Rechnen mit Zahlen oder Rechnen mit Buchstaben

Numerische Mathematik und Mathematische Logik in der Informatik

von Dirk Siefkes

Überblick
Sind Computer unvorstellbar schnelle Rechner oder verarbeiten sie Symbole und sind damit intelligent? Die beiden Auffassungen haben sich nebeneinander entwickelt, mit merkwürdigen Verknüpfungen und zeitlichen Verschiebungen, haben aber nicht zu unterschiedlichen Realisierungen geführt. Das trägt dazu bei, dass wir beim Programmieren Mensch und Maschine so fraglos „hybridisieren". Ich gehe den alten, historisch schwer fassbaren Verstrickungen nach, um das neue Phänomen des „allgegenwärtigen" Computers und der „definierenden" Wissenschaft Informatik besser zu verstehen.

Abstract
Computers – are they unimaginably fast computing machines, or do they handle symbols and thus are intelligent? These two concepts developed separately, connected in strange ways, but on differing time lines. They did not, however, produce different realizations. This fact is part of the reason that in programming we „hybridize" man and machine without much questioning. I follow these old entanglements in order to better understand the new phenomena of the „ubiquitous" computer and its „defining" science.

***

Einleitung
Einigkeit besteht, dass die Informatik ein Kind von Mathematik und Nachrichtentechnik sei. So beginnen historische Darstellungen[1] im Allgemeinen mit Rechenverfahren und Rechengeräten, setzen mit frühen Rechenautomaten und anderen Maschinen, die durch Kode gesteuert werden, fort und gelangen so geradlinig zu den ersten Computern der 1930er und 1940er Jahre. Gelegentlich wird dabei die enge Wechselwirkung zwischen mathematischer und technischer Entwicklung betont.[2] Mit dem Aufkommen der Informatik als eigenständiger Disziplin rückt der Unterschied zwischen den Eltern in den Hintergrund. Die Entwicklung der Informatik war und ist von dauernden

1 Z.B. Petzold, Hartmut: Moderne Rechenkünstler. Die Industrialisierung der Rechentechnik in Deutschland, München 1992; Naumann, Friedrich: Vom Abakus zum Internet. Die Geschichte der Informatik, Darmstadt 2001.
2 Coy, Wolfgang: Industrieroboter, Berlin 1985.

Auseinandersetzungen geprägt, ob Maschinen oder Formalismen wichtiger seien, aber der Gegensatz wird selten thematisiert. (Laut wird nur immer die Kluft zwischen Theorie und Praxis betont, die doch quer dazu liegt und allen Wissenschaften, wenn nicht unserer Kultur überhaupt, eigen ist.) Lange Zeit wurden Programmiersprachen als Nachweis der geglückten Verschmelzung angesehen.[3] Und mit dem Durchbruch des Internet als „definierender Technologie“[4] wird der Computer samt seinen Programmen zum Medium erklärt, das weder formal noch technisch, sondern „einfach überall“ sei.

Ich möchte deswegen der Rolle des Technischen und des Formalen in der Entstehung der Informatik genauer nachgehen. Dabei schließe ich an neuere Arbeiten an: Heidi Schelhowe[5] versteht die Metamorphose des Computers von der Maschine über das Werkzeug zum Medium als einen Wandel im Gebrauch und in der Sichtweise: Wir verlieren das Maschinenhafte am Computer, als dem Menschen nicht adäquat, aus dem Blick und nehmen es damit nicht mehr wahr. Die Anfänge dieser Entwicklung haben wir in einem Interdisziplinären Forschungsprojekt „Sozialgeschichte der Informatik“ aufgearbeitet.[6] Peter Eulenhöfer[7] rückt in seiner Untersuchung das Formale in den Mittelpunkt: Informatiker benutzen Formalismen, um geistige Tätigkeiten und Computer-Abläufe in Korrespondenz zu setzen. Geistige Vorgänge werden schematisiert und dann formal beschrieben, um sie auf die Maschine bringen zu können; maschinelle Abläufe werden formal beschrieben und dann als geistige Bewegungen interpretiert, um sie verstehen zu können. Die beiden gegensätzlichen Bereiche des Menschlichen und des Maschinellen werden durch das Formale scheinbar vereint, „hybridisiert“. Dabei tritt auf jeweils charakteristische Weise ein Bereich in den Vordergrund und verdeckt den anderen mehr oder weniger. Die formalen Beschreibungen, die solches Ver-

3 Bauer, Friedrich Ludwig, Was heißt und was ist Informatik? In: IBM-Nachrichten 24, 1974, S. 333-337; Zemanek, Heinz: Was ist Informatik? In: Elektronische Rechenanlagen 13, 1971, S. 157-161.

4 Coy, Wolfgang: Defining Discipline, in: Freksa, Christian, Matthias Jantzen u. Rüdiger Valk (Hg.), Foundations of Computer Science – Potential, Theory, Cognition, Heidelberg 1997, S. 21-36.

5 Schelhowe, Heidi: Das Medium aus der Maschine. Zur Metamorphose des Computers, Berlin 1997.

6 Eulenhöfer, Peter, Dirk Siefkes, Heike Stach u. Klaus Städtler: Die Konstruktion von Hybridobjekten als Orientierungsmuster in der Informatik (Forschungsbericht des Fachbereichs Informatik 97-23, TU Berlin), Berlin 1997; Eulenhöfer, Peter, Dirk Siefkes u. Heike Stach: Sozialgeschichte der Informatik, in: FIFF-Kommunikation 2, 1998, S. 3f. u. 29-46; Eulenhöfer, Peter, Dirk Siefkes, Heike Stach u. Klaus Städtler (Hg.): Sozialgeschichte der Informatik. Kulturelle Praktiken und Orientierungen, Wiesbaden 1998; Annette Braun, Peter Eulenhöfer, Dirk Siefkes, Heike Stach, Klaus Städtler (Hg.): Pioniere der Informatik, Heidelberg 1999.

