No hay *Software* y otros ensayos sobre filosofía de la tecnología

Friedrich Kittler

Editorial
Universidad de Caldas

© Comité Editorial
Editorial Universidad de Caldas, 2017

Título:
No hay *Software* y otros ensayos sobre filosofía de la tecnología. Friedrich Kittler
Autor: Friedrich Kittler

Primera Edición
500 ejemplares
Junio de 2017

ISBN: 978-958-759-160-6

Todos los derechos reservados por la Editorial Universidad de Caldas

Coordinación de la colección:	Liliana Villescas Guzmán Felipe César Londoño López
Editores académicos	Alejandro Duque Andrés Burbano
Traducción:	Mauricio González Rozo
Gestión Editorial:	Editorial Universidad de Caldas
Revisión:	Luis Miguel Gallego Sepúlveda
Diseño y diagramación:	Luis Osorio Tejada / Editorial Universidad de Caldas

Todos los derechos reservados. Prohibida la reproducción total o parcial, por cualquier medio, sin permiso expreso de la Editorial Universidad de Caldas

Kittler, Friedrich

No hay Software y otros ensayos sobre filosofía de la tecnología / Friedrich Kittler, traductor Mauricio González Roso. Manizales: Editorial Universidad de Caldas, 2017.
138 páginas, fotos; 21x27 cm. – (Colección Diseño Visual)

Notas: Incluye Índice, bibliografía, índice temático y onomástico, biografía del autor.

ISBN 978-958-759-160-6

I.Filosofía de la ciencia 2. Filosofía de la tecnología – Ensayos 3. Tecnología y civilización I. Duque, Alejandro II. Burbano, Andrés III. González Rozo, Mauricio

CDD 601/K62

CEP –Banco de la República-Biblioteca Luis Ángel Arango

No hay *Software*

«El mundo de Oriente está explotando», *The Eastern World, it is explodin'*, cantaba Barry McGuire. La primera vez lo hizo en los salvajes años sesenta, buscando disuadir a sus amigos (entonces, por vía del vinilo o de la cinta magnética de 8 pistas) de su vana creencia en que *no* estamos ante la «víspera de la destrucción». La segunda vez, luego de que un brillante *remake* electrónico lo llevara a lo más alto de la tabla de *hits* de la Emisora de las Fuerzas Armadas Norteamericanas o Armed Forces Network en Dharan, sus palabras (ahora por vía de ondas ultracortas) buscaron disuadir a los guerreros de la Operación Tormenta del Desierto de la no menos vana creencia de que *sí* estamos —tanto ellos como nosotros— en la víspera de la destrucción...[1]

McGuire (o más bien el procesador de señal digital que logró borrar, sin dejar rastro, su negación inmortalizada por vía fonográfica) tenía todavía razón la segunda vez, pero solo porque las explosiones ya no contaban más. Carecía casi completamente de importancia si lo que volaba por los aires eran torres petroleras o misiles Scud (hijos legítimos del Reich como inmediatos descendientes del V-2, el *Vergeltungswaffe-2*

[1] Agradezco a Wolfgang Hagen (Radio Bremen), quien preparó una comparación textual de las dos versiones de la canción de McGuire, *Eve of Destruction*, para sus oyentes durante una trasmisión en vivo. N. Trad. Sobre *Eve of destruction* –la canción escrita por P. F. Sloan y grabada por Barry Mc.Guire en julio de 1965– ver: https://es.wikipedia.org/wiki/Eve_of_Destruction – o escuchar: https://www.youtube.com/watch?v=qfZVu0alU0I

Nazi). Oriente puede seguir adelante y explotar. Todo lo que importa en el momento presente es lo que suceda en el mundo occidental: primeramente y ante todo, la implosión de las altas tecnologías y como resultado, la implosión de un conjunto de escenas de significantes que de otro modo serían llamadas (dicho con Hegel) «espíritu del mundo» o *Weltgeist*. Sin las tecnologías computacionales no habría desconstrucción, declaró Derrida una vez en Siegen. Escrituras y textos ya no existen más en espacios y tiempos perceptibles, sino más bien en las celdas de los transistores computacionales. Y puesto que desde hace al menos tres décadas, las acciones heroicas de Silicon Valley han logrado reducir las dimensiones de las celdas de los transistores al rango submicrónico (esto es, a menos de un micrómetro), nuestra actual escena de la escritura[2] solo puede ser descrita por medio de la geometría fractal: como la auto-semejanza de las letras que a lo largo de unas seis décadas se extiende desde vallas publicitarias del tamaño de una casa hasta un mapa de bits del tamaño de un transistor. Si en los comienzos alfabéticos de la historia misma, solo unas dos o tres décadas separaban al camello de los caracteres hebreos que designaban el animal, ahora que todos los signos han sido miniaturizados a escala molecular, el acto de la escritura se ha desvanecido.

Como todos sabemos, aún si nadie quiere decirlo, nadie escribe ya más. La escritura -ese tipo particular de *software*- se desgastó hace mucho en medio de la confusión incurable entre uso y mención. Aún en los tiempos en los que Hölderlin compuso sus himnos, la mera mención del rayo era evidencia suficiente de que este podría ser usado para la poesía[3]. En contraste hoy en día, luego de la transformación de este mismo rayo en electricidad, la escritura humana tiene lugar a través de inscripciones que no solo se consumen en silicio por medio de rayos de electrones litográficos, sino que más aun -en contraste con todos los dispositivos de escritura a lo largo de la historia- son ellos mismo capaces de leer y escribir.

