Digitale Schriftlichkeit. Eine Einleitung

Martin Bartelmus, Alexander Nebrig

»L'homme pense et ses actions sont machinales.«[1]

In den Literatur- und Sprachwissenschaften meint digitale Schriftlichkeit Praktiken des Schreibens und Lesens in einer digitalen und vernetzten Umgebung, die sich von analogen Schriftpraktiken durch Interaktivität und Multimedialität unterscheiden. Die neuen technischen Möglichkeiten der sozialen Interaktion und der elektronischen Vernetzung von Schrift sowie ihrer Kombination mit anderen Medien sind in dieser Intensität im Analogen nie da gewesen. Der qualitative Unterschied entsteht durch ein drittes Merkmal, das Henning Lobin als Hybridität bezeichnet.[2] Es bedingt die anderen beiden insofern,

1 S. Karsakof [d.i. Semen Nikolaievič Korsakov]: Aperçu d'un procédé nouveau d'investigation au moyen de machines à comparer les idées. Avec deux planches. St. Petersbourg: De l'imprimerie de la III-me section de la chancellerie privée de sa Majesté Impériale 1832, 1. Auf Deutsch lautet der erste Satz der Schrift: »Der Mensch denkt und seine Handlungen sind maschinell.« (Übersetzung von AN, das schwer zugängliche Original wird zitiert nach dem digitalisierten Exemplar der Russischen Staatsbibliothek in Moskau, OR Ф.137 к.167 ед.36) Vgl. zu Korsakov Wladimir Velminksi/Wolfgang Ernst: *Semën Karsakov. Ideenmaschine. Von der Homöopathie zum Computer*. Berlin: Kadmos 2008, die sich entgegen allen Schreibungen und allein auf der Basis der französischen Transkription des Namens von 1832 für die Schreibung ›Karsakov‹ entscheiden.

2 Genau genommen sind »Automatisierung, Datenintegration und Vernetzung« für Lobin »Triebkräfte[] der Digitalisierung« und »Hybridität, Multimediali-

als Interaktivität und Multimedialität im digitalen Raum ihre beeindruckenden Leistungen aus der Automatisierung der Schrift schöpfen und zu einer Hybridisierung von Mensch und Maschine sowohl beim Lesen als auch beim Schreiben führen. In vorliegender Einleitung konzentrieren wir uns auf dieses Merkmal. Es geht also nicht um digitale Schriftkommunikation von und für Menschen, weder um soziales Lesen noch um multimediale Semantiken, Literatur im digitalen Raum oder Fake News. Vielmehr stehen jene Formen digitaler Schriftlichkeit infrage, die entweder ganz oder teilweise auf maschineller Seite zu verorten sind: Programme, Codes und Algorithmen. Menschen nutzen die digitale Schreib- und Lesetechnik nicht nur, sondern gestalten sie mit und überantworten Schreibprozesse den Maschinen. Digitale Schriftlichkeit menschlicher Akteure:innen basiert darauf, dass Maschinen lesen und schreiben, dass maschinelle Schriftlichkeit unseren Raum des Les- und Schreibbaren strukturiert. Es gibt eine Schriftlichkeit der digitalen Schriftlichkeit, eine Potenzierung, die analogen Schriftkulturen unbekannt ist.

Walter Ong deutet die Erfindung der Schrift als Technologisierung der Sprache.[3] Die Gleichsetzung von Mündlichkeit mit Natürlichkeit und Schriftlichkeit mit Künstlichkeit erfolgt auf der Annahme, dass orale Sprache nicht als Kunst im Sinne einer Technik erlernt werden muss, sondern sich wie von selbst in jedem Individuum entwickelt. Schrift dagegen impliziert eine technische Theorie der Sprache. Für die digitale Schriftlichkeit muss Ongs These weiter differenziert werden. Denn die analoge Technologisierung der Rede als Schrift hat anders als ihre digitale Fortsetzung noch keinen doppelten Boden, ist eine Technik erster Ordnung. Anders gesagt: Digitale Schrift funktioniert nur deshalb, weil sie eine weitere Schriftebene schafft, welche die sichtbare

tät und Sozialität« von jenen bewirkte Tendenzen (Henning Lobin: *Engelbarts Traum. Wie der Computer uns Lesen und Schreiben abnimmt*. Frankfurt a.M./New York: Campus Verlag 2014, 96).

3 Walter J. Ong: *Orality and Literacy. The Technologizing of the Word*. London: Methuen & Co 1982.

Schrift im Besonderen und die digitalen Medien im Allgemeinen nochmals technisiert. Es kommt neben der Technologisierung der Sprache zu einer Technologisierung der Schrift.

Diese Schrifttechnik zweiter Ordnung ist menschengemacht, verändert die Menschen und vollzieht Operationen, an denen sie nicht beteiligt sind. Somit kristallisieren sich drei Aspekte digitaler Schriftlichkeit heraus: Erstens erfinden Menschen Schriften, um mit Maschinen zu kommunizieren und sie in ihren Dienst zu nehmen. Programme werden geschrieben und gelesen, um Computer zu benutzen. Zweitens gibt es eine Form digitaler Schriftlichkeit, die sich ganz ohne menschliches Zutun vollzieht, da Maschinen auch mit Maschinen kommunizieren und Prozesse ausführen, damit wir lesen und schreiben können. Drittens geht es um die alltägliche Erfahrung, dass der digitale Code zwar menschengemacht ist, aber seine eigenen Affordanzen besitzt. Maschinen adressieren ihre menschlichen Nutzer:innen, Algorithmen ›programmieren‹ unser Verhalten. Unter diesen drei Aspekten sollen im Folgenden die drei Grundoperationen digitaler Schriftlichkeit vorgestellt werden. Programmieren betrifft die Mensch-Maschine-Kommunikation, Prozessieren maschinelle Vorgänge des Lesens bzw. Auslesens und Codieren die dem digitalen Code inhärente Eigenlogik, wie sie im Algorithmus zur Sprache kommt.

1. Programmieren: Mit Maschinen sprechen

Der Turing-Test, ob diese Einleitung maschinell verfasst ist, kann mittlerweile selbst wieder von einer Maschine durchgeführt werden. Die Leistungsfähigkeit, die Textgeneratoren in den letzten Jahren erreicht haben, zeigt sich darin, dass auch Künstliche Intelligenz (KI) eine Schwierigkeit damit hätte, zu beweisen, ob ein Text von einem Chatbot oder einem Menschen geschrieben worden ist. Es ist nicht mehr

erkennbar, ob Übersetzer:innen oder Autor:innen aus Fleisch und Blut einen Text verantworten oder eine KI.[4]

Als erste Annäherung an das Programmieren, das eine zentrale Praxis auf dem Gebiet der digitalen Schriftlichkeit darstellt, erweist sich Alan Turing noch in einer weiteren Hinsicht als Impulsgeber. Mit seinem Konzept der *paper machine*[5] begreift er nicht die Maschine als Menschen, sondern den Menschen als Maschine. Es sei möglich, so Turing 1948,[6] eine Rechenmaschine nachzuahmen, indem man Handlungsanweisungen aufschreibt und sie von einem Menschen ausführen lässt. Eine solche Verbindung von Menschen und aufgeschriebenen Anweisungen, die man auch kurz Programm nennen kann, bezeichnet Turing als Papiermaschine:

> »Es ist möglich, den Effekt einer Rechenmaschine zu erreichen, indem man eine Liste von Handlungsanweisungen niederschreibt und einen Menschen bittet, sie auszuführen. Eine derartige Kombination eines Menschen mit geschriebenen Instruktionen wird ›Papiermaschine‹ genannt. Ein Mensch, ausgestattet mit Papier, Bleistift und Radiergummi sowie strikter Disziplin unterworfen, ist in der Tat eine Universalmaschine.«[7]

---

4 Das hat natürlich auch Konsequenzen für das menschliche Schreiben. Vgl. Crystal Chokshi: In Other Words: Smart Compose and the Consequences of Writing in the Age of AI. In: *Culture Maschine* 20 (2021): https://culutremachine.net/vol-20-machine-intelligences/ (zuletzt abgerufen am 21.12.2023): *Die Herausgeber haben sich nach besten Wissen und Gewissen bemüht, sicherzustellen, dass die URLs für externe Websites, auf die in diesem Buch verwiesen werden, zum Zeitpunkt der Drucklegung korrekt und aktiv sind. Die Herausgeber übernehmen jedoch keine Verantwortung für die Websites und können keine Garantie dafür geben, dass eine Website aktiv bleibt oder dass der Inhalt angemessen ist oder bleiben wird.*

5 Vgl. ferner die Variationen der Turing-Maschine: Oswald Wiener/Manuel Bonik/Robert Hödicke: *Eine elementare Einführung in die Theorie der Turing-Maschinen*. Wien/New York: Springer 1998.

6 Alan Turing: Intelligente Maschinen. In: ders.: *Intelligence Service. Schriften*. Hg. von Bernhard J. Dotzler und Friedrich Kittler: Berlin: Brinkmann & Bose 1987, 81–113.

7 Turing: Maschinen, 91.

Die Papiermaschine zeichnet sich aus durch ihren Universalismus, endloses Papier und unendlich viel Tinte.[8] Man könnte versucht sein, mit der Papiermaschine eine:n Schauspieler:in zu assoziieren, der/die das Stück eines/einer Dramatikers:in aufführt. Ist eine Tragödie ein Programm, eine niedergeschriebene Instruktion? Die Frage lässt sich ebenso auf eine musikalische Komposition und ihre Aufführung beziehen. Doch es besteht ein Unterschied. Ein niedergeschriebenes Drama wird nicht ausgeführt, sondern aufgeführt. Die Performanz ist dem Programm, das allein syntaktische Regeln befolgt, unbekannt. Till Heilmann weist darauf hin, dass dem Programm die »pragmatische und semiotische Dimension« der Schrift abgehe:

> »Erst wenn eine Schrift auf ihre bloßen syntaktischen Formen reduziert ist und diese Formen durch physikalische Zustände, beispielsweise als Spannungspotentiale in Transistoren, ›anschreibbar‹ sind, können sie in einem Computer die regelgeleiteten Kaskaden von Schaltvorgängen auslösen, an deren Ende der durch eben diese Schaltvorgänge veränderte Zustand des Computers als Ergebnis der Programmierung ›ablesbar‹ ist.«[9]

Ohne Code kein Klang, keine Sprache, keine Schrift und keine Bilder, die auf Bildschirmen erscheinen oder von Computern ausgegeben werden können. Code ist eine Art *Lingua franca* geworden, »not only of computers but of all physical realitiy«, wie N. Katherine Hayles konstatiert.[10] Die Turing-Maschine steht für sie am Anfang eines »computational re-

8 Bernhard J. Dotzler: *Papiermaschine. Versuch über COMMUNICATION & CONTROL in Literatur und Technik*. Berlin: Akademie Verlag 1996, 7.