7 Eulenhöfer, Peter: Die formale Orientierung der Informatik. Zur mathematischen Tradition der Disziplin in der Bundesrepublik Deutschland. Dissertation, FB Informatik, TU Berlin 1999.

stecken ermöglichen, wurden im Projekt *Hybridobjekte* genannt, der Vorgang selbst *Hybridisierung*. Mit demselben Ansatz beschreibt Heike Stach[8] die Entwicklung der Programmierung allgemeiner als Hybridisierung verschiedener Dualismen: Geist und Materie, Lebewesen und Maschine, Text und Ding, bewegt und bewegend. Die Art und Weise, wie Informatiker hybridisieren, ändert sich mit der Zeit, sie hängt von den kulturellen und technischen Orientierungen innerhalb und außerhalb der Wissenschaft ab und beeinflusst die Produkte ihrer Arbeit.

Die Idee, dass Menschen und Maschinen nicht gegensätzlich, sondern verwandt seien, dass der Mensch eine Maschine mit Seele sei oder die Maschine zum Leben gebracht werden könne, wurzelt tief in unserer Kultur.[9] Und die Verbreitung des Computers hat solche Vorstellungen ebenfalls verbreitet und die Diskussion darum verschärft. Gemäß anerkannten Theorien aus Biologie und Psychologie beruht lebendige Entwicklung auf der Wechselwirkung zwischen schematisch sich wiederholenden Vorgängen im Lebewesen und ihrer „Interpretation" in der Umgebung. Die Befürworter z.B. der „Künstlichen Intelligenz" oder der „Sozionik" setzen auf, wie ich meine, unzulässige Weise diese „Schemata" mit Computerprogrammen gleich und vergessen dabei die „Interpretation" oder denken sie sich mitprogrammiert.[10]

Unter der Hand ist damit neben dem Technischen und Formalen das Menschliche in der Informatikgeschichte aufgetaucht. Die Informatik als Sprössling dreier Eltern? Wohl kaum. Immer wieder wird betont, wie wichtig Psychologie, Linguistik, Soziologie oder andere Geistes- und Sozialwissenschaften für die Informatik seien,[11] aber es gibt keine einzelne Wissenschaft, die „das Menschliche" (oder „das Geistige und Soziale" oder „das Kulturelle") vertritt. So haben es die Vertreter der beiden juristischen Eltern leicht, unerwünschte Verwandte von der Kinderstube fernzuhalten. Mathematiker machen sich selten Gedanken über die kulturellen Verwicklungen ihrer Formalismen; so bestreiten Theoretische Informatiker die Notwendigkeit einer über das Mathematische hinausgehenden Theorie der Informatik.[12] Und Ingenieure überlassen solche Fragen gern ihren Kollegen aus Philosophie, Soziologie oder

8 Stach, Heike: Zwischen Organismus und Notation. Zur kulturellen Konstruktion des Computer-Programms, Wiesbaden 2001.

9 Vgl. Bammé, Arno, Günter Feuerstein, Renate Genth, Eggert Holling, Renate Kahle u. Peter Kempin: Maschinen-Menschen Mensch-Maschinen, Reinbek b. Hamburg 1983.

10 Vgl. Siefkes, Dirk: Die Rolle von Schemata in der Informatik als kultureller Entwicklung (Forschungsbericht des Fachbereichs Informatik 99-6, TU Berlin), Berlin 1999.

11 Vgl. Coy, Wolfgang, Frieder Nake, Jörg-Martin Pflüger, Arno Rolf, Jürgen Seetzen, Reinhard Stransfeld u. Dirk Siefkes (Hg.): Sichtweisen der Informatik, Wiesbaden 1992; Floyd, Christiane, Reinhard Budde, Reinhard Keil-Slawik u. Heinz Züllighoven (Hg.): Software Development and Reality Construction, Heidelberg 1992; Friedrich, Jürgen, Thomas Herrmann, Max Peschek u. Arno Rolf (Hg.): Informatik und Gesellschaft, Heidelberg 1995.

12 Vgl. Coy et al. (wie Anm. 11), Nake, Frieder, Arno Rolf u. Dirk Siefkes (Hg.): Informatik – Aufregung zu einer Disziplin. Tagung Heppenheim 2001 (Uni Hamburg, FB Informatik, Bericht 235), Hamburg 2001 (im Internet: http://tal.cs.tu-berlin.de/siefkes/Heppenheim).

Ethik und den Politikern.[13] Zwar gibt es – einmalig in den Ingenieurwissenschaften! – ein Fachgebiet „Informatik und Gesellschaft“,[14] dieses wird jedoch von den „eigentlichen“ Informatikern meist als Öffnung zum Druckausgleich nach draußen betrachtet, nicht als Verbindung zu anderen Wissenschaften. Wenn aber Hybridisierung so zentral für die Informatik ist, dann scheint es angebracht, die Rollen, die mathematische Formalismen als Mittel der Hybridisierung gespielt haben und spielen, näher anzusehen.

Rechnen

Lange Zeit bestand die Entwicklung der Mathematik in der Entwicklung von Rechenverfahren und allgemeiner von Formalismen zur Formulierung solcher Verfahren. Zu berechnen war der Gang der Gestirne, die Fläche von Äckern, der Inhalt von Gefäßen. Probleme und Verfahren wurden komplizierter, die Korrektheit der Verfahren war zu beweisen, die Beweise wurden aufwändiger. Aber was ein Rechenverfahren ist, wurde nicht hinterfragt. Das änderte sich erst im 20. Jahrhundert. Auf dem Mathematikerkongress in Paris 1900 nannte David Hilbert als eine der zehn wichtigsten Aufgaben der Mathematik, zu klären ob es ein allgemeines Verfahren gebe, mit dem man alle mathematischen Probleme lösen könnte. Man sollte ein solches Verfahren angeben oder beweisen, dass es keines gibt. Das war eine Aufgabe für die mathematische Grundlagenforschung, die in dieser Zeit mit dem Ziel entstand, mathematische Grundbegriffe wie Menge, Relation, Funktion oder Beweis zu präzisieren. Da ‚Beweis‘ hierbei eine zentrale Rolle spielte, etablierte sich das Gebiet als „Mathematische Logik“. Um Hilberts Frage negativ beantworten zu können, müsste man definieren, was Berechnungsverfahren oder, mathematisch ausgedrückt, berechenbare Funktionen sind. Beweise kann man nur für mathematisch präzise Begriffe führen, nicht für intuitiv gebrauchte, so selbstverständlich der Gebrauch auch ist. Hilbert sagt davon nichts; er spricht von „finiten Verfahren“, das scheint ihm klar genug.