Entonces, el acto final de la escritura en la historia, puede haber ocurrido ya a finales de los años setenta, cuando un equipo de ingenieros de Intel, bajo la dirección del Dr. Marcian E. Hoff, dispersó algunas docenas de metros cuadrados[4] de papel de dibujo por el piso de un garaje vacío en Santa Clara y bosquejó sobre él la arquitectura de *hardware* para el primer microprocesador integrado. En un segundo paso -esta vez mecánico-, la disposición manual de dos mil transistores y sus conexiones fue reducido al tamaño de una uña en un chip de verdad. En un tercero, un equipo electro-óptico escribió el diseño al imprimirlo en silicio. Cuarto, luego de que el producto acabado (el 4004, que ha provisto el prototipo para todos los microprocesadores desde entonces)[5] fuera empleado en la nueva máquina de cómputo del cliente japonés de Intel, nuestra escena de escritura postmoderna pudo comenzar.

Entretanto, dada la complejidad del *hardware* en los microprocesadores de la actualidad, las técnicas manuales de diseño no tienen chance ya. Para desarrollar la siguiente generación de computadores, los ingenieros no usan ya dibujos sino más bien *Computer-Aided Design* (literalmente: diseño asistido por computador): las capacidades geométricas de la generación de calculadoras más reciente son suficientes para mapear la topología de sus sucesores. De este modo, «los pies de aquellos que te cargarán están» una vez más «ante la puerta»[6].

Y sin embargo, Marcian E. Hoff pudo ofrecer con sus primitivos blueprints un ejemplo casi perfecto de una máquina de Turing. A partir de la disertación de Turing de 1937, todo acto de cómputo -sea realizado por seres humanos o por máquinas- puede ser formalizado como un conjunto finito de comandos que operan a través de una banda infinitamente larga de papel y sus signos discretos. El concepto de Turing de una tal máquina de papel[7], cuyas operaciones solo abarcan escritura y lectura, avanzar y retroceder, ha podido demostrar el equivalente matemático para todas las funciones calculables y proveer para ello un sentido literalmente maquinal que pueda desplazar la inocente designación profesional de 'ordenador' o 'computador'[8]: Las máquinas universales de Turing solo requieren ser surtidas con la descripción (el programa) de cualquier otra máquina para poder imitar a esta otra en sus efectos. Y puesto que ya desde Turing es posible hacer abstracción de las diferencias en cuanto a *hardware* entre dos dispositivos, la así llamada hipótesis de Church-Turing[9] apunta a declarar en su forma más estricta (la física) a la naturaleza misma como máquina universal de Turing.

Como tal, esta afirmación ha tenido el efecto de duplicar la implosión del *software* mediante la implosión del *hardware*. Desde que se han podido implementar los computadores, a partir de 1943 con tubos al vacío y desde 1949 con transistores, ha surgido el problema acerca de cómo describir y cómo leer de alguna manera estas máquinas universales de lectoescritura, ellas mismas ilegibles. Como bien se sabe, la solución para ello se denomina '*software*', esto es, el desarrollo de lenguajes de programación de alto nivel (o más estandarizados). El antiguo monopolio de las lenguas o lenguajes cotidianos para ser ellos mismos su propio metalenguaje y de este modo no admitir a ningún otro como su otro, ha colapsado para dar lugar a una nueva jerarquía de los lenguajes de programación. Entretanto, esta Torre de Babel[10] postmoderna se extiende ahora desde

simples códigos de comando, cuya extensión lingüística es aún una configuración de *hardware*, pasando por *assembler*, cuya extensión es exactamente la misma de aquellos códigos de comando, hasta los llamados lenguajes estándar (*Hochsprachen*), cuya extensión pasa por todos los desvíos posibles a través de intérpretes, compiladores y *linkers*, y se llama a su vez *assembler*. Escribir hoy, como ocurre en el desarrollo del *software*, implica una serie infinita de auto-similitudes, según lo ha dado a pensar la geometría fractal. Solo que, en contraste con el modelo matemático, sigue siendo imposible alcanzar o tener acceso a todas estas capas en sentido físico-fisiológico. Ya a partir del cine y el gramófono, las tecnologías modernas de los medios se encuentran fundamentalmente dispuestas para burlar y evitar a la percepción sensorial. Ya no podemos por principio saber lo que está haciendo nuestra escritura, y mucho menos en el caso de la programación digital.

Para ilustrar esta situación, basta con evocar casos cotidianos como el del procesador de palabras con el cual redacto estas palabras. Que el *genius loci* de Palo Alto, que ha producido el primer y más elegante entre los sistemas operacionales, disculpe a un súbdito de la Corporación Microsoft por limitar su ejemplo al más tonto entre todos los sistemas operacionales.

