; por ejemplo, funciones aritméticas, y construir con todo eso una computadora. No importa si nuestros interrup-

tores son realmente interruptores o son bulbos o transistores, pero el hecho de que se pudieran usar transistores en vez de los tubos de vacío hizo las computadoras más pequeñas, más baratas y más confiables.

Probablemente el lector se estará preguntando ahora el por qué usar la base 2, por qué no base 10 a la que estamos acostumbrados. Hay tres razones fundamentales: la primera es que gracias a Frege, a Boole, a Shannon y algunos otros, sabemos que todo lo que nos interesa calcular, se puede hacer con base en funciones booleanas, que surgen de la lógica convencional en la que cualquier cosa sólo puede ser falsa o verdadera.

Para explicar la segunda razón pensemos en una diana de tiro al blanco: un conjunto de círculos concéntricos donde el objetivo es tirar un dardo desde cierta distancia d, que atine en el centro. Ahora modifiquemos el juego, supongamos que tenemos una diana de radio $r = 10$ cm. dividida en 10 círculos concéntricos formando franjas de igual anchura y que la intención del juego es atinarle a una de las franjas en particular: cada vez que un jugador tira dice antes a que franja le quiere atinar. ¿Qué pasa si ahora en vez de dividir los 10 centímetros de radio en diez franjas lo hacemos sólo en dos y repetimos el juego? Seguramente todos lo jugadores atinaran con mayor frecuencia que antes a la franja que desean. Lo mismo ocurre con nuestros dispositivos electrónicos. Supongamos que tenemos un dispositivo que nos entrega por un cable un cierto voltaje que nos dice el resultado de alguna operación, nuestro dispositivo debe entregar un voltaje en un cierto rango, digamos entre 0 y 5 volts. Si lo que debe entregar es uno de diez posibles valores entonces debemos dividir el intervalo de 0 a 5 volts en diez partes iguales, cada una de 0.5 volts. Si un día hace mucho calor y el dispositivo quiere entregar 3.3 volts porque ha calculado el número 6 como su resultado es probable que nos entregue 3.6 volts lo que interpretamos como un 7. Si el número entregado por el dispositivo es el dígito más significativo de un número de 6 dígitos entonces el cálculo salió mal por un millón de unidades. Si en cambio tenemos el intervalo de 0 a 5 volts dividido en sólo dos partes el dispositivo tiene mayor margen de error sin que los resultados se alteren.

La tercera y última razón por la que las computadoras trabajan en binario es porque los algoritmos para calcular en binario son más simples, lo que además los hace más fáciles de traducir a circuitos. Por ejemplo en base dos sólo hay una tabla de multiplicar.

Shockley tenía un carácter muy difícil y era el líder del equipo constituido por él mismo, Brattain y Bardeen. Estos dos últimos fueron quienes descubrieron el transistor, pero su diseño, llamado "transistor de contacto", no era muy útil, funcionaba a veces y era muy frágil, así que el celoso Shoc-

kley, a espaldas de sus compañeros, rediseñó el transistor haciéndolo más robusto y susceptible de ser fabricado en gran escala; esto lo llevó a disputarse con ellos los derechos en la patente y la discordia dió al traste con uno de los equipos científicos más productivos que han existido. El equipo se desintegró, sólo se volvieron a hablar en buenos términos en un restaurante de Estocolmo en 1956, la noche del día que recibieron el Nobel. Bardeen volvería en 1972 a Estocolmo a recibir un segundo premio Nobel, esta vez por sus investigaciones en superconductividad. Shockley dejó AT&T en 1954 y decidió fundar su propia compañía para desarrollo de semiconductores, la *Shockley Semiconductor*; para fundarla eligió un lugar cálido y agradable, que atrajera a jóvenes talentos, el lugar donde su madre tenía su casa: Palo Alto California, cerca de la Universidad de Stanford. Ése sería el comienzo de lo que hoy día es *Silicon Valley*.

En 1956 entró en funcionamiento la primera computadora transistorizada, la TX-0 del Laboratorio Lincoln del MIT a cargo de Kenneth Olsen. Para ese entonces Olsen ya había tenido mucha experiencia en el diseño de equipos de cómputo; en 1950 había trabajado para IBM en un proyecto con el departamento de defensa norteamericano: Whirlwind, diseñado para monitorear el espacio aéreo y responder a posibles ataques en tiempo real. En 1957 Olsen fundó su propia compañía (para variar) la *Digital Equipment Corporation* (DEC), que produjo el *Procesador de Datos Programable* (*Programmed Data Processor*) número 1, en siglas PDP-1; la segunda computadora transistorizada comercial de la historia. La primera fue producida en 1959 por IBM, la IBM 7090, que, a pesar de tener un diseño muy inferior y un precio muy superior a la PDP, dominó el mercado durante muchos años. La 7090 era lo que se solía llamar un *mainframe*, un sistema de tiempo compartido de gran capacidad de almacenamiento y procesamiento, lo más representativo de los sistemas de cómputo de entonces.

Los *mainframes* eran máquinas muy costosas y su mercado estaba restringido a las grandes corporaciones, las dependencias gubernamentales con alto presupuesto y la milicia; aquéllos que podían darse el lujo de hacer un enorme gasto en el equipo mismo y en su mantenimiento, en la edificación de cuartos refrigerados provistos de instalaciones eléctricas de alta demanda y en la contratación de personal especializado capaz de administrar, programar y operar el equipo. Los grandes gabinetes llenos de focos de colores y sonidos *sui generis* que se ven en las películas de los sesenta, son una imagen bastante extraña para nosotros hoy día, pero se apegan bastante al aspecto que exhibían los *mainframes* de antaño.

Con su PDP-1, DEC inició un segmento nuevo en el mercado de equipos de cómputo: el de las *minicomputadoras*. Típicamente con menos poder

de cómputo que los *mainframes*, pero un costo mucho menor en todos los rubros, las *minis* se hicieron de sus propios clientes: las pequeñas empresas, los laboratorios universitarios, centros de investigación científica, pequeñas dependencias gubernamentales, etcétera. Aquellos que requerían de modesto poder de cómputo o que no podían costearse un *mainframe*. Las *minis* servían además como equipo auxiliar de las grandes computadoras, lo que hacía posible dedicar éstas a labores prioritarias mientras las *minis*, en el papel de esclavas, hacían labores menores cuyos resultados podrían ser luego transferidos a la computadora maestra, así que los dueños de *mainframes* también estaban interesados.

Con el tiempo las *minis* cobraron mayor poder y mayor versatilidad sin incrementar su costo notablemente, se las vería compitiendo dignamente con los grandes dinosaurios, hasta que el nicho que éstos ocupaban se hizo tan pequeño que se extinguieron. Pero la victoria sería breve. DEC creó las *minis* y moriría con ellas: fue comprada en 1998 por *Compaq* luego de una larga decadencia que comenzó en 1981, por razones que pronto develaremos.

8

Macro-efectos de micro-cosas

Incierto es el futuro e ingenuo aquél que trata de predecirlo. Cuando era estudiante de sexto semestre de bachillerato, un compañero se acercó a mi grupo de amigos para plantearnos su disyuntiva: su padre estaba dispuesto a comprarle una computadora, ¿cuál debía comprar?, podía ser una Commodore Amiga o una IBM PC ¿A quién se le ocurre considerar a la IBM PC como opción frente a una Commodore? pensábamos nosotros: "La IBM no tiene mucho futuro", "¿dónde has visto una?", "Commodore, en cambio, está por todas partes", sentenció mi amigo Arturo.

La primera vez que entré en contacto con una computadora fue alrededor de 1984. A regañadientes, como todo adolescente, acompañe a mi madre al "super", a la tienda de autoservicio que solía visitar cada semana. Allí había un *stand* con un producto nuevo que no había visto antes, se trataba de una computadora personal: una Commodore 16, había algunos folletos de publicidad y tome uno. La cosa esa tenía 16 Kbytes de memoria (yo no tenía idea de lo que eso significaba) y podía conectarse al televisor, como la consola de videojuego de mis primos (un Atari 2600). Toda la computadora estaba dentro de un único componente con teclas, no como las que conocemos hoy en día con el teclado aparte. Decía que se podía programar en un lenguaje BASIC y que se podían correr programas educativos y de administración financiera del hogar.

Poco después entré al bachillerato y uno de mis compañeros, Arturo, que luego se convirtió en mi mejor amigo, sabía programar un poco en ese lenguaje BASIC; al parecer su hermano le había enseñado algunas cosas. Arturo estaba muy emocionado de lo que se podía hacer con una de esas computadoras y me llevó a la tienda para mostrarme un pequeño programa que había hecho y que solamente mostraba un mensaje en la pantalla (algo

como el usual "Hello world"). Me interesé en el asunto y le pregunté si sabía cómo hacer algunas cosas más, él le preguntó a su hermano; juntos aprendimos lo elemental de BASIC, viendo y preguntando. Nos embarcamos en el proyecto de hacer un juego inspirado en una de nuestras películas favoritas: "Star Wars". Nuestro juego ponía una nave imperial (un "Tie-fighter") y una mira en la pantalla; cuando la nave estaba en el centro de la mira y se disparaba con la barra espaciadora la eliminábamos y poníamos otra nave, así *ad infinitum*. Luego le añadimos cosas, le dimos movimientos aleatorios a la nave, sonidos, fragmentábamos la nave cuando era destruida, en fin; amablemente los empleados de la tienda nos prestaron una unidad de cinta para guardar nuestro programa y no tener que volverlo a teclear en nuestra siguiente visita. La unidad de cinta usaba "cassettes" de audio convencionales. Luego compramos un par de libros de programación en BASIC para Commodore y aprendimos a acceder directamente a la memoria, terminamos programando en un pariente cercano del lenguaje de máquina binario llamado lenguaje ensamblador. Hicimos lo que se conoce como un *cargador*, un programa que lee programas y los coloca en la memoria de la computadora para que se puedan ejecutar.

Más tarde en esa misma tienda aparecieron otros modelos: Commodore 64, 64C, Amiga. La 64 tenía 64 Kb de memoria. A veces sentíamos limitados los 16 Kb del modelo anterior, así que nos pareció maravilloso que el nuevo modelo incluyera un océano interminable de memoria: 64 Kb era una cantidad increíble, nunca nos la acabaríamos. Para el tercer año del bachillerato ya eramos un grupo de 5 *nerds* expertos en Commodore y en BASIC, aunque no debería confesarlo: "Es prácticamente imposible enseñar buen estilo de programación a estudiantes que han sido expuestos previamente a BASIC, como programadores potenciales están mentalmente mutilados sin esperanza de regenerarse", sentenció Dijkstra. Yo quiero pensar que está equivocado, aunque eso de creer que el viejo, experimentado e inteligente Dijkstra, con todo el currículum posible en computación, está equivocado... me hace sospechar.

Pero el comienzo de esta historia está donde nos quedamos en el capítulo anterior. Habíamos dejado a Shockley luego de fundar su nueva compañía en lo que sería *Silicon Valley*, en la que había contratado a mucho personal brillante, lo que la hacía potencialmente productiva. Pero, como ya dijimos, no era muy placentero trabajar para Shockley; era extremadamente arrogante, y de hecho hizo añicos su reputación con declaraciones acerca de sus teorías que relacionaban la capacidad intelectual con la raza y la eugenesia. Así que en 1957 ocho de sus mejores empleados se fueron a fundar otra com-

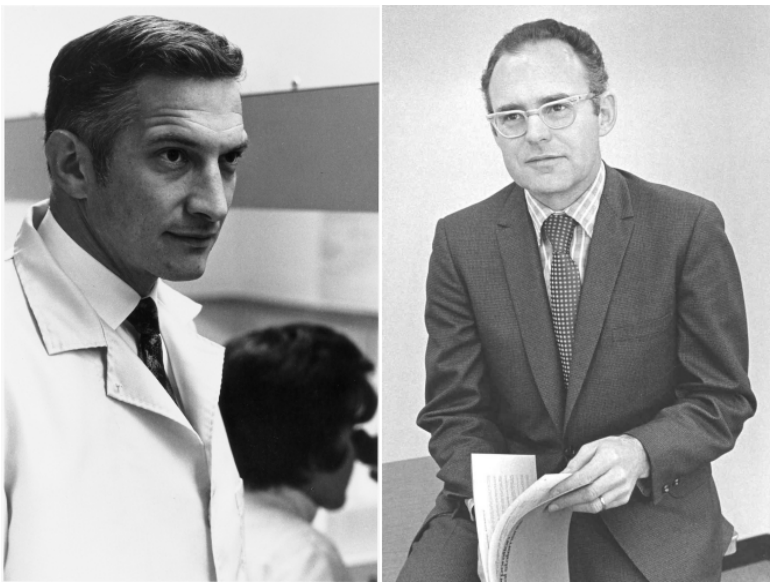

Figura 8.1: Robert Noyce (izquierda) y Gordon Moore (derecha).

pañía, de hecho una división de la *Fairchild Camera and Instruments* que se llamó *Fairchild Semiconductor*, también en Palo Alto.