Über 20 Jahre vor dem Hilbertschen Vortrag hatte Richard Dedekind „primitive Rekursion“ benutzt, um Funktionen über den natürlichen Zahlen zu definieren. So führte er sukzessive Addition, Multiplikation, Exponentiation und weitere wichtige Funktionen auf die Nachfolgerfunktion zurück. Guiseppe Peano gründete darauf sein Axiomensystem für die natürlichen Zahlen, das nur Formeln zur Definition von Null und Nachfolger und für die vollständige Induktion umfasste. Lange Zeit galt daher ‚primitiv rekursiv‘ als geeignetes Verfahren zur Definition berechenbarer Funktionen. In den 1920er Jahren entwickelte Thoralf Skolem damit seine „berechenbare Zahlentheorie“. Erst 1928 fand der Hilbert-Schüler Wilhelm Ackermann eine Funktion, die nicht primitiv rekursiv, aber berechenbar ist. Sein Verfahren

13 Vgl. MacKenzie, Donald u. Judy Wajcman (Hg.): The Social Shaping of Technology, Buckingham 1999.

14 Vgl. Friedrich et al. (wie Anm. 11).

wurde schnell verallgemeinert, und immer größere Klassen berechenbarer Funktionen kamen ans Licht. Konnte es eine umfassende Definition geben, die dem ausufernden Prozess ein Ende setzte?

In der ersten Hälfte der 1930er Jahre wurden gleich mehrere solcher Definitionen vorgeschlagen. In den USA hatte Alonzo Church schon länger mit seinem „λ-Kalkül" (griech. lambda) an einer Grundlegung der Zahlentheorie gearbeitet.[15] Mit geeigneten Auswertungsregeln waren die im Kalkül definierbaren Funktionen berechenbar, und Church und seine Schüler konnten alle wichtigen Funktionen in dem Kalkül definieren. Daraufhin vermutete Church 1934, dass das für alle berechenbaren Funktionen gehe. Er trug die These 1935 auf einer Konferenz vor, allerdings für eine andere gleichwertige Definition: Kurt Gödel, aus Wien in die USA emigriert, hatte 1934 in einer Vortragsreihe die Funktionen, die durch Auswerten gewisser verallgemeinerter primitiv rekursiver Gleichungssysteme gewonnen werden, „rekursiv" genannt. So entstand die heute so genannte *Churchsche These:* Alle berechenbaren[16] Funktionen sind rekursiv. Wenig später definierte Churchs Schüler Stephen Kleene die „μ-rekursiven" Funktionen (griech. my), die mit primitiver Rekursion und einem Auswahloperator auskommen, und die „allgemein rekursiven", für die er beliebige rekursive Gleichungssysteme auf geschickte Weise auswertete.[17] Alle diese Definitionen stellten sich als äquivalent heraus; das heißt, die Funktionenklassen waren gleich. Das war ein starkes Argument für die Churchsche These. Trotzdem scheint, nach Martin Davis[18], Gödel noch nicht überzeugt gewesen zu sein. Alle diese Definitionen beruhten auf bekannten mathematischen Berechnungsverfahren; konnte es nicht ganz andere geben, die auf bisher ungeahnte Weise zu rechnen erlaubten?

Im Frühjahr 1936 kam eine solche ganz andere Definition aus England. Alan Turing beschrieb Rechnen als Schritte einer gedachten Maschine, die auf einem beliebig langen Band hin- und herläuft und Symbole druckt, verändert oder löscht.[19] Nichts anderes tut ein Mathematiker, der mit Bleistift und Papier rechnet, argumentierte Turing, der sich von Kind an für Maschinen begeistert hatte.[20] Und wirklich ist es nicht schwer, mathematische Verfahren wie Aus-

15 30 Jahre später entstand daraus die Programmiersprache LISP.

16 Früher sagte man auch „effektiv berechenbar", um zu betonen, dass es um effektiv ausführbare Rechenverfahren (Algorithmen) geht, nicht um Heuristiken.

17 Daraus wurden später funktionale algorithmische oder Programmiersprachen wie FP (Backus), LCF (Scott/Milner), REC (Siefkes, Moschowakis), OPAL (Pepper).

18 Davis, Martin: Why Gödel Didn't Have Church's Thesis, in: Information and Control 54, 1982, S. 3-24.

19 Turing, Alan: On Computable Numbers with an Application to the Entscheidungsproblem, in: Proceedings of the London Mathematical Society 42, 1936, S. 230-265 – Deutsch in Dotzler, Bernhard, u. Friedrich Kittler (Hg.): Alan Turing, Intelligence Service, Berlin 1987, S. 18-60.

20 Hodges, Andrew: Alan Turing. The Enigma, Berlin 1989.

wertungen im λ-Kalkül auf „Turingmaschinen“ (wie Church sie gleich 1937 in einer Besprechung der Arbeit Turings nannte) zu simulieren.

Es soll Gödel gewesen sein, der sofort die Bedeutung des Ansatzes von Turing erkannte. Er hatte 1931 bewiesen, dass die Zahlentheorie nicht axiomatisierbar ist, und dafür mathematische Formeln und Beweise mühsam mit Hilfe von Primzahlzerlegungen in natürliche Zahlen kodiert, ähnlich wie man es heute in der Kryptographie tut. Er musste kodieren, weil die Berechnungsverfahren, die er kannte, nur auf Zahlen anwendbar waren. Wer wie Turing Rechnen auf Symbolmanipulation zurückführte, konnte sich das Kodieren in Zahlen ersparen. Nichtsdestoweniger ließ Turing seine Maschinen (reelle) Zahlen berechnen. Erst 1946 führte William van Orman Quine Wörter statt Zahlen als Struktur zum Rechnen ein.[21]

Tatsächlich war der Unentscheidbarkeitsbeweis in Turings Arbeit, für den er seinen Formalismus eigentlich entwickelt hatte, weniger kompliziert als der entsprechende bei Church. Davis vermutet, dass erst Turings Definition und die Tatsache, dass sie zu den anderen äquivalent ist, Gödel von der Churchschen These überzeugt hat. Turing gewann seine Maschinen nicht aus spezifischen mathematischen Methoden für spezifische mathematische Probleme; Turingmaschinen führen die elementaren Operationen aus, die in jedem Rechenverfahren vorkommen. Allgemeiner kann man ‚berechenbar‘ nicht definieren. Heutzutage wird daher die Churchsche These oft so formuliert (*Turingsche These)*: Alle effektiv lösbaren Probleme sind mit Turingmaschinen lösbar.

