Es gibt keine Software

The Eastern World is exploding, sang Barry McGuire. Beim erstenmal, in den wilden sechziger Jahren, um all seinen Freunden über Vinyl oder Autokassettenrekorder den Glauben auszureden, daß wir nicht am Vorabend der Zerstörung leben. Beim zweitenmal, nach einem brillanten elektronischen Remake, das seinen alten Vinylsong zum digitalen Spitzenreiter von AFN Dharan beförderte, um westlichen Desert Storm-Kriegern über UKW den Glauben auszureden, daß sie oder wir am Vorabend der Zerstörung leben[1] ...

McGuire (oder vielmehr der digitale Signalprozessor, der eine phonographisch verewigte Negation wieder spurlos löschen konnte) hat recht behalten, aber nur, weil Explosionen gar nicht zählen. Ob Ölbohrtürme oder Scud-Raketen, diese reichsunmittelbaren Enkel der V 2, in die Luft gehen, spielt keine Rolle. Der Osten mag ruhig explodieren, denn einzig zählt, was in der Westlichen Welt gegenwärtig stattfindet: eine Implosion zunächst der Hochtechnologien und in ihrer Folge auch einer Signifikantenszene, die sonst womöglich noch immer Weltgeist hieße. Ohne Computertechnik keine Dekonstruktion, sagte Derrida in Siegen. Schriften und Texte (unter Einschluß des Textes, den ich eben vorlese) existieren mithin nicht mehr in wahrnehmbaren Zeiten und Räumen, sondern in den Transistorzellen von Computern. Und da die Heldentaten von Silicon Valley es in den letzten drei Dekaden geschafft haben, die Transistorzellengröße auf Dimensionen im Submikronbereich, also unter einen Mikrometer zu drücken, kann unsere aktuelle Schreibszene nurmehr in Begriffen der fraktalen Geometrie beschrieben werden: als Selbstähnlichkeit der Buchstaben über einige sechs Dekaden, von der haushohen Firmenreklame bis zur gleichen, aber transistorgro-

1 Mit Dank an Wolfgang Hagen/Radio Bremen, der den Textvergleich zwischen den zwei Versionen von *Eve of Destruction* seinen Radiohörern bei laufender Sendung vorführte.

ßen Bitmap. Am alphabetischen Beginn von Geschichte überhaupt lagen zwischen einem Kamel und dem Gamel, seinem hebräischen Buchstaben, gerade zweieinhalb Dekaden; mit der Miniaturisierung aller Zeichen auf molekulare Maße dagegen ist der Schreibakt selber verschwunden.

Wie wir alle wissen und nur nicht sagen, schreibt kein Mensch mehr. Schrift, diese seltsame Art Software, laborierte wohl an ihrer unheilbaren Verwechslung von Gebrauch und Erwähnung. Bis in die Tage von Hölderlins Hymnen scheint die bloße Erwähnung etwa eines Blitzes noch hinreichende Evidenz für seinen möglichen poetischen Gebrauch gewesen zu sein.[2] Heute dagegen, nach der Verwandlung dieses Blitzes in Elektrizität, läuft menschliches Schreiben durch Inschriften, die nicht nur mittels Elektronenlithographie in Silizium eingebrannt, sondern im Unterschied zu allen Schreibwerkzeugen der Geschichte auch imstande sind, selber zu lesen und zu schreiben.

Letzter historischer Schreibakt mag es folglich gewesen sein, als in den späten Siebzigern ein Team von Intel-Ingenieuren unter Leitung von Dr. Marcian E. Hoff einige Dutzend Quadratmeter[3] Zeichenpapier auf leergeräumten Garagenböden Santa Claras auslegte, um die Hardware-Architektur ihres ersten integrierten Mikroprozessors aufzuzeichnen. Dieses manuelle Layout aus zweitausend Transistoren und ihren Verbindungskanälen wurde in einem zweiten, nun aber mechanischen Schritt auf die Daumennagelgröße des realen Chips verkleinert und drittens von elektro-optischen Geräten ins Silizium geschrieben. Und nachdem das Endprodukt, der 4004 als Prototyp aller seitherigen Mikroprozessoren, viertens auch noch seinen Platz in den neuen Tischrechnern von Intels japani-

2 Vgl. in Bälde Thomas Hafki, *Elektrizitat und Literatur. 1750 – 1816.*

3 1978, beim Design des Prozessors Intel 8086, sollen diese Blaupausen 64 m² Millimeterpapier bedeckt haben. Vgl. Klaus Schrodl, 1990, *Quantensprung.* DOS, Heft 12, S. 102 f.

schem Auftraggeber eingenommen hatte, konnte unsere postmoderne Schreibszene gerade eben beginnen.

Mittlerweile nämlich, bei der Hardware-Komplexität gegenwärtiger Mikroprozessoren, haben manuelle Entwurfstechniken längst keine Chance mehr. Um die jeweils nächste Computergeneration zu entwickeln, hilft den Ingenieuren kein Zeichenpapier weiter, sondern nur noch Computer Aided Design: Die geometrischen Fähigkeiten der jeweils letzten Rechnergeneration reichen eben hin, um die Topologie ihrer Nachfolgergeneration zu entwerfen. So stehen die Füße derer, die dich hinaustragen werden, einmal mehr vor der Tür.