9 Till A. Heilmann: *Textverarbeitung. Eine Mediengeschichte des Computers als Schreibmaschine*. Bielefeld: transcript 2012, 65.

10 N. Katherine Hayles: *My Mother Was a Computer. Digital Subjects and Literary Texts*. Chicago/London: The University of Chicago Press 2005, 15.

gime«,[11] in dem und durch das Code ontologisch wird und das gleichzeitig »as metaphor and means« funktioniert.[12]

»Computation is a scriptorial process, and the computer is a scripting machine«,[13] so Louisa Shen, und auch Florian Cramer versteht Computerprogramme als Sprache und Schrift erzeugende Maschinen auf der Basis von Codes: »The digital computer is a symbolic machine that computes syntactical language and processes alphanumerical symbols; it treats all data – including images and sounds – as textual, that is, as chunks of coded symbols.«[14] Steven Connor bestimmt ferner Schreiben als eine Art der Maschinerie und fragt: »If all writing is a kind of machinery, why might it be plausible to see every machine as a kind of writing?«[15] Schreiben basiert auf einer Differenz von natürlicher und maschineller Sprache,[16] wobei letztere nicht ohne erstere denkbar und zudem ihrerseits aufgeteilt ist in »languages in which algorithms are expressed and software is implemented.«[17] Für die Mensch-Maschine-Kommunikation sind ferner sogenannte *Assembler*, also Übersetzungsprogramme notwendig, wobei »ein Programm in Assemblersprache sehr einem Programm in Maschinensprache [ähnelt], das für Menschen lesbar gemacht worden ist.«[18] *Compiler* wiederum sind Programme, die mehr Übersetzungsarbeit leisten. *Interpreter* dagegen übersetzen nicht mehr in Maschinensprache, sondern »lesen eine Zeile und führen

11 Hayles: *My Mother*, 18. Hayles geht dabei von Stepahn Wolframs Buch *A New Kind of Science* aus.

12 Hayles: *My Mother*, 20f.

13 Louisa Shen: GUIding the Overwrite. In: James Gabrillo/Nathaniel Zetter (Hg.): *Articulating Media. Genealogy, Interface, Situation*. Open Humanities Press 2023, 78–95, hier: 78.

14 Florian Cramer: Language. In: Matthew Fuller (Hg.): *Software Studies. A Lexicon*. Cambridge MA/London: MIT Press, 168–173, hier: 171.

15 Steven Connor: How to Do things with Writing Machines. In: Sean Pryor/David Trotter (Hg.): *Writing, Medium, Machine. Modern Technographies*. O.O.: Open Humanities Press 2016, 18–34, hier: 18.

16 Vgl. für eine kurze Skizze der Maschinensprache: Douglas R. Hofstadter: *Gödel, Escher, Bach ein Endloses Geflochtenes Band*. München: DTV ^{9}2003, 310f.

17 Cramer: Language, 168.

18 Hofstadter: *Gödel, Escher, Bach*, 312.

sie sofort aus.«[19] Das hat zur Folge, dass Schriftlichkeit der digitalen Schriftlichkeit, also die Technisierung der Schrift, die Zeichen der Welt weiter umcodiert: »Verhalf das Alphabet dazu, den Körper der Welt (Soma) in ein Zeichen (Sema) zu verwandeln, geht der digitale Code weiter und löst nun auch die Welt der Zeichen (Sema) in Digits (bits) auf.«[20]

Was den bzw. die Schauspieler:in – um weiter im Kontext von Körper und Code zu bleiben – vom Computer unterscheidet, und damit auch ganz grundsätzlich alle Rechenmaschinen vor Turings Papiermaschine von denen von heute, ist eine »Auftrennung in Funktionsgruppen«.[21] Für Bernhard Dotzler beginnt der Computer dort, »wo Funktionsgruppen sich aufspalten.«[22] Diese Auftrennung, die nicht vom Körper der Schauspielerin geleistet wird, bezeichnet die Funktionen: Rechnen, Speichern, Übertragen – Computing, Control, Communication.[23]

In diesem Kontext kann der Untertitel des vorliegenden Bandes – *Programmieren, Prozessieren und Codieren von Schrift* – auch anders formuliert werden: Vorschreiben, Fortschreiben und Einschreiben. Diese Trias meint nicht einfach nur »Textverarbeitung«, also das Schreiben normalsprachlicher Texte mithilfe des Computers.[24] Es markiert das Weichwerden der Wörter als ihre Codier- und Decodierbarkeit. Damit geht die Unterscheidung von ›elektrisch‹ und ›elektronisch‹ einher. Letzteres meint, dass elektrisch codiert und decodiert wird.[25] Diskrete bzw. digitale Schriftlichkeit ist nicht auf mechanisches, sondern auf maschinenartiges Schreiben bezogen, auf die Schrift einer idealen Maschine.[26]

---

19 Hofstadter: *Gödel, Escher, Bach*, 314.

20 Martin Burkhardt/Dirk Höfer: *Alles und Nichts. Ein Pandämonium digitaler Weltvernichtung*. Matthes & Seitz: Berlin 2015, 49.

21 Vgl. Dotzler: *Papiermaschine*, 32.

22 Dotzler: *Papiermaschine*, 30.

23 Vgl. Dotzler: *Papiermaschine*, 41.

24 Textverarbeitung bzw. »word-processing« war ab 1967 als Begriff in Umlauf und wurde durch IBMs 1961 vorgestellten »electric typewriter« populär gemacht. Vgl. Connor: Writing Machines, 28.

25 Connor: Writing Machines, 29.

26 Vgl. Connor: Writing Machines, 31f.

Digitale Schriftlichkeit forciert die skripturalen Prozesse in und an der Maschine und weniger Be-Schreibungen wie Franz Reuleaux' *Kinematische Zeichensprache* oder Charles Babbages *Mechanical Notation*, zwei Versuche, »Maschinen als das mechanische Getriebe, das sie sind, zu notieren«.[27] Semën N. Korsakov, als Gegenentwurf zu Babbage, entwarf 1832 »ein Lochkartenverfahren […], das Datenverarbeitung zum Zweck von *intelligence* einsetzt«,[28] und konzipierte damit eine Ideenmaschine. Ziel ist eine »begriffsvergleichende Maschine«, die »ganz im Sinne Alan Turings eine scheinbar geistige intellektuelle Tätigkeit vollständig durch ein mechanisches Äquivalent« ersetzt.[29] Auch für diese Maschine bedarf es einer Programmierung, folglich einer Anschrift mithilfe von »Datenbanken und Tabellen«.[30]

Die digitale Schriftlichkeit versteht sich demnach als eine Form der Schriftlichkeit, die sich jenseits der technischen Vorläufer von digitalen Computern, den *Machinae Arithmeticae*, verortet und sich dezidiert als prozessuale Materialität der Schreibweise einer neuen Konfiguration zwischen ›Control‹ und ›Communication‹ darstellen lässt.[31] Damit schließt die digitale Schriftlichkeit an die Schreibstuben und ihre Begriffe ›Kontor/Comptoir‹ an,[32] wird als Funktion der Buchführung sichtbar in dem Sinne, dass Buchführung eben »Grundelement schriftbasierter Gesellschaften« ist.[33] Von hier aus ist es nicht mehr weit bis zum Kernbegriff des Computerzeitalters »computus« (als Programmierkunst) und dann zum »Zauberwort« Algorithmus.[34]

---

27 Dotzler: *Papiermaschine*, 48.

28 Wolfgang Ernst: Ein medienarchäologisches Schicksal. In: Wladimir Velminski/Wolfgang Ernst: *Semën Karsakov. Ideenmaschine. Von der Homöopathie zum Computer*. Berlin: Kadmos 2008, 7–48, hier: 20.

29 Ernst: Schicksal, 27.

30 Ernst: Schicksal, 27.

31 Angeregt von N. Catherine Hayles: Print is Flat, Code is Deep: The Importance of Media-Specific Analysis. In: *Poetics Today* 25/1 (2004), 67–90.

32 Dotzler: *Papiermaschine*, 66.

33 Dotzler: *Papiermaschine*, 66.

34 Vgl. Dotzler: *Papiermaschine*, 69.

Schriftlichkeit gehört existenziell zum Diskurs um Datenverarbeitung, in dessen Zentrum das Wort Information steht. Information ist nicht nur Nebenprodukt von Prozessen, sondern gleichsam auch abstrakt und messbar.[35] Claude Shannon wird gemeinsam mit Warren Weaver in *The Mathematical Theory of Communication* die Disziplin der Informationstheorie begründen,[36] indem er das Problem des Rauschens in elektronischer Kommunikation als Ausgangspunkt nimmt, um Information an Medialität zu knüpfen.[37] Gregory Bateson definiert Information als die Differenz, die Differenz macht.[38]

Programmieren, Prozessieren und Codieren gehören zum Spektrum um ›Control‹ und ›Communication‹ und schließen den Befehl,[39] also ›Command‹ mit ein. Aufgrund der Befehlsstruktur wird die Botschaft ohne jegliches Mitdenken vom maschinellen Empfänger verstanden, als wäre er ein ›Vollidiot‹: »Writing in a computer programming language is phrasing instructions for an utter idiot.«[40] Der Programmierspezialist Eben Moglen, der in den 1970er Jahren für IBM arbeitete, wertet das Programmieren der Kommandozeile dagegen mit der Bemerkung auf: »›What we were doing with computers was making languages that were better than natural languages for procedural thought‹«.[41]

Dadurch ist – im wahrsten Sinne des Wortes – ein für die Schriftlichkeit bestimmender Diskurs angesprochen. Denn Sprache und Sprechen

35 Vgl. Ted Byfield: Information. In: Matthew Fuller (Hg.): *Software Studies. A Lexicon*. Cambridge MA/London: MIT Press, 125–131, hier: 126.

36 Vgl. dazu: Claude E. Shannon: *Ein/Aus. Ausgewählte Schriften zur Kommunikations- und Nachrichtentheorie*. Hg. von Friedrich Kittler/Peter Berz/David Hauptmann/Axel Roch. Berlin: Brinkmann & Bose 2000.