Der amerikanische Logiker Emil Post hatte schon in den 1920er Jahren einen Berechnungsformalismus entwickelt, der mit Turings praktisch identisch ist: Ein Arbeiter läuft an einer langen Reihe von Schachteln hin und her und setzt oder löscht darin Marken nach einem vorgegebenen Programm – eine Turingmaschine mit nur einem Symbol. Publiziert hat er die Idee erst 1936, als Church ihm Turings Arbeit ankündigte.[22] Inzwischen ist die Church-Gödel-Turingsche These allgemein akzeptiert. Meist werden drei Begründungen gegeben:

(1) Äquivalenz: Alle Vorschläge zur mathematischen Präzisierung der Berechenbarkeit – es gibt Dutzende – haben sich als äquivalent erwiesen.

(2) Robustheit: Man kann alle diese Definitionen stark verändern, ohne dass sie „Berechnungskraft“ verlieren oder gewinnen. So kann man Turingmaschinen auf einem, auf zwei, auf einem halben (das heißt, nur nach einer Seite unbeschränkten) oder auf 17 Bändern rechnen lassen; ebenso auf 2-dimensionalen Rechenfeldern oder im n-dimensionalen Raum; man

21 Quine, Willard van Orman: Concatenation as a Basis for Arithmetic, in: Journal of Symbolic Logic 11, 1946, S. 105-114.

22 Post, Emil L.: Finite combinatory processes – Formulation 1, in: Journal of Symbolic Logic 1, 1936, S. 103-104 – auch in Davis, Martin (Hg.): The Undecidable, New York 1965, S. 289-291.

kann sie statt Buchstaben ganze Wörter lesen und schreiben lassen; man kann sie deterministisch oder nichtdeterministisch rechnen lassen – die Probleme, die man damit lösen kann, ändern sich nicht.

(3) Universalität: Man kann in allen diesen Formalismen „universelle" Funktionen definieren, aus denen man durch Variation von Parametern alle berechenbaren Funktionen erhält. Z.B. gibt es universelle Turingmaschinen, die beliebige Turingmaschinen simulieren, wenn sie deren Programm als Eingabe erhalten.

Die Äquivalenz fiel schon 1936 ins Gewicht, die anderen Begründungen kamen später. Entscheidend war wohl 1936 die Gleichwertigkeit der mathematischen Beschreibungen mit der maschinenhaften. Rechnen ist nicht das Ausführen mathematischer Operationen nach mathematischen Methoden, sondern das mechanische Manipulieren von Symbolen nach vorgegebenen Regeln.

Rechenautomaten und Hybridisierung

Nachdem Maschinen im 19. Jahrhundert die Fabriken erobert hatten, durchdrang Maschinendenken in der ersten Hälfte des 20. Jahrhunderts alle gesellschaftlichen Bereiche, wie Bettina Heintz schön beschreibt.[23] Frederick Taylor begründete 1912 die Zerlegung von Arbeitshandlungen in kleinste Schritte, die dann von Maschinen oder Menschen ausgeführt werden können, mit „wissenschaftlichen Prinzipien" –, genau wie Turing 1936 die „Maschinisierung der Kopfarbeit"[24] psychologisch rechtfertigte, während Post direkt Taylors Vision folgte. Die Massen bejubelten die maschinenhaften Aufmärsche der Armeen faschistischer (und anderer) Staaten wie die maschinenhaften Bewegungen der Revuegirls. Zur Olympiade in Berlin 1936 wurde die „maschinenhafte Präzision" der sportlichen Körper verherrlicht. Charlie Chaplin lehnte sich 1936 mit dem Film *Modern Times* gegen die Maschinisierung des Lebens auf. Aber im selben Jahr machte Turing die Maschine in der Mathematik salonfähig, indem er sie abstrakte Berechnungen ausführen ließ.

Ebenfalls 1936 entwarf Konrad Zuse in Deutschland den ersten Computer im heutigen Sinne des Wortes. Er war Bauingenieur und wollte die ermüdenden numerischen Berechnungen, die er zu machen hatte, von einer Maschine ausführen lassen. Er baute daher einen „Rechenautomaten", der numerische Ausdrücke automatisch auswertet; die „Rechenpläne", die die Auswertung steuern, sind von den Rechenbögen der großen Rechenbüros abgeguckt. Mit ihnen hybridisiert Zuse also die Arbeit der Rechenkräfte und das Rattern der Maschine. Beim Entwurf der Schaltungen erfand Zuse die Aussagenlogik neu und erweiterte seine Rechenpläne um logische Bedin-

23 Heintz, Bettina: Die Herrschaft der Regel, Frankfurt a.M. 1993, Kap. 4.

24 Nake, Frieder: Informatik und die Maschinisierung von Kopfarbeit, in: Coy et al. (wie Anm. 11), S. 181-201.

gungen, die er aber erst 1950 technisch realisieren konnte. Auch seine weiteren Maschinen waren ausschließlich für numerische Berechnungen angelegt. Er träumte aber von Anfang an von beliebigen Anwendungen wie Schachspielen oder Verwaltungsaufgaben[25] und unterschied ganz früh zwischen „algebraischen" und „logistischen" Rechenautomaten.[26] Während der erzwungenen Arbeitspause zu Kriegsende entwarf er den „Plankalkül", eine universelle Programmiersprache. Er wusste nichts von den Berechnungsformalismen der Logiker und auch nichts von den amerikanischen Computern, die in der Zwischenzeit entwickelt worden waren.[27]