Und doch hatte Marcian E. Hoff schon mit seinen primitiven Blaupausen das fast vollkommene Beispiel einer Turingmaschine gegeben. Seit Turings Dissertation von 1937 kann jeder Rechenakt, ob bei Menschen oder Maschinen, formalisiert werden als eine abzählbare Menge von Befehlen, die über einem unendlich langen Papierband und seinen diskreten Zeichen arbeiten. Turings Konzept einer solchen Papiermaschine,[4] deren Operationen nur Schreiben und Lesen, Vorrücken und Zurückgehen umfassen, hat sich als mathematisches Äquivalent aller berechenbaren Funktionen erwiesen und dafür gesorgt, daß die unschuldige Berufsbezeichnung Computer vom maschinellen Wortsinn völlig verdrängt worden ist.[5] Universale Turingmaschinen brauchen nur mit der Beschreibung

4 Vgl. Alan M. Turing, 1937, *Über berechenbare Zahlen. Mit einer Anwendung auf das Entscheidungsproblem.* In: Turing, 1987, S. 40f.

5 Wie weit diese Verdrängung fortgeschritten ist, demonstrieren die Verfasser oder Setzer von Intels *Programmer's Reference Manuals:* Aus dem Gleitkommabefehl *f2xm1*, also der Zweierpotenz einer Eingangsgroße minus eins, wird nach seiner Übersetzung ins Alltagsenglische nicht etwa »Compute 2^x-1«, sondern »Computer 2^x-1«. Vgl. Intel Corporation, 1989, *387 DX User's Manual. Programmer's Reference.* Santa Clara/CA., S. 4–9, wie auch Intel Corporation, 1990, *i486 Microprocessor. Programmer's Reference Manual.* Santa Clara/CA., S. 26–72.

(dem Programm) einer beliebigen anderen Maschine gefüttert zu werden, um diese Maschine effektiv zu imitieren. Und weil seit Turing von den Hardware-Unterschieden zwischen beiden Geräten erstmals abstrahiert werden darf, läuft die sogenannte Church-Turing-Hypothese in ihrer strengsten, nämlich physikalischen Form darauf hinaus, die Natur selber zur Universalen Turingmaschine zu erklären.

Diese Behauptung als solche hat den Effekt gehabt, die Implosion der Hardware durch eine Explosion der Software zu verdoppeln. Seitdem Computer implementiert werden können, ab 1943 auf Röhrenbasis, ab 1949 auf Transistorbasis, besteht auch das Problem, die universalen, selber aber unlesbaren Schreib-Lese-Maschinen irgendwie zu beschreiben und zu lesen. Seine Lösung heißt bekanntlich Software, also Entwicklung höherer Programmiersprachen. Das uralte Monopol der Alltagssprachen, ihre eigene Metasprache zu sein und damit keinen Anderen des Anderen mehr zu haben, ist zusammengebrochen und einer neuen Hierarchie der Programmiersprachen gewichen. Dieser postmoderne Turm von Babel[6] reicht mittlerweile von schlichten Befehls-Codes, deren linguistische Exten-

6 Vgl. Wolfgang Hagen, 1989, *Die verlorene Schrift. Skizzen zu einer Theorie der Computer.* In: *Arsenale der Seele. Literatur- und Medienanalyse seit 1870.* Hrsg. Friedrich A. Kittler und Georg Christoph Tholen, München, S. 221: »In der Sprachstruktur der Neumannschen Maschinenlogik liegt also schon der prinzipielle Auseinanderfall von Software und Software-Dokumentation begründet, und so türmt sich seit 1945 ein babylonischer Programm-Turm von Computer-performances auf, deren Benutzung mit der sinnvollen Veranstaltung einer Maschinensprache nichts mehr zu tun hat. Ein Software-Turm mit undokumentierten Fehlern, heillos verworrenen Dialekten und einer Anhäufung von sprachlichen Akten, die niemand mehr nachvollziehen kann.« In einem weniger präzisen, dafür aber desperaten Bild formuliert ein UNIX-Experte: »Fast alle großen Betriebssysteme zeichnen sich nach einem gewissen Alter durch einen hohen ›Verschmutzungsgrad‹ aus. Sie wuchern in alle Richtungen und machen den Eindruck, als seien sie Trümmer, die nur noch

sion noch eine Hardwarekonfiguration ist, über Assembler, dessen Extension genau jene Befehls-Codes sind, bis zu sogenannten Hochsprachen, deren Extension nach allen möglichen Umwegen über Interpreter, Compiler und Linker wiederum Assembler heißt. Schreiben heute ist also auch als Softwareentwicklung eine schier endlose Kette von Selbstähnlichkeiten, wie die fraktale Geometrie sie entdeckt hat. Nur daß es, im Unterschied zum mathematischen Modell, eine physisch-physiologische Unmöglichkeit bleibt, all diese Schichten noch zu erreichen. Moderne Medientechnologien sind, schon seit Film und Grammophon, grundsätzlich daraufhin angelegt, die Sinneswahrnehmungen zu unterlaufen. Wir können schlichtweg nicht mehr wissen, was unser Schreiben tut, und beim Programmieren am allerwenigsten.

Zur Illustration dieser Lage genügen aber schon alltäglichere Fälle, etwa das Textverarbeitungsprogramm, dem meine Wörter entstammen. Der Genius loci von Palo Alto, der die ersten wie auch die elegantesten Betriebssysteme hervorgebracht hat, möge verzeihen, daß ein Untertan der Microsoft Corporation seine Beispiele auf das dümmste unter ihnen allen beschränkt.