Para poder procesar textos, esto es, para llegar a ser uno mismo una máquina de papel en un IBM AT que corre bajo Microsoft DOS, la compra de un paquete de *software* comercial es el primer ítem del negocio. En segundo lugar, algunos de los archivos en este paquete deben tener la extensión .EXE o .COM, de otro modo un procesador de palabras no puede correrse bajo DOS. Los archivos ejecutables y solo ellos mantienen una relación extraña y particular con sus nombres propios. Por un lado, ellos exhiben grandilocuentes nombres auto-referenciales tales como *WordPerfect*; por otro, con la omisión de vocales ellos conforman acrónimos más o menos crípticos como WP. Por todo ello, el nombre completo sirve solo para las estrategias publicitarias que emprenden en lenguas cotidianas las casas productoras de *software*; y ello es así puesto que el *Disk Operating System* alias DOS no puede leer nombres de archivos de más de ocho letras o caracteres. De este modo, acrónimos o abreviaciones impronunciables que se han liberado de las vocales, y que son la revocación de una innovación elemental introducida en la antigua Grecia, resultan ser no solo necesarias para la escritura postmoderna sino también perfectamente suficiente para ejecutar su tarea. Más aún, por primera vez desde su invención, ellas parecen dotar al alfabeto de fuerzas mágicas de nuevo. La abreviación WP realiza exactamente lo que dice. A diferencia no solo de la palabra *WordPerfect* sino también de solemnes palabras ya desgastadas o vaciadas de la vieja Europa (tales como «espíritu» o incluso «palabra»), los archivos de computador ejecutables abarcan todas las rutinas y datos que son necesarios para que ellos alcancen su realización. El acto de escritura consistente en teclear las letras 'W', 'P' y 'Enter' en el teclado no hace perfecta a la Palabra, pero sí hace correr al *WordPerfect*. Tales son los triunfos el *software* permite alcanzar.

---

2 N.Trad. Este es un guiño a un título de Jacques Derrida: «Freud y la escena de la escritura», en: *La escritura y la diferencia* (1967).

3 Ver: Thomas Hafki, *Franklin — Frankenstein. Zum Verhältnis von Elektrizität und Literatur* [*Franklin — Frankenstein. Sobre la relación entre electricidad y literatura*]. Tesis de Maestría, Bochum, 1993.

4 En 1978, cuando se estaba diseñando el procesador 8086 de Intel, se decía que los blueprints ocupaban un espacio de 64 metros cuadrados de papel gráfico. Ver: Klaus Schrödl, «Quantensprung», *DOS* 12 (1990), p. 102 s.

5 N.Trad. *...desde entonces* –se sobreentiende– *hasta la actualidad*, es decir, el año en que Friedrich Kittler escribió su texto (1992), muy distante ya de 'nuestra actualidad' (2017) en un mundo donde la tecnología cambia aceleradamente.

6 N.Trad. La referencia bíblica es a Hechos de los Apóstoles 5, 9-10. Ella va mediada, sin embargo, por el guiño implícito a Hegel, quien remite a este pasaje como analogía de la dinámica histórica por la cual la nueva filosofía supera y desplaza a la anterior: «...a estas filosofías les son aplicables otras palabras del Evangelio, las que el apóstol Pedro dice a Safira, mujer de Ananías: "Los pies de quienes han de sacarte de aquí están ya a la puerta." La filosofía que ha de refutar y desplazar a la tuya no tardará en presentarse, lo mismo que les ha ocurrido a las otras.» (F.W. G. Hegel. *Lecciones sobre Historia de la Filosofía I*. FCE, México, 1995, p. 23).

7 Ver: Alan M. Turing, «On Computable Numbers with an Application to the *Entscheidungsproblem* [Sobre los números computables, con una aplicación al problema de la decisión]», en: *Proceedings of the London Mathematical Society 42* (1937), pp. 230 – 65.

8 N.Trad. Acceso abierto en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.247.3451&rep=rep1&type=pdf

9 Cuán profundo es este desplazamiento pudo demostrarlo el editor o tipógrafo del *Programmer's Reference Manuals* de Intel: la instrucción de comando *[2xm1* –esto es, de una magnitud de input al cuadrado menos uno–, por ejemplo, al ser traducida al inglés cotidiano, no se lo hace como «Compute 2^x-1» sino «Computer 2^x-1». Ver: Intel, *387 DX User's Manual. Programmer's Reference* (Santa Clara, 1989), pp. 4 – 9; ver así mismo: *Intel, i486 Microprocessor. Programmer's Reference Manual* (Santa Clara, 1990), pp. 26 – 72. N.Trad. Ver Tesis de Church-Turing en Wikipedia: https://es.wikipedia.org/wiki/Tesis_de_Church-Turing

10 Ver: Wolfgang Hagen, *La escritura perdida. Esbozo para una teoría del computador*, en: *Arsenale der Seele. Literatur- und Medienanalyse seit 1870*. Ed. Por Friedrich Kittler y Georg Christoph Tholen (Munich: Fink 1989), p. 221: «Así, en la estructura lingüística de la máquina lógica de von Neumann se establece ya la divergencia de principio entre software y manuales de software, y así se comienza a erigir a partir de 1945 una babilónica torre de programación de la performancia computacional, cuyo empleo no tiene ya en absoluto que ver con el arreglo y organización significativa de un lenguaje maquinal. Se trata de una torre de software con errores indocumentados, dialectos insalvablemente mezclados y confundidos y amontonamiento de actos de lenguaje que ya nadie más puede realizar.» Con una imagen menos precisa pero por ello mismo más desesperada, un experto de UNIX lo formula así: «Casi todos los sistemas operacionales están marcados, después de cierta edad, por un alto 'grado de contaminación'. Ellos crecen en todas las direcciones y dan la impresión de ser ruinas que solo con mucho esfuerzo pueden mantenerse juntas en pie.» Horst Drees, *UNIX. Ein umfassendes Kompendium für Anwender und Systemspezialisten*. Haar: Markt & Technik, 1988, p. 19). El experto de Unix es muy demasiado cortés como para comprometer con nombre propio las excrecencias de una firma como Microsoft Corporation.