Die erste allgemeine Beschreibung einer solchen Maschine gab 1945 der ungarische Mathematiker und Chemieingenieur John von Neumann. Als Jude hatte er sich 1933 – schon berühmt für seine Leistungen in vielen mathematischen Gebieten (einschließlich der Mathematischen Logik) und deren Anwendungen – endgültig in den USA niedergelassen und hatte gleich bei Kriegsausbruch angeboten, mit seinen wissenschaftlichen Fähigkeiten zur Verteidigung gegen Nazi-Deutschland beizutragen. So war er Berater in wichtigen militärischen Forschungsvorhaben geworden, insbesondere beim Atombombenprojekt in Los Alamos, und war auf der Suche nach Rechenleistungen auf eine Gruppe gestoßen, die an dem Computer ENIAC arbeitete. In einem Bericht[28] beschreibt er ein Nachfolgeprojekt der Gruppe, die EDVAC, und stellt dabei die Prinzipien auf, nach denen bis heute unsere „von-Neumann-Computer" gebaut werden. Als Mathematiker geht es ihm um numerische Anwendungen, sein „large-scale automatic computing device" ist also wie bei Zuse ein maschinelles Abbild eines Mathematikers mit „organs" für Arithmetik, Steuerung, Speicherung und Ein/Ausgabe. Ebenso vertraut ist ihm aus der Ausbildung und aus der Arbeit in praktischen Projekten die Sicht des Ingenieurs; technische Überlegungen nehmen einen breiten Raum ein und bestimmen wichtige Entwurfsunterscheidungen. Die Hybridisierung von Mathematiker und Maschine wird daher nicht problematisiert, Zahldarstellungen werden direkt zu Speicherkonfigurationen, numerische Operationen zu „Kodes" für Änderungen des Speicherinhalts. Weitergehende Anthropomorphismen benutzt er nicht, auch nicht die Berechenbarkeitstheorie der Logiker, an der er selbst früher gearbeitet hat – insbesondere nicht die Arbeit Turings, obwohl er sie 1938 begutachtet hat. Sein Computer soll numerische Berechnungen, nicht mathematische Ableitungen ausführen.

25 Zuse, Konrad: Der Computer – mein Lebenswerk, Berlin 1993.

26 Ders.: Rechenplangesteuerte Rechengeräte für technische und wissenschaftliche Anwendungen. Manuskript, 19 S., Berlin 1943.

27 Mehr dazu in dem schönen Band Hellige, Hans-Dieter (Hg.): Geschichten der Informatik. Visionen, Paradigmen, Leitmotive, Heidelberg 2004.

28 Neumann, John von: First Draft of a Report on the EDVAC, in: Taub, Abraham H. (Hg.): John von Neumann. Collected Works, Bd. 5, Oxford 1963, S. 1-31.

Mit einer ganz anderen Hybridisierung, die er ausdrücklich als „Neuronen-Analogie“ einführt, verknüpft von Neumann dagegen die Arbeitsweise des Gehirns und des Rechners auf der untersten technischen Ebene. Neurophysiologie und behavioristische Denkweise sind ihm aus Diskussionen um eine neue Wissenschaft bekannt, die Norbert Wiener später Kybernetik nennt[29] und in der „control and communication in the animal and in the machine“ einheitlich behandelt werden sollen. Von Neumann zitiert hier – die einzige Referenz in dem Papier – die Arbeit von Warren McCulloch und Walter Pitts,[30] einem Neurophysiologen und einem Mathematiker, die neuronale Strukturen zu logischen Netzen abstrahieren und beweisen, dass man genau diese mit aussagenlogischen Formeln mit einem Zeitparameter beschreiben kann. Auf die gleiche Weise abstrahiert von Neumann von den Eigenschaften konkreter Schaltungen und macht so neuronale Strukturen und technische Schaltungen vergleichbar. Damit kann er für Auswahl und Aufbau der Schaltelemente neurophysiologisch wie technisch argumentieren, in fließenden Übergängen. Sein maschineller Computer ist also so universell[31] wie der menschliche, er kann im Prinzip dasselbe, ist nur um Größenordnungen weniger komplex. Die Beschränkung auf *einen* Prozessor, den berühmten „Von-Neumann-Flaschenhals“, begründet er technisch; man kann aber spekulieren, ob ihn nicht unbewusst damalige kybernetische Vorstellungen vom Gehirn als einer Maschine und philosophische vom Denken als einem Bewusstseinsstrom ebenso beeinflusst haben.

Von Neumann benutzt also logische Netze als Formalismus, um menschliches Gehirn und Computer auf der elementarsten Ebene zu hybridisieren. Die selbstverständliche Hybridisierung von menschlichem und maschinellem Rechnen ist davon völlig getrennt. Deswegen zitiert er Turing nicht, der die Ebenen nicht unterscheidet.[32] In anschließenden Arbeiten 1946-47 mit anderen wird die stillschweigende Gleichsetzung von numerischem Algorithmus und Maschine-Kode problematisiert, und Flussdiagramme und symbolische Adressierung werden als erste Brücken in die entstandene Lücke gestellt. Man kann die frühe Geschichte der Programmierung – vom Kodieren über die ersten höheren Programmiersprachen bis hin zu funktionalen Programmiersprachen – als das Bemühen verstehen, den Weg zwischen menschlichem Rechnen und rechnender Maschine in beiden Richtungen gangbar zu machen.[33] Je mächtiger die Maschine wird, desto größere Teile

29 Wiener, Norbert: Cybernetics, Cambridge/Mass. 1948 – Deutsch: Kybernetik. Regelung der Nachrichtenübertragung im Lebewesen und in der Maschine, Düsseldorf 1963.

30 McCulloch, Warren u. Walter Pitts: A Logical Calculus of the Ideas Immanent in Nervous Activity, in: Anderson, James u. Edward Rosenfeld (Hg.): Neurocomputing, Cambridge/Mass. 1943, S. 18-28.