Dos de los hombres que dejaron a Shockley fueron Robert Noyce y Gordon Moore, que rápidamente entraron en reñida competencia con Jack Kilby de *Texas Instruments* por la miniaturización de los transistores y de hecho más que eso, la miniaturización de circuitos enteros. Había jugosos contratos en juego, sobre todo con la NASA y con el Departamento de Defensa norteamericano. Estados Unidos estaba en plena guerra fría, los soviéticos habían lanzado el Sputnik en octubre de 1957, la guerra de Corea había puesto las cosas peligrosas a principios de esa década, se necesitaba más y mejor tecnología para hacer frente a "la amenaza comunista". Todo apuntaba a dispositivos de control (recordemos a Wiener) que pudieran ponerse en un cohete o dentro de un misil balístico donde el espacio es reducido y el peso un factor crítico.

Kilby y Noyce lograron casi al mismo tiempo el circuito integrado. A principios de 1959 Kilby solicitó su patente, que fue autorizada en 1964, y cinco meses después lo hizo Noyce, que la recibió en 1961. Ahora la carrera espacial era posible y también un sin número de nuevas aplicaciones. Luego Noyce, Moore y Andrew Grove dejaron Fairchild en 1968 para fundar otra

Figura 8.2: Marcian (Ted) Hoff.

compañía dedicada a hacer nuevos tipos de circuitos integrados. El nombre elegido por ellos fue un acrónimo de *Integrated Electronics*: *Intel*.

Algunas de las cosas que se dedicaba a hacer *Intel* eran circuitos integrados (chips de ahora en adelante) de memoria, los que eran utilizados por calculadoras de bolsillo. Uno de los clientes de *Intel* era una compañía japonesa de nombre *Busicom*, líder en el mercado de calculadoras. *Busicom* encargó a *Intel* un nuevo chip para una línea de calculadoras más ambiciosa que las existentes hasta ese momento (finales de la década de 1960): pretendían hacer una calculadora que incorporara funciones matemáticas avanzadas, difíciles de calcular con circuitos hechos a la medida; asignaron la tarea a un graduado de Stanford: Marcian Hoff (o Ted Hoff, como se le conoce generalmente). A Hoff se le ocurrió matar dos pájaros de una sola pedrada: había que diseñar un chip que hiciera cosas muy complicadas para hacerlas en *hardware*, con electrónica pues; así que por qué no hacer mejor un chip de propósito más general, un chip programable, al que pudiera decírsele en *software* qué hacer y cómo, así se lograba cumplir con lo que Busicom necesitaba y además se tenía un producto muy versátil, adaptable a las necesidades de futuros clientes. Hoff se allegó de un equipo de ingenieros, Federico Faggin y Stan Mazor entre ellos. Juntos diseñaron el *Intel* 4004, el primer microprocesador de la historia.

Junto con el 4004 se diseñaron otros chips auxiliares. El 4001 era lo que suele llamarse *memoria de sólo lectura* y servia para guardar programas que debían permanecer inalterados durante la vida útil de la calculadora, como los relacionados con las funciones básicas de la misma. El 4003 tenía los registros: pequeños espacios de memoria en los que los procesadores guardan los datos con los que se llevan a cabo las operaciones y los resultados generados por estas. El 4002, el chip de memoria de lectura y escritura, para almacenar datos y/o programas modificables. En conjunto la familia del 4004 ofrecía la funcionalidad completa de una pequeña computadora. Éste era un procesador de 4 bits, lo que significa que ese era el tamaño de los datos (escritos en binario) con los podía hacer operaciones aritméticas.

Luego llegó otro cliente a *Intel*, una compañía dedicada a hacer terminales, los dispositivos con monitor y teclado desde donde los usuarios de los sistemas de tiempo compartido podían interactuar con la computadora central. La compañía se llamaba *Computer Terminal Corporation* que después cambió su nombre a *Datapoint*. Esta vez el cliente requería de chips que fueran capaces de controlar la terminal y el flujo de datos desde y hacia la computadora central. Así que Hoff y Mazor le propusieron al cliente usar un microprocesador para resolver el problema. En abril de 1972 el nuevo procesador diseñado para Datapoint fue liberado comercialmente, por cierto que Datapoint no lo utilizó, decidió usar *hardware* más convencional. En buena medida, el 8008 fue diseñado teniendo en mente las necesidades de control de terminales y envío y recepción de datos entre una terminal y el computador central, lo que implica un procesamiento intensivo de cadenas de caracteres, así que era un procesador de 8 bits (los usuales para representar caracteres en esos tiempos). Igual que su predecesor, el 8008 requería de chips de soporte.

Para 1973 ya había varias compañías que ofrecían microprocesadores. En 1974 *Intel* anunció el 8080, un microprocesador de 8 bits que requería de menos chips adicionales de soporte, operaba a 2 MHz, lo que lo hacía un orden de magnitud más rápido que el 8008 y podía manejar hasta 64 Kb de memoria. Mientras se diseñaba el 8080 uno de los miembros del equipo de desarrollo, Faggin, se salió de *Intel* y fundó su propia compañía (¡qué raro!), que produjo su propio procesador de 8 bits, compatible con el 8080, el famoso Z80 de Zilog.

Iniciaba la década de los 70's, toda una generación de jóvenes buscaba alternativas: ideológicas, políticas, sexuales, artísticas, tecnológicas. A fines de la década anterior esta búsqueda frenética había llegado a un violento climax, lo mismo en París, que en Berlín o la Ciudad de México. La ansiedad de la juventud que buscaba nuevos aires, sus propios aires, había

hecho explosión. Esa generación gestó una revolución en todos los ámbitos cuestionando y tirando a la basura todo lo convencional. Se impuso el amor libre, las quemas de sostenes, el uso de la píldora anticonceptiva, el LSD, el rechazo al imperialismo y la guerra, el "peace and love", el rock y un cambio radical en el modo de concebir a las computadoras, quiénes las usan y para qué. Todo ello es herencia de esa generación.

A principios de la década de 1960 el usuario típico de un sistema de cómputo era un ingeniero o un científico. El sistema sería usado para hacer estadística, control de transacciones financieras, cálculos científicos relacionados con la milicia, la ingeniería aeronáutica o el pronóstico del clima. A fines de la década de 1980 el usuario típico es cualquier persona: un estudiante, una secretaria, un diseñador gráfico, un poeta o un músico y el ámbito de uso es tan amplio que no vale la pena mencionar aplicaciones particulares. A mediados de los 70's a ningún fabricante de computadoras se le ocurría una razón por la que alguien quisiera tener una computadora en casa, evidentemente nadie necesita o desea hacer cálculo balístico de artillería en su sala, pero si cambias el uso, cambias a los usuarios. Sólo jóvenes en busca de alternativas podrían haberse dado cuenta de ello: se requería de computadoras no convencionales, corriendo aplicaciones no convencionales para usuarios no convencionales.

En 1974 comenzó la revolución de las computadoras personales. En ese año tres diferentes revistas del tipo "hágalo usted mismo" promovieron kits de construcción de pequeñas computadoras personales: *QST*, en su número de marzo anunció la Scelbi-8H, basada en el 8008 de *Intel*; en la portada del mes de julio de *Radio-Electronics* decía: "Construya la Mark-8: Su minicomputadora personal", ésta también estaba basada en el 8008; *Popular Electronics* no se quiso quedar atrás de la competencia y ese mismo año hizo un trato con Edward Roberts, el dueño de una compañía de Albuquerque dedicada a hacer instrumentos de telemetría, *Micro Instrumentation and Telemetry Systems* (MITS), para que construyera una computadora personal que pudiera venderse a través de la revista. Roberts hizo un diseño basado en el nuevo procesador de *Intel*, el 8080, y su hija de 12 años lo bautizó con el nombre de Altair. La Altair 8800 fue anunciada en el número de enero de 1975 de *Popular Electronics* y podía comprarse ya armada o en piezas para armarla uno mismo. Costaba alrededor de 400 dólares (desarmada), un orden de magnitud más barata que las minicomputadoras de DEC, claro que sin sistema operativo ni software alguno, pero barata.

La Altair además estaba diseñada para recibir conexiones de otros dispositivos electrónicos, algo que potencialmente les agradaría a los aficionados a la electrónica que constituían el universo de lectores de la revista. En el

Figura 8.3: Portada de *Popular Electronics* de enero de 1975.

artículo de las páginas interiores, Roberts señaló 23 aplicaciones distintas de la máquina. Pero en realidad era difícil considerarla como algo serio: para programarla había que hacerlo en lenguaje de máquina (binario) manipulando decenas de *switches* en el frente de la máquina y la ejecución del programa consistía en una serie de focos que se prendían y apagaban. Pero el entusiasmo de los aficionados superó las dificultades. La Altair pronto recibió soporte de numerosos grupos de aficionados, de otras revistas, de pequeñas compañías y de tiendas de electrónica; luego de algún tiempo ya había teclados disponibles, conectores para televisor (que hacía las veces de monitor), lectoras de cinta de papel para alimentar programas y datos y unidades de cinta de audio para leer y escribir en "cassettes" convencionales, ancestros de la unidad de cintas que utilizamos mi amigo y yo en la Commodore.

Cuando salió el número de enero de *Popular Electronics*, un par de muchachos en Massachusetts también se entusiasmaron; el mayor de ellos tenía 21 años y el menor 19, se hicieron amigos durante el bachillerato e hicieron algunas travesuras juntos, por ejemplo, penetrar ilegalmente en algunos sistemas de cómputo. Eran el prototipo de lo que hoy día suele llamarse *hacker*. El mayor le mostró al otro el número de la revista don-

Figura 8.4: William Gates (izquierda) y Paul Allen (derecha).

de aparecía la Altair y juntos decidieron hacer un intérprete del lenguaje BASIC para la máquina. El mayor de los muchachos se llama Paul Allen, el menor William Gates III. Gates estaba estudiando (por presión de sus padres) la carrera de leyes en Harvard, donde Aiken construyera la Mark I treinta años antes. No tenían una Altair disponible, lo único que tenían a su alcance era la PDP-10 del centro de cómputo de Harvard, así que escribieron el intérprete de BASIC en la PDP de acuerdo con las especificaciones del 8080, el programa resultante lo mandaron perforar en una cinta de papel que Allen se llevó a Albuquerque para hacerle una demostración a Roberts.

Roberts se interesó en el intérprete y entonces Gates y Allen fundaron una compañía que ya había hecho su primer negocio: *Micro-Soft*, que retuvo los derechos del intérprete y le concedió licencia de venta del mismo a MITS, lo que por cierto sentó precedente.

Al año siguiente, en 1976, muchas compañías introdujeron sus propios modelos de computadoras personales y Gates y Allen pudieron vender las licencias de BASIC a nuevos clientes: Tandy, por ejemplo, que ese año sacó al mercado su *Radio Shack* TRS-80, basada en el microprocesador Zilog Z80, que, como el lector recordará, era compatible con el *Intel* 8080. Otra compañía que compró la licencia del intérprete de BASIC fue una empresa canadiense, la *Commodore*, que también en 1976 lanzó su modelo PET basado en el procesador de 8 bits 6502 de la compañía *MOS Technologies*, el

Figura 8.5: Curiosa fotografía de Bill Gates luego de un arresto por exceso de velocidad en Albuquerque. Tengo muchos amigos que gozarán viéndola.

mismo procesador que usaba la Commodore 16 y que había sido diseñado por Charles Peddle un año antes; Peddle salió de la compañía *Motorola*, del equipo de diseño del procesador Motorola 6800, por lo que el 6502 es casi idéntico al 6800, pero más barato. En 1976 *Commodore* compró *MOS Technologies*, la compañía de Peddle, con lo que garantizaba el más bajo costo de sus propios procesadores. La Commodore 64, que tanto disfrutamos mi amigo del bachillerato y yo, fue lanzada en 1982 y tenía un procesador 6510 de *MOSTech* como se solía llamar a la fábrica y seguía usando el intérprete de BASIC de *Microsoft* (ya sin el guión). Eventualmente *Commodore* tuvo problemas financieros y en 1994 la compañía *Gateway 2000 Computers* compró lo que quedaba de ella.

Otra compañía que sacó al mercado su propia línea de computadoras personales, que también usaba procesadores de *MOSTech* y el intérprete de *Microsoft* fue *Apple*. Dos amigos: Steve Jobs y Steve Wozniak se reunían con un club de entusiastas de las computadoras personales, el club *Homebrew* (lo que significa algo como "bebida alcoholica casera"), de hecho Wozniak había construido una versión mejorada de la Altair, y decidieron hacer su propio diseño. Ellos también habían hecho travesuras, fabricaban pequeños dispositivos electrónicos que hacían posible intervenir la red telefónica de tal forma que era posible hacer llamadas de larga distancia sin pagar, por ejemplo. En el garage de Wozniak se entregaron a su labor y de allí surgió la primera Apple, la Apple I; cuando la vió Paul Terrell, un miembro del club

Homebrew al que asistían Jobs y Wozniak, les hizo un pedido de cincuenta. Wozniak vendió su Vocho y sus calculadoras programables, ambos dejaron sus trabajos (Wozniak era programador de *Hewlett Packard* y Jobs trabajaba para *Atari*) y fundaron *Apple Computer Corporation* en abril de 1976. Ésa fue sólo una de las 23 compañías que se desprendieron del club de aficionados *Homebrew* hasta su desintegración en 1986.