La documentación en papel (*Paperware*) más o menos inflacionaria que acompaña al *software*, para no dejar atrás a la línea de comando[11], redobla estos poderes mágicos. Los manuales de *software* de uso corriente, dado que tienen que franquear el abismo entre lenguajes formales y cotidianos, entre electrónica y literatura, presentan el paquete de programación como un agente lingüístico con el poder absoluto de dar órdenes sobre recursos de sistema, rangos de direcciones (address spaces) y parámetros de *hardware* del computador en cuestión; el WP, al ser activado por la línea de comando con el argumento X, cambia la pantalla de modo A en el modo B, se inicia con el ajuste C, finalmente retorna a D, etc[12].

Sin embargo, todas las acciones que ejecuta el WP, de acuerdo con lo que yace escrito en el papel, son enteramente virtuales, puesto que cada una de ella tiene que correr 'bajo' DOS, como se lo dice de manera tan acertada. Pero en términos fácticos, lo único que trabaja es el sistema operativo, o más precisamente su shell: COMMAND.COM busca en el búfer del teclado un nombre de archivo de 8 bytes, traduce las direcciones relativas del archivo que éste (quizás) encuentra en direcciones absolutas, carga esta versión modificada del *buffer* de memoria externa en una memoria RAM de silicio, y finalmente asigna la ejecución del programa (lo que ocurre por un tiempo limitado) a las primeras líneas de código de un esclavo llamado *WordPerfect*.

Pero el mismo argumento de línea de comando puede ser empleado también en contra de DOS, puesto que en último análisis el sistema operacional trabaja como una simple expansión de un sistema básico de *input/output* llamado BIOS. Ninguno de los programas de aplicaciones podría iniciarse, ni siquiera el microprocesador base podría hacerlo, si algunas de las funciones elementales -que por razones de seguridad han sido fundidas en silicio y por lo tanto forman parte del *hardware* indeleble- no poseyeran, por así decir, la habilidad del Barón de Münchhausen para sacarse del barro jalándose de su propia trenza[13]. Cada transformación material de entropía en información, de un millón de células de transistores adormecidas en diferencias de voltaje eléctrico, necesariamente presupone un evento material llamado «Reset»[14].

En principio, este descenso del *software* al *hardware*, de niveles superiores a niveles inferiores de observación, puede perdurar a lo largo de décadas. Incluso códigos operacionales elementales, a pesar de sus promesas metafóricas (p.e. *call* o *return*), se reducen a manipulaciones de signos absolutamente locales y de este modo (para decirlo con Lacan) a significantes con potenciales eléctricos. Toda formalización, según lo define David Gilbert, tiene el efecto de desmontar a la teoría, simplemente porque «la teoría ya no es más un sistema de enunciados significativos, sino un sistema de *pro*posiciones como secuencia de palabras que son a su turno secuencia de letras. Por lo tanto, solo con referencia a la forma decimos qué combinaciones de palabras son proposiciones, qué proposiciones son axiomas y qué proposiciones se siguen como consecuencia inmediata de otras»[15].

Cuando las significaciones se solidifican en proposiciones, y las proposiciones en palabras, y las palabras en letras, ya no hay más *software*. Más aún: no lo habría si los sistemas computacionales no tuvieran que coexistir como hasta ahora en un ámbito constituido por lenguajes cotidianos. Sin embargo, a partir de que una doble invención griega[16] fuera introducida, este entorno tiene una consistencia doble: éste consiste de letras y monedas, *letters* and *litters*. Estas buenas razones económicas han podido eliminar entretanto la humildad de Alan Turing, quien en la edad de piedra de la era computacional prefirió leer el *output* de máquinas que se expresan en binario que en decimal[17]. Por el contrario, la llamada filosofía de la también llamada comunidad computacional pone todas sus apuestas en eclipsar u ocultar el *hardware* detrás del *software*, los significantes electrónicos detrás de las interfaces humano-máquina. Con toda la debida filantropía, los manuales de uso[18] para lenguajes de programación de alto nivel advierten sobre el colapso mental que acarrearía el escribir funciones trigonométricas en lenguaje *assembler* o ensamblador[19]. Con toda condesciende cortesía, los manuales de BIOS (*Basic Input/Output System*), así como sus autores (técnicos altamente especializados) se hacen a la tarea de «ocultar los particulares que controlan el *software* de base de tu programa»[20]. Pensándolo un poco más allá: sin que esto en últimas se distinga tanto de las jerarquías de los ángeles en el medioevo, las funciones de sistema operacional tales como COMMAND.COM seguirían el encargo de esconder el BIOS, programas de aplicación tales como WordPerfect ocultan el sistema operacional, así sucesivamente hasta que al final de todos los tiempos dos adaptaciones fundamentales en el diseño computacional (o en el concepto de ciencia del Pentágono) hayan conducido todo este sistema secreto a su clausura exitosa.

En primer lugar, a un nivel intencionadamente superficial, fueron desarrolladas interfaces gráficas de uso, las cuales (dado que ocultan las operaciones o actos-de-escritura que son indispensables para la programación) hacen la máquina en su totalidad le sea inaccesible a los usuarios. Ni siquiera el compendio de gráficas computacionales autorizado de IBM escondería el hecho de que sus interfaces de uso no hacen más

rápida y más eficiente la programación del sistema de lo que lo harían las meras líneas de comando[21]. En segundo lugar, en conexión inmediata con el ADA[22], el lenguaje de programación del Pentágono, se desarrolló a nivel microscópico, a partir del *hardware* mismo, un nuevo modo de operativo denominado *Protected Mode*. De acuerdo con el *Microprocessor Programming Manual* de Intel, éste tiene como único propósito el prevenir que *untrusted programs* y *untrusted users* puedan tener acceso a recursos del sistema tales como canales de *input/output* o al núcleo del sistema operacional[23]. En sentido técnico, sin embargo, ningún usuario es digno de confianza; en *Protected Mode* (tal y como éste domina, digamos, en UNIX) ya no se le permite a ninguno de sus usuarios tener control sobre sus máquinas.