31 S. oben die Begründungen zur Churchschen These.

32 Vgl. dazu Heintz (wie Anm. 23), Kap. 6.

33 Vgl. dazu Eulenhöfer et al. 1997 (wie Anm. 6); dies. 1998 (wie Anm. 6), Eulenhöfer (wie Anm. 7), Stach (wie Anm. 8); Siefkes (wie Anm. 10).

des Weges übernimmt sie, desto mehr tritt sie gleichzeitig – nur scheinbar paradox – in den Hintergrund, verschwindet durch Hybridisierung hinter menschlichen Fähigkeiten. Auch wenn Rechnen bald zu fast beliebigen geistigen Fähigkeiten und der Computer zur Datenverarbeitungsanlage verallgemeinert werden, bleibt vielen Informatikern als Erbe der Herkunft aus Mathematik und Ingenieurwissenschaft der unerschütterliche Glaube, die Computerisierung eines Lebens- (Arbeits-, Spiel-) Bereiches bringe, wenn sie nur gut gemacht werde, eine Verbesserung, aber keine grundsätzliche Veränderung. Getreu der Tradition ihrer Elterndisziplinen meinen sie, Informatiker müssten nur technisch und formal sauber arbeiten, menschlich seien sie ebenso wenig involviert wie beim Entwerfen eines mathematischen Formalismus oder einer Telefonanlage. Ich halte das für einen Irrglauben. Mathematiker hybridisieren geistige Vorgänge mit formalen Schritten, Nachrichtentechniker hybridisieren Kommunikationsprozesse mit dem Austausch technischer Signale. Informatiker kombinieren beides und potenzieren so die Wirkung. Sie hybridisieren mit ihrer Arbeit geistig-körperliche menschliche Tätigkeiten, also Menschen als ganze, mit maschinellen Abläufen; sie greifen viel stärker in geistige und soziale Zusammenhänge ein als Mathematiker und Nachrichtentechniker. Wenn sie das nicht beachten, handeln sie blind. Natürlich können sie nicht überall kompetent sein. Aber sie können sich auf Projekte beschränken, bei denen sie sich gemeinsam mit „Nutzern“ und zuständigen Wissenschaftlern engagieren. Dadurch gäbe es nicht weniger Arbeit für Informatiker, aber mehr sinnvolle.[34]

Von Neumann und Turing

Erst 1948 setzt von Neumann seine Rechenautomaten mit Turings Maschinen in Beziehung. Einen Vortrag auf der ersten Kybernetik-Konferenz[35] beginnt er mit der Feststellung, Automaten und natürliche Organismen seien funktional so ähnlich, dass Mathematiker und Naturwissenschaftler für ihre Untersuchungen dieser Objekte voneinander lernen könnten. Gegenwärtig scheitere das daran, dass natürliche Organismen viel komplexer seien als Automaten und dass es für derart komplexe Gebilde noch keine Theorie gebe. Es könne weiterhelfen, Automaten sich selber reproduzieren zu lassen und diese Evolution zu studieren. Dazu fasst er im letzten Abschnitt des Papiers seine Automaten als Turingmaschinen auf. Wie diese ihre eigenen Beschreibungen kopieren und ausführen können (s.o. „Universalität“), sollen jene ihre Beschreibungen kopieren und sich so vermehren. Dabei verschwimmt die Unterscheidung zwischen Maschinen und Beschreibungen,

34 Vgl. dazu den Schluss von Siefkes, Dirk: Sinn im Formalen? Wie wir mit Menschen und Formalismen umgehen, in: Coy et al. (wie Anm. 11), S. 97-114.

35 Neumann, John von: The General and Logical Theory of Automata, in: Taub (wie Anm. 28), S. 288-328 – Deutsch: Die allgemeine und logische Theorie der Automaten. Kursbuch 8, Frankfurt a.M. 1967.

die er vorher genau diskutiert; unklar bleibt auch, wie so komplexere Automaten entstehen sollen: durch Fehler beim Kopieren, aber durch welche Selektion?

Turing selbst hat 1936-1938 am MIT gearbeitet und bei Church in mathematischer Logik promoviert. Ein Angebot für eine neue Stelle bei von Neumann schlug er wegen der politischen Lage aus und kehrte nach England zurück.[36] Dort arbeitete er während des Krieges für den Geheimdienst an der Entschlüsselung des Kodes der Deutschen Wehrmacht für die U-Boote. Die Maschinen, die dafür gebaut wurden, würde man heute „special purpose computer" nennen. Nach dem Krieg war er in der staatlichen Computerentwicklung angestellt. Seine Vorbilder waren die EDVAC von Neumanns und andere amerikanische „number-cruncher", nicht seine eigenen Papier-und-Bleistift-Maschinen von 1936. 1948 geriet er in Konflikt mit seinen Vorgesetzten, weil die Presse nach einem Interview mit ihm die noch unfertigen Produkte als Wundermaschinen, „giant brains", der Öffentlichkeit vorstellte. Er schrieb eine Rechtfertigung, die er 1950 überarbeitete und unter dem Titel *Computing Machinery and Intelligence* veröffentlichte.[37] Um die strittige Frage „Can machines think?" zu klären, schlägt er zwei „imitation games" vor, bei dem (1) ein Mann und eine Frau, (2) eine Maschine und ein Mensch daran unterschieden werden sollen, wie sie auf Fragen schriftlich antworten. „Wir wollen eine Maschine intelligent nennen, wenn sie im Spiel (2) den Frager (‚who may be of either sex') ebenso oft täuschen kann wie ein Mann im Spiel (1)", ist seine Definition. Turing sagt voraus, dass bis zum Ende des Jahrhunderts Maschinen solche Intelligenz erreicht haben werden. Die 50 Jahre sind um, und rund um die Welt spielen Computer-Freaks Turing-Tests – vom Typ (2), nicht vom Typ (1) – auf Konferenzen und im Internet. Über die Ergebnisse ist man sich nicht einig. Für mich sind zwei andere Punkte in Turings Papier bemerkenswert: Wie von Neumann zur gleichen Zeit glaubt Turing nicht, dass man intelligente Maschinen direkt konstruieren könne. Das Programmieren sei zu aufwändig. (Ähnlich spekuliert von Neumann, die Beschreibung neuronaler Strukturen sei so aufwändig, dass für komplexere Strukturen die kürzeste Beschreibung „die Struktur selbst" sein könne.) Deswegen könne man intelligente Maschinen nur durch Erziehung gewinnen. Während bei von Neumann an der Evolution maschineller Intelligenz nur Automaten beteiligt zu sein scheinen, die in großer Zahl durcheinanderschwimmen, entwickeln sich Turings Maschinen unter menschlichen Strafen und Belohnungen. Beide gehen auf den Lernprozess nicht näher ein.