37 Byfield: Information, 128.

38 Gregory Bateson: *Ökologie des Geistes. Anthropologische, psychologische, biologische und epistemologische Perspektiven*. Frankfurt a.M.: Suhrkamp 1985, 488. Vgl. Jacques Derrida führt seinen Begriff der *différance* als »irreduzible Abwesenheit der Intention« ein, um »die allgemeine graphematische Struktur jeder ›Kommunikation‹ zu behauptet.« Vgl. Jacques Derrida: *Limited Inc.* Wien: Passagen Verlag ²2023, 41.

39 Vgl. Hofstadter: *Gödel, Escher, Bach*, 310.

40 Cramer: Language, 171.

41 Shen: GUIding, 88. (Moglen and Worthington 2000).

vom Befehl her zu denken, wie es in der Geistesgeschichte kontinuierlich versucht wird, markiert die Opposition von Stimme und Schrift, Schriftlichkeit und Mündlichkeit, in deren Zentrum der Phonologozentrismus erscheint, und bildet gleichzeitig den Horizont, vor dem sich eine digitale Version dieser Opposition abzeichnet. Der Futurist Velimir Chlebnikov imaginiert 1916: »Wir können jede Handlung, jede Form mit einer Zahl benennen, und durch das Erscheinenlassen einer Zahl auf dem Glas eines Lämpchens können wir sprechen. […] Besonders geeignet ist die Zahlensprache für Radiotelegramme. Zahlenreden.«[42] Dank der Technologie haben wir es heute mit einem arithmetischen Phonologozentrismus zu tun.

Die ›Kommandostruktur‹ der digitalen Kommunikation mit Computern basiert nicht nur auf Kommandozeilen wie bei MS-DOS, sondern geht auf die Lochkarten der Webstühle zurück. Dadurch sind der digitalen Schriftlichkeit eine Form der pastoralen Weisung, einer *Talking Cure*, aber auch ein Sprechakt eingeschrieben, die sich nun nicht mehr unmittelbar an Menschen, sondern an Maschinen richten. Diese überführen die Menschen aus der Disziplinargesellschaft in eine Kontrollgesellschaft.[43] Programmieren ist voll von einer Sprache der Rede und nicht einer der Schrift: »call, query, confirm, dialogue.«[44] Gleichzeitig sind diese mündlichen Sprechakte als Diktate rückübersetzt in das schriftliche Kommando und machen eine Redesimulation sichtbar: »Beschreiben und Schreiben wird eins: Programmieren.«[45]

Die Programmierkunst, auf die sich die digitale Schriftlichkeit bezieht, ist auch historisch in gewisser Weise neu und unterscheidet sich von arithmetischen Maschinen und anderen Befehlsketten. Sie ist gekennzeichnet durch die Programmierbarkeit, wie sie als erstes von Ada Lovelace in Bezug auf die Analytical Engine (1833/34) von Babbage

42 Velimir Chlebnikow: Brief an zwei Japaner, In: ders.: *Werke.* Teil 2. Berlin: Suhrkamp 2022, 245–248, hier: 247f.

43 Gilles Deleuze: Postskriptum über die Kontrollgesellschaften. In: ders.: *Unterhandlungen* 1972–1990. Frankfurt a.M.: Suhrkamp [5]2014, 254–262.

44 Shen: GUIding, 92.

45 Burkhardt/Höfer: *Alles und Nichts*, 19.

beschrieben wurde.[46] Programmierbarkeit bedeutet hier die »discretization (or disciplining) of hardware.«[47] Daraus leitet sich eine Form des Genießens ab, die sich im Schreiben von Programmen offenbart und sich als Macht(-gefühl) oder Souveränitätssimulation der Programmierer:innen äußert.[48]

Donald E. Knuths 1984 vorgestelltes *literate programming* kehrt die Vorzeichen des Genießens beim Programmieren um. Das Adjektiv »literate« bedeutet in diesem Kontext, dass Literatur anders dokumentiert (wird) als Code. Knuth, der eine bessere Dokumentation von Programmen fordert,[49], wirbt für einen *human turn* in der Informatik: »let us concentrate rather on explaining to *human beings* what we want a computer to do.«[50] Seit Lovelace und Turing ist also viel passiert: Der Computer muss nicht mehr angeschrieben werden, sondern (erneut) der Mensch. Menschen müssen darüber informiert werden, wie das ominöse ›Wir‹ den Computer dazu bringt, das zu tun, was das Wir will. Es geht nicht nur um das Kommentieren von Code, sondern auch um das Einbinden der Erklärung und des Codes in eine spezifische literarische, sogar philosophisch-politische Form: den Essay.[51] Im Zentrum des Essays steht – jedenfalls für Knuth – die Konstitution eines künstlerischen Subjekts: »I suddenly have a collection of programs that seem quite beautiful to my own eyes, and I have a compelling urge

46 Dotzler: *Papiermaschine*, 536

47 Wendy Hui Kyong Chun: Programmability. In: Matthew Fuller (Hg.): *Software Studies. A Lexicon*. Cambridge MA/London: MIT Press, 224–228, hier: 225.

48 Chun: Programmability, 227.

49 Donald E. Knuth: Literate Programming. In: *The Computer Journal* 27.2 (1984), 97–111, hier: 97.

50 Knuth: Literate Programming, 97.

51 Auch Mark Marino geht in seiner Konzeption der Critical Code Studies auf Don Knuth ein und versteht das Programme-Schreiben als essayistische Praktik. Vgl. Mark C. Marino: *Critical Code Studies*. Cambridge MA: MIT Press 2020, 41. Auch Max Bense interessiert sich für die Form des Essays. Vgl. Max Bense: *Über den Essay und seine Prosa*. In: *Merkur* 3 (1947): https://www.merkur-zeitschrift.de/max-bense-ueber-den-essay-und-seine-prosa-75-jahre-merkur/ (zuletzt abgerufen am 15.11.2023).

to publish all of them so that everybody can admire these works of art.«[52] Der Glaube an die Literarizität der Programmiersprache, also ihre Fähigkeit, Literatur zu werden, wird mithilfe einer ontologischen Konzeption als »idea« begründet.[53] Der/die Programmierer:in hat die Metamorphose zum/r Künstler:in beinahe überstanden. Beinahe, denn Knuths Programme wuchern, werden immer mehr, sodass der Speicher nicht mehr ausreicht: »my office will be encrusted with webs of my own making.«[54] Die Konsequenz ist, in die Welt auszuufern, in Zeitschriften zu diffundieren und damit medien-archäologisch eine neue Gedächtnis- und Erinnerungsschicht zu produzieren.

Mit dem neuen Schreiben geht ein spezifischer Genuss (»joy«[55]) einher, der vorher – beim traditionellen Coding/Programmieren – nicht in dieser Form bestanden hat. Der Genuss führt dazu, dass Knuth in der Lage ist, »to write programs as they should be written.«[56] Wie sollen also Programme geschrieben werden? »My programs are not only explained better than ever before; they also are better programs, because the new methodology encourages me to do a better job.«[57] Das »better« bleibt unbestimmt, und gleichzeitig impliziert das »literate« sein Gegenteil als moralisches Totschlagargument: »surely nobody wants to admit writing an *illiterate* program.«[58] Auch Programmierer:innen stehen also in einem Aufschreibesystem, das ihre Lese- und Schreibfähigkeit seinerseits codiert, normiert und kontrolliert.

Begriffsgenealogisch gehen dem Programmieren die Begriffe ›Coding‹ oder »planning the computation« voraus.[59] Das Codieren meint, dass Information, die codiert werden soll, digital zur Verfügung steht, d.h., sie muss »diskretisiert« und »quantisiert« werden.[60] Oder anders

52 Knuth: Literate Programming, 109.
53 Knuth: Literate Programming, 111.
54 Knuth: Literate Programming, 109.
55 Knuth: Literate Programming, 97.
56 Knuth: Literate Programming, 97.
57 Knuth: Literate Programming, 97.
58 Knuth: Literate Programming, 97.
59 Heilmann: *Textverarbeitung*, 62.
60 Heilmann: *Textverarbeitung*, 196.

formuliert: Wir zählen abzählbare Zeichen,[61] damit diese wiederum gespeichert, übertragen und verarbeitet, d.h. prozessiert werden können. Dabei ist entscheidend, dass nicht nur die gängigen Buchstaben digitalisiert werden, sondern auch das Leerzeichen, Satzzeichen und andere Befehle.

Frieder Nake beschreibt Zeichen im Anschluss an Charles Sanders Peirce als algorithmische Zeichen.[62] Das hat zur Folge, dass das Zeichen für Mensch und Maschine prozessier- bzw. lesbar wird.[63] Wie Hayles bemerkt, haben wir es mit »flackernden Signifikanten« zu tun, einer wie Till A. Heilmann erklärt, ständig changierenden Kette durch maschinelle En- und Decodierprozesse verknüpfter Markierungen.[64] Diese flackernden Zeichen sind nicht (nur) metaphorisch zu verstehen, sondern auch materiell. Für Turing ›flackerten‹ Zeichen und Text noch nicht.[65]

Mit dem Flackern der Zeichen geht eine neue ontologische Dimension einher, die Auswirkungen auf das Schreiben als Programmieren hat. Die Frage, wer was schreibt, wenn programmiert wird, ist insofern relevant, als sich mit der Programmierbarkeit »die Geschichte des Worts in die der Idee des Algorithmus« verwandelte.[66] Der/die Programmierer:in erscheint als Autor:in eines Programms, das eine gewisse Abgeschlossenheit und damit Werkcharakter erhält. Der/die Coder:in dagegen liefert variable Codesegmente, die sowohl Programmierer:innen als auch Computer nutzen können. Denn der Code kann auch einfach hinterlegt werden, sodass der Computer je nach Programm auf diese Codesegmente zurückgreift und damit eigenständig ›schreibt‹. Hardware und Soft-

61 Heilmann: *Textverarbeitung*, 199.

62 Heilmann: *Textverarbeitung*, 230 und Frieder Nake: Das algorithmische Zeichen und die Maschine. In: Hansjürgen Paul/Erich Latniak (Hg.): *Perspektiven der Gestaltung von Arbeit und Technik. Festschrift für Peter Brödner*. München: Rainer Hampp 2004, 203–221.

63 Heilmann: *Textverarbeitung*, 231.

64 Heilmann: *Textverarbeitung*, 217f.

65 James Purdon: Teletype. In: Sean Pryor/David Trotter (Hg.): *Writing, Medium, Machine. Modern Technographies*. O.O.: Open Humanities Press 2016, 120–136, hier: 133.