Um Texte zu prozessieren, also selbst zur Papiermaschine auf einem IBM AT unter Microsoft DOS zu werden, steht zunächst der Kauf eines kommerziellen Softwarepakets ins Haus. Zweitens müssen ein paar Dateien aus diesem Paket die Dateiextensionsnamen .EXE oder .COM tragen, andernfalls eine Textverarbeitung unter DOS nie starten könnte. Ausführbare Dateien und nur sie unterhalten nämlich ein seltsames Verhältnis zu ihrem Eigennamen. Auf der einen Seite tragen sie vollmundig autoreferenzielle Namen wie etwa WordPerfect, auf der anderen Seite ein mehr oder minder kryptisches, weil vokalloses Akronym

mühsam zusammengehalten werden.« (Horst Drees, 1988, *UNIX. Ein umfassendes Kompendium für Anwender und Systemspezialisten.* Haar, S. 19.) Der UNIX-Experte ist zu höflich, um in die Wucherungen einen Firmeneigennamen wie Microsoft Corporation einzuflechten.

wie etwa WP. Der volle Name dient allerdings nur den notwendigerweise noch immer alltagssprachlichen Reklamestrategien der Softwarehäuser, und zwar deshalb, weil das Disk Operating System alias DOS Dateinamen mit mehr als acht Buchstaben gar nicht lesen könnte. Deshalb sind unaussprechliche, von Vokalen tunlichst befreite Abkürzungen oder Akronyme, dieser Widerruf einer elementaren griechischen Innovation, für postmodernes Schreiben nicht nur notwendig, sondern auch völlig hinreichend. Mehr noch, sie scheinen dem Alphabet erstmals seit seiner Erfindung wieder magische Kräfte zuzuführen. Das Kürzel WP nämlich tut, was es sagt. Im Unterschied nicht nur zum Wort WordPerfect, sondern auch zu leeren alteuropäischen Wörtern wie Geist oder Wort umfassen ausführbare Computerdateien alle Routinen und Daten, die zu ihrer Realisierung notwendig sind. Der Schreibakt, auf einer AT-Konsole die Tasten W, P und Enter anzutippen, macht zwar das Wort nicht vollkommen, startet aber doch einen aktuellen Lauf von WordPerfect. Solche Triumphe gewährt Software.

Woraufhin die beiliegende, mehr oder minder inflationäre Paperware, um nicht hinter der Kommandozeile zurückzubleiben, die Zauberkräfte noch verdoppelt. Gängige Softwarehandbücher, weil sie ja den Abgrund zwischen formalen und alltäglichen Sprachen, Elektronik und Literatur überbrücken müssen, präsentieren ihr Programmpaket als linguistischen Agenten, dessen Allmacht über Systemressourcen, Adreßräume und Hardware-Parameter des betroffenen Computers schlechthin gebietet: WP, von der Kommandozeile mit Argument X aufgerufen, würde den Bildschirm von Modus A nach B schalten, in Einstellung C beginnen, am Ende nach D zurückkehren usw.[7]

7 Das einzige mir bekannte Gegenbeispiel stammt nicht zufällig aus Richard Stallmans Free Software Foundation, die ja überhaupt dem Software-Copyright einen ebenso heroischen wie verzweifelten Kampf angesagt hat. Das Gegenbeispiel lautet: »When we say that ›*C-n* moves down vertically one line‹ we are glossing over a distinction that is irrelevant in ordinary use but

Nur sind alle Taten, die Agent WP laut Paperware vollbringt, gänzlich virtuell, weil jede Einzelaktion, wie es so treffend heißt, »unter« DOS zu laufen hat. Faktisch arbeitet nur das Betriebssystem und näherhin seine Shell: COMMAND.COM durchsucht den Tastaturpuffer nach einem 8-Byte-Dateinamen, übersetzt die relativen Adressen einer eventuell gefundenen Datei in absolute, lädt diese modifizierte Version aus dem externen Massenspeicher ins Silizium-RAM und übergibt die zeitweilige Programmausführung schließlich den ersten Code-Zeilen eines Sklaven namens WordPerfect.

Dasselbe Argument kann aber auch gegen DOS gewandt werden, weil das Betriebssystem in letzter Analyse als bloße Erweiterung eines basalen Input/Output-Systems namens BIOS arbeitet. Kein einziges Anwenderprogramm, ja nicht einmal das zugrunde liegende Mikroprozessorsystem könnte jemals starten, wenn ein paar elementare Funktionen, die aus Sicherheitsgründen in Silizium gebrannt sind, also Teil der unlöschbaren Hardware bilden, nicht über Münchhausens Fähigkeit verfügten, sich am eigenen Schopf aus dem Sumpf zu ziehen.[8] Jede materielle Transformation von Entropie in Information, von einer Million schlummernder Transistorzellen in elektrische Spannungsdifferenzen setzt notwendig ein materielles Ereignis namens Reset voraus.

Im Prinzip könnte dieser Abstieg von Software zu Hardware, von höheren zu niedrigeren Beobachtungsebenen über beliebig viele Dekaden laufen. Sogar die elementaren Code-Operationen, trotz ihrer metaphorischen Fähigkeiten wie etwa Call oder Return, reduzieren sich auf absolut lokale Zeichenmanipulationen und damit, Lacan sei's ge-

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, 1988, GNU Emacs Manual. 6. Aufl. Cambridge/MA., S. 19.)