Esta imparable marcha triunfal del *software* es una extraña inversión de la demostración de Turing según la cual no puede haber problemas que sean calculables (en sentido matemático) que una mera máquina no pueda resolver. Precisamente en el lugar de esta máquina, la hipótesis física de Church-Turing, precisamente al equiparar el *hardware* físico con los algoritmos para calcularlo, creó una laguna que puede venir a ocupar exitosamente el *software*, y de cuya oscuridad pudo beneficiarse.

Después de todo, los lenguajes de programación de alto nivel operan de modo muy similar a como lo hacen las así llamadas funciones unidireccionales[24] de la más reciente criptografía matemática[25]. En su forma estándar, tales funciones se dejan calcular con una inversión justificable de tiempo; por ejemplo, cuando tiempo operacional (*Maschinenzeit*) se incrementa solo en expresiones polinómicas de complejidad funcional. Por otro lado, sin embargo, el costo temporal para la operación inversa, esto es, para calcular de vuelta parámetros de entrada a partir de los resultados de la función, se incrementaría exponencialmente, y por ende insostenible, respecto de la complejidad de la función. En otras palabras, las funciones unidireccionales protegen a los algoritmos de a sus propios resultados.

Esta propiedad criptográfica le viene como anillo al dedo al *software*. Ella le ofrece una manera cómoda para evadir aquello que la demostración de Turing muestra: que el concepto de propiedad intelectual ha devenido imposible, sobre todo en lo que respecta a los algoritmos. Ahora, el hecho mismo de que el *software* no tenga existencia como capacidad independiente de las máquinas ha logrado solo incrementar su insistencia en cuanto medio comercial (o norteamericano). Todas las licencias, *dongles* y patentes que han sido registrados para WP o WordPerfect demuestran la funcionalidad misma de funciones unidireccionales. En desdén de todo honor matemático, tribunales norteamericanos han llegado a ratificar las pretensiones de *copyright* sobre algoritmos.

Así, no resulta sorprendente el hecho de que la más recientemente y al más alto nivel, al de IBM para decirlo con nombre propio, se haya abierto la caza para fórmulas matemáticas que puedan determinar la diferencia en complejidad (la ecuación de Kolmogorov) entre un algoritmo y su *output*. En los buenos tiempos pasados de la teoría de señales de Shannon, el máximo de información coincidía hasta cierto grado con el máximo de

→

11 N.Eds. La interfaz de línea de comandos o interfaz de línea de órdenes (en inglés, command-line interface, CLI) es un método que permite a los usuarios dar instrucciones a algún programa informático por medio de una línea de texto simple.

12 No por accidente, el único contraejemplo proviene de Richard Stallman, cuya *Free Software Foundation* le ha declarado la guerra frontal al copyright del software en general, una batalla tan justa y heroica como desesperada y destinada al fracaso. El contraejemplo dice: «When we say that '*C-n* moves down vertically one line' we are glossing over a distinction that is irrelevant in ordinary use but is vital in understanding how to customize Emacs. It is the function *next-line* that is programmed to move down vertically. *C-n* has this effect because it is bound to that function. If you rebind *C-n* to the function *forward-word* then *C-n* will move forward by words instead.» (Richard Stallman, *GNU Emacs Manual* Cambridge/Mass., 1988, p. 19.)

13 Este gesto puede entenderse como una traducción libre para el inglés *booting* (Nota de F. Kittler).

14 N.Trad. Según el *Oxford English Dictionary*, en la electrónica se entiende por Reset el evento por el cual un dispositivo binario ingresa en un estado representado por el numeral 0.

15 Stephen C. Kleene, citado en: Robert Rosen, «Effective Processes and Natural Law», *The Universal Turing Machine. A Half-Century Survey*, ed. Rolf Herken, Hamburg, Berlin, Oxford 1988, p. 527.

16 Ver: Johannes Lohmann, «Die Geburt der Tragödie aus dem Geiste der Musik [El nacimiento de la tragedia a partir del espíritu de la música]», en: *Archiv für Musikwissenschaft* (1980), p. 174.

17 Ver: Andrew Hodges, *Alan Turing: the Enigma*. New York, Simon & Schuster, 1983, p. 399.

18 Ver: *TOOL Praxis: Assembler-Programmierung auf dem PC, Ausgabe 1*. Würzburg, 1989, p. 9.

19 N.Trad. Ver Lenguaje Ensamblador en Wikipedia: https://es.wikipedia.org/wiki/Lenguaje_ensamblador

20 Nahajyoti Barkaliti, *The Waite Group's Macroassembler Bible*. Carmel, Ind. Howard H. Sams, 1989, p. 528.

21 Ver: James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes, *Computer Graphics. Principles and Practice*. Reading/Mass., 1990, p. 398.

22 Sobre la conexión entre Pentagon, Ada y el iAPX 432 de Intel, el primer microprocesador en Protected Mode, el cual dio origen, al fracaso económico, de la industria estándar, desde el modelo 80286 hasta el 80486, ver: Glenford J. Myers, «Overview of the Intel iAPX 432 Microprocessor», en: *Advances in Computer Architecture*. New York: John Wiley & Sons, 1982, pp. 335 – 44 (agradezco la referencia a Ingo Ruhmann, de Bonn). Quien desee entender el fracaso, debe meditar acerca de la siguiente afirmación: «The 432 can be characterized as a three-address-storage-to-storage architecture, there are no registers visible to programs» (p. 342).