66 Dotzler: *Papiermaschine*, 71.

ware lassen sich dadurch unterscheiden, aber auch ineinander übersetzen.

Computer können nicht nur mit digitalem Code programmiert werden, sondern auch mit auf Papier gelochten Programmen. Erst mit Textterminals ab den 1960er Jahren und damit mit sogenannten Editoren lässt sich Code für Programme am und mit bzw. für den Computer schreiben. Das bedeutet, dass sich Codes in ihrer Komplexität unterscheiden. Sogenannte höhere Programmiersprachen schaffen immer mehr Distanz zwischen Mensch und Maschine, auch wenn sie die Kommunikation deutlich steigern. Ihre Idealisierung erfährt die absolute Kommunikation dann in der Kybernetik,[67] die, wie Norbert Wieners Publikation von 1948 im Titel formuliert, Kontrolle und Kommunikation aller nichtmenschlicher Aktanten regelt: *Cybernetics or Control and Communication in the Animal and the Machine*.

Dafür braucht es einen Übersetzer, sogenannte Compiler, die Quellcode in den Objektcode übertragen. Wir haben es meistens mit zwei Textsorten zu tun: dem Programm als Text, also dem Quelltext, sowie dem Maschinencode.[68] Entweder man sagt wie Jörg Pflüger, dass im Anfang »der Text« war,[69] oder man sieht wie Jean Baudrillard mit dem Computer die Ära des Codes beginnen: »[T]he era of the signified and of the function is over: it is the era of the signifier and of the code that begins.«[70] Dieser Befund, den Baudrillard 1972 in einem ökologischen Kontext vorstellt, betont gleichzeitig die Verschiebung von einer analogen zu einer digitalen Schriftlichkeit. Anstelle von Fortschritt sind

67 Vgl. Peter Janich: *Was ist Information?* Frankfurt a.M.: Suhrkamp 2006, 48–57 und zur Geschichte der Kybernetik: Thomas Rid: *Maschinendämmerung. Eine kurze Geschichte der Kybernetik*. Berlin: Propyläen 2016.

68 Heilmann: *Textverarbeitung*, 94.

69 Jörg Pflüger: Writing, Building, Growing. Leitvorstellungen der Programmiergeschichte. In: Hans Dieter Heilige (Hg.): *Geschichten der Informatik. Visionen, Paradigmen, Leitmotive*. Berlin: Springer 2004, 275–320, hier: 281.

70 Jean Baudrillard: Design and Environment. Or, The Inflationary Curve of Political Economy. In: *The Universitas Project. Solutions for a post-technological Society*. New York: Museum of Modern Art 2006, 50–66, hier: 61.

Produkte wie ChatGPT nur Aktualisierungen von theoretischen Vorschriften, die vor mehr als 50 Jahren verfasst wurden. Ähnliches gilt für den *Data Driven Turn* der Digital Humanities. Henning Schmidgen und Bernhard Dotzler weisen in ihrem Buch *Foucault digital* auf den Umstand hin, dass die Geschichtswissenschaft schon 1969 mit Daten arbeitete. Michel Pêcheuxs *Analyse automatique du discours* dachte nicht nur an die Datenverarbeitung, sondern auch an eine Maschine, »die in einem emphatischen Wortsinn zu *lesen* und *deshalb* die Menschen, die ›ein Erlernen des Lesens‹ hinter sich haben, zu ersetzen imstande ist.«[71] Baudrillard scheint auf einen solchen Diskurs hinzuweisen und damit diesen als Lektüresimulation aufzudecken, der sich hinter den Verheißungen der Kybernetik verbirgt.

2. Prozessieren: Eine andere Form des Lesens

Im Zentrum digitaler Schriftlichkeit steht immer wieder die Mensch-Maschine-Kommunikation, aber auch die Kommunikation zwischen Maschinen. Programme, Codes bzw. Software dienen der Vermittlung und Übersetzung. Für diese Kommunikation bedarf es einer eigenständigen Materialität von Interfaces, die aus Keyboards,[72] Bildschirmen, spezifischen Kathodenstrahlröhren, Lichtstiften,[73] der Entertaste,[74] der Maus, aber auch aus einfachen Lochkarten aus Papier bestehen kann.[75] Prozessieren bzw. Maschinenlesbarkeit ist stets das Ziel. Der Fokus liegt nicht allein auf der oder dem *User:in*, sondern auf der Maschine,

71 Bernhard J. Dotzler/Henning Schmidgen: *Foucault, digital*. Lüneburg: Meson Press 2022, 71.

72 Shen: GUIding, 79: »The keyboard has been a mainstay of programming and processing, as has paper (albeit with varying degrees of importance as memory and display components have evolved).«

73 Shen GUIding, 80.

74 Connor: Writing Maschine, 30.

75 Vgl. zu der Materialität digitaler Speichermedien: Wolfgang Ernst: *Das Gesetz des Gedächtnisses. Medien und Archive am Ende (des 20. Jahrhunderts)*. Berlin: Kadmos 2007, 205–260.

die benutzt wird.[76] Maschinen, vor allem vernetzte, automatisieren Kommunikationsprozesse nicht einfach, sie vervielfachen diese auch, sodass Sadie Plant konstatiert: »paralleles Prozessieren automatischer Kommunikation, verschalteter Leitungen, wiederholter Operationen; Muster und Netzwerke, die sich wie Unkraut vermehren.«[77] Alles in allem erscheint dabei der Begriff der digitalen Kommunikation dürftig – eine Chance für die digitale Schriftlichkeit.

Programmieren, Prozessieren und Codieren, also die maschinenartigen Schreib- und Leseprozesse, die nicht nur auf Menschen, sondern auch und besonders auf Maschinen bezogen sind, gehören einer Ordnung der Grammatologie an, die Catherine Malabou als allgegenwärtiges ›motor scheme‹ bezeichnet.[78] Darin wäre digitaler Computercode eine Art »worldview«, wie Hayles erklärt, die zugleich Funktion und Bedeutung von Schrift und Sprache beeinflusst und *vice versa*.[79] Grammatologie und »Computation«, so Hayles, stehen sich aber gegenüber.[80]

Von Ada Lovelace bis Knuth spielen die klassisch literaturwissenschaftlichen Kategorien von Autor:innenschaft,[81] Werk und Schrift zentrale Rollen, die sich im Zeichen des *motor schemes* ›Grammatologie‹ mithilfe von *Rhetorical Code Studies*[82] markieren und zwischen *Critical* und *Source Code Studies* in Hinblick auf die Frage nach einer prozessontologischen digitalen Schriftlichkeit neu verorten lassen können und auf die Begriffe Code, Programme, Software und Algorithmus ausgreifen. Dabei lassen sich Programmieren, Prozessieren und Codieren mit den

76 Connor: Writing Maschine, 31.

77 Sadie Plant: *nullen + einsen. Digitale Frauen und die Kultur der neuen Technologien*. Berlin Verlag 1998, 122.

78 Wir verweisen hier auf unsere Einleitung zum Band *Schriftlichkeit. Agentialität, Aktivität und Aktanten von Schrift*. Bielefeld: transcript 2022.

79 Hayles: *My Mother*, 16.

80 Hayles: *My Mother*, 17.

81 Vgl. Kathleen Fitzpatrick: The Digital Future of Authorship. Rethinking Originality. In: *Culture Machine* 12 (2011): https://culturemachine.net/the-digital-humanities-beyond-computing/ (zuletzt abgerufen am 21.12.2023).

82 Vgl. Kevin Brock: *Rhetorical Code Studies. Discovering Arguments in and around Code*. Ann Arbor: University of Michigan Press 2019.

textuellen Ebenen von Bedeutung, Lesbarkeit und Zeichenhaftigkeit parallelisieren.

Prozessieren verweist dabei auf eine Lesbarkeit und ein Lesen, mithin auf eine vermeintliche Widerstandslosigkeit des programmierten Codes. Prozessieren hängt zudem spezifisch vom »compiling« ab, also der Interpretation des Codes.[83] Dazu bedarf es auch neuer Praktiken des digitalen Lesens, nicht nur als Lesen im Digitalen. Hayles schlägt die Unterscheidung von »close«, »hyper« und »machine reading« vor.[84] Insbesondere letztere integriert das algorithmische Auslesen in den erweiterten Begriff der Alphabetisierung.[85]

Für ein solches Lesen als Prozessieren muss jedoch jeder Code und jedes Programm auf gewisse Weise (ab-)geschlossen sein. Abgeschlossen wird Code im Übergang vom Quelltext zum Programmcode via Compiler. Das verleiht dem Programm Werkcharakter, was bedeutet, dass es nicht mehr ohne Gewaltanwendung umgeschrieben werden kann.[86] Die Übersetzung von Programmcode in Maschinencode kann wiederum nicht umgekehrt werden. Diese Irreversibilität erlaubt es, den Quellcode mit einem Copyright zu versehen und unzugänglich zu machen. Im Copyright zeigt sich die Notwendigkeit der Schließung des Codes als abgeschlossenes Werk, das wiederum distribuiert und damit auf Computern installiert werden kann. Erst so lassen sich Code/Programme von Maschinen prozessieren, sodass der/die User:in diese verwenden kann. Der Blick auf die Schriftlichkeit des Codes zeigt, dass diese Schließung selbst nicht Teil der Schrift des Codes ist, sondern seines ideologischen Gebrauchs.

---

83 Vgl. Hayles: *My Mother*, 59.

84 N. Katherine Hayles: *How we think. Digital Media and Contemporary Technogenesis.* Chicago/London: The Chicago University Press 2012, 55–80.

85 Hayles: *How we think*, 11. Vgl. ferner: Anette Vee. *Code Literacy. How Computer Programming is Changing Writing.* Cambridge MA/London: MIT Press 2017.

86 Umgekehrt stellt die Schriftlichkeit des Codes den Werkcharakter des Programms infrage, worauf Markus Krajewski hinweist. Markus Krajewski: Against the Power of Algorithms Closing, Literate Programming, and Source Code Critique. In: *Law Text Culture* 23 (2019), 119–133, hier: 123.

Mit der Geschlossenheit von Programmen geht zudem eine Art Werkcharakter einher, der etwas benötigt, was analogen Texten fremd zu sein scheint. Sie müssen gewartet werden. Damit Programme ausgeführt werden können, bedürfen sie der permanenten Wartung. Durch die Signatur in Bezug auf ihren Quellcode wird dem Programmcode die Notwendigkeit zum Update eingeschrieben, was wiederum die Schließung des Programms aufschiebt, immer weiter in die Zukunft verschiebt und gleichzeitig einen Abschluss erst ermöglicht. Nur die Wartung garantiert das ordnungsgemäße Prozessieren, sprich die Ausführung eines Programms.