8 Dies möge als freie Übersetzung von Booting durchgehen.

klagt, auf Signifikanten elektrischer Potentiale. Alle Formalisierung in Hilberts Wortsinn hat den Effekt, Theorie abzuschaffen, einfach weil »die Theorie ein System nicht mehr bedeutsamer Aussagen ist, sondern ein System von Sätzen als Wortfolgen, welche Wortfolgen ihrerseits Buchstabenfolgen sind. Deshalb kann man allein aufgrund der Form unterscheiden, welche Wörterkombinationen Sätze sind, welche Sätze Axiome und welche Sätze als unmittelbare Folgen aus anderen hervorgehen.«[9]

Wenn Bedeutungen zu Sätzen, Sätze zu Wörtern, Wörter zu Buchstaben schrumpfen, gibt es auch keine Software. Oder vielmehr: Es gäbe sie nicht, wenn Computersysteme nicht bislang in einer Umgebung aus Alltagssprachen koexistieren müßten. Diese Umwelt besteht jedoch seit einer berühmten und zweifachen griechischen Erfindung[10] aus Buchstaben und Münzen, letters and litters. Diese guten ökonomischen Gründe haben die Demut Alan Turings, der in der Steinzeit des Computerzeitalters lieber binären als dezimalen Maschinen-Ausdruck las,[11] mittlerweile gründlich ausgerottet. Die sogenannte Philosophie der sogenannten Computergemeinschaft setzt im Gegenteil alles daran, Hardware hinter Software, elektronische Signifikanten hinter Mensch-Maschine-Schnittstellen zu verdecken. In aller Menschenfreundlichkeit warnen Programmierhandbücher für Hochsprachen vor der geistigen Zerrüttung, die beim Schreiben trigonometrischer Funktionen in Assembler ausbräche.[12] In aller Liebenswürdigkeit übernehmen BIOS-Prozeduren (und deren Fachautoren) die Funktion, »die Einzelheiten der Steuerung zugrunde liegender Hard-

9 Stephen C. Kleene, zitiert in: Robert Rosen, 1988, *Effective Processes and Natural Law.* In: Herken, 1988, S. 527.

10 Vgl. Johannes Lohmann, 1980, *Die Geburt der Tragödie aus dem Geiste der Musik.* Archiv für Musikwissenschaft, 37, S. 174.

11 Vgl. Andrew Hodges, 1983, *Alan Turing: the enigma.* New York, S. 399.

12 Vgl. *TOOL Praxis,* 1989, *Assembler-Programmierung auf dem PC, Ausgabe 1.* Würzburg, S. 9.

ware vor Ihrem Programm zu verstecken«[13]. Weitergedacht, würden also, nicht viel anders als im Gradualismus mittelalterlicher Engelshierarchien, Betriebssystemfunktionen wie COMMAND.COM das BIOS verbergen, Anwenderprogramme wie WordPerfect das Betriebssystem usw. – bis schließlich in den allerletzten Jahren zwei fundamentale Umstellungen im Computerdesign (oder im Wissenschaftskonzept des Pentagon) dieses ganze Geheimsystem seiner erfolgreichen Schließung zugeführt haben.

Zunächst einmal wurden, auf einer mit Absicht oberflächlichen Ebene, brauchbare graphische Schnittstellen entwickelt, die, weil sie die zur Programmierung immer noch unumgänglichen Schreibakte verstecken, eine ganze Maschine ihren Benutzern entziehen. Denn nicht einmal das IBM-autorisierte Computergraphikkompendium gibt vor, daß computergraphische Benutzeroberflächen die Systemprogrammierung schneller oder effizienter als schlichte Kommandozeilen machen würden.[14] Zweitens entstand

13 Nabajyoti Barkalati, 1989, *The Waite Group's Macroassembler Bible.* Indiana/IL., S. 528.

14 Vgl. James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, 1990, *Computer Graphics. Principles and Practice.* 2. Aufl. Reading/MA., S. 397 f.: »Direct [graphic] manipulation is sometimes presented as being the best user-interface style. It is certainly quite powerful and is especially easy to learn. But the Macintosh interface can be slow for experiented users in that they are forced to use direct manipulation when another style would generally be faster. Pointing the file ›Chapter 9‹ with direct manipulation requires the visual representation of the file to be found and selected, then the Print command is involved. Finding the file icon might evolve scrolling through a large collection of items. If the user knows the name of the file, typing ›Print Chapter 9‹ is faster. Similarly, deleting all files of type ›txt‹ requires finding and selecting each such file and dragging it to a trash can. Much faster is the UNIX-style command ›rm *.txt‹, which uses the wildcard * to find all files whose names end in ›.txt‹.« Aus alldem folgt schließlich die niederschmetternde, weil auf den Zweck von Computern selbst bezogene Untertreibung: »Some applications, such as programming, to not lend themselves to direct manipulation.«

in unmittelbarem Zusammenhang mit ADA,[15] der Pentagon-Programmiersprache, aber auf der mikroskopischen Ebene von Hardware selber, eine neue Prozessorbetriebsart namens Protected Mode, die nach Auskunft von Intels *Microprocessor Programming Manual* den einzigen Zweck verfolgt, »untrusted programs« und »untrusted users« von jedem Zugriff auf Systemressourcen wie Eingabe/Ausgabe-Kanäle oder Operationssystemkern abzuhalten. Vertrauensunwürdig in diesem technischen Sinn sind aber Anwender überhaupt, die im Protected Mode (wie er etwa unter UNIX herrscht) ihre Maschine gar nicht mehr steuern dürfen.