Cuando las computadoras no poseían un uso práctico para las personas comunes y corrientes, los miembros de *Homebrew* y de muchos otros clubes las construían sólo por diversión; los pretextos eran muchos: balancear sus gastos, guardar recetas de cocina o su catálogo de libros o de discos de música. Ciertamente eran sólo pretextos, realmente lo hacían por diversión, por la misma razón por la que alguien adquiere un modelo a escala para armar o un trenecito de juguete: por placer, porque uno se divierte haciéndolo y viéndolo moverse y probando sus posibilidades; es un placer lúdico, nada tiene que ver con que sea útil. Los miembros de *Homebrew* compartían los programas que elaboraban ellos mismos u otras personas, en el contexto era lo más natural, nadie allí veía las cosas como negocio; como en cualquier otro club de aficionados hay intercambio de material. En febrero de 1976 el club recibió una carta de un indignado individuo que les decía que estaban mal, que esos programas que se intercambiaban eran el trabajo de alguien más y que eso era un robo, el autor de la carta era alguien que había cambiado de bando recientemente, alguien que habiendo retado al *establishment* ahora era parte de él: Bill Gates.

Las computadoras personales pronto comenzaron a ser realmente útiles y se convirtieron en productos comercialmente exitosos cuando dejaron de ser simples piezas de *hardware* y adquirieron una "personalidad" propia dada por el *software*, es decir los programas de aplicación. Las computadoras personales proliferaron. Un conjunto de aficionados, mayoritariamente adolescentes, había cambiado radicalmente el panorama de la industria de la computación: el control del mercado ya no era privilegio exclusivo de las grandes empresas, ahora era algo más terrenal, pero aún faltaban por librarse muchas batallas.

La Apple I no fue ciertamente un éxito, pero en 1977 la Apple II sí lo fue. La Apple II estaba basada en el procesador MOSTech 6502 y usaba también cinta de audio como medio de almacenamiento, pero luego Wozniak le diseñó una unidad de disco flexible de $5\,\frac{1}{4}$ pulgadas; lo mejor llegó cuando en 1979 la Apple II comenzó a venderse junto con VisiCalc, un programa de los comúnmente llamados "hojas de cálculo". Eso redundó en grandes ventas para *Apple*.

Figura 8.6: La Apple II.

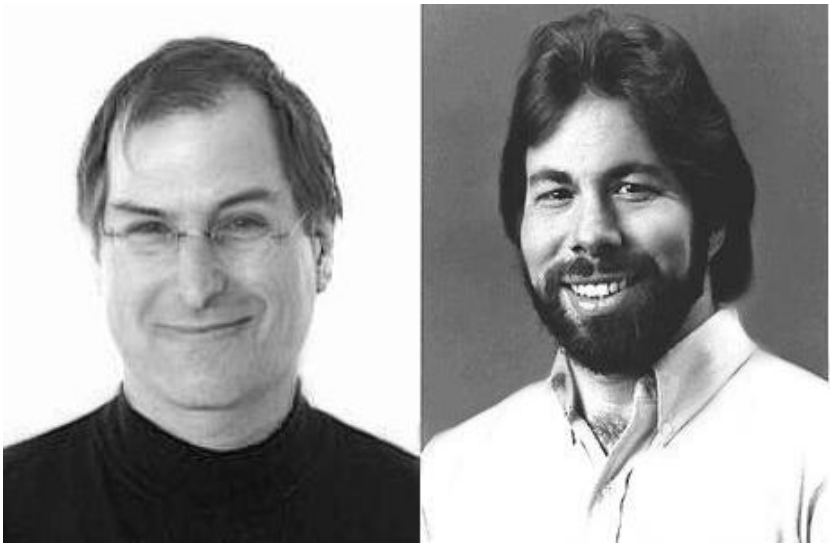

Figura 8.7: Steve Jobs (izquierda) y Steve Wozniak (derecha).

En 1976 un individuo llamado Gary Kildall y su esposa, Dorothy McEwen fundaron una compañía con el bastante ridículo nombre de *Intergalactic Digital Research*. El principal producto de la compañía era un programa que originalmente había sido hecho para controlar el disco de una computadora IBM 360 y que se había adaptado para ser utilizado en microcomputadoras; el programa en cuestión se llamaba CP/M por las siglas de *Control Programm for Micros*. IMSAI una compañía que fabricaba un modelo mejorado de la Altair, la IMSAI 8080, encargó a Kildall una versión de su CP/M para ser usado como sistema operativo para su computadora. La máquina tenía unidades de disco flexible de 8 pulgadas, lo usual en la época, de donde podría leerse el programa de Kildall, que de hecho controlaría las mismas unidades de disco y el resto de los recursos de la máquina, es decir haría las veces de un *sistema operativo*, que en el caso de las computadoras personales tiene una labor mucho más simple que en el caso de un sistema grande multiusuario. La IMSAI 8080 y CP/M se volvieron un estándar de facto en el medio de las computadoras personales o microcomputadoras, como suele llamárseles y como Kildall tenía que estar constantemente migrando su programa a diferentes máquinas con distinto hardware, decidió hacer las cosas bien: programó todo lo que tenía que ver con el hardware en rutinas que podían ser llamadas desde el programa principal, que hacía cosas más generales, más abstractas (por decirlo de algún modo); así cada vez que había que migrar CP/M a otra computadora sólo tenía que reescribir las rutinas y no todo el sistema. Al conjunto de rutinas les puso por nombre BIOS (*Basic Input/Output System*), nombre que aún utilizamos.

En 1981 el auge de las computadoras personales hizo pensar a algunos directivos de IBM que allí había expectativas de buenos negocios. Hasta ese momento IBM, como todas las compañias "serias" de computadoras, se había dedicado al diseño y fabricación de los grandes *mainframes*, con soporte para múltiples usuarios simultáneos y muchos recursos compartidos por todos ellos. Demasiado barullo despertó al gigante. Intentaron comprar *Apple*, pero Jobs y Wozniak no la quisieron vender. Se desarrolló entonces una computadora basada en un descendiente del procesador 8080: el 8088 de *Intel*. El 8080 era un procesador de 8 bits, en cambio el 8088, al igual que su primo el 8086, eran procesadores de 16 bits[1]. IBM no quería apostar a la incertidumbre y no quería invertir mucho en desarrollos nuevos para un producto que no sabían qué tan bien se iba a vender ni durante cuánto tiempo, así que decidieron que el software sería elegido de entre los que ya

[1]Normalmente el número de bits de un procesador se refiere al tamaño de los números con los que puede hacer operaciones aritméticas.

Figura 8.8: Gary Kildall, creador de CP/M.

tenían un mercado hecho. Jack Sams, un ejecutivo de IBM, fue a Seattle en el verano de 1980 a visitar a Gates (*Microsoft* se mudó de Albuquerque a Seattle poco tiempo después de ser fundada). En la primera plática no hubo nada concluyente, Sams le dijo a Gates que IBM planeaba construir una computadora personal y que estaban analizando la posibilidad de usar su intérprete de BASIC, pero su descripción de la máquina no fue exacta (algunos piensan que intencionalmente), la descripción de Sams era de una máquina bastante menos poderosa que la que IBM diseñaba, lo que fue señalado por Gates como una desventaja importante. Al final de la reunión Sams le dijo a Gates que *posiblemente* lo llamarían después. En efecto llamaron después para concertar una cita entre un equipo de abogados de IBM y los de *Microsoft* y otra entre el equipo técnico de IBM y el de *Microsoft*; como Gates era ambas cosas acabó llevando a la reunión a otros cuatro empleados de su compañia, para no verse tan desvalido. IBM adquirió la licencia del BASIC de *Microsoft* y Gates quedó como consultor del proyecto y cuando se le pidió una sugerencia acerca del software del sistema mandó a IBM con Kildall. Kildall y Gates tenían un acuerdo verbal: ambos eran líderes en su ramo, *Microsoft* hacía lenguajes y *Digital Research* sistemas operativos, ninguno se metería en el campo del otro.

Figura 8.9: La PC de IBM.

IBM habló entonces con Kildall para explorar la posibilidad de migrar su CP/M a una máquina de 16 bits, hasta ese momento CP/M sólo funcionaba en máquinas de 8 bits, así que era algo de lo que había que preocuparse. Cuando los ejecutivos de IBM llegaron a visitar a Kildall en Monterey California él no estaba, de hecho sabía de la cita y a pesar de ello decidió salir a probar su nuevo planeador, se sabía dueño de la situación, su producto era un estándar y los tipos trajeados de IBM podían esperar, así que a los ejecutivos los atendió su esposa, quien al ver las cláusulas del contrato propuesto pensó que podían meterse en un callejón sin salida, por lo que amablemente les indicó la salida.

IBM se desesperó y le propuso a Gates que *Microsoft* proveyera el sistema operativo. Gates encontró una compañía llamada *Seattle Computer Products* donde un tipo llamado Timothy Patterson había escrito una especie de sistema operativo pensando en el 8086 (el primo del 8088 que IBM usaría). Gates le compró a la pequeña compañía los derechos del programa, que Patterson había bautizado QDOS (*Quick and Dirty Operating System* o *Sistema Operativo Rápido y Sucio* en español), pagó por ello 175 mil dólares y luego utilizó QDOS para hacer MSDOS, sin duda uno de los programas que mayor influencia ha tenido en el mundo.

IBM lanzó su computadora personal, la IBM PC, en 1981 y con esto marcó más de una década la pauta en el mundo de las computadoras personales. Se vendieron millones. Con cada nueva generación de procesadores de *Intel* la IBM PC se actualizaba, cambiaba de nombre (por ejemplo IBM AT cuando se incorporó el 80286 de *Intel*) e incrementaba su velocidad, pero el esquema general de funcionamiento era el mismo y las nuevas versiones podían seguir corriendo versiones anteriores de los mismos programas. Se volvió un estándar de facto en muchos aspectos y aún hoy en día, luego de 22 años, las normas que estableció en la industria de la computación siguen teniendo vigencia o, por lo menos, subyacen en las actuales. En 1982 la revista *Time* eligió a la IBM PC para el reconocimiento que tradicionalmente ocupa "el hombre del año"; realmente la IBM PC fue, para bien o para mal, LA computadora personal de dos décadas.

Poco tiempo después del lanzamiento de la IBM PC, Kildall se percató de su error al ignorar a IBM y decidió sacar su versión de CP/M para 16 bits, pero otra vez se equivocó: CP/M era un estándar de facto, así que pensó que la gente pagaría gustosa los 240 dolares que costaba en vez de los 60 de MSDOS, todo con tal de tener el sistema operativo más conocido. Por supuesto la apuesta le salió mal y no fue el único al que le ocurrió, John Roach el presidente de *Tandy* declaró "No creo que esto sea significativo" refiriéndose a la IBM PC. ¿Alguien ha visto una computadora *Tandy* últimamente? digamos... en los últimos 15 años[2].

La única compañía que, en ocasiones, le hizo la competencia a IBM fue *Apple*. En 1984 lanzaron su modelo Macintosh[3]. La Mac, como se le solía llamar, era una máquina innovadora, poseía una interfaz gráfica muy fácil de usar. Los íconos que conocemos en los sistemas de hoy día, los folders que se abren, los menús que se despliegan, el mouse, todo ello apareció en el mundo comercial con la Mac. Al parecer la idea original no fue de la gente de *Apple*; cuenta la leyenda que Jobs hizo una visita a los laboratorios de *Xerox* (Xerox-PARC) y que allí quedó impresionado por el desarrollo de una interfaz gráfica para una máquina llamada Xerox Star, lo que le dio la idea de la interfaz de usuario y toda la filosofía WYSIWYG (*What You See Is*

[2]Tandy dejó de existir en 1987 cuando fue adquirida por una compañia llamada AST. Su último modelo de computadora, la Tandy 4000, era 100% compatible con IBM.

[3]El lanzamiento de la Mac se llevó a cabo con un anuncio televisivo que fue transmitido una sola vez: durante el Super Tazón. El anuncio hace alusión a la novela de George Orwell "1984" en la que el mundo entero es controlado por un tirano omnisciente y omnipotente llamado "el gran hermano" (*Big Brother*). Por supuesto que, sin decirlo, el "gran hermano" del comercial es IBM (a quien suele llamarsele *Big Blue*). El anuncio terminaba diciendo algo como: *Conozca la nueva Apple Macintosh y vea por qué 1984 no será como "1984".*

Figura 8.10: Apple Macintosh.

What You Get o *lo que ves es lo que obtienes*) de la Mac. Los defensores de Jobs dicen que esto no es verdad. En todo caso es innecesario defender la creatividad de Jobs, a fin de cuentas la Mac no sólo era una interfaz gráfica, era una máquina bien pensada. De hecho para diseñarla Jobs formó un equipo de desarrollo especial; *Apple* había crecido mucho y era ahora una gran empresa con tiempos de desarrollo muy largos y mucha burocracia, se había perdido aquel dinamismo y la pasión que fructificaron en el garage de Wozniak. Jobs y su equipo diseñaron el principal producto de la compañía al margen de ésta, eso le causaría problemas más tarde.