Die Schriftlichkeit der Wartung ernst nehmen, bedeutet, nicht nur die Oberfläche des Textes bzw. des Gewebes zu betrachten, nicht nur das Interface oder die Kommandozeile zu analysieren,[87] sondern auch den »Prozess des Webens« zu berücksichtigen, wie er in technischen Medien der Bild- und Textproduktion wie »Siebdruck, Druckerpressen, Druckmatrizen, Fotografie und Schreibmaschine« zugrunde liegt.[88]

Neben dem Gutenberg-Universum erscheint so noch das Jacquard-Universum des Webstuhls, dessen kurze Geschichte mit Basile Bouchon 1725 beginnt, 1728 von Jean-Baptiste Falcon durch die Lochkarte fortgesetzt, mit Joseph-Marie Jacquard 1801 seinen Namen erhält und von Hermann Hollerith 1884 perfektioniert wird. Und dieses Jacquard-Universum ließe sich wiederum in Phasen einteilen: die *Machinae Arithmeticae* (Wilhelm Schickard), Calculation Engines (Babbage) und dann die Computer.[89]

Das Webstuhl-Universum provoziert das anthropozentrische Denken. Es liefert eine weitere Kränkung des Menschen, da es darauf hinweist, dass menschliches Denken von einer Maschine geleistet werden kann. Zeitgleich zu den Arbeiten von Babbage und Korsakov wird Georg

87 Vgl. dazu Lori Emerson: *Reading Writing Interfaces. From the Digital to the Bookbound*. Minneapolis: University of Minnesota Press 2014.

88 Vgl. Plant: *nullen + einsen*, 76.

89 Vgl. Dotzler: *Papiermaschine*, 43. Aber: »die Machinae Arithmeticae [sind] keine Computer-Vorläufer []; [] keine CONTROL interveniert, keine COMMUNICATION – kein Algorithmus.« (Dotzler: *Papiermaschine*, 49).

Wilhelm Friedrich Hegel mittels der idealistischen Philosophie die Abwehr dieser Mechanisierung des Denkens vorbereiten.[90]

Die beiden medien- und schrifttheoretischen Universen, die sich zeitversetzt ausbilden, stehen nicht in Konkurrenz, sondern sind in ihrer materiellen wie ideellen Konzeption aufeinander bezogen. Anstelle der Software ›Schrift‹, die dem Gutenberg-Universum entstammt, beginnt Ada Lovelace das Jacquard-Universum zu programmieren. Anstelle des Buchdrucks adressiert die Webetechnik[91] eine maschinelle Literatur; anstelle von Papier und beweglichen Lettern definieren Garn und Lochkarten die Materialität digitaler Schriftlichkeit. Daraus ergibt sich das (textile) Prozessieren[92] als spezifische Lesetechnik.

In Knuths *literate programming* manifestiert sich das Webstuhl-Universum nun im Raum des Programmierens. Den Konnex verdankt dieser Raum dem Namen seiner Programmiersprache: WEB, also Gewebe. Weil dieses Drei-Buchstaben-Wort noch nicht für das Programmieren benutzt wird, gibt es nicht nur den Eigennamen der Sprache, sondern auch das Konzept: WEB, das Gewebe, Netz, das verfilzt oder (ver-)und gewebt werden kann. Damit schreibt sich Knuths *literate programming* in die Geschichte eines Webstuhl-Universums ein. Denn Knuths Programmieren zeichnet sich durch einen doppelten Vorgang aus: durch *weaving* und *tangling*, weben und verfilzen.[93] Der ›gewebte‹ Code, den die Maschine ausführt, wird ›verfilzt‹. Zwei Schreib- und Leseoperationen lassen die Tradition des Webstuhls aufleben, von denen alle Programmierer:innen abstammen. *Weaving* und *tangling* sind Schreibmodi, verweisen aber auch auf zwei unterschiedliche Weisen

90 Vgl. Ernst: Schicksal, 41.

91 Vgl. Plant: *nullen + einsen*, 19.

92 Zitiert nach Ernst: Schicksal, 24. Vgl. Birgit Schneider: Diagramm und bildtextile Ordnung. In: *Bildwelten des Wissens*, 3.1 (2005), 9–20.

93 Stiegler hebt die Praxis des Webens auch für die tertiäre Retention hervor. Er schreibt: »This weaving, for which tertiary retention is irreducible, and constitutes noesis insofar as it makes [fait] (the) différance at once as exteriorization, reproduction and discernment, is also what undermines [défait] noesis: it is as such that writing is a pharmakon.« Bernard Stiegler: *The Neganthropocene*. O.O.: Open Humanities Press 2018, 229.

des Prozessierens, die von ihren jeweiligen Adressaten:innen abhängen: Mensch und Maschine. An sie ist nicht nur die Wartung der Maschine gekoppelt, sondern auch die des Codes, damit das fortdauernde Funktionieren als Kommunikation gewährleistet bleibt. Im digitalen Kontext von Programmieren, Prozessieren und Codieren ist das Netz nicht Metapher, sondern bestimmt das, was Buchdruck- vom Webstuhl-Universum unterscheidet. Im Gutenberg-Universum ist jedes Zeichen adäquat verräumlicht, während im Webstuhl die Zeichen miteinander verbunden sind.

Beide Universen teilen eine Gemeinsamkeit, nämlich die Konfiguration der Nutzer:innen als Subjekte, die prozessieren, und damit einem Skript folgen müssen, um die Maschinen zu bedienen.[94] Markus Krajewski und Cornelia Vismann heben den Verwendungszusammenhang des Quellcodes hervor: »Man kann aus ihm mit einem Klick entweder einen Text in Form eines Handbuchs oder einer Dokumentation für andere Entwickler erstellen, oder aber ebenfalls per Knopfdruck eine ausführbare Datei generieren.«[95]

Krajewski spricht von einer »Source Code critique«, die auf den Quellcode als doppelten Text setzt: als Kommentar des Codes, der den Code beinhaltet. Ziel ist die Analyse des »extensive interplay of code and commentary in order to make the algorithms themselves more transparent and thus more comprehensible.«[96] Die elementaren Praktiken sind dabei Lesen, Kommentieren und Modifizieren: »The source code would thus contain the individual commands and data structures together with their documentation or philological apparatus.«[97] Quellcode, das gilt es zu beachten, enthält zudem immer mehr als nur den Code: Dort werden auch Kommentare der Programmierer:innen verzeichnet.[98]

94 Shen: GUIding, 88.

95 Markus Krajewski/Cornelia Vismann: Kommentar, Code und Kodifikation. In: *Zeitschrift für Ideengeschichte* (2009), 5–16, hier: 12.

96 Krajewski: Against Power, 128.

97 Krajewski: Against Power, 130.

98 Joasia Krysa/Grzesiek Sedek: Source Code. In: Matthew Fuller (Hg.): *Software Studies. A Lexicon*. Cambridge MA/London: MIT Press, 236–242, hier: 237.

Wendy Hui Kyong Chun ergänzt die Perspektive um einen kritischen Einwand. Für sie ist der Quellcode eine Art Fetisch.[99] Mit dem Wortspiel »sourcery«, bestehend aus *source* (Quelle) und *sorcery* (Zauberei), verweist Chun einerseits auf die Konstruktion einer Quelle als Ausgangspunkt maschineller Agency als auch auf das Prozessieren als etwas, das ›von selbst‹ geschieht. Demgegenüber konzipiert Chun Code als »re-source«, um erstens den »gap« zwischen Code und Ausführung zu verstehen, zweitens »entropy, noise and decay« von Code zu denken und drittens Code zu historisieren und zu kontextualisieren.[100]

Beide methodischen Zugänge (Critical und Source Code Studies) behalten in ihrem Kern ein seit Friedrich Schleiermacher bekanntes Postulat bei: Hermeneutik darf nicht mechanisch, oder in unserem Fall, digital werden, damit sich ein Verstehen als aktives Ereignis einstellt.[101]

Krajweski bezieht sich für seine Praxis der Source Code Critique auf Knuths *The Art of Computer Programming*. Knuth konzipiert ein *literate programming*, »das sich als dichte Verbindung von algorithmischen Strukturen und ihren Kommentaren zeigt.«[102] Das doppelte Schreiben von Weben und Verfilzen in Knuths *literate programming* inklusive des ökonomischen Kalküls seiner Korrigierbarkeit ermöglicht nämlich »style«. Und dieser entwickelt sich.[103] Programmieren in WEB wird zu einer artistischen Angelegenheit, die dem Programmieren wieder einen geniehaften, auktorialen Pinselstrich zurückgibt. Hier schreibt sich explizit der/die Programmierer:in als Autor:in ein. Eine alte literarische

99 Wendy Hui Kyong Chun: On ›Sourcery‹, or Code as Fetish. In: *Configurations* 16/3 (2008), 299–324.

100 Chun: Sourcery, 321f.

101 Vgl. Dotzler: *Papiermaschine*, 16: »Die Auslegung wird hier zur Rechnung, die Hermeneutik, indem sie die lebendige Operation darstellen will, zum puren Mechanismus«, wird Dilthey mit Rücksicht auf jene Rückschau, die ihm den Blick eingesenkt hat, kritisieren, um dagegen an eben jene neue Position zu erinnern, wie Schleiermacher sie definiert hat: »daß sich das Mißverstehen von selbst ergibt und das Verstehen auf jedem Punkt muß gewollt und gesucht werden«.

102 Krajewski/Vismann: Kommentar, 12.

103 Knuth: Literate Programming, 108.

Autor:innen-Funktion wird in das Konzept des Programmierens eingeschmuggelt. Dieser Schmuggel erlaubt es aber auch, eine Vorschrift für Autor:innenschaft zu setzen: »you have the freedom to say anything you want, and this freedom entails the responsibility of deciding what to say.«[104] Daraus resultiert, dass die Programme weniger Fehler aufweisen, weil ein starkes Autor:innensubjekt gezwungen ist, »to clarify my thoughts as I do the programming.« Das große »I« ist dabei schwerer zu täuschen.[105] Der Schmuggel hat also einen Hang zur Wahrheit.