Dieser unaufhaltsame Siegeszug der Software ist eine seltsame Umkehrung von Turings Beweis, daß es keine im mathematischen Sinn berechenbaren Probleme geben kann, die eine schlichte Maschine nicht zu lösen vermöchte. An der genauen Stelle dieser Maschine hat die physikalische Church-Turing-Hypothese, weil sie ja die physische Hardware mit den Algorithmen zu ihrer Berechnung gleichsetzt, eine Leerstelle geschaffen, die die Software erfolgreich besetzen konnte, nicht ohne dabei von ihrer Dunkelheit zu profitieren.

Denn schließlich arbeiten Programmierhochsprachen, je höher und alltagssprachlicher ihr babylonischer Turm wächst, ganz wie die sogenannten Einwegfunktionen der jüngsten mathematischen Kryptographie.[16] In ihrer Stan-

15 Über den Zusammenhang zwischen Pentagon, ADA und Intels iAPX 432, dem ersten Mikroprozessor im Protected Mode, dessen ökonomischem Scheitern dann der Industriestandard vom 80286 bis zum 80486 entsprang, vgl. Glenford Myren jr., 1982, *Overview of the Intel iAPX 432 Microprocessor.* In: *Advances in Computer Architecture.* New York, S. 335-344. (Mit Dank an Ingo Ruhmann/Bonn.) Wer dem Scheitern näherkommen will, meditiere eine Assemblersitzung lang über den Satz: »The 432 can be characterized as a three-address-storage-to-storage architecture, there are no registers visible to programs.« (S. 342)

16 Zum folgenden vgl. Patrick Horster, 1982/1985, *Kryptologie: eine Anwendung der Zahlentheorie und Komplexitätstheorie.* Mannheim – Wien – Zürich, S. 23 – 27.

dardform lassen sich solche Funktionen mit vertretbarem Zeitaufwand berechnen, etwa wenn die Maschinenlaufzeit nur in polynomischen Ausdrücken der Funktionskomplexität anwächst. Dagegen würde der Zeitaufwand für die inverse Form, also um aus dem Ergebnis der Funktion auf ihre Eingangsparameter zurückzuschließen, in exponentiellem und mithin untragbarem Verhältnis zur Funktionskomplexität steigen. Einwegfunktionen, mit anderen Worten, schützen Algorithmen vor ihrem eigenen Ergebnis.

Für Software kommt diese kryptographische Eigenschaft wie gerufen. Sie bietet einen bequemen Weg, den Sachverhalt zu umgehen, daß nach Turings Beweis das Konzept geistigen Eigentums unmöglich und bei Algorithmen am unmöglichsten geworden ist. Eben daß Software als maschinenunabhängige Fähigkeit nicht existiert, läßt sie als kommerzielles oder amerikanisches Medium nur um so mehr insistieren. Alle Lizenzen, Dongles und Patente, die für WP wie auch für WordPerfect angemeldet sind, beweisen die Funktionalität von Einwegfunktionen. Amerikanische Gerichte haben kürzlich sogar, jeder mathematischen Ehre zum Trotz, Copyright-Ansprüche auf Algorithmen bestätigt.

So nimmt es nicht wunder, daß jüngst auch die höchste Ebene, IBM mit Namen, die Jagd auf mathematische Formeln eröffnet hat, die über Shannons Informationsmaß und Kolmogorows Informationsgehalt hinaus Algorithmen nach ihrer Originalität bewerten könnten. Gerade daß in den guten alten Tagen der Informationstheorie maximale Information und maximales Rauschen einigermaßen zusammenfielen, Würfelwurfserien also mathematisch prämiert wurden, wirft die IBM Shannon vor. Aber auch das Kolmogorow-Maß, das den kürzesten unter allen möglichen Algorithmen zur Erzeugung eines konstanten Outputs prämiert, befriedigt sie noch nicht. Kolmogorow zufolge würde die ganze Mühsal bei Errechnung eines trigonometrischen oder astronomischen Tabellenwerks in den schlichten Gleichungen, die ihm zugrunde liegen, ja wie-

der verschwinden. Also ist das neue IBM-Maß logischer Tiefe, in seiner Verwechslung von Sitzfleisch und Einfall, wie folgt definiert:

> Der Wert einer Nachricht [...] scheint weder in ihrer Information (ihren absolut unvorhersagbaren Teilen) noch in ihrer offensichtlichen Redundanz (wörtlichen Wiederholungen, ungleichen Bitfrequenzen) zu liegen, sondern vielmehr in etwas, was begrabene Redundanz heißen könnte – in Teilen, die nur mit Schwierigkeiten vorhersagbar sind, und Sachverhalten, die der Empfänger im Prinzip hätte herausfinden können, ohne sie gesagt zu bekommen, aber nur mit beträchtlichem Aufwand an Geld, Zeit oder Rechenleistung. Mit anderen Worten, der Wert einer Nachricht ist der Betrag an mathematischer oder andersartiger Arbeit, den ihr Sender plausiblerweise aufgebracht und den ihr Empfänger nicht noch einmal zu vollbringen hat.[17]