Steve Jobs reclutó en 1983 a John Sculley, que trabajaba para .jpgi-Cola, para ocupar el puesto de presidente de *Apple*. Cuando ocurrió lo de la Mac, Sculley se enfadó y logró convencer al consejo de administración de que había que acotar cuidadosamente el poder de Jobs ... y Jobs se fue. Salió de *Apple* en 1985 vendiendo sus acciones en un millón de dólares. ¿Qué haría ahora?

Probablemente el lector haya visto alguna vez las películas de Disney: "Toy Story", "A Bug's Life" ("Bichos" en español), "Toy Story 2" (para mi gusto, mejor que la primera parte) o "Monsters Inc."; películas que siempre han estado a la vanguardia en las técnicas de animación por computadora. Éstas fueron producidas por una compañía llamada *Pixar Animation Stu-*

dios usando técnicas revolucionarias nunca antes intentadas; las escenas de la última película ("Monsters Inc.") fueron producidas por un *cluster*[4]: un conjunto de 250 servidores Sun Enterprise 4500, cada uno con 14 procesadores Ultra Sparc II (lo que hace un total de ¡3500 procesadores en el cluster!). Bien, ahí está parte de la respuesta a la pregunta formulada arriba. Steve Jobs fue co-fundador de *Pixar* en 1986.

Pero eso es sólo parte de la respuesta. En 1988 fundó otra compañía: NeXT, dedicada a hacer estaciones de trabajo, computadoras multiusuario pequeñas, pero más poderosas que una computadora personal. La Mac había sido diseñada con base en el procesador Motorola 68000 y las NeXT continuaron, de alguna manera, la estirpe; también utilizaban procesadores de Motorola: el 68020 y 68030, por ejemplo. Como toda estación de trabajo que se respete, las NeXT utilizaban sistema operativo Unix y además, por supuesto, una interfaz gráfica atractiva y fácil de usar, también venían con una gran cantidad de aplicaciones, tanto de software libre como de software desarrollado por NeXT. En 1997 las finanzas de NeXT no iban del todo bien, sus máquinas eran caras y la competencia, encabezada por *Sun Microsystems* era fuerte, así que *Apple* compró NeXT, como para reconciliarse con Jobs, que volvió a ser el presidente de *Apple* ya sin Sculley por ahí. Los últimos derroches de creatividad de Jobs son las iMac, que, como siempre ha ocurrido con *Apple*, son buenas, bien diseñadas, pero caras y con un mercado cautivo siempre fiel.

Éste no es el caso de aquellas compañías que no pudieron ver que el futuro le pertenecía mayoritariamente a las computadoras personales y a las compatibles con IBM en particular. Ya mencionamos el caso de *Tandy* y el de *Commodore*; hay varios más, pero probablemente el más dramático es el de DEC que en 1981 ocupaba el lugar número 34 de las empresas más rentables de Estados Unidos y en 1997, poco antes de ser comprada por *Compaq*, estaba por debajo del número 300; la única compañía que fue capaz de personificar a David frente al Goliat de IBM, pagó con su vida el privilegio de ser diferente y pronosticar mal el futuro de la computación personal.

Hoy en día las computadoras personales son tan comunes que ya no nos sorprende verlas en cualquier parte. En países con mayores ingresos *per capita* están en prácticamente todos los hogares. Ahora casi cualquiera tiene a su alcance un esclavo confiable que hace parte importante del trabajo tedioso y con el que se puede hacer mucho más: jugar, oir música, ver películas,

[4]Un *cluster* es un conjunto de computadoras interconectadas por una red de alta velocidad que colaboran coordinadamente para lograr una tarea común.

leer, acceder a una gran cantidad de información, platicar con amigos en otra parte del mundo y hasta verlos, en fin, lo que habíamos mencionado como "usos no convencionales" y que ahora son los más. La omnipresencia de la computadora hoy en día es consecuencia de la invención del microchip, del microprocesador y del juego de muchachos de hacer con eso microcomputadoras, formar microempresas que crecen, que valen millones de dólares y llevan lo *micro* a todo el orbe: oficinas, fábricas, laboratorios, escuelas y casas; que cotizan en el NASDAQ y que si pierden o ganan arrastran consigo las economías de paises enteros: macro-efectos de micro-cosas.

En 1943 Thomas Watson, el presidente de IBM dijo: "Yo creo que hay un mercado mundial para, posiblemente, cinco computadoras más" y en 1977 Ken Olsen, el fundador de DEC: "No hay razón alguna por la que alguien quiera tener una computadora en su casa". Como dije al comenzar este capítulo: Incierto es el futuro e ingenuo aquél que trata de predecirlo.

9

Una gigantesca metáfora

En 1957 en plena guerra fría, la Unión Soviética puso en órbita al primer hombre. El triunfo tecnológico soviético hizo que los norteamericanos pusieran mayor énfasis en su propio desarrollo en ese rubro. Por ello fue creada la agencia de proyectos de investigación avanzada o ARPA (*Advanced Research Projects Agency*), dependiente del departamento de defensa (por lo que ocasionalmente se le cambia el nombre a DARPA, poniendo la "D" de *Defense*).

Por supuesto, en gran medida, el desarrollo tecnológico debería estar relacionado con el desarrollo de sistemas de cómputo que habían probado su utilidad para menesteres bélicos. A cargo del proyecto de cómputo de ARPA quedó un hombre llamado Joseph Licklider, un psicólogo con estudios de posgrado cuyo interés en las computadoras iba más allá del entusiasmo. Con la intención de matar dos pájaros de una sola pedrada, Licklider fue puesto a cargo de dos programas, uno de ciencias del comportamiento, para el que evidentemente estaba capacitado y el de cómputo, para el que no era tan evidente su elección. Licklider veía en las computadoras un medio para extender las capacidades del ser humano, herramientas para ampliar el poder analítico de la mente, esperaba que en el futuro se diera una especie de relación simbiótica entre las computadoras y los seres humanos para la solución de problemas. Licklider tenía muy buenas relaciones en el mundo académico, así que en buena medida sus esfuerzos se encaminaron a vincular a la comunidad científica de ciencias de la computación con ARPA. Para cuando dejó su oficina el nombre de ésta había cambiado de *Command and Control Research* a *Information Processing Techniques* (IPTO).

Figura 9.1: Paul Baran (izquierda) y Donald Davies (derecha).

En 1966 quedó Robert Taylor a cargo de IPTO, él había hecho investigación en psicoacústica, igual que Licklider. Taylor estaba preocupado porque los numerosos científicos asociados a proyectos de IPTO requerían cada vez de más y más recursos de cómputo, casi cada investigador quería su propia computadora y éstas no eran nada baratas así que ¿por qué no pensar en compartirlas? Sería bueno pensar en que varios científicos que realizaban trabajos similares y requerían de recursos similares los compartieran. Por supuesto estos científicos estaban en lugares geográficamente distantes en sus propias universidades, así que había que pensar en proveer de ligas de comunicación entre las máquinas.

Otros dos científicos trabajaban en la idea de comunicar computadoras. Paul Baran, un norteamericano de origen polaco cuyo padre poseía una tienda de abarrotes en Boston, había trabajado para la *Eckert-Mauchly Computer Corporation* y para *Remington-Rand* mientras estudiaba para obtener el grado de maestro en ingeniería en UCLA, donde interrumpió sus estudios de doctorado. Mientras trabajaba en RAND se interesó en un problema: imaginemos todos los sistemas de cómputo de defensa, los sistemas de control de lanzamiento de misiles entre ellos, conectados en una red de comunicaciones y luego imaginemos que las líneas de comunicación sufren desperfectos debido a un ataque nuclear; si la red de comunicaciones no es

suficientemente robusta entonces queda anulada la posibilidad de responder al ataque sufrido. Eso es una tentación para el enemigo, atacar primero asegura que no habrá respuesta. La condición necesaria para que la capacidad de respuesta del país se mantenga en operación es que los sistemas de comunicación para armas estratégicas sean capaces de sobrevivir un ataque y ésa no era precisamente la situación: las redes de comunicación de larga distancia en Estados Unidos para esa época eran muy vulnerables. Baran empezó a trabajar en el problema de construir una infraestructura de comunicaciones robusta.

Parte de la solución encontrada por Baran consistía en una red de comunicaciones donde hubiera muy alta redundancia, donde los mensajes que desde un punto A se enviaran hasta un punto B pudieran viajar a través de rutas diferentes, para que en caso de que una ruta se estropeara fuera posible usar la alternativa o alguna de las alternativas. Era además necesario que los mensajes pudieran ser fragmentados en paquetes pequeños, de forma que si se rompía la comunicación entre dos sitios por una ruta, fuera posible desde B notar que algo andaba mal (porque al menos algunos paquetes se habrían recibido luego de cambiar su ruta) y fuera también posible reenviar los paquetes perdidos. La esencia de una red de conmutación de paquetes.

Esta idea también fue explorada desde el punto de vista teórico por Leonard Keinrock, quien en 1961 publicó el primer artículo sobre el tema *Flujo de Información en Grandes Redes de Comunicación* (*Information Flow in Large Communication Nets*).

Del otro lado del Atlántico un hombre completamente diferente (había sido un estudiante excelente y constante) había llegado a la misma conclusión por motivos completamente diferentes. Donald Davies había estado en el equipo de Alan Turing en 1947 diseñando una computadora ultra rápida y luego de una estancia de un año en el MIT estaba en Inglaterra en 1965 pensando en todo el potencial de cómputo que podría obtenerse si pudieran conectarse diversos equipos y pudieran interactuar. Una posibilidad es conectar estos equipos a través de la ya existente red telefónica, pero esta alternativa no es buena. La interacción de computadoras genera una comunicación que se lleva a cabo en ráfagas de transmisión de datos, muy diferente de la uniformidad necesaria en la red de comunicación telefónica, donde se requiere mantener un canal dedicado exclusivamente a la, presumiblemente uniforme, comunicación entre un emisor y un receptor. Cada vez que dos personas hablan por teléfono utilizan un canal de comunicación dedicado exclusivamente a esa conversación, la red telefónica cierra los circuitos necesarios para que sea posible seguir continuamente la señal entre los puntos de comunicación a lo largo de un cable, es una red de conmutación

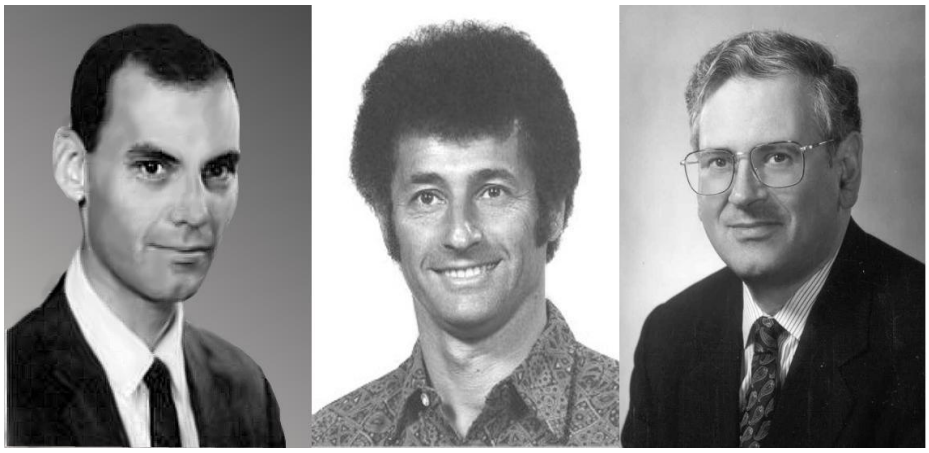

Figura 9.2: Larry Roberts (izq), Leonard Kleinrock y Robert Kahn (der).

de circuitos (*circuit-switched network*), mientras que, en el caso de comunicaciones entre computadoras más bien se requiere que viajen a través de la red esas ráfagas de comunicación que se dan cada cierto tiempo. Se puede pensar entonces en transmitir paquetes de datos, la misma idea de Baran.

Taylor, ignorante aún de las ideas de Baran y Davies, elaboró su proyecto de crear una red de computadoras y logró vendérselo a Herzfeld, el jefe supremo en ARPA, a cargo del proyecto quedó un hombre llamado Larry Roberts, un investigador del laboratorio Lincoln del MIT al que costó mucho trabajo convencer y que tenía muchos amigos y conocidos en el medio científico y de ingeniería.

Ya inmerso en los problemas inherentes a la labor de comunicar computadoras, Roberts entró en contacto durante 1966 y 1967 con las ideas de Baran y Davies. Contaba también con un buen número de personas capaces a su lado (entre ellos Leonard Kleinrock, a quien Roberts le dio el cargo de Oficial del *Network Measurement Center*, instancia encargada de medir el desempeño de la red), Douglas Engelbart, un científico del *Stanford Research Institute*, Jon Postel, un estudiante de posgrado en UCLA, Wes Clark y Frank Heart. En Octubre de 1966 Roberts elaboró el primer plan de la red de ARPA (que en adelante llamaremos ARPANET) y lo puso en blanco y negro en un documento titulado *Hacia una red cooperativa de computadoras de tiempo compartido* (*Towards a Cooperative Network of Time-Shared Computers*).