Das Ich mit großem I (wie es auch Knuths *literate programming* voraussetzt) hat aber nicht immer die absolute Kontrolle. Die Les- bzw. Prozessierbarkeit des Werks ist stets bedroht, und zwar von ganz unscheinbaren nicht-digitalen Entitäten. So hat eine Motte schon einmal eine Rechenmaschine lahmgelegt.[106] Grace Hopper entdeckte beim Versuch, das Problem der Maschine zu lösen, den ersten »bug«. Seit diesem Vorfall werden Computer und ihre Netzwerke von Insekten bevölkert. Die Schrift kehrt über den »bug« (den Parasiten) als Öffnung wieder in die abgeschlossene Tätigkeit des (Programm- bzw. Code-)Prozessierens ein. Das »De-Bugging« wird im Kontext der digitalen Schriftlichkeit zu einem entscheidenden Einstiegspunkt der kritischen Auseinandersetzung mit Code als Schrift und der Materialität der Maschine, weil Prozessierung, also Leseprozesse unterbrochen und neu gestartet werden können und müssen.

3. Codieren: Vom einzelnen Zeichen zum Algorithmus

Erst als Programme auch als abgeschlossene Texte in höheren Programmiersprachen geschrieben wurden und Inhalte bereitstellten, konnte Codieren wiederum einen Teilaspekt des Schreibprozesses bezeichnen. Code ist von Rede und Schrift dadurch unterschieden, »in that it exists in clearly differentiated versions that are executable in a process that

104 Knuth: Literate Programming, 108.
105 Knuth: Literate Programming, 108.
106 Plant: *nullen + einsen*, 134.

includes hardware and software and that makes obsolete programs literally unplayable unless an emulator or archival machine is used.«[107] Das bedeutet auch, dass Codes Bausteine eines Programms sind wie Sätze in einem Text. Code ist hierarchisiert: C++ zum Beispiel ist in ASCII geschrieben, wobei der ASCII-Code wiederum in Maschinensprache übersetzt wird.[108]

Im Grunde meint Codieren, ein Zeichen mittels eines Codes in ein anderes Zeichen zu verwandeln. Dieser Prozess wird allerdings schnell standardisiert, sodass sich Bausteincodes ergeben, die wiederum zum Programmieren verwendet werden können. Code ist in der meisten Software, die wir im Alltag benutzen, ›alter‹ Code, der um neue Bestandteile ergänzt wird,[109] ganz so wie bestimmte Elemente in Narrativen immer wieder benutzt werden, um etwas verständlich auszudrücken oder damit ein Text ›funktioniert‹. Die Assoziation liegt nahe, hier von Intertextualität des Codes zu sprechen. Doch nicht allein anhand seiner Bausteine kann der Code erkannt werden. Friedrich Kittlers Skepsis, wir könnten »schlichtweg nicht mehr wissen, was unser Schreiben tut, und beim Programmieren am allerwenigsten«,[110] muss nicht geteilt werden. Durchaus kann mittels Wartungsarbeit beschrieben werden, was die Schrift respektive der Code tun, mithin was digitale Schriftlichkeit tut. Es geht nicht mehr um den Schreibprozess, sondern um einen Schrift- oder Codeprozess, der von Menschen und

107 Hayles: *My Mother,* 52.

108 Hayles: *My Mother,* 57. Damit unterscheidet sich C++ von Programmiersprachen wie FORTRAN oder BASIC. Zu unterscheiden sind ferner prozedurale von objekt-orientierten Programmierungen (Hayles: *My Mother*, 57), die entsprechend der Syntax natürlicher Sprache konzipiert sind. Prozedurale Programmsprachen dagegen verwenden sogenanntes »early binding«, verbinden also Programm, Compiler und Maschine bevor das Programm eigentlich läuft (Hayles: *My Mother*, 59).

109 Marisa Leavitt Cohn: Keeping Software present: Software as a Timely Object for STS Studies of the Digital. In: Janet Vertesi/David Ribes: *digitalSTS. A Field Guide for Science & Technology Studies*. Princeton UP 2019, 423–446.

110 Friedrich Kittler: Es gibt keine Software. In: ders.: *Draculas Vermächtnis. Technische Schriften*. Leipzig: Reclam 1993, 225–242, hier: 229.

Maschinen gelesen werden kann. Das Geschriebene gewährleistet das Anschreiben, das Adressieren meint, und adressiert wird die Software genauso wie die Hardware. Um zu erfahren, was nun der Code tut, bedarf es gewisser Vorsichtsmaßnahmen. Erstens stellt die Adressierung der Software eine Souveränitätssimulation dar, die das eigentliche Schreiben des Codes nicht mehr betrifft. Die digitale Schriftlichkeit untersucht und verkörpert zugleich die Amalgamierung von Soft- und Hardware, jene Reduktion der Schrift auf »Maschinenzwänge«,[111] wie sie in jedem Compiler stattfindet. Entscheidend nämlich ist, inwiefern Hardware Heimat eines Schreibsystems sein kann.[112] Folglich ist eine »Mineralogie des Geistes«[113] nur mit Computern zu beschreiben. Silizium und Siliziumoxid und noch viele andere seltenen Erden führt Jussi Parikka in einer Geologie der Medien zusammen.[114]

Zweitens also muss die genuine Materialität des Codes berücksichtigt werden. Schriftlichkeit hat eine offensichtliche materielle Dimension: Papier, Tinte, Schreibgeräte. Und dennoch ist gerade der Schriftlichkeit das Prozessuale und Emergente von Schrift und Schreiben eingetragen, sodass auch digitale Schriftlichkeit eine spezifische Verbindung zu Prozess und Materialität unterhält: Computer, Siliziumchips, Serverfarmen, Plastik, seltene Erden etc. auf der einen Seite, ein manchmal auch KI-gestütztes Produzieren von Text auf der anderen Seite, das auch seine neuronale Erinnerungsspur kennt: das Engramm.[115] Im analogen wie

111 Kittler: Es gibt keine Software, 237.

112 Kittler: Es gibt keine Software, 239.

113 Emanuele Coccia: *Das Gute in den Dingen*. Berlin: Merve 2017, 17.

114 Jussi Parikka: *A Geology of Media*. Minneapolis: University of Minnesota Press 2015. Vgl. auch Sarah Pink/Elisanda Ardèvol/Débora Lanzeni: Digital Materiality. In: dies.: *Digital Materialities. Design and Anthropology*. London/New York: Roudledge 2016, 1–26, hier: 10, die im Kontext der Begriffe Design, Anthropologie und Umwelt die Materialität des Digitalen nicht als *a priori* gegeben sehen oder sie als Endpunkt einer Analyse begreifen, sondern »as a process, and as emergent«.

115 Vgl. zu den Engrammen Shintaro Miyazaki: Medien-Archäo-Graffiti. In: Moritz Hiller/Stefan Höltgen (Hg.): *Archäographien. Aspekte einer radikalen Medienarchäologie*. Schwabe Verlag: Basel 2019, 13–20, hier: 14.

digitalen Fall, die sich nur graduell unterscheiden und sich historisch verorten lassen, benötigt eine Grammatologie des Codes auch eine Betrachtung der Materialität.[116] Eine digitale Grammatologie ist nicht ohne ihre Hardware zu denken. Codes, Programme und Software werden so zu digitalen Objekten, wie der Technikphilosoph Yuk Hui argumentiert.[117]

Drittens gilt es, die Gesten und Praktiken des Codes zu analysieren. Im Feld der Science & Technology Studies wird zunehmend Software als soziale Praxis[118] in den Blick genommen und damit in das Paradigma der Kommunikation und Interaktion eingeschrieben: »Software as a whole is not only ›code‹ but a symbolic form involving cultural practices of its employment and appropriation.«[119] Carl DiSalvo bestimmt, wie Software funktioniert, wie folgt, nämlich »through associations, interactions, and performances between and among individuals, groups, organizations and code.«[120] So zeigt sich eine spezifische Affordanz der Software, die von ihrer Codierung abhängig ist. Marissa Leavitt Cohn verortet Software »somewhere in the gray middle, somewhere between the excess of code's present performance, and the inevitable decay and obsolescence of software diagrams, documentation, and representations.«[121] Cohn gibt folglich das Diktum aus, Software ›präsent‹ zu halten. Diese Vorstellung des ständigen Updatens von Software setzt nicht nur Übersetzungs- und Interpretationsprozesse in Gang, sondern

116 Vgl. Laura Forlano: Materiality. Introduction. In: Janet Vertesi/David Ribes: *digitalSTS. A Field Guide for Science & Technology Studies*. Princeton UP 2019, 11–16, hier: 13.

117 Yuk Hui: *On the Existence of Digital Objects*. Minneapolis: University of Minnesota Press 2016.

118 Carl DiSalvo: Software. Introduction. In: Janet Vertesi/David Ribes: *digitalSTS. A Field Guide for Science & Technology Studies.* Princeton UP 2019, 365–368, hier 365.

119 Cramer: Language, 173. Und weiter heißt es: »But since writing in a computer control language is what materially makes up software, critical thinking about computers is not possible without an informed understanding of the structural formalism of its control languages.«

120 DiSalvo: Introduction, 365.

121 Cohn: Keeping Software present, 425.

stellt ins Zentrum, was Kittler die Korrigierbarkeit der Schrift nennt: »Die Korrektur von Geschriebenem scheint zunächst ein subjektiv willkürlicher Eingriff in den bereits geschriebenen Text. Aber sie gehorcht doch zugleich dem Text; sie tilgt, was nicht in ihn hinein gehört.«[122] Hier schließen wir wieder an die Wartungsarbeit als Sorge der digitalen Schriftlichkeit an, die dann auch eine politische Dimension erreicht, sobald das Codieren automatisiert wird.

Hatten wir bereits davon gesprochen, dass Codieren das Umformen von Zeichen meint, so lässt sich auch diese Prozedur automatisieren. Auf diese Weise ergeben sich Algorithmen. Algorithmus (eigentlich Algorismus),[123] die Latinisierung des Eigennamens des Universalgelehrten Abu Dscha'far Muhammad ibn Musa al-Chwārizmī,[124] der ein Buch über den Gebrauch indischer Zahlzeichen schrieb,[125] bezeichnet »[e]ine Regel, die erlaubt, eine Operation, die ausschließlich mit Zeichen arbeitet, als eine effektive Prozedur zu realisieren.«[126] ALGORITHM = LOGIC + CONTROL.[127] Die Formel bezeichnet, so Dotzler, dass jeder Algorithmus auf seine Effizienz hin spezifiziert werden kann und muss. Was zur Folge hat, dass es »keinen Algorithmus [gibt] ohne Befehlsgewalt.«[128]

Mit der Turingmaschine rücken Algorithmen ins Zentrum der »computing science«.[129] Algorithmen gehören der »pragmatic dimen-

122 Friedrich Kittler: *Baggersee. Frühe Schriften aus dem Nachlass*. Paderborn: Fink 2015, 168.

123 Vgl. Dotzler: *Papiermaschine*, 69.

124 Krajewski: Against Power, 122 und: Friedrich Kittler: Die Endlichkeit der Algorithmen: https://archive.transmediale.de/de/content/die-endlichkeit-der-algorithmen (zuletzt abgerufen am 23.11.2023).