IBMs Maß logischer Tiefe in seiner mathematischen Strenge könnte mithin die alten, notwendig ungenauen Alltagssprachbegriffe von Originalität, Autorschaft und Copyright sämtlich ersetzen, also auch prozedural einklagbar machen. Nur leider ist gerade der Algorithmus zur Originalitätsberechnung von Algorithmen überhaupt selber turingunberechenbar.[18]

In dieser tragischen Lage hat das Strafrecht, zumindest in Deutschland, den Begriff des geistigen Eigentums an einer ebenso immateriellen Software fallengelassen und Software statt dessen als »Sache« definiert. Die Feststellung des Bundesgerichtshofs, derzufolge kein Computerprogramm ohne entsprechende elektrische Ladungen in Siliziumschaltkreisen je laufen würde,[19] beweist einmal mehr, daß der virtuellen Unentscheidbarkeit zwischen Software und

17 Charles H. Bennett, 1988, *Logical Depth and Physical Complexity.* In: Herken, 1988, S. 230.

18 Mit Dank an Oswald Wiener/Dawson City.

19 Vgl. M. Michael König, 1991, *Sachlich sehen. Probleme bei der Überlassung von Software.* c't, Heft 3, S. 73.

Hardware keineswegs nur, wie Systemtheoretiker so gern glauben würden, ein Wechsel der Beobachterperspektive zugrunde liegt.[20] Gute Gründe sprechen vielmehr für die Unabdingbarkeit und folglich auch die Vorgängigkeit von Hardware.

Denn die Maschine mit unbegrenzten Ressourcen in Zeit und Raum, mit unendlichem Papiernachschub und grenzenloser Rechengeschwindigkeit hat es nur einmal gegeben: in Turings Papier *Über berechenbare Zahlen mit einer Anwendung auf das Entscheidungsproblem*. Allen physikalisch machbaren Maschinen dagegen setzen diese Parameter strikte Grenzen im Code selber. Die Unfähigkeit von Microsoft DOS, Dateinamen von mehr als acht Buchstaben wie etwa WordPerfect zu erkennen, beleuchtet auf ihre triviale und obsolete Art nicht nur ein Problem, das zu immer größeren Inkompatibilitäten zwischen den verschiedenen Generationen von 8-Bit-, 16-Bit- und 32-Bit-Mikroprozessoren geführt hat. Sie verweist auch auf eine prinzipielle Unmöglichkeit der Digitalisierung, den Körper der reellen Zahlen, also die ehedem so genannte Natur, zu berechnen.[21]

Das heißt aber, in den Worten des Los Alamos National Laboratory: »Wir benutzen digitale Computer, deren Ar-

20 Eher könnte man, wie in einem Brief Dirk Baeckers vom 15.4.1991, »vermuten, daß die Unterscheidung zwischen hardware und software eine Unterscheidung ist, die den Wiedereintritt der Unterscheidung zwischen Programmierbarkeit und Nichtprogrammierbarkeit in den Bereich des Programmierbaren zu betreuen hat. Sie steht gleichsam für die Berechenbarkeit der Technik, und in dem Sinne für die Technik selbst. Sie kann dafür nur stehen, weil die ›Einheit‹ des Programms nur realisiert werden kann, wenn Gleichung und Berechnung jeweils so auf zwei Seiten verteilt werden, daß immer nur eine Seite operativ zur Disposition steht und die andere Seite konstant gehalten werden kann.«

21 Ich verstehe also nicht, wie Turings berühmtes Papier, nachdem es im ersten Satz »berechenbare Zahlen in Kürze als diejenigen reellen Zahlen beschrieb«, »deren Dezimalausdrücke mit endlichen Mitteln errechnet werden können« (Turing 1937,

chitektur uns in Form einer physikalischen Maschine mit all ihren künstlichen Beschränkungen gegeben ist. Wir müssen kontinuierliche algorithmische Beschreibungen erst auf Beschreibungen reduzieren, die auf einem Gerät, dessen fundamentale Operationen abzählbar sind, codiert werden können. Wir erreichen das auf dem Weg vielfältiger Zerstückelungen, die üblicherweise Diskretisierung heißen. Der Compiler schließlich reduziert dieses Modell auf eine binäre Form, die weitgehend von Maschinenzwängen bestimmt wird.

Das Ergebnis ist gegenüber dem ursprünglichen Problem ein diskretes und synthetisches Mikrowelt-Abbild, dessen Struktur willkürlich durch ein Differenzierungsschema und eine beliebig gewählte Rechnerarchitektur festgelegt wird. Der einzige Überrest des vormaligen Kontinuums ist der Einsatz einer Zahlenbasis-Arithmetik, deren Eigentümlichkeit ungleiche Gewichtungen der Bits und deren Folge für nichtlineare Systeme trügerische Singularitäten sind.