Clark fue el primero en sugerir una idea que, por cierto, había sido también una de las conclusiones de Davies en Inglaterra. ¿Cómo comunicar diferentes computadoras entre sí? incluso computadoras muy dispares tanto en hardware como en software del sistema, lo que significa en términos coloquiales, que "hablan diferentes idiomas"[1]. Podría pensarse en conectar la computadora A con la B, por ejemplo, lo que significa dotar a ambas de los dispositivos electrónicos y de los programas para que cada una traduzca, en sus propios términos, lo que su contraparte le dice. Si ahora se pretende añadir una nueva computadora C, completamente diferente de A y B, habría que pensar en dotarla también de la electrónica y los programas para comprender a todas las máquinas que ya estaban en la red y a estas de lo propio para comprender a la recién llegada. Continuar añadiendo computadoras nuevas a la red preexistente transforma el problema de comunicación en algo inmanejable: en principio debemos tener programas diferentes para conectar cualesquiera dos tipos diferentes de computadoras. La solución de Clark fue genial: por qué no pensar en resolver el problema de comunicar eficazmente cada tipo de computadora: A, B y C, con un solo tipo estándar de computadora pequeña y cercana que fuera la que, en realidad efectuara la comunicación a larga distancia con otra computadora pequeña idéntica y que también se comunicara con alguna o algunas otras computadoras grandes y cercanas de propósito general, la idea era entonces construir computadoras pequeñas de propósito específico que fueran capaces de comunicarse entre sí a larga distancia y que ofrecieran la posibilidad de comunicarse también, a corta distancia, con otras computadoras de propósito general llamadas "anfitriones" (*hosts* en inglés). A las computadoras pequeñas de propósito específico se les denominó procesadores de interfaz de mensaje (*Interface Message Processors*, IMP).

Ahora el principal problema consistía en definir cómo debería efectuarse la comunicación anfitrión-IMP y la comunicación IMP-IMP y luego preocuparse por construir los IMP's. Para finales de julio, Roberts y su equipo ya tenían un borrador decente de los requerimientos de la red en general y de los IMP's en particular. El documento sintetizaba las ideas de Kleinrock (teoría, análisis probabilístico), Baran, Davies (redundancia de rutas, fragmentación de los datos en paquetes) y Clark (la red basada en una subred de IMP's). Se sometió entonces el documento a los posibles proveedores para decidir qué compañía construiría los IMP's. IBM y CDC contestaron que no era posible construir tales cosas porque no había computadora al-

[1] Formalmente hablando diríamos que siguien convenciones diferentes para representar sus datos y sus instrucciones.

guna que fuera suficientemente grande como para llevar a cabo la labor de un IMP y que fuera suficientemente barata como para pensar en tener un gran número de ellas constituyendo la subred de comunicación; en total se recibieron alrededor de una docena de propuestas. A finales de 1968 se decidió que la compañía agraciada fuera *Bolt Benarek y Newman* (BBN), una pequeña firma de consultores de Cambridge cuyo primer proyecto había sido el diseño de la acústica del edificio de las Naciones Unidas. BBN eligió una minicomputadora Honeywell (DDP-516 con 12K de memoria) como la que habría de hacer las funciones del IMP. El senador Edward Kennedy felicitó a BBN mediante un telegrama por haber ganado la licitación para construir el "*interfaith*" (en vez de "interface") *message processor*.

El equipo de ARPA se puso a trabajar con el de BBN, encabezado por Robert Kahn, un profesor de ingeniería eléctrica del MIT con licencia en BBN; además se integró un grupo de estudiantes de cuatro universidades, las que albergarían los primeros cuatro anfitriones, UCLA (concretamente el *Network Measurement Center* de Kleinrock), Stanford (concretamente el *Stanford Research Institute* o SRI), UC en Santa Barbara y Utah. Este grupo de estudiantes graduados (que constituyeron el NWG o *Netwok Working Group*) se reunían con frecuencia, discutían y escribían luego las minutas correspondientes. Los estudiantes suponían la existencia de un equipo de diseño de protocolos constituido por relevantes y doctos profesionales; éste no existía, pero Steve Crocker, el miembro del grupo encargado de redactar las minutas, temia que estas resultaran ofensivas para el imaginario equipo de diseño, que podía pensar que un conjunto de muchachos irreverentes estaban pretendiendo inmiscuirse en su trabajo, así que decidió hacer evidente que los documentos eran borradores, algo que requería de revisión y se sometía a consideración de todos, de allí surgió el nombre *Request for Comments* o RFC que aún poseen los documentos donde se especifican los protocolos de Internet. El primer RFC salió a la luz en abril de 1969 y trata sobre el software elemental de comunicación que debe poseer el anfitrión.

Luego de arduos trabajos, en octubre de 1969 se pudieron interconectar el SRI y UCLA. Un mes más tarde se incorporó la Universidad de California en Santa Barbara y para fines de año finalmente Utah. Todas a través de líneas de 50Kbps (kilo bits por segundo) de AT&T. El primer protocolo de conexión anfitrión-anfitrión proveía de la capacidad elemental de entrar en sesión en una máquina desde otra y se le llamó Telnet; todavía se usa, aunque cada vez menos debido a cuestiones de seguridad.

En 1971 surgió FTP, uno de nuestros conocidos protocolos para transferir archivos, que también tiende a usarse cada vez menos por las mismas razones de Telnet.

En 1972 un ingeniero de BBN, Ray Tomlinson, escribió un programa experimental para transferir archivos entre computadoras y luego decidió modificarlo para transferir un mensaje escrito por el usuario de un sistema al espacio de archivos asociado a otro usuario en una máquina diferente (el destinatario del mensaje). Esto se convirtió luego en el correo electrónico y de la necesidad de Tomlinson de separar el nombre del usuario al que iba destinado el mensaje, del nombre de la máquina en la que dicho usuario estaba registrado surgió el uso actual de la "@"[2].

En 1973 se añadió la primera conexión internacional a ARPANET, el *University College* de Londres y en ese mismo año Robert Kahn y Vinton Cerf diseñaron un nuevo protocolo para la red. Este diseño fue publicado en 1974 en un artículo titulado *Un protocolo para la interconexión de redes de paquetes* (*A Protocol for Packet Network Interconnection*) y el protocolo se llamó TCP (*Transmission Control Program*; poco después se reemplazaría "program" por "protocol" como lo conocemos ahora). En 1978 se decidió cambiar el diseño de TCP y partirlo en dos protocolos. El primero es IP (*Internet Protocol*), encargado de fragmentar los datos a enviar formando paquetes de tamaño apropiado y añadiendo los datos de remitente y destinatario, además de decidir la ruta inmediata de envío (el primer "salto" de los que hay que dar entre máquinas para que el paquete llegue a su destino). El segundo es el propio TCP, que ahora se encarga de hacer confiable la transmisión de datos, asegurando que los paquetes que los contienen sean recibidos por el destinatario y recuperados en el orden correcto.

A lo largo de su vida ARPANET, que ahora conocemos como Internet, ha evolucionado y ha ido cambiando su configuración y sus mecanismos de funcionamiento, entre ellos sus protocolos. Ya mencionamos las transformaciones de TCP. También Telnet y FTP han sufrido varias transformaciones a lo largo de los años. Actualmente a la familia de protocolos de Internet (que exceden los 100) se les conoce como TCP/IP genéricamente.

En 1990 Tim Berners-Lee, un programador del Centro Europeo de Investigación de Partículas en Suiza (CERN por sus siglas en francés), elaboró una propuesta para resolver un problema típico del CERN. En el centro (o laboratorio, como prefiere autonombrarse) laboran varios miles de investigadores en diversas áreas relacionadas con la física de partículas elementales. Es frecuente que ingresen nuevos investigadores o que algunos de los que ya están se cambien ocasionalmente del área de trabajo en que están a algu-

[2]Por cierto que la "@" causó controversia, Tomlinson la eligió porque en el teclado del teletipo que utilizaba existía y sin embargo Tenex, el sistema operativo de la máquina en que trabajaba, no lo utilizaba. Pero los usuarios de otro sistema, Multics, utilizaban el símbolo para cancelar la línea de comando completa en que aparecía.

Figura 9.3: Tim Berners-Lee.

na otra relacionada, esto significa que permanentemente hay personas que desean saber acerca de los recursos disponibles (dispositivos de medición, software, material impreso, etc.) para trabajar en el área que les interesa y en la que aún no se encuentran bien adaptados. También desean saber qué cosas ya se han hecho o intentado hacer, cuáles fueron los resultados de algunos experimentos que se hicieron hace años; para obtener esta información se recurre frecuentemente a lo que podríamos llamar "la tradición oral", lo que los colegas dicen en los pasillos o en la sala de café. Algo de esta valiosa información se pierde irremediablemente o es sumamente difícil de obtener. A pesar de que esta información se almacene en dispositivos como discos de computadora o cintas, luego de algún tiempo, con los cambios de hardware, algo se pierde. Berners-Lee escribió en su propuesta: "La estructura real de la organización [refiriéndose al CERN] es una telaraña [web en inglés] cuyas interconexiones evolucionan con el tiempo. En este ambiente, cuando llega una nueva persona o alguien toma una nueva tarea, normalmente tiene pocas pistas acerca de las personas con las que es útil hablar. La información acerca de los recursos existentes y cómo encontrarlos viaja a través de los corredores como chisme y ocasionalmente en alguna publicación informativa y los detalles acerca de lo que se necesita hacer se esparcen de manera similar. Tomando todo esto en consideración, el resultado es notablemente

exitoso, a pesar de los malos entendidos y la duplicidad de esfuerzos", las anotaciones entre corchetes fueron añadidas por el autor de estas páginas y no aparecen en el manuscrito original.

Para resolver el problema se necesita un sistema de manejo de información que haga posible que ésta pueda crecer y evolucionar junto con la organización, un sistema en el que la información no permanezca estática sino que sea capaz de modelar, de alguna manera, esa maraña de relaciones cambiantes que se establecen. Berners-Lee dice: "...una telaraña de notas con ligas (como referencias) entre ellas es mucho más útil que un sistema con una jerarquía fija. Cuando se describe un sistema complejo, muchas personas recurren a diagramas con círculos y flechas. [Esto] le deja a uno la libertad para describir relaciones entre las cosas, algo que no se puede hacer con tablas, por ejemplo". En el mismo documento Berners-Lee hace notar que el problema del CERN es, de hecho, un problema del mundo entero.

En síntesis, la solución propuesta por Berners-Lee es un sistema donde sea posible modelar la relación entre diversos documentos mediante ligas entre ellos. Este es el concepto de *hipertexto* que resulta tan común hoy en día.

El proyecto de Berners-Lee fue aprobado y para octubre ya tenía a su disposición una computadora (de hecho un cubo) de NeXT donde para fines de diciembre de 1990 ya tenía corriendo un programa de manejo de hipertexto llamado *World Wide Web* (telaraña de tamaño mundial) enfatizando el símil que anotó en su propuesta.

La información manejada por el sistema de Berners-Lee estaba constituida exclusivamente por archivos de texto contenidos en varias computadoras; desde un documento residente en una computadora A era posible hacer referencia (tener una liga) a otro documento físicamente almacenado en otra computadora B. El acceso del segundo documento a partir del primero era transparente para quien decidía seguir la referencia. Debajo del sistema había un protocolo que permitía que el usuario no se percatara de la transferencia de información que tenía lugar y que hacia posible el acceso, este protocolo es el conocido HTTP (*Hypertext Transfer Protocol*).

En 1992 un grupo de estudiantes de la Universidad de Illinois en Urbana (UIUC), encargados de mantener la conectividad internacional al centro de supercómputo de la universidad, el *National Center for Supercomputing Applications* (NCSA), encabezado por Marc Andersen, se interesaron en el protocolo utilizado por Berners-Lee para su sistema en el CERN y decidieron proporcionarle una interfaz amigable, lo que lo haría más útil para personas no entrenadas en el uso de computadoras. Al ver los resultados algunas personas les solicitaron el mismo programa para computadoras personales

basadas en Intel o Macintosh y finalmente decidieron dejar el programa disponible a todo público en Internet en 1992, el programa se llamó *Mosaic*. Para 1993 un millón de personas ya había hecho uso de la mancuerna Mosaic-HTTP. En 1994 Andersen, ya graduado, fundó su propia compañía para desarrollar su programa de interfaz gráfica para web, entre otras cosas. La compañía se llama *Netscape*.

En 1970 la red de ARPA creció a un tasa de un nodo por mes. En 1980 eran 213 anfitriones, en 1984 se sobrepasó la barrera de los 1000 anfitriones, para 1989 se rebasaron los 100,000, en 1992 el millón, en 1996, ya con la telaraña mundial montada sobre Internet, los 10 millones, en julio de 2000 eran poco más de 93 millones de anfitriones.