Von Neumann entwickelt seine Vision automatischer Intelligenz nicht für die Maschinen, an denen er arbeitet, sondern für Turingmaschinen; er denkt theoretisch, nicht praktisch. Turing dagegen – das ist das zweite Be-

36 Hodges (wie Anm. 20).

37 Turing, Alan: Computing Machinery and Intelligence, in: Mind 59, 1950, S. 433-460 – Deutsch in Dotzler/Kittler (wie Anm. 19), S. 148-182.

merkenswerte – schreibt ausdrücklich über reale Computer. Denen, nicht seinen Papiermaschinen, traut er Intelligenz zu. Deswegen wohl wird sein Papier, und nicht das von von Neumann, als der Ursprung der Künstlichen Intelligenz (KI) angesehen. Wie er sich die Programmierung der Zöglinge vorstellte, ob als numerische oder symbolverarbeitende, wissen wir nicht.

Zahlen und Buchstaben in der Informatik

Für die symbolische KI dagegen beruht Intelligenz auf der Verwendung von Symbolen, und Computer sind symbolverarbeitende Maschinen. In den 1950er Jahren wurden erste Programme entwickelt, die „wie Menschen" aus (z.B. geometrischen) Aussagen logische Schlüsse ziehen können sollen. Als der Mathematiker John McCarthy 1960 diese Ansätze erweitern und vereinheitlichen wollte, verwarf er FORTRAN als ungeeignet und entwickelte eine Sprache, der nicht Zahlen und numerische Funktionen, sondern Listen und Listenoperationen zugrundeliegen. Seine Gruppe machte daraus die Programmiersprache LISP, die für viele Jahre die beherrschende Sprache zur Textverarbeitung darstellte und bis heute in der KI eine wichtige Rolle spielt. LISP-Programme sind keine Turingmaschinen; die arbeiten auf unstrukturierten Symbolfolgen. Aber McCarthy hatte Turings Arbeit[38] studiert und daraus die Idee gewonnen, dass Computer nicht bloß große Rechenautomaten sind.[39]

In den letzten zehn Jahren sind symbolverarbeitende Programme in der KI weitgehend durch „(künstliche) neuronale Netze" verdrängt worden. Das sind aus elementaren Schaltungen zusammengesetzte Netze, die sich auf „Belohnung oder Bestrafung" hin „selbst ändern" und so ein vorgegebenes (Ein/Ausgabe-) „Verhalten" stochastisch „lernen". Sie sind für bestimmte Anwendungen symbolischen KI-Programmen weit überlegen und spielen z.B. in der Mustererkennung eine große Rolle. Hier werden, wie in von Neumanns Neuronenanalogie, neurophysiologische Vorgänge und Abläufe in elektronischen Schaltungen mit Hilfe Mathematischer Logik und Wahrscheinlichkeitstheorie hybridisiert. Wieviel „Intelligenz" dabei wirklich erzielt wird, ist strittig. Aber niemand käme mehr auf die Idee, wie Turing Computerprogrammen, die „rechnen" – also Zahlen verarbeiten –, Intelligenz zuzuschreiben. Meist werden heute – in der neuronalen wie in der symbolischen KI – der Entwicklung „lernender Maschinen" explizite Lerntheorien aus Neurophysiologie oder Kognitiver Psychologie zugrundegelegt – merkwürdigerweise nicht aus der Entwicklungspsychologie, obwohl dort beginnend mit Piaget und Vygotsky die Entwicklung kindlicher Intelligenz im Wechselspiel zwischen biologischen Prozessen und sozialen Vorgängen gründlich untersucht wird.[40]

38 Turing (wie Anm. 19).
39 Stach (wie Anm. 8).
40 Siefkes (wie Anm. 10).

Auch in der Theoretischen Informatik hat sich Turing nicht unumstritten durchgesetzt. In den meisten Lehrbüchern werden Turingmaschinen als elementares Modell des Computers eingeführt, obwohl das fürs Programmieren wenig hilfreich ist, weil sie ganz anders arbeiten. In der Strukturellen Komplexitätstheorie werden Turingmaschinen als Berechnungsmodell benutzt, weil sie für die genaue Analyse von Problemen und für Beweise am flexibelsten und bequemsten sind; deswegen hatte Turing selbst sie eingeführt. In der Theorie der Formalen Sprachen hat man jahrzehntelang mit Turingmaschinen gearbeitet, weil man unter Chomskys linguistischem Einfluss Computerprogramme als Folgen von Symbolen und nicht als (Folgen von) Formeln bzw. Termen ansah. In der Theorie effizienter Algorithmen dagegen werden statt Turingmaschinen Registermaschinen benutzt. Man kann sie als über beliebigen Datenstrukturen arbeitend ansehen, so wie man das heute in der Softwarespezifikation mit realen Computern macht. Registermaschinen liefern daher realistischere Ergebnisse für die Analyse als Turingmaschinen. Mathematisch gesehen sind beide Modelle gleichwertig, weil man sie wechselseitig simulieren kann. Nur die Aufwände variieren, je nachdem, wie viel man in die Datenstruktur der Registermaschine hineinsteckt.

Heute scheint die Unterscheidung zwischen „numerischen" und „logistischen" Computern, zwischen Zahl- und Symbolverarbeitung überholt. Wir manipulieren beim Programmieren nicht nur Zahlen oder Symbole, sondern beliebige „Daten", und modellieren Probleme in „Datenstrukturen". In der Objektorientierten Programmierung werden daraus beliebige Systeme von „Objekten", die miteinander kommunizieren und agieren wie Lebewesen. Im Netz treiben solche Objekte als „Agenten" ihr Spiel. Auf dem Bildschirm lassen sich solche Welten graphisch darstellen und manipulieren. Bilder sind suggestiv. Deswegen meinen wir – Entwickler wie Benutzer –, auf dem Bildschirm die Wirklichkeit zu haben.[41] Technik und Formalismen liefern so mächtige Werkzeuge fürs Programmieren, dass die Maschine und die Symbole dabei verschwinden. Wir hybridisieren, ohne es zu merken. Das erleichtert die Arbeit ungemein. Je bequemer aber Programmieren wird, je weniger wir uns durch Technik oder Formalismen eingeengt fühlen, desto leichter verfallen wir dem alten Irrglauben, Probleme durch bloßes Programmieren „lösen", dass heißt zum Verschwinden bringen zu können. Weil uns die neuen Objekte der Programmierung so natürlich vorkommen, scheint Softwareentwicklung per se menschengerecht zu sein.