125 Dotzler: *Papiermaschine*, 69.

126 Sybille Krämer: Wieso gilt Ada Lovelace als die ›erste Programmiererin‹ und was bedeutet überhaupt ›programmieren‹? In: dies. (Hg.): *Ada Lovelace: Die Pionierin der Computertechnik und ihre Nachfolgerinnen*. München: Fink 2015, 75–90, hier: 79.

127 Dotzler: *Papiermaschine*, 8: Die Formel geht auf Robert Kowalski zurück.

128 Dotzler: *Papiermaschine*, 8.

129 Andrew Goffey: Algorithm. In: Matthew Fuller (Hg.): *Software Studies. A Lexicon*. Cambridge MA/London: MIT Press, 15–20, hier: 16.

sion of programming« an, weil sie etwas tun.[130] Ihre Agentialität ist entscheidend. Mit anderen Worten: Algorithmen sind Sprechakte. Andrew Goffey begreift Algorithmus sogar als ›Aussage‹ im Sinne Michel Foucaults.[131] Während Foucault Aussagen an Subjekte koppelt, argumentiert Goffey, dass Algorithmen als Aussagen einen transversalen Maschinendiskurs konfigurieren, in denen Menschen und Maschinen als Subjekte artikuliert werden und aufeinander einwirken.[132]

Sybille Krämer bestimmt Algorithmen ferner anhand von vier Merkmalen: Allgemeingültigkeit, Finitheit, Eindeutigkeit, Variabilität.[133] Sie weist darauf hin, dass ein Algorithmus noch kein Programm sei: »Algorithmen sind das Rohmaterial für die Programmierung, das also, was die Schnürsenkel beim Schuhezubinden und was die Backzutaten für das Backen sind.«[134] Kittler definiert Algorithmen mit drei Begriffen: »Determiniertheit«, gleiche Bedingungen liefern immer wieder denselben Output, »Determinismus«, die Abfolge der Schritte ist immer gleich, und »Terminierung«, der »Algorithmus hält für jede Eingabe nach endlich vielen Schritten an«.[135] Glaubt man Kittler, hat Leibniz den Algorithmus eingeführt: »Leibniz hat 1684 den Calculus, die Differentialrechnung, ganz formal in ihren Grundregeln beschrieben und das dann Algorithmus genannt.«[136] Krämer präzisiert mit Bezug auf Leibniz und definiert Kalkül:

> »Ein Kalkül ist eine beschränkte Menge von Zeichen in Form eines Alphabets, Formationsregeln geben an, wie die Grundzeichen zu Ausdrücken (*expressiones*) aneinander gefügt werden, und Transformati-

130 Goffey: Algorithm,16f.

131 Goffey: Algorithm, 17.

132 Goffey: Algorithm, 19.

133 Krämer: Wieso gilt, 77f.

134 Krämer: Wieso gilt, 79.

135 Kittler: https://archive.transmediale.de/de/content/die-endlichkeit-der-algorithmen (zuletzt abgerufen am 23.11.2023).

136 Kittler: https://archive.transmediale.de/de/content/die-endlichkeit-der-algorithmen (zuletzt abgerufen am 23.11.2023).

onsregeln stellen Beziehungen zwischen den Ausdrücken her, damit diese ineinander umgeformt werden können.«[137]

Nach dem klassisch zu nennenden Handbuch *Introduction to Algorithms* ist der Algorithmus eine Funktion, die Werte nimmt (Input) und Werte ausgibt (Output).[138] Daraus resultieren zwei Fragen: Erstens, welche Input-Daten werden verwendet, und zweitens, wie funktioniert der Algorithmus? Beide Fragen umkreisen das Feld der digitalen Schriftlichkeit und ihrer Bedeutung für eine kritische Theorie der Mensch-Code-Schrift-Beziehung. Es ist nicht so wichtig, ob die *Blackbox* »Algorithmus« geöffnet wird. Denn wichtiger ist, zu verstehen, welchen Input der Algorithmus bekommt und woher der Input kommt.[139]

Algorithmen sind vor allem »networked information algorithm«.[140] Damit ist nicht nur der Algorithmus in der Vorstellung der Informatik gemeint, als Methode mit Input bestimmten Outcome zu generieren. Vielmehr wird der Algorithmus als Bestandteil eines Akteur-Netzwerks mit dem Menschen aufgefasst. Zu diesem Netzwerk gehören dann nicht nur Menschen und Maschinen, sondern normierte und institutionalisierte Algorithmen und Codes, menschliche Praktiken ihrer Anwendung sowie die daraus wiederverwendbaren Daten über die Nutzer:innen.[141]

Für die digitale Schriftlichkeit ist die Schrift nicht mehr nur wie bei Leibniz binär, also ausgehend vom chinesischen I Ching als »ideographic writing«[142] konzipiert und damit nicht mehr durch 1 und 0 bestimmt,

137 Krämer: Wieso gilt, 81.

138 Thomas H. Cormen/Charles E. Leiserson/Ronald L. Rivest/Clifford Stein: *Introduction to Algorithms*. Cambridge, MA: The MIT Press ⁴2022.

139 Mike Ananny: Toward an Ethics of Algorithms: Convening, Observation, Probability, and Timeliness. In: Science, Technology, & Human Values 4/1 (2016), 93–117, hier: 98.

140 Ananny: Ethics, 97.

141 Ananny: Ethics, 98.

142 Hui: *Digital Objects*, 16. Zu einer grundlegenden Kritik des Binären im Digitalen siehe: Aden Evens: *The Digital and Its Discontents*. Minneapolis: University of Minnesota Press 2024.

sondern »as the capacity to process data.«[143] Und auch Yuk Hui erkennt, dass Algorithmen an unseren Gedanken mitschreiben. Der Mensch ist dabei gestellt, im Sinne von Martin Heideggers Begriff des *Gestells*, in ein *Semantic Web* nach Tim Berners-Lee oder in Edgar Todds *relational database*. Damit ermöglichen Algorithmen eine tertiäre Protention der Schrift.[144] Hat Bernard Stiegler in Bezug auf Derridas Kritik an Husserl und den phänomenologischen Konzepten von primärer und sekundärer Retention eine dritte Form herausgearbeitet, wendet Yuk Hui dieses Ergebnis nun auch auf die Protention an, wodurch mit »digital writing, synchronization has become much more powerful, becoming the very foundation of what we call digital cultures.«[145]

Und weil Algorithmen in einem Beziehungsgeflecht zu ihren *User:innen* stehen, wie bewusst dieses Verhältnis auch immer sein mag, verändern sie sich mit ihrer Benutzung.[146] Es kommt zu einer Personalisierung, einer Kooperation zwischen Mensch und Maschine, Input und Output.[147] Das heißt, Benutzen und Schreiben fällt hier in eins. Aber auch das digitale Archiv spielt bei der Konfiguration von Subjekten und einer digitalen Kultur, wie Wolfgang Ernst analysiert hat, eine tragende Rolle.[148]

Algorithmen, die aus in sich geschlossenen Befehlsketten bestehen,[149] können als Code bezeichnet werden. Damit dieser Code ausgeführt werden kann, bedarf es erneut eines Körpers,[150] sprich ei-

143 Hui: *Digital Objects*, 25.

144 Hui: *Digital Objects*, 38.

145 Hui: Writing, 22.

146 Vgl. Dazu: Ulises Ali Mejias: *Off the Network. Disrupting the Digital World*. Minneapolis: University of Minnesota Press 2013.

147 Nick Seaver: Knowing Algorithms. In: Janet Vertesi/David Ribes: *digitalSTS. A Field Guide for Science & Technology Studies*. Princeton UP 2019, 412–422, hier: 415 und 419.

148 Vgl. Wolfgang Ernst: *Digital Memory and the Archive.* Minneapolis: University of Minnesota Press 2012. Wolfgang Ernst: *Das Rumoren der Archive*. Berlin: Merve 2002, 129–140.

149 Krajewski: Against the Power, 122.

150 Krajewski: Against the Power, 122.

nes Computers. Die Ausführbarkeit verändert aber den Algorithmus, wodurch der Schreibprozess nicht mehr an ein intentionales Autor:innensubjekt gekoppelt ist, sondern sich als ein Akteur-Netzwerk nichtmenschlicher und menschlicher *User:innen* erweist. Problematisch wird diese Beziehung erst dann, wenn es, wie Stiegler und McKenzie Wark kritisieren, asymmetrisch und intransparent wird.[151] Dann gilt tatsächlich jene erste Erwähnung in lateinischer Sprache, die den Algorithmus als Herrschaftszeichen einführt: »*Dixit Algoritmi: laudes deo rectori nostro*«.[152]

Gesetz und Befehl sind dem bzw. der Programmierer:in und dem bzw. der Coder:in nicht fremd, ganz im Gegenteil: »Code is law«.[153] Sie arbeiten am Algorithmus, der eine spezifische Ausführbarkeit gewährleistet. Der Umweg, der für eine andere Interpretation nötig ist, geht über eine Diskursanalyse, die »nach dem Algorithmus sucht«, wie Philipp Sarasin meint, d.h. nach dem Programmcode für eine spezifisch wiederkehrende Folge von Aussagen bzw. für die Möglichkeitsbedingungen von Aussagen und ihrer Ausschlussmechanismen.

Und weil *User:innen* »für Regierungen wie für Medienindustrien grundsätzlich nicht vertrauenswürdig [sind] und von den Potentialen fortgeschrittener Technik ferngehalten werden [müssen]«,[154] wird es umso erforderlicher, dass sich die Geisteswissenschaften mit den Skripten, dem Codieren, Programmieren und Prozessieren nicht nur von Computern, sondern auch der »Wetware« Mensch beschäftigen. Denn wir interpretieren zwar schon immer Daten, aber seit der Aufklärung – dem Befehl zum Selbstdenken – selten Befehle.