Pero Internet no sólo ha crecido en tamaño, también ha crecido en la cantidad de información y temas disponibles. En 1992 en Internet había casi exclusivamente información relacionada con el área de la computación: software, documentación de programas, artículos. Ocasionalmente era posible encontrar algo ajeno a las computadoras o a la computación, porque esencialmente los usuarios de la red eran personas del área. Con el advenimiento de herramientas de acceso poderosas y fáciles de utilizar, cada vez más personas interesadas en diversas áreas empezaron a usar Internet. Hoy en día es posible encontrar información de casi cualquier cosa: historia, economía, noticias, arte, pasatiempos, sitios comerciales, ciencia, política. Prácticamente cualquier cosa.

Con frecuencia se alzan voces que dicen que Internet es un depósito de basura mundial y algunas incluso claman para que se legisle sobre lo que es puesto en la red. Ciertamente es posible encontrar en Internet sitios dedicados al racismo, la explotación sexual de menores, pornografía y fundamentalismo religioso, pero también hay, y son más, los sitios dedicados a poesía, pintura, historia, deportes o ciencia (y de vanguardia). Esas páginas han sido elaboradas por personas que no buscan beneficio propio, son una manifestación de generosidad, han sido hechas porque alguien cree que es bueno que otra persona, que ni siquiera conoce, se entere de lo ocurrido en la guerra de los cien años o lea poesía japonesa o comprenda cómo se demostró el último teorema de Fermat. Internet es la síntesis de lo humano, una gigantesca metáfora del mundo: como todas las creaciones humanas, es un reflejo de lo que somos, lo deleznable y lo sublime.

William Blake lo dijo con mejores palabras. En uno de sus poemas se lee:

> La misericordia tiene corazón humano;
> la compasión, humano rostro;
> el amor, divina forma humana
> y la paz, humanos atavíos.

y en otro:

> La crueldad tiene corazón humano
> y la envidia humano rostro;
> el terror reviste divina forma humana
> y el secreto lleva ropas humanas.

10

Una mirada al horizonte

A fines de mayo de 1953 Edmund Hillary y el sherpa Tenzing Norgay lograron por primera vez llegar a la cumbre del monte Everest, llamado Chomolungma por los tibetanos y Sagarmatha ("diosa madre" en sánscrito) por los nepaleses. Luego de arribar a la cúspide y felicitarse mutuamente, Hillary se quitó la mascarilla de oxígeno y al volverse hacia Tenzing se percató de que éste estaba orando y enterrando algunos alimentos en la nieve como ofrenda a la diosa madre por haberle permitido acceder a su santuario, nunca antes hollado por pies humanos. Cuando arribaron al campamento de altura donde los esperaba George Lowe, su compañero de expedición, Hillary exclamó exultante: "Bueno George, ¡Vencimos al bastardo!"[1]. Al correr de los años se entabló una absurda batalla acerca de quién de los dos escaladores había llegado primero a la cima, Tenzing procuró permanecer callado ante las declaraciones de Hillary, que incluso ofendían su dignidad, y nunca negó que fuera Hillary el primero en llegar (aunque desmintió las versiones en las que el aparecía como un fardo que hubo que izar hasta la cima). Sin embargo todo este chisme indigno sí llegó a afectar a Tenzing que enfermó de alcoholismo a causa de la depresión que lo afectó hasta el final de sus días.

Esta pequeña anécdota nos permite contrastar dos idiosincrasias opuestas, dos concepciones diferentes del mundo y del hombre. La actitud occidental muy al estilo de la descrita de principios del siglo, en la que el mundo es el enemigo a vencer, "el bastardo" que hay que derrotar y el hombre es el domador, la cúspide de la creación, el dueño de todo cuanto le rodea. La misma actitud que ha caracterizado a la cultura occidental desde hace

[1] "Well George, we knocked the bastard off!"

siglos, primero éramos el centro del universo, luego Copérnico y Galileo nos hicieron ver que no, pero por lo menos éramos el centro de la creación, el mundo fue creado para nosotros, hasta que Darwin nos demostró que no. Nos la hemos pasado regateando con la ciencia en busca de obtener siempre una posición tan privilegiada como sea posible. Hemos llegado al extremo de decir que como en general el género humano no es privilegiado, algún subconjunto de él si lo es: los arios, los blancos, los croatas, los de mi mismo sexo, los de mi misma religión, en fin...

Opuesta a esta concepción del mundo y del hombre está la humildad de Tenzing: "soy parte integral del mundo, vivo en él por gracia de la naturaleza (encarnada en multitud de dioses y diosas), quien me provee de lo que necesito y quien puede aplastarme con su poder cuando lo desee". Percatarse de esto hace al hombre más humilde y más sabio, más sabio cuanto más ignorante se sabe, la paradoja de Sócrates. Afortunadamente esa actitud ha cobrado fuerza entre los científicos modernos. En general ya no queremos forzar a la naturaleza a entregarnos sus secretos, ya no la obligamos a adaptarse a lo que creemos que es, ya no creemos que la tierra es el centro del universo porque así debe ser, ahora pensamos que es mejor conocerla bien para obtener sus secretos, que sólo es posible conocer el mundo estudiando cómo se comporta, que nuestras teorías deben describir lo que vemos o están mal. Nos percatamos de que es mejor hacer el amor que practicar una violación.

De esta actitud humilde han surgido algunas cosas interesantes en computación. Todas ellas son, en cierta forma, una respuesta a la pregunta ¿qué podemos aprender de la naturaleza?

"Aunque el ingenio humano puede lograr infinidad de inventos, nunca ideará ninguno mejor, más sencillo y directo que los que hace la naturaleza, ya que en sus inventos no falta nada y nada es superfluo" escribió Leonardo de Vinci[2]. Un siglo después Johannes Goldschmidth, mejor conocido por su nombre latino *Fabricius ab Aquapendente* (quien se disputara con Galileo el descubrimiento de las manchas solares), afirmaría anticipándose a Darwin: "La naturaleza perpetúa aquello que resulta lo mejor" (*De Motu Locali Animalium*). Todos los seres vivos que habitamos en este planeta somos, de alguna manera, obras casi perfectas. Cuando nos percatamos de la vasta complejidad que somos, de cómo nuestro cuerpo es una enredada madeja de relaciones delicadamente establecidas entre los distintos órganos, no podemos más que maravillarnos, y otro tanto ocurre cuando percibimos al resto de los seres vivos de la misma manera.

[2] *Cuaderno de Notas. La vida y estructura de las cosas, el embrión*

A mediados de la década de los 90's un equipo de ingenieros aeronáuticos se puso a estudiar el problema de cómo terminar un ala de avión de tal forma que no generara turbulencias. Los flujos turbulentos en ingeniería aerodinámica e hidrodinámica no son buenos porque significan una pérdida de energía del móvil que los genera y suelen aparecer en los extremos de las alas de avión. Sobre el ala y debajo de ella el flujo de viento es uniforme, lo que se suele llamar *laminar*, pero al terminar abruptamente el ala el flujo laminar se vuelve turbulento, se generan remolinos en los que el avión gasta parte de su combustible inútilmente. Para resolver el problema, generalmente se colocan unas placas perpendiculares a la superficie del ala, pero esta solución no es del todo efectiva.

Los ingenieros se preguntaron ¿cómo hará la naturaleza para resolver el problema? ¿Cómo hacen las aves, también pierden energía? Eligieron un ave que volara grandes distancias sin abastecimiento, de tal forma que fuera crítico para ella conservar su energía. La cigüeña fue la elección. Luego colocaron un ala de cigüeña en un túnel de viento e investigaron cómo se comportaba el flujo de aire en el ala a diferentes velocidades. Las cigüeñas tienen, al final de las alas, unas plumas que se abren cuando el ave vuela, como los dedos de una mano extendida. Los científicos se percataron de que dichas plumas se abrían más mientras mayor fuera la velocidad del viento y también se percataron de que se generaban remolinos en cada una de ellas. Resultó que estos dos fenómenos combinados tenían como efecto global generar una fuerza ascendente que contribuía a mantener al ave en vuelo. La cigüeña no sólo pierde muy poca energía en el flujo turbulento del final de sus alas, sino que además gana sustentación aprovechando este flujo que normalmente es un problema. "... cuanto más informados e iluminados estemos acerca de las obras de Dios, más inclinados estaremos a encontrarlas excelentes y totalmente conformes a cuanto se hubiera podido desear" diría Leibniz (*Discours de Metaphysique*).

La naturaleza (o Dios como prefiere Leibniz) genera seres óptimos, seres perfectamente adaptados a su entorno constituido por una infinidad de circunstancias: temperatura, presión atmosférica, precipitación, velocidad del viento, altitud, nivel de insolación, depredadores, alimentos, etc. Cada ser vivo es perfecto para vivir en su entorno, fuera de él perecería, su adaptación es perfecta porque, a lo largo de miles de generaciones se ha perfeccionado a pequeños saltos infinitesimales, lo que vemos hoy en día es el resultado acumulado de miles de experimentos exitosos que han ido refinando paulatinamente alguna creación primigenia. Los experimentos fallidos, probablemente algunos órdenes de magnitud más numerosos que los exitosos, no los vemos. Los individuos resultantes perecieron compitiendo al lado

de otros más aptos para sobrevivir. "La selección natural obra solamente mediante la conservación y acumulación de pequeñas modificaciones heredadas, provechosas todas al ser conservado" escribió Darwin dando nombre a este proceso[3].

Estas "modificaciones heredadas", señaladas por Darwin como las generadoras de organismos mejores, son llamadas hoy en día mutaciones y constituyen el motor de la evolución. Un organismo mutante ha sufrido una modificación que lo hace diferente al resto de sus congéneres. Esta modificación puede ser inconveniente para él (la falta de algún miembro útil de su cuerpo, por ejemplo), pero puede ocurrir también que la modificación le confiera alguna cualidad que le permite sobrevivir más fácilmente que al resto de los individuos de su especie; este organismo tendrá mayor probabilidad de reproducirse y heredará a sus descendientes la característica que le dió ventaja. Con el tiempo, gracias a la competencia, los organismos que en un principio eran raros se volverán comunes a costa de la desaparición del "modelo anterior". Se habrá dado entonces un paso en el proceso evolutivo.

Esta cualidad del proceso natural de la evolución (generar organismos óptimos sobre los que influyen infinidad de variables), llamó la atención de algunos científicos en las décadas de los 50's y 60's. Un alemán, Rechenberg, introdujo en 1965 lo que denominó *Evolutionsstrategie* o estrategias evolutivas, como un método para optimizar funciones de varias variables que definían dispositivos tales como perfiles de alas de avión. En 1966 otros investigadores, Fogel, Owens y Walsh, se dieron a la tarea de dejar evolucionar máquinas de estados finitos sometiendo sus diagramas de transición a cambios aleatorios (mutaciones), creando con ello lo que se conoce como *programación evolutiva*. También en los 60's John Holland, junto con algunos de sus colegas y alumnos de la Universidad de Michigan, desarrolló lo que se conoce como *Algoritmos Genéticos* (AG's). El objetivo de Holland, sin embargo, no era inventar un método de optimización basado en los mecanismos de la evolución natural, sino elaborar un estudio formal acerca del fenómeno de adaptación en sistemas naturales y artificiales. Es decir, definir un modelo para la adaptación; el algoritmo genético definido por Holland en su trabajo es un intento de abstraer las características esenciales del proceso evolutivo como se observa en la naturaleza, con vistas a importarlo a sistemas de cómputo.

Actualmente los AG's son preferentemente utilizados como métodos de búsqueda de soluciones óptimas para problemas en los que hallar esas soluciones, ya sea exactamente por métodos analíticos, o aproximadamente por

[3] *El origen de las especies, selección natural*

métodos tradicionales basados en el cálculo diferencial e integral y el álgebra lineal, es difícil. Han sido usados con éxito en la solución de problemas de optimización combinatoria, optimización de funciones reales y como mecanismos de aprendizaje de máquina (*machine learning*). Esto último les ha ganado un lugar en el campo de la inteligencia artificial.

El descubrimiento de Watson y Crick de la estructura del ADN (ácido desoxirribonucléico) nos ha proporcionado buenas pistas acerca de cómo opera la naturaleza para lograr tales maravillas. Decidir la forma y características de un individuo óptimo para vivir en un cierto medio es complicado, así que lo primero es *codificar* las diferentes formas y características de alguna manera manipulable. Para eso es justamente el ADN, el código genético de los seres vivos.

Nosotros somos lo que somos en función de las proteínas que producimos. Éstas determinan nuestro color de ojos, estatura, color de piel, capacidad de resistir los rayos del sol, en fin, todas nuestras características o casi todas. Qué proteínas producimos es justamente lo que se encuentra codificado en nuestros genes mediante largas moléculas de ADN. Estas moléculas están constituidas exclusivamente por cuatro substancias elementales llamadas bases, *adenina*, *timina*, *citosina* y *guanina*. La adenina sólo puede ensamblarse con la timina, y la citosina sólo con la guanina. Cada conjunto de tres bases (tripleta) determina un aminoácido, que son los constituyentes de las proteínas. Manipular este código, tripletas de bases, es mucho más sencillo que manipular formas, colores, estaturas, etc. así que la naturaleza simplifica su problema usando el código genético. Éste es el primer paso.