23 Ver: F. Kittler, «Protected Mode», en: *Computer, Macht und Gegenwehr [Computador, Poder y Defensa]*. Eds. Ute Bernhardt e Ingo Ruhmann, Bonn 1991, pp. 34-44.

24 N.Trad. Ver: https://en.wikipedia.org/wiki/One-way_function

25 Sobre lo que sigue, ver: Patrick Horster, *Kryptologie*. Mannheim, Wien, Zürich, Institut Wissenschaftsverlag, 1985, pp. 23-27.

ruido[26]. En contraste, la nueva magnitud de profundidad lógica adoptada por IBM se define así:

> El valor de un mensaje (...) no parece residir tanto en su información (sus partes absolutamente impredecibles) ni en su obvia redundancia (repeticiones literales, desiguales frecuencias de bits), sino más bien en lo que podría llamarse su redundancia enterrada - en partes que solo son predecibles con dificultad, cosas que el receptor podría en principio haber averiguado sin que se le haya dicho, pero solo bajo una inversión considerable de dinero, tiempo, o rendimiento de cómputo. En otras palabras, el valor del mensaje es la cantidad de trabajo matemático u otro que su emisor plausiblemente ha dedicado, y que su receptor no tiene entonces que efectuar una vez más[27].

Las magnitudes de profundidad lógica de IBM, en su rigor matemático, podrían reemplazar también a los anticuados términos cotidianos, necesariamente imprecisos, de *originalidad*, *autoría* y *copyright*, y con ello hacer que estos también recobren fuerza de ley. Ahora bien, precisamente el algoritmo para calcular la originalidad de los algoritmos en general resulta ser incalculable aún según los métodos de Turing[28].

En esta situación trágica, el derecho penal, al menos en Alemania, ha decidido abandonar el concepto de propiedad intelectual sobre algo no menos inmaterial que éste como lo es el *software*, y en lugar de ello ha definido el *software* como una *cosa* (*Sache*). La declaratoria del Tribunal Supremo Federal (*Bundesgerichtshof*) según la cual ningún programa computacional podría correr sin la respectiva carga eléctrica en circuitos de silicio[29], corrobora una vez más que el carácter virtualmente indistinguible entre *software* y *hardware* difícilmente se basa, como los teóricos de sistemas gustosamente están dando a creer, en un cambio de perspectiva del observador[30]. Antes bien, hay buenas razones que hablan a favor de su dependencia frente al *hardware*, y por ende de la antecedencia (aprioricidad) del *hardware* frente a aquél.

Una máquina con recursos ilimitados de tiempo y espacio, con reservas infinitas de papel y velocidad de cómputo ilimitada, ha existido tan solo una vez: en el ensayo de Turing ya citado, «Sobre los números computables con una aplicación al *problema de la decisión*»[31]. En contraste, todas las máquinas físicamente construibles están limitadas por parámetros estrictos dentro de su propio código. La incapacidad de Microsoft DOS para reconocer nombres de archivo cuya extensión sea superior a los ocho caracteres (p.e. *WordPerfect*) no solo ilustra a su manera un problema obsoleto y trivial que ha conllevado a incompatibilidades crecientes entre diferentes generaciones de microprocesadores, de los de 8 bits a los de 16 y 32 bits. Ella también apunta a la imposibilidad de principio para la digitalización, esto es, para computar el cuerpo de los números reales, es decir, aquello que solíamos llamar 'naturaleza'[32].

Pero, dicho en palabras del Laboratorio Nacional de Los Alamos[33], esto significa que:

> Nosotros usamos computadores digitales cuya arquitectura nos es dada en la forma de una pieza física de maquinaria, con todas sus restricciones artificiales. Nosotros debemos reducir una continua descripción algorítmica a una codificable sobre un aparato cuyas operaciones fundamentales sean contables, y esto lo hacemos por vía de diversas formas de talado en piezas a lo cual generalmente se llama discretización. Usando diferencias finitas, elementos, o algún esquema similar, se construye un algoritmo con un conteo N de operación y se lo traduce a algún lenguaje de más alto nivel. Entonces, el compilador (*compiler* o convertidor de datos) prosigue a reducir este modelo a una forma binaria ampliamente determinada por restricciones a las que la máquina constriñe. El resultado es una imagen microcósmica, discreta y sintética, del problema original, cuya estructura es arbitrariamente fijada por un esquema diferencial y una arquitectura computacional escogida al azar (*at random*). El único residuo del continuum es el uso de aritmética base (*radix*), la cual tiene la propiedad de ponderar desigualmente los bits, siendo así para sistemas no lineales fuente de singularidades espurias.
>
> Esto es lo que nosotros de hecho hacemos cuando computamos un modelo del mundo físico con aparatos físicos. Este no es el proceso idealizado y sereno que imaginamos cuando tratamos por lo general acerca de las estructuras fundamentales de la computación, y muy lejos de las máquinas de Turing[34].

No se trata entonces ya de seguir la hipótesis Church-Turing para así «inyectar un carácter algorítmico al comportamiento del mundo físico para lo cual, sin embargo, no hay evidencia»[35]. Si el mundo no surge de un juego de dados de Dios, el comportamiento algorítmico de las nubes o de las olas marinas no excluye, sino que más bien incluye, el hecho de que sus moléculas operan como computadores de su propia actividad. Por el contrario, sería un asunto de calcular el «precio de la calculabilidad» misma. Esta decisiva capacidad de los computadores claramente no tiene nada que ver con el *software*; ella depende solo del grado en el que un ítem de *hardware* dado pueda alojar algo así como un sistema de escritura.