Genau das tun wir, wenn wir ein Modell der physischen Welt mit physischen Geräten erstellen. Es ist nicht jener idealisierte und serene Prozeß, den wir üblicherweise beim Argumentieren über fundamentale Rechenstrukturen ausmalen, und von Turingmaschinen weit entfernt.«[22]

Es ginge also nicht mehr an, die physikalische Church-Turing-Hypothese weiterzuverfolgen und damit »ins Verhalten der physikalischen Welt ein algorithmisches Verhalten zu injizieren, für das es keinerlei Evidenz gibt«[23]. Wenn die Welt nicht aus Gottes Würfelwurf entsteht, schließt das algorithmische Verhalten von Regenwolken oder Meereswellen nicht ein, sondern aus, daß ihre Moleküle als Computer der eigenen Tätigkeit arbeiten. Umgekehrt käme alles

S. 19), und daraufhin die Menge der berechenbaren Zahlen als abzählbar definierte, schließlich π als »Grenzwert einer berechenbar konvergenten Folge« zur berechenbaren Zahl ernennen konnte (S. 49 f.).

22 Brosl Hasslacher, 1988, *Algorithms in the World of Bounded Resources.* In: Herken, 1988, S. 421 f.

23 Hasslacher, 1988, S. 420.

darauf an, den »Preis der Programmierbarkeit« selber zu berechnen. Diese entscheidende Fähigkeit von Computern hat ersichtlich nichts mit Software zu tun; sie hängt einzig und allein vom Grad ab, in dem eine jeweilige Hardware dergleichen wie ein Schreibsystem beherbergen kann. Als Claude Shannon 1937 »in der wohl folgenreichsten Magisterarbeit, die je geschrieben wurde«[24], den Nachweis führte, daß schlichte Telegraphenrelais die gesamte Boolesche Algebra implementieren können, war ein solches Aufschreibesystem installiert. Und als der integrierte Schaltkreis, in den frühen Siebzigern aus Shockleys Transistor abgeleitet, auf ein und demselben Chip das Element Silizium, diesen kontrollierbaren Widerstand, mit seinem eigenen Oxid, diesem fast idealen Isolator, kombinierte, konnte die Programmierbarkeit der Materie, ganz wie Turing prophezeit hatte, »die Kontrolle übernehmen«[25]. Software, wenn es sie denn gäbe, wäre bloß ein Milliarden-Dollar-Geschäft rund um eines der billigsten Elemente auf Erden. Denn in ihrer Verbindung auf dem Chip sorgen Silizium und Siliziumoxid für nahezu perfekte Hardware. Einerseits arbeiten Millionen von Schaltungselementen unter denselben physikalischen Bedingungen, was vor allem für den kritischen Parameter Chiptemperatur entscheidend ist und exponentiell anwachsende Abweichungen der Transistorspannung verhindert; andererseits bleiben diese Millionen von Schaltungselementen voneinander elektrisch isoliert. Einzig diese paradoxe Beziehung zwischen zwei physikalischen Parametern, der thermischen Kontinuität und der elektrischen Diskretisierung, ermöglicht es integrierten Digitalschaltkreisen, nicht einfach endliche Automaten zu sein wie so viele andere Dinge auf Erden, sondern jene Universale Diskrete Maschine zu ap-

24 Friedrich-Wilhelm Hagemeyer, 1979, *Die Entstehung von Informationskonzepten in der Nachrichtentechnik. Eine Fallstudie zur Theoriebildung in der Technik in Industrie- und Kriegsforschung.* Diss. phil. FU Berlin, S. 432.

25 Turing, 1959, *Intelligente Maschinen. Eine häretische Theorie.* In: Turing, 1987, S. 15.

proximieren, in die der Name ihres Erfinders Turing längst untergetaucht ist.

Diese Strukturdifferenz kann sehr leicht illustriert werden. Zum Beispiel »ist ein Kombinationsschloß ein endlicher Automat, kann aber nicht in eine Basismenge von elementaren Komponenten zerlegt werden, die zur Simulation eines beliebigen physikalischen Systems auch wieder neu konfiguriert werden könnten. Folglich ist das Kombinationsschloß nicht strukturell programmierbar, und in diesem Fall kann es effektiv programmierbar nur in dem eingeschränkten Sinn heißen, daß sein Zustand gesetzt werden kann, um eine eingeschränkte Klasse von Verhaltensweisen zu bewirken.« Demgegenüber »ist ein Digitalcomputer, der zur Simulation eines Kombinationsschlosses eingesetzt wird, strukturell programmierbar, weil dieses Verhalten durch Synthese aus einer kanonischen Menge elementarer Schaltkomponenten erreicht wird«.[26]

Schaltkomponenten aber, seien es Telegraphenrelais, Elektronenröhren oder schließlich Siliziumtransistoren, zahlen für ihre Zerlegbarkeit oder Diskretisierung einen Preis. Abgesehen vom trivialen, weil diskreten Fall Textverarbeitung, der jedoch hinter all den anderen wissenschaftlichen, militärischen und industriellen Computereinsatzgebieten nachgerade verschwindet, stehen Digitalrechner als einzige »Ja-Nein-Organe im strengen Sinne des Wortes«[27] weiterhin einer kontinuierlichen Umwelt aus Wolken, Kriegen und Wellen gegenüber. Diese Lawine großer und reeller Zahlen, wie Ian Hacking sagen würde, bewältigen sie aber nur durch additive Anfügung von immer mehr Schaltelementen, bis aus den 2000 Transistoren des Intel 4004 die 1,2 Millionen des momentanen Intel-Flaggschiffs 80486 geworden sind. Es läßt sich aber mathematisch zeigen, daß die Wachstumsrate möglicher Vernetzungen zwischen diesen Elementen und damit die Rechenleistung als