A la hora de resolver un problema concreto, el conjunto de posibles soluciones se codifica; de esta forma cada posible solución consiste en una cadena de símbolos y el algoritmo genético manipulará estas cadenas de modo análogo como la naturaleza manipula nuestros códigos genéticos. Seleccionará preferentemente cadenas que correspondan a mejores alternativas de solución de entre un conjunto de posibles soluciones para reproducirlas y mezclarlas y así generar un nuevo conjunto que resultará mejor al anterior; a este conjunto se le suele llamar población, y a cada propuesta de la población, individuo, por analogía con los sistemas de seres vivos. Aun cuando en la naturaleza el proceso evolutivo tome varios millones de años, en una computadora es posible simular la evolución de una población inicial a lo largo de cientos de generaciones en unos cuantos minutos.

En su trabajo Holland propone una manera de seleccionar individuos y de cruzarlos y aunque existen muchas otras propuestas, las de Holland constituyen aún hoy la base de muchos desarrollos teóricos y prácticos en el área. El algoritmo de Holland con algunas modificaciones menores suele

llamarse *Algoritmo Genético Simple* (SGA). En el SGA se considera que los códigos genéticos están en binario, algo natural si se piensa almacenarlos en la memoria de una computadora. Explicado en detalle el proceso de un SGA es:

1. Decidir cómo codificar el dominio del problema.
2. Generar un conjunto aleatorio (población inicial) de N posibles soluciones codificadas al problema; a éste se le llamará la población actual.
3. Calificar cada posible solución (individuo) de la población actual.
4. Seleccionar dos individuos de la población actual con una probabilidad proporcional a su calificación.
5. Tirar un "volado sesgado"[4] (con probabilidad p_c cae águila).
6. Si cayó águila mezclar los códigos de los dos individuos seleccionados para formar dos híbridos a los que llamaremos *nuevos individuos.*
7. Si cayó sol llamamos a los individuos seleccionados *nuevos individuos.*
8. Por cada bit de cada nuevo individuo tirar otro "volado sesgado" (con probabilidad p_m cae águila).
9. Si cae águila cambiar el bit en turno por su complemento.
10. Si cae sol el bit permanece inalterado.
11. Incluir a los dos nuevos individuos en una *nueva población.*
12. Si la nueva población tiene ya N individuos llamarla *población actual* y regresar al paso 3 a menos que se cumpla alguna condición de terminación.
13. Si no, regresar al paso 4.

La condición de terminación, a la que se hace referencia en el paso 12, puede definirse de muchas maneras. Se puede fijar un número máximo de generaciones en las que se pretende correr el algoritmo, o puede decidirse hacer alto cuando la mayoría de la población, digamos el 85%, tenga una

[4]Tirar un volado es lanzar una moneda al aire. "Sesgado" significa que no es igualmente probable que caiga "águila" (cruz) o "sol" (cara). En términos técnicos a esto se le denomina un *experimento de Bernoulli.* Sólo pueden ocurrir exclusivamente dos eventos, uno con probabilidad p y otro con probabilidad $1 - p$

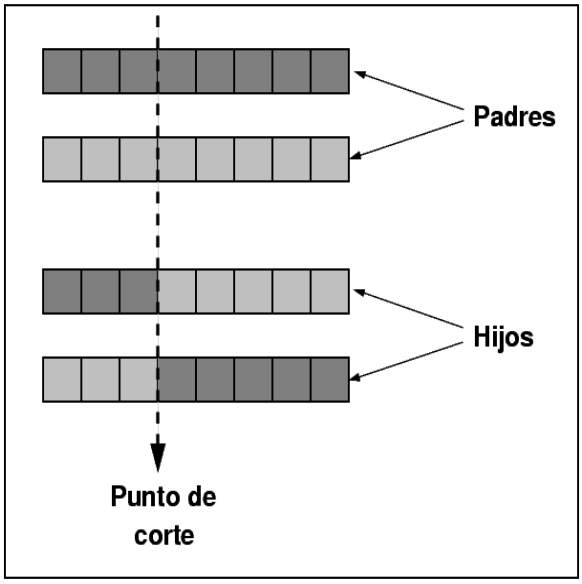

Figura 10.1: Cruzamiento de un punto (*1-point crossover*).

calificación que esté dentro de 0.6 desviaciones estándar de la media. En fin, opciones hay muchas, generalmente la decisión depende del problema o de preferencias personales acerca de cuándo es conveniente detenerse.

En el paso 4 se menciona que hay que seleccionar dos individuos con probabilidad proporcional a su calificación. Este tipo de *selección proporcional* es también llamado de "rueda de ruleta" (*roulette wheel selection*) por lo siguiente: supóngase que se suman las calificaciones de todos los individuos de la población y esta suma es considerada el 100% de una circunferencia; luego a cada individuo se le asigna el trozo de ésta, una "rebanada del pastel" que le corresponde según su aportación a la suma de las calificaciones.

Aún queda por aclarar cómo es que se mezclan los códigos de dos individuos para formar dos híbridos. En general hay muchas maneras de hacerlo, al lector mismo se le pueden ocurrir muchas; sin embargo, la que se usa en el SGA y una de las más populares es el cruzamiento de un punto (*1-point crossover*). En este tipo de cruzamiento, dados dos individuos, se elige aleatoriamente un punto de corte entre dos bits cualesquiera del cromosoma, esto define segmentos izquierdos y derechos en cada genotipo, se procede entonces a intercambiar los segmentos derechos (o los izquierdos indistintamente) de cada individuo, de esto resultan, al igual que en el caso del cruzamiento de cromosomas, dos híbridos.

Los algoritmos genéticos, las estrategias evolutivas y la programación genética constituyen lo que se conoce como computación evolutiva y han adquirido gran popularidad. Actualmente es una de las áreas más investigadas en computación y sin duda será la base para futuros desarrollos; un punto de partida interesante para la exploración de nuevos rumbos. No sería descabellado pensar, por ejemplo, que nuestras futuras computadoras vayan aprendiendo cosas a lo largo de su existencia, tal como lo hacemos nosotros y muchos animales. Quizás en un futuro no muy lejano nuestras computadoras sean realmente personales, en el sentido de ir adaptando su comportamiento de acuerdo a lo que aprenden acerca de las labores típicas de su dueño y el modo peculiar que tenga de hacerlas y tampoco sería descabellado pensar en que este aprendizaje se lleve a cabo usando computación evolutiva. Incluso el hardware del futuro podría auto-evolucionar.

Ésta no es la única vertiente basada en el funcionamiento de sistemas vivos. A principios de la década de los 90's el italiano Marco Dorigo escribió su tesis de doctorado en el Politécnico de Milán, cuyo título traducido al español es: *Optimización y aprendizaje por medio de algoritmos naturales.* Ése fue el punto de partida para lo que se ha llamado *sistemas de hormigas.*

Las hormigas no son seres muy inteligentes, sólo saben hacer unas cuantas cosas sencillas y están programadas para hacer ésas y sólo esas cosas, el libre albedrío es algo desconocido en su mundo. Sin embargo, las labores que puede llevar a cabo una colonia de hormigas son sorprendentes igual que las que son capaces de hacer otros insectos sociales como las abejas o las termitas. En todos estos casos el individuo es, por sí solo, casi insignificante, el catálogo de actividades que es capaz de realizar es muy limitado; sin embargo la colonia es un ente mucho más inteligente, un arquitecto, un planificador, un administrador excelente. El comportamiento inteligente es propiedad de la colonia y no de sus elementos constitutivos. En estos casos se dice que el comportamiento organizado que exhibe la colonia es una propiedad *emergente* (de emerger, surgir espontáneamente). Nadie lo puso ahí, se da solo.

Se han hecho experimentos con sistemas similares, organismos artificiales sencillos que siguen unas cuantas reglas y que, en conjunto, exhiben un comportamiento organizado. En el MIT se elaboraron una serie de robots muy simples que sólo sabían caminar en una línea recta en dirección aleatoria y cuando se topaban con un objeto lo levantaban y caminaban con él en otra dirección; cuando estaban cargando algo y se topaban con otro objeto dejaban el que traían consigo en ese lugar y volvían a caminar en una dirección aleatoria (si chocaban con algo mayor, como una pared, sólo cambiaban

de dirección). Se pusieron varios de estos robots en un área restringida con algunos objetos que los robots podían levantar. Al principio se empezaron a hacer pequeños cúmulos de objetos acarreados por los robots y luego de un tiempo construyeron un solo cúmulo. Se han simulado también termitas o cardúmenes de peces.

Dorigo ha podido resolver con sus hormigas artificiales problemas que en computación se consideran difíciles, como el conocido problema del agente viajero. Este problema consiste en determinar el viaje redondo más corto posible para un agente viajero que debe regresar a la misma ciudad A de donde partió, luego de recorrer todas y cada una de las ciudades de un conjunto, sin pasar más de una vez por cada ciudad.

Las hormigas reales se guían por su sentido del olfato. Todos hemos notado cómo las hormigas se forman en hileras interminables por donde transitan de ida y vuelta incansablemente. Esa ruta esta impregnada del olor de una sustancia producida por cada hormiga, una feromona. Mientras mayor sea el olor a feromona en una dirección, más atractiva será ésta para una hormiga, mientras más hormigas pasen por un lugar, mayor será el olor a feromona de ese lugar. Así que si ponemos a caminar hormigas en las rutas del agente viajero, el número de hormigas que circulan por la ruta más corta en un intervalo de tiempo determinado será mayor que el de las que han circulado por las rutas más largas (pasan más hormigas por unidad de tiempo), así que el olor en esa ruta tenderá a hacerse más poderoso y las hormigas tenderán a elegirlo más frecuentemente que a cualquier otro camino. Luego de un tiempo casi todas las hormigas circulan por el camino más corto.

En 1994 Leonard M. Adleman, reconocido miembro de la comunidad de ciencia de la computación en el mundo, hizo un experimento que inició una nueva área de investigación. Adleman había hecho algo de investigación acerca del SIDA pero estaba preocupado porque no tenía mucho éxito al tratar de comunicarse con el resto de la comunidad científica que investigaba en esa misma área, así que se decidió a estudiar algo de biología molecular. Durante su periodo de estudio leyó el libro de Watson y Crick: *The Molecular Biology of the Gene*, cuando llegó a la parte referente a la polimerasa, la enzima que se encarga de copiar la molécula de ADN, notó la similitud entre su funcionamiento y el de una máquina de Turing.

Ya mencionamos que la timina sólo puede pegarse a la adenina y la citosina sólo a la gunanina, así que la polimerasa, cada vez que se encuentra una adenina en la cadena existente pone una timina en la que está construyendo y viceversa; y cada vez que se encuentra una citosina pone una guanina y viceversa. En esencia pues, la polimerasa lee, uno a uno, los símbolos que

constituyen la cadena molecular de entrada y escribe en otra cadena molecular, la de salida, los símbolos complementarios. El pequeño alfabeto de la polimerasa posee cuatro símbolos diferentes, podría pensarse que es demasiado pequeño, pero el modelo de la máquina de Turing sólo necesita dos (sistema binario otra vez) y es capaz de calcular todo lo calculable según la tesis de Church. Así que la polimerasa es, esencialmente, un dispositivo de cómputo simple.

Construyendo moléculas con secuencias de bases específicas y utilizando algunos procedimientos químicos, electroquímicos y enzimas, Adleman pudo resolver un caso particular del problema del agente viajero con 14 ciudades.

Las moléculas de ADN ocupan muy poco espacio, en un centímetro cúbico de volumen de ADN hay tanta información como en varios millones de discos compactos actuales (al menos le caben unos 700 millones de caracteres a cada uno). Además se puede efectuar un cálculo en cada una de los millones de moléculas presentes en ese volumen al mismo tiempo si se tiene el número necesario de enzimas, que también ocupan poco espacio. Por si esto fuera poco, la energía necesaria para efectuar trillones de operaciones elementales en una cadena (pegar bases) es insignificante. Todo esto hace pensar que es posible construir computadoras que utilicen moléculas como el ADN para hacer cálculos útiles, *computación molecular*.

Por supuesto hay muchos problemas técnicos que resolver antes de poder construir una computadora molecular, por ejemplo el cálculo llevado a cabo por Adleman tardó siete días en terminar, debido en gran parte a las mediciones y los procedimientos electroquímicos que hay que efectuar. Sin embargo es un área prometedora: Richard J. Lipton de Princeton y Daniel Boneh de Stanford han delineado un procedimiento por el que es posible que una computadora de ADN descifre los mensajes codificados con el esquema DES (*Data Encription Standard*), uno de los esquemas criptográficos más usuales. Es un buen sistema de cifrado de información porque sería muy tardado para una de nuestras actuales computadoras buscar cuál, de entre muchos millones de palabras clave, fue la utilizada para cifrar un mensaje. Pero en una computadora de ADN estas posibilidades pueden explorarse todas a un tiempo.

En varias partes del mundo, tanto empresas de manufactura de computadoras como universidades, tienen programas de investigación en computación molecular. Lo mejor está por venir.