Cuando en 1937, Claude Shannon, en «la más exitosa tesis de maestría que haya sido nunca antes escrita»[36], ofreció una demostración de que los más simples relevadores o relés de telégrafo podían implementar toda el álgebra de Boole, un sistema tal de escritura/grabación (*Aufscreibsystem*) fue establecido. Y cuando el circuito integrado, derivado del transistor de William Shockley a comienzos de los años setenta, pudo combinar en uno y el mismo chip la resistencia controlable de silicio con su propio óxido como un 'insulador' o aislante casi perfecto, la programabilidad de la materia pudo «tomar el control»[37], según lo había pronosticado Turing. Y de este modo, el *software* -si es que alguna vez él ha existido- no sería otra cosa que un negocio de billones de dólares revoloteando alrededor de uno de los elementos más baratos sobre la tierra. Pues en su conexión con el chip, silicio y óxido de silicio procuran un *hardware* casi perfecto. Por una parte, millones de elementos de circuito integrado trabajan bajo las mismas condiciones físicas, lo cual es decisivo, sobre todo, para el parámetro crítico de la temperatura del chip y previene el incremento exponencial de desviaciones en la resistencia de transistor. Por otra parte, estos millones de elementos de circuito integrado permanecen eléctricamente aislados unos de otros. Solo esta relación paradójica entre dos parámetros físicos, continuidad térmica y discreción eléctrica, hace posible que circuitos digitales integrado no sean solo autómatas finitos, como tantas otras cosas sobre la tierra, sino que se aproximen a la Máquina Universal Discreta que desde hace tiempo se ha denominado bajo el nombre de «Turing», su inventor.

Esta diferencia estructural puede ilustrarse fácilmente. Por ejemplo,

> un candado de combinación es un autómata finito, pero no puede descomponerse en un conjunto de base de componentes elementales que puedan ser reconfigurados para simular un sistema físico cualquiera. Como consecuencia, no es estructuralmente programable y en este caso resulta ser solo programable en el sentido limitado a que su estado pueda ser dispuesto para alcanzar una clase limitada de comportamientos. En contraste, un computador digital usado para simular un candado de combinación es estructuralmente programable puesto que el comportamiento es alcanzado mediante síntesis a partir de un conjunto canónico de componentes elementales de circuitos[38].

Pero los componentes de circuitos -ya se trate de relés de telégrafo, tubos de electrones o, finalmente, transistores de silicio- pagan un alto precio por su separabilidad desarticulable o discretización (*Zerlegbarkeit oder Diskretisierung*). Aparte del caso trivial (por ser un caso discreto) del procesador de palabras, el cual queda a la zaga si se apunta a otros ámbitos científicos, militares e industriales altamente computarizados, las calculadoras digitales siguen confrontando como único «órgano de sí-o-no en sentido estricto de la palabra»[39] un entorno continuo de nubes, olas y guerras. Esta avalancha de números gigantes y reales, como diría Ian Hacking, puede solo ser dominada mediante la anexión aditiva de cada vez más elementos de circuitos -así hasta que los 2000 transistores

---

26 Ver: Friedrich Kittler: «Relación Señal-Ruido [Signal-Rausch-Abstand]», en: *Materialität der Kommunikation*, eds. Hans Ulrich Gumbrecht y K. Ludwig Pfeiffer. Frankfurt/M., 1988, pp. 342-359.

27 Charles H. Bennett, «Logical Depth and Physical Complexity», en: *The Universal Turing Machine*. Ed. Rolf Herken (ver nota # 15), p. 230.

28 Agradezco la observación a Oswald Wiener, Dawson City.

29 Ver: Michael König, «[Mirar a las cosas en concreto. Problemas en el abandono del software.] Sachlich sehen. Probleme bei der Überlassung von Software.», en: *c't*3, 1990, p. 73.

30 Se podría más bien entonces, como me lo ha indicado una carta personal de Dirk Baecker, «...suponer que la distinción entre *hardware* y *software* indica una diferencia a la cual se le ha encomendado el relanzar la diferencia entre programabilidad y no-programabilidad en el ámbito mismo de lo programable. Ella está a la vez comprometida con la calculabilidad de la tecnología y en éste sentido a favor de la técnica misma. Y solo puede estarlo, en la medida en que la 'unidad' del programa puede solo realizarse si igualación y cómputo se encuentran respectivamente repartidos en dos lados, de tal modo que siempre solo uno de ellos se encuentre operativamente a disposición mientras que el otro lado pueda ser retenido como constante.» (Carta del 15 de abril de 1991).

31 N.Trad. «On Computable Numbers with an Application to the *Entscheidungsproblem* » (ver nota # 7).

32 Por ende, no termino de comprender cómo Turing pudo entonces comenzar su famoso ensayo con la declaración según la cual «los 'números computables pueden describirse brevemente como los números reales cuyas expresiones como decimal son calculables por medios finitos...» (Turing, «On Computable Numbers ...», p. 230, ver nota # 7), para luego definir el conjunto de números calculables como contables, y finalmente llamar a ϖ «el límite de una secuencia convergente de modo computable», un número computable (256).