Von Neumann oder Turing?

Computer sind im Wesentlichen nach wie vor die Maschinen, die von Neumann 1945 beschrieben hat. Sie können aber Symbole manipulieren, also

41 Ein schönes Beispiel sind manche Beiträge zur „virtuellen Lehre"; vgl. z.B. Schinzel, Britta, Jörg Busse u. Dirk Siefkes: Bildung und Computer. Themenschwerpunkt FIfF-Kommunikation 1/01, Berlin 2001.

sind sie intelligent[42] und können alles.[43] Aber wie soll das zusammenhängen? Wie eh und je befördern sich mathematische und technische Entwicklung wechselseitig, und ‚früher' und ‚später' sind kaum zu benennen. Fasziniert von Maschinen im menschlichen Alltag sah Turing[44] das Maschinelle in den kalkülisierten Berechnungen der Logiker und realisierte das, was er sah, in einer formalen Maschine. Dass diese Maschinen universell sind, bewies er, indem er sie kodierte und ihren eigenen Kode manipulieren ließ. Inspiriert von den schematisierten Vorgängen in Rechenbüros sah von Neumann[45] das Maschinelle in den numerischen Berechnungen der Mathematiker, Physiker und Chemiker und verbesserte und abstrahierte mit dem, was er sah, die schon existierenden realen Maschinen. Dass seine Maschinen alles können, begründete er zunächst[46] mit einer Analogie zum menschlichen Gehirn und erst später[47] mit Bezug auf Turing, in dem er reale Maschinen durch formale Beschreibungen ersetzte und sie ihren eigenen Kode manipulieren ließ, so dass sie sich selbst reproduzieren und so entwickeln können. Motiviert von seinen Erfahrungen mit kode-brechenden realen Maschinen arbeitete Turing zur gleichen Zeit an der Entwicklung von Computern. Dass sie zur Intelligenz erziehbar seien, begründete er mit den erstaunlichen Fähigkeiten der von-Neumann-Maschinen. Die heutigen Vorstellungen von Wissenschaftlern oder Politikern, von Technikverehrern oder -verächtern, darüber, was Computer können und nicht können, haben ihren Ursprung in diesen alten Verstrickungen von Mythos und Wirklichkeit, von technischen Fakten und formalen Theorien und menschlichen Visionen.

Das heißt nicht, dass Turing und von Neumann zusammen den Computer erfunden hätten. In beiden Personen spiegelt und verdichtet sich eine Vielzahl von Strömungen ganz unterschiedlicher Art. Dass es für eine grundlegende Erfindung mehr braucht als ein oder zwei Genies, zeigt der Vergleich mit Konrad Zuse. Zuse hat seit 1936 technische und formale Ideen entwickelt und umgesetzt, die denen Turings bzw. von Neumanns ebenbürtig oder überlegen und teilweise ganz ähnlich waren. Trotzdem ist er praktisch ohne Einfluss geblieben und hat wenig zur Computerentwicklung beigetragen. Das hatte viele Gründe: Zuse war ein Einzelgänger, ein technischer Tüftler, der Computer „aus Faulheit"[48] bauen wollte, aber weder Geld noch Beziehungen noch eine formale mathematische Ausbildung hatte und vom Militär abgewiesen wurde. Er hat an dem Projekt bis an sein Lebensende und mit großem Erfolg gearbeitet; aber seine Arbeiten wurden erst be-

42 Turing (wie Anm. 37).
43 Vgl. Neumann (wie Anm. 28); Turing (wie Anm. 19).
44 Turing (wie Anm. 19).
45 Neumann (wie Anm. 28).
46 Ebd.
47 Neumann (wie Anm. 35).
48 Zuse (wie Anm. 25).

kannt, als die Entwicklung in den USA schon viel weiter war. Von Neumann war ein weltberühmter Wissenschaftler und Berater der US-Regierung, als er mit der Computerentwicklung begann. Sein Motiv (Rechenleistung für militärische Zwecke) war fachlich und politisch höchst anerkannt. Er war nicht nur ein wissenschaftliches, sondern auch ein kommunikatives Genie (ein „Partylöwe", heißt es), der in wechselnden Teams mit den unterschiedlichsten Menschen zusammenarbeitete. Turing war menschlich ebenfalls ein Einzelgänger,[49] aber als Wissenschaftler genügend in Beziehungen eingebunden und mit Mitteln versehen, um seine Visionen verbreiten und wie besessen verfolgen zu können.[50]

Wissenschaftliche und technische Erfindungen können sich wie alle Neuerungen nur durchsetzen, wenn sie in lokalen Zentren Erregungen bilden, die sich ausbreiten können.[51] Einzelne Menschen tragen solche Erregungen, sie nehmen sie auf und geben sie weiter, verändern sie vielleicht dabei. Das Zusammentreffen zweier unterschiedlicher mathematischer Sichten – geschult an logischen Kalkülen bzw. an numerischen Verfahren – auf Menschen und Maschinen, die rechnen, hat ein solches Zentrum gebildet. Die Erregung klingt bis heute nach. Wir verstehen die heutige Aufregung um die Informatik besser, wenn wir ihr nachgehen.

Anschrift des Verfassers: Prof. Dr. Dirk Siefkes, TU Berlin, Franklinstr. 28/29, D-10587 Berlin, E-mail: Siefkes@cs.tu-berlin.de.

49 Enzensberger, Hans Magnus: Mausoleum. 37 Balladen aus der Geschichte des Fortschritts, Frankfurt a.M. 1975, S. 122-123.

50 Hodges (wie Anm. 20).

51 Siefkes, Dirk: Formale Methoden und kleine Systeme. Lernen, leben und arbeiten in formalen Umgebungen, Braunschweig, Wiesbaden 1992; ders. et al. 1998 (wie Anm. 6).