En 1982 apareció publicado un artículo en el *International Journal of Theoretical Physics* titulado "Simulando Física en Computadoras" (*Simulating Physics with Computers*). El autor era Richard Feynman, por ese entonces profesor de física en el Caltech. Feynman fue uno de los hombres

del proyecto Manhattan en Los Alamos, era egresado del MIT y de Princeton y ganó el premio Nobel en física en 1965 por su trabajo en electrodinámica cuántica. Además era un buen ejecutante de los bongoes, descifrador de jeroglíficos mayas y en 1986, casi un par de años antes de morir, fue el detective que descubrió la causa del desastre del transbordador espacial Challenger. Un hombre versátil, genial y con un excelente sentido del humor.

En el artículo, Feyman escribe que al parecer no es posible hacer simulaciones eficientes de sistemas propios de la mecánica cuántica en una computadora y especula que posiblemente en una computadora basada en los principios de la mecánica cuántica estas simulaciones se podrían llevar a cabo eficientemente. Esto lleva naturalmente a pensar en que si una computadora basada en principios cuánticos puede hacer algo, en particular simulaciones de mecánica cuántica, más eficientemente que una computadora convencional, entonces es posible que la computación en general se lleve a cabo más eficientemente en una de tales computadoras.

En 1994 Peter Shor demostró que esta especulación es cierta. En el 35^o Simposium del IEEE acerca de los fundamentos de las ciencias de la computación, presentó un artículo: *Algoritmos para Computación Cuántica: Logaritmo Discreto y Factorización* (*Algorithms for Quantum Computation: Discrete Log and Factoring*).

El problema del logaritmo discreto consiste en lo siguiente: dados tres números enteros b, x y m, encontrar un cuarto número k tal que: $b^k \equiv x \pmod{m}$, es decir, si se divide b^k entre m sobran x unidades. Al conjunto de los enteros $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ se le llama *el conjunto de las clases residuales módulo* m y son todos los posibles residuos de dividir cualquier número entre m. Este conjunto tiene propiedades interesantes cuando m es un número primo, de esos que sólo se pueden dividir por ellos mismos y por 1. En este caso encontrar el número k mencionado arriba es un problema difícil. Si m es además un número grande, de varios cientos de dígitos, entonces el problema es *muy difícil* en el sentido de que no puede ser resuelto, en general, por una computadora (aunque sea muy rápida) en un tiempo razonable.

El problema de la factorización, un poco simplificado, es el siguiente: supongamos que tenemos dos números primos p y q muy grandes en uno de estos conjuntos de clases residuales $\mathbb{Z}_m$, y que obtenemos el número n como el residuo de la división del producto pq entre m. Esto de hecho es obtener el *producto módulo* m. Si ahora le damos a alguien n y le pedimos que encuentre los dos primos p y q que lo produjeron, se tardará, en general, mucho tiempo, más que el que lleva de existir el universo según nuestros cálculos, aun con ayuda de una computadora. El teorema fundamental de

la aritmética dice que *cualquier* entero es expresable de manera única como el producto de números primos. Pero cuando hay que encontrarlos no es posible hacerlo eficientemente, al menos no en general.

Estos dos problemas son los representantes típicos de una clase de procesos que es fácil efectuar en un sentido pero muy difícil de llevar a cabo a la inversa. Dado un rompecabezas de 5000 piezas armado, es fácil desarmarlo, pero es, en general, muy difícil armarlo. A este tipo de procesos o funciones se les denomina de un solo sentido (*one-way functions*) y son de uso práctico en nuestros modernos sistemas criptográficos, esos que se encargan de que el número de nuestra tarjeta de crédito no pueda ser obtenido a partir de nuestra conexión de Internet con un sitio donde compramos algo.

En su artículo Shor muestra un algoritmo que podría ser ejecutado en una computadora cuántica y que resuelve los problemas mencionados en un tiempo computacionalmente razonable. De existir computadoras cuánticas nuestros actuales sistemas de cifrado de información dejarían de ser seguros. Por supuesto, de existir computadoras cuánticas seríamos también capaces de establecer nuevos sistemas criptográficos.

Así pues, en el paradigma de la computación cuántica es posible resolver eficientemente problemas que en nuestro actual paradigma de la computación no podemos. Por primera vez nos hallamos ante algo que nos hace dudar de nuestra hipótesis de trabajo fundamental, la tesis de Church-Turing que establece que todo aquello que es computable puede ser calculado por una máquina de Turing.

En nuestras computadoras actuales un registro de longitud de n bits contiene siempre uno y sólo uno de 2^n posibles números expresables en n bits. Si estuviéramos hablando de $n = 2$ entonces un registro de dos bits puede tener uno de los siguientes valores: 00, 01, 10 y 11. Pero si un registro utilizara las propiedades cuánticas de la materia podría estar simultáneamente en *todos* los estados posibles, de hecho en una superposición de los estados mencionados arriba. Un solo registro de longitud n está simultáneamente en 2^n estados, el número de estados es una función exponencial de la longitud del registro. Cada vez que se hiciera un cálculo usando el registro es como si se hicieran realmente 2^n cálculos diferentes al mismo tiempo, en paralelo.

Con frecuencia nos impresiona el poder de nuestras actuales computadoras paralelas o nuestros *clusters* de computadoras que poseen muchos procesadores trabajando al mismo tiempo. Sin embargo los problemas realmente difíciles de la computación no son efectivamente solubles en tiempo razonable ni por esas máquinas y esto es porque la complejidad de los problemas difíciles crece exponencialmente en función de la complejidad de los datos de entrada. Es decir, si una máquina resuelve un caso del problema con 2

datos en 4 segundos entonces tarda 8 segundos con 3 datos, 16 segundos con 4 datos, 1024 segundos con 10 datos y poco más de 1 millón de segundos con 20 datos. ¡No hay computadora que alcance!

Pero si pudiéramos tener computadoras "exponencialmente paralelas" como serían las computadoras cuánticas, capaces de procesar en un solo paso una cantidad exponencial de información, entonces pudiera ser que nuestros problemas difíciles dejaran de serlo.

Actualmente la computación cuántica se desarrolla en dos frentes. Uno es el teórico, que provee de modelos formales de computación y algoritmos ejecutables en computadoras cuánticas calculando sus grados de complejidad; en este rubro David Deutsch en 1985 formuló el equivalente cuántico de una máquina de Turing y en 1989 un modelo formal alternativo llamado *redes computacionales cuánticas*. En el otro frente, el tecnológico, en 1989 Charles Benett y su grupo de trabajo en IBM construyó el primer prototipo de una computadora cuántica, justamente para intercambiar claves criptográficas. Otros prototipos fueron construidos y probados en la Universidad Johns Hopkins en 1996 por el grupo encabezado por James Franson y en el laboratorio de Los Alamos en 1998 por el grupo encabezado por Richard Hughes y W. Buttler. Se han hecho otros muchos experimentos con más o menos buenos resultados, pero el problema fundamental radica en que, para que las cosas funcionen bien, la computadora debe estar completamente aislada de cualquier influencia externa ajena a las necesarias para operar. Cualquier partícula elemental, cualquier perturbación energética, un insignificante fotón, da al traste con todo el sistema.

La computación cuántica, la molecular, la evolutiva y la vida artificial son, sin dudarlo, importantes puntos de partida para el futuro. Quizás en este momento, en el inicio de un nuevo siglo, sea aún más importante hacerlo notar. Después de todo fue durante el siglo pasado que vimos surgir la ciencia de la computación y en sólo unos decenios ha alcanzado dimensiones insospechadas. Las matemáticas y la física tienen siglos de existencia, son grandes montañas de conocimiento que la humanidad ha construido piedra por piedra para poder ver más lejos en el horizonte y sin embargo nos siguen sorprendiendo todos los días, seguimos acarreando piedras; lo mejor de la ciencia de la computación está aún por venir. Quizás a fines del siglo XXI estudiemos la máquina de Turing como quien estudia la mecánica de Newton hoy en día, como un modelo válido bajo ciertas restricciones, como un caso particular de una teoría mucho más general. Probablemente las futuras generaciones sonrían al pensarnos ingenuos por no percatarnos de una gran roca que estaba allí y nunca vimos, como sonreímos cuando nos enteramos de que los griegos pasaron de largo junto al álgebra sin verla.

Primero nos paramos sobre nuestros pies y miramos el horizonte, luego, para poder ver más lejos, nos colocamos sobre los hombros de gigantes; después construimos una montaña de teoría que nos sustente para ver más lejos aún. Parados en la cima de nuestra modesta montaña sólo podemos elucubrar acerca de lo que hay más allá del horizonte, para poder verlo realmente sólo hay una posibilidad: volar un poco y después... continuar construyendo la montaña.

Bibliografía

[1] Adleman, Leonard M., "Computing with DNA", *Scientific American*, agosto 1998, pp 34-41.

[2] Barners-Lee, Tim, *Information Management: A Proposal*, CERN, marzo 1989, mayo 1990. Accesible en: `http://www.w3.org/History/1989/proposal.html`.

[3] Barrow, John D., *La Trama Oculta del Universo*, Crítica, 1992, Col. Drakontos.

[4] Bell, E. T., *Men of Mathematics*, Simon and Schuster, 1965.

[5] Bonabeau, Eric y Guy Théraulaz, "Swarm Smarts", *Scientific American*, Marzo 2000, pp 55-61.

[6] Boole, George, *An Investigation of the Laws of Thought*, Dover Publications Inc., 1958, 424 pp.

[7] Boyer, Carl B., *A History of Methematics*, 2a. ed., John Wiley and Sons. 1989.

[8] Cohen, John, *Los Robots en el Mito y en la Ciencia*, Grijalbo, 1969, Col. Dina.

[9] Copeland, J. y D. Proudfoot, "Alan Turing's Forgotten Ideas in Computer Science", *Scientific American*, abril 1999, Vol. 280, No. 4.

[10] Chesterton, Gilbert K., *Pequeña Historia de Inglaterra*, Espasa-Calpe, 1946.

[11] Dawson, J., "Gödel and the Limits of Logic", *Scientific American*, junio 1999, Vol. 280, No. 6.

[12] Dewdney, A. K., *The New Turing Omnibus*, Computer Science Press, 1993.

[13] Dorigo, Marco, "Ant Colony System: A Cooperative Learning Approach to the traveling Salesman Problem", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, abril 1997, pp 53-65.

[14] Freeman, James A. y David M. Skapura, *Neural Networks: Algorithms, Applications ans Programming Techniques*, Addison-Wesley, 1991, Computation and Neural Systems Series.

[15] Garcés, Contreras Guillermo,*Pensamiento Matemático y Astronómico en el México Precolombino*, Instituto Politécnico Nacional, 1990, 360 pp.

[16] Gershenfeld, Neil e Isaac L. Chuang, "Quantum Computing with Molecules", *Scientific American*, junio de 1998, pp. 66-71.

[17] Goldstine, Herman H., *The Computer: From Pascal to Von Neumann*, Princeton University Press, 1972.

[18] Greniewski, Henryk, *Cibernética sin Matemáticas*, Fondo de Cultura Económica, 1978.

[19] Hafner, Katie y Matthew Lyon, *Where Wizards Stay Up Late, The Origins of the Internet*, Simon & Schuster, 1996.

[20] Haugeland, John, *La Inteligencia Artificial*, Siglo XXI, 1988.

[21] Hilbert, David, *Sobre los Problemas Futuros de las Matemáticas*, en Comunicación Interna No. 7, Departamento de Matemáticas, UNAM, 1980, Serie Divulgación.

[22] Hofstadter, Douglas, *Gödel, Escher, Bach: An Eternal Golden Braid*, Basic Books, 1979.

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

[24] Lovett, Cline Barbara, *Los Creadores de la Nueva Física*, FCE, 1992, Breviarios 134.

[25] Moreau, René,*The Computer comes of Age*, MIT press, 1984.

[26] Nagel, Ernest y James R. Newman, *Gödel's Proof*, New York University Press, 1986, 118 pp.

[27] Nelson, R. J., *Introduction to Automata*, John Wiley & Sons, 1968.

[28] Rawlins, Gregory J. E., *Compared to What? An Introduction to the Analysis of Algorithms*, Computer Science Press, 1992.

[29] Rosenblatt, Frank, "The Perceptron: A Probabilistic Model for Information Storage and Organization", *Psychological Review*, vol 65, 1958, pp. 386-408.

[30] Shurkin, Joel, *Engines of the Mind*, W. W. Norton & Co., 1996.

[31] Smith, David E., *History of Mathematics*, Vol I, Dover Publications Inc., 1951.

[32] Turnbull, Herbert W. " Los Grandes Matemáticos", en *Sigma, El mundo de las Matemáticas*, James R. Newman, editor, Tomo I, Grijalbo, 1997.

[33] Vazirani, Umesh, "Introduction to Special Section on Quantum Computation", *SIAM Journal of Computing*, 26(5), pp. 1409-1410, octubre de 1997.

[34] Willerding, Margaret F., "Sistemas Antiguos de Numeración", en *Antología de Matemáticas*, Universidad Nacional Autónoma de México, 1983, pags. 42-52, Lecturas Universitarias.

[35] Williams, Colin P., *Explorations in quantum Computing*, Springer Verlag (TELOS), 1997.

[36] Williams, Michael R., *A History of Computing Technology*, 2a ed., IEEE Computer Society Press, 1998, 426 pp.

Índice de figuras

Índice analítico