---

151 Stiegler: *Neganthropocene* und McKenzie Wark: *Das Kapital ist tot.* Berlin: Merve 2021.

152 Kittler: Endlichkeit: https://archive.transmediale.de/de/content/die-endlichkeit-der-algorithmen (zuletzt abgerufen am 15.11.2023): »Es sagt der Algorithmus gelobt sei Gott, unser Herr«.

153 Goffey: Algorithm, 19.

154 Heilmann: *Textverarbeitung*, 45.

Anstelle von Masterdiskursen müsste im 21. Jahrhundert die Rede von Masteralgorithmen sein,[155] die unseren Alltag aus Überwachung, Nudging, Social Media etc. prägen. Es wird dabei zur Aufgabe, diese Abhängigkeit, wie Nick Seaver erklärt, im Modus der Anagnorisis (wieder) zu erkennen[156] oder zu reparieren.

Eine Reparaturpolitik gegenüber dem Algorithmus fordern Julia Velkova und Anne Kaun. Sie hat das Austricksen von Algorithmen im Blick, rekurriert also einerseits auf die algorithmische Anagnorisis Seavers als auch auf die kapitalismuskritischen Einwürfe von Stiegler und Wark. Was sich abzeichnet, ist ein diskursiver Bruch in der Bedeutung von Algorithmen. Waren Algorithmen einst Bestandteile von Programmen, sind sie heute zunehmend eigenständige Akteure, die wie Quasi-Programme, Datenverarbeitung im Digitalen steuern und strukturieren. Algorithmen werden zu machtvollen Objekten stilisiert, auch durch kritische Forschung, die ihre politische, sozio-ökonomische und ästhetische Bedeutung betonen, interpretieren und analysieren.[157] Die digitale Schriftlichkeit der Codes zu analysieren heißt auch, nach dem »Verhalten«[158] von Schrift, spezieller von Code und Algorithmen zu fragen.

Der Zusammenhang von Stimme und Schrift wird auch im Kontext von Coding wichtig, da Algorithmen Stimmen »verstärken oder unterdrücken«.[159] Stimme ist im weiteren Sinne gemeint, im Sinne von »eine Stimme haben« oder »sich Gehör verschaffen«. Algorithmen haben also einen »material effect on end users«.[160] Dagegen bleibt das Coding, die

155 Vgl. Pedro Dominges: *The Master Algorithm. How the Quest for the Ultimate Learning Machine Will Remake Our World.* New York: Basic Books 2015.

156 Seaver: Knowing Algorithms, 414f.

157 Julia Velkova/Anne Kaun: Algorithmischer Widerstand. Medienpraxis und Reparaturpolitik. In: Daniel Neugebauer (Hg.): *Haut und Code. Publikationsreihe Das Neue Alphabet*. Band 5. Leipzig: Spector Books 2021, 59–82, hier: 63f. Vgl. auch weiterführend: Seb Franklin: *The Digitally Disposed. Racial Capitalism and the Informatics of Value*. Minneapolis: University of Minnesota Press 2021.

158 Velkova/Kaun: Algorithmischer Widerstand, 65.

159 Velkova/Kaun: Algorithmischer Widerstand, 66.

160 Goffey: Algorithm, 15.

Schrift des Algorithmus sekundär, als Hilfsmittel diese Stimme zu verstärken oder abzuschwächen.

Die Reparaturpolitik zielt darauf ab, die programmierten Algorithmen durch ein Andersnutzen in neue Bahnen zu lenken, sodass Rückwärtsbildersuchen, überhaupt Suchanfragen etc. diversere Ergebnisse zeitigen. Dabei geht es nicht darum, in die Schriftlichkeit des Codes einzugreifen und diesen umzuschreiben, sondern vielmehr in der Souveränitätssimulation Interface-Anwendungstaktiken zu entwickeln, die die Nutzung verbessern.[161] »›Reparatur‹ ist hier eine Metapher, die für den symbolischen Akt der Korrektur einer empfundenen ›Störung‹ eines algorithmischen Systems steht, womit auch die dominante Position des Systems selbst in Frage gestellt werden kann.«[162] Die Reparaturpolitik ist zudem an die Praxis des Einschreibens als Hacking gekoppelt,[163] gerade dort, wo die »Repair-Initiativen« im Sinne des Technikphilosophen Gilbert Simondon »vom unfertigen und verletzten Objekt her« gedacht werden.[164]

Folglich findet sich bereits bei Ada Lovelace nicht nur eine Ökonomie des Kalküls, wenn sie von der Optimierung des Codes spricht,[165] sondern auch eine potenzielle Reparierbar- und Korrigierbarkeit, die der Optimierung immanent ist. Zwar kann Optimierung im neoliberal-kapitalistischen Sinne als Effizienzmaximierung verstanden werden, und vor

161 Vgl. Velkova/Kaun: Algorithmischer Widerstand, 79.

162 Velkova/Kaun: Algorithmischer Widerstand, 80.

163 Zum Hacking vgl. u.a. McKenzie Wark: *Hacker Manifesto. A Hacker Manifesto*. Aus dem Englischen von Dietmar Zimmer. München: C.H. Beck 2005 und Sophie Toupin: Feministisches Hacking. Widerstand durch das Schaffen neuer Räume. In: Cornelia Sollfrank (Hg.): *Die schönen Kriegerinnen. Technofeministische Praxis im 21. Jahrhundert*. Wien: Transversal texts 2018, S. 33–58.

164 Barbara Eder: *Das Denken der Maschine. Marx, Mumford, Simondon*. Wien/Berlin: Mandelbaum 2023, 55.

165 Vgl. Bernhard J. Dotzler: Anmerkungen der Übersetzerin. Charles Babbage und Ada Augusta Lovelace in Kooperation. In: Sybille Krämer (Hg.): *Ada Lovelace. Die Pionierin der Computertechnik und ihre* Nachfolgerinnen. Paderborn: Fink 2015, 53–68, hier: 65.

allem geht es um eine Ökonomie des Kalküls, wenn die Zeit zur Berechnung minimiert werden soll, aber allein mit der Manipulation des Codes, der nicht von sich aus bereits ›perfekt‹ ist, eröffnet sich eine Möglichkeit des Eingreifens und des Widerstands. Und dieses Moment ist für die digitale Schriftlichkeit konstitutiv, weil sich hier ihre dekonstruktive Haltung erneuert.

In dieser Perspektive wird deutlich, welche Rolle eine digitale Schriftlichkeit spielen kann, und zwar in Bezug auf eine kritische (Quell-)Codeanalyse in soziopolitischen und kapitalistischen Zusammenhängen. Historisch lässt sich die generalisierende Amalgamierung von Schriftlichkeit und Digitalität als Grammatisierung mit Beginn des *World Wide Web* datieren: 1993.[166] Wie schon bei den ›networked information algorithms‹ geht es um das Zusammenwirken von nichtmenschlichen und menschlichen Akteur:innen. Mit Stiegler lassen sich die Schreibweisen näher bestimmen und unterscheiden.[167] Als erste Schreibweise kann das Codieren von ›networked information algorithms‹ bezeichnet werden, die von ihrer Benutzung ständig weiterentwickelt, mithin ›gefüttert‹ werden. Dieses Codieren ordnet alle menschliche Tätigkeit dem Paradigma des *content creating* unter,[168] das auch Wark analysiert.[169] Durch die digitale Grammatisierung entstehen neue Klassen, wie die »Vektorialistenklasse«, die den Vektor der Informationsflüsse besitzt und kontrolliert, und einem »Kognitariat«, das neue Information produziert. Dazu zählen Schriftsteller:innen ebenso wie Wissenschaftler:innen sowie viele andere mehr, die täglich etwas im Internet posten, verbreiten oder auf andere Art und Weise vernetzt sind.[170] Die andere Schreibweise bezeichnet ein Codieren als Produzieren von sogenannter ›free software‹, die nicht kapitalisiert und

166 Stiegler: *Neganthropocene*, 144.
167 Stiegler: *Neganthropocene*, 144.
168 Stiegler: *Neganthropocene*, 144.
169 Vgl. Wark: *Das Kapital*.
170 Wark: *Das Kapital*, 69.

damit nicht mit einem bzw. einer spezifischen Autor:in und als Werk mit einem urheberrechtlich geschützten Produkt einhergehen.[171]

Weil aber die erste Schreibweise der digitalen Grammatisierung die dominierende ist, entstehen so weitere prekäre Arbeitsfelder wie das der Commercial Content Moderators (CCMs), die »im Auftrag von externen Social-Media-Zulieferern und Drittanbietern an der Illusion einer unbedarften Medienpräsentation« arbeiten.[172] Diese CCMs durchsuchen die einzelnen sozialen Plattformen nach Bildern und Beiträgen, deren Inhalte »Bombenanschläge und Enthauptungen, Gewalt im Drogenkrieg, Spam, pornografische Bilder aus globalen Sexindustrien, Selbstmordnachrichten und Hilferufe von allen erdenklichen Orten der Welt« zeigen.[173] Diese ›Reinigungsarbeit‹ (im Gegensatz zur Politik der Reparaturarbeit) wird gerade nicht von Algorithmen übernommen. Sie zeigt nicht nur, wie sehr der Mensch als Arbeitskraft ausgebeutet wird, sondern erinnert einmal mehr an den materiellen Aufwand der scheinbar immateriellen digitalen Kommunikation.

4. Die Beiträge

Gabriele Gramelsberger macht in ihrem Beitrag *Von ›bits‹ zu ›words‹. John von Neumanns linguistic turn als Ursprung digitaler Schriftlichkeit* die Bedeutung der Schriftlichkeit für die Maschinensprache deutlich. Mittels der Transformationsschübe Mechanisierung und Elektrifizierung wird Sprache für Maschinen gebräuchlich. John von Neumann ist es, der Maschinencodes insofern normierte, als er ihre Sprache als Microcode vereinheitlichte. Diese »mnemotechnische Revolution des Maschinencodes«, so Gramelsberger, erlaubt es, Maschinen und Menschen sprachlich zu vernetzen. Das Ergebnis sind »autooperative Schriften«, die einen deklarativen Zeichengebrauch inaugurieren. Dadurch werden die protodigitalen Bedingungen der digitalen Schriftlichkeit sichtbar,

171 Stiegler: *Neganthropocene*, 144.

172 Eder: *Das Denken*, 17.

173 Eder: *Das Denken*, 17.

die sich aus dem *linguistic turn* von Neumanns ableiten. Darin liegt allerdings ein Problem: Das »a-word-at-a-time-thinking« des *linguistic turns* führt zu einer Fülle an Daten, die nicht mehr vom anthropomorphen Schriftgebrauch