26 Michael Conrad, 1988, *The Prize of Programmability.* In: Herken, 1988, S. 289.

27 Vgl. John von Neumann, 1951/1967, *Allgemeine und logische Theorie der Automaten.* Kursbuch Nr. 8, S. 150.

solche eine Quadratwurzelfunktion zur Obergrenze hat. Das System, anders gesagt, kann »nicht mit polynomischen Zuwachsraten des Problemumfangs mithalten«[28], um von exponentiellen Raten ganz zu schweigen. Eben die Isolation zwischen digitalen oder diskreten Elementen, die seine Funktionsfähigkeit zumindest bei weder tropischen noch arktischen Bedingungen sicherstellt, beschränkt auch das Ausmaß möglicher Vernetzungen auf die lokalen Nachbarn eines jeden Elements. Bei globalen Wechselwirkungen hingegen, wie Digitalchips sie nur in ihrer Thermik kennen, könnte die Vernetzbarkeit »den geltenden Kräftegesetzen«[29] und der kombinatorischen Logik zufolge bis zu einer oberen Schranke ansteigen, die bei der Quadratzahl aller beteiligten Elemente läge.

Genau diese optimale Vernetzbarkeit aber zeichnet auf der anderen oder physikalischen Seite nichtprogrammierbare Systeme aus. Aufgrund ihrer globalen Wechselwirkungen können solche Systeme, ob nun Wellen oder Wesen, polynomische Zuwachsraten an Komplexität aufweisen, deshalb aber auch nur von Maschinen berechnet werden, die nicht selber den Preis der Programmierbarkeit zahlen müßten. Ganz offenbar würde dieser hypothetische, aber bitter notwendige Maschinentyp reine Hardware darstellen: ein physisches Gerät, das in einer Umgebung aus lauter physischen Geräten arbeitet und nur derselben Beschränkung seiner Ressourcen wie sie untersteht. Software im üblichen Sinn einer immer machbaren Abstraktion gäbe es nicht mehr. Die Prozeduren einer solchen Maschine, wiewohl sie algorithmischer Verschriftung weiterhin offenstünden, müßten wesentlich auf einem materiellen Substrat arbeiten, dessen Vernetzbarkeit wechselnde Rekonfigurationen seiner Zellen erlauben würde. Und obwohl »auch dieses Substrat, mithilfe von Simulationen, in algorithmischen Ausdrücken beschrieben werden könnte, ist seine Charakterisierung doch von so unermeßlicher

28 Conrad, 1988, S. 293.
29 Conrad, 1988, S. 290.

Bedeutung für die Effektivität [...] und mit der Hardware-Auswahl so eng verknüpft«[30], daß seine Programmierung mit der von approximierten Turingmaschinen nichts mehr gemein haben wird.

Solch dringend notwendige und wohl nicht mehr allzu ferne Maschinen, wie sie von der aktuellen Informatik diskutiert und von der Chipindustrie auch schon angenähert werden,[31] dürften einige Beobachteraugen Dubrovniks wohl in die Versuchung führen, das vertraute Antlitz des Menschen, evolutionär verkleidet oder auch nicht, in ihnen wiederzufinden. Mag sein. Gleichzeitig aber befolgt unsere nicht minder vertraute Silizium-Hardware schon heute viele der Anforderungen an hochvernetzte nichtprogrammierbare Systeme. Zwischen ihrer Million von Transistorzellen finden eine Million-im-Quadrat von Wechselwirkungen immer schon statt: Elektronendiffusion und quantenmechanische Tunneleffekte laufen über den ganzen Chip. Nur behandelt die gegenwärtige Herstellungstechnik solche Interaktionen als Systemschranken, physikalische Nebeneffekte, Störquellen usw. All das Rauschen, das unmöglich zu verhindern ist, doch wenigstens zu minimieren –: genau das ist der Preis, den die Computerindustrie für strukturell programmierbare Maschinen entrichten muß. Die umgekehrte Strategie, das Rauschen zu maximieren, fände nicht nur den Weg zurück von IBM zu Shannon; sie wäre wohl auch der einzige Weg zu jenem Körper reeller Zahlen, der ehedem Chaos hieß.

»Cant't you understand what I'm tryin' to say«, heißt es – ohne Remake – in *Eve of Destruction*.

30 Conrad, 1988, S. 304.

31 So griff etwa das erste integrierte Neuronale Netzwerk, ausgerechnet aus Intels diskretem Chip-Imperium und, soweit ich sehen kann, zum zweitenmal in der ganzen Firmengeschichte nach dem ziemlich hybriden Signalprozessor i2920, auf schlichte analoge Operationsverstärker zurück.

Friedrich Kittler

Draculas Vermächtnis

Technische Schriften

RECLAM VERLAG LEIPZIG

ISBN 3-379-01476-1

© Reclam Verlag Leipzig 1993 (für diese Ausgabe)
Quellen- und Rechtsnachweis am Schluß des Bandes

Reclam-Bibliothek Band 1476
1. Auflage, 1993
Reihengestaltung: Hans Peter Willberg
Umschlaggestaltung: Friederike Pondelik unter Verwendung
der Computergrafik »Tanz der Silikone« von Werner Drescher
Printed in Germany
Satz: Schroth Fotosatz GmbH Limbach-Oberfrohna
Druck und Binden: Offizin Andersen Nexö Leipzig GmbH
Gesetzt aus Meridien

Inhalt