33 N.Trad. Ver: https://es.wikipedia.org/wiki/Laboratorio_Nacional_de_Los_Álamos

34 Brosl Hasslacher, «Beyond the Turing Machine», en: *Universal Turing Machine* [ver nota # 15], p. 391.

35 Ibid, p. 389.

36 Friedrich-Wilhelm Hagemeyer, *El surgimiento de los conceptos de la informática en la tecnología noticiosa. Un caso de estudio para la formación de la teoría tecnológica en la investigación sobre la industria y la guerra.* [Die Entstehung von Informationskonzepten in der Nachrichtentechnik. Eine Fallstudie zur Theoriebildung in der Technik in Industrie- und Kriegsforschung.] Disertación Doctoral (mecanografiada). Berlín, 1979, p. 432.

37 Alan Turing, «Intelligent Machinery: A Heretical Theory», *The Essential Turing: Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life*, ed. B. Jack Copeland. Oxford University Press, 2004, p. 475.

38 Michael Conrad «The Price of Programmability», en: *Universal Turing Machine* [ver nota # 14], pp. 264-265.

39 John von Neumann, «General and Logical Theory of Automata», en: *Collected Works*. Oxford: Pergamon, 1963, V, p. 296 ss.

que componen el Intel 4004, hayan llegado a ser 1.2 millones para el *flagship* Intel 80486. Sin embargo, se puede demostrar matemáticamente que la rata creciente de conexiones posibles entre estos elementos, y con ello el rendimiento computacional como tal, tiene como su límite superior una función de raíz cuadrada. En otras palabras, el sistema «no puede seguir el ritmo a tasas de crecimiento polinómico en cuanto al alcance de los problemas»[40], por no decir tasas exponenciales. Este mismo aislamiento entre elementos digitales o discretos, el cual garantiza su funcionamiento al menos bajo condiciones no tropicales ni árticas, también limita la dimensión de posibles conexiones al entorno local de un chip dado. Por el contrario, bajo condiciones de interacciones o efectos recíprocos globales, como la que experimentan los chips digitales solo en su condición térmica, la conectividad «según las leyes de la fuerza en vigor»[41] y siguiendo la lógica combinatoria podría incrementarse hasta un límite superior dispuesto por el valor al cuadrado de todos los elementos involucrados.

Pero es esta conectividad óptima lo que, por el otro lado (el físico), distingue precisamente a los sistemas no-programables. Sobre la base de esta interacción global, tales sistemas, sean olas/ondas o seres, pueden desplegar ratas de crecimiento polinómico en cuanto a su complejidad; más por ello mismo pueden ser solo calculadas por máquinas que no tuvieran que pagar ellas mismas el precio de la programabilidad. Evidentemente, este tipo hipotético (pero agriamente necesario) de máquina presentaría un *hardware* puro: un dispositivo físico que trabajaría en un entorno netamente de aparatos físicos y que estaría sometido solo a las mismas limitaciones de recursos a los cuales estos están sometidos. El *software*, en el sentido convencional de una abstracción siempre realizable, no existiría ya más. Los procedimientos para tal máquina, aún si permanecieran abiertos para una grafía algorítmica, tendrían que operar esencialmente sobre un sustrato material, cuya conectividad permitiera la reconfiguración alternante de sus celdas. Y aún cuando «el sustrato pueda describirse también en términos algorítmicos por medio de simulación», su «caracterización resulta ser de una importancia tan grande para (...) la efectividad y estar conectada de manera tan estrecha con la elección del *hardware*»[42] que su programación tendría poco en común con la de aquellas que tienden a aproximarse a la máquina de Turing.

Tales máquinas urgentemente requeridas, que no están muy lejos ya en tanto que son motivo actual de discusión entre informáticos y la industria del chip se está acercando también a ellas,[43] puede llevar los ojos de algunos observadores de Dubrovnik a la tentación de querer reencontrar el familiar rostro del hombre en ellas, evolutivamente travestido o no. Puede ser. Pero al mismo tiempo nuestro no menos familiar *hardware* de silicio se encuentra ya siguiendo muchas de las exigencias de sistemas no programables de alta conectividad. Entre su millón de celdas de transistor, ya siempre un millón al cuadrado de interacciones tienen lugar: difusión de electrones y efectos de túnel mecánico-cuánticos[44] corren a todo lo largo del chip; solo que la tecnología manufacturera trata tales interacciones hoy en día como limitaciones del sistema, como efectos colaterales físicos, como fuentes de interferencia o ruido, etc. Todo este ruido, que es imposible de evitar, al menos se debe minimizar; tal es el precio que la industria computacional debe pagar por las máquinas estructuralmente programables. La estrategia inversa, la de maximizar el ruido, no solo permitirá reencontrar el camino de vuelta de IBM a Shannon; ella sería también el único camino hacia aquel cuerpo de números reales que anteriormente se llamaba «caos».

«¿No entiendes lo que estoy tratando de decirte?», *Don't you understand what I'm tryin' to say?*, nos recuerda —cantando sin *remake*— Barry McGuire, en su *Eve of Destruction* o «Víspera de la destrucción».

1992

---

40 M. Conrad «The Price of Programmability», p. 268.

41 Ibid, p. 265.

42 Ibid, p. 279.

43 Así, la primera red neuronal integrada –salida del discreto imperio del chip de Intel y, h donde veo, por segunda vez en toda la historia de la compañía, en el altamente hib procesador de señal i2920– se remonta a amplificadores operacionales netamente anál [N.Trad.Ver: https://es.wikipedia.org/wiki/Amplificador_operacional]

44 N.Trad.Ver Efecto túnel en Wikipedia: https://es.wikipedia.org/wiki/Efecto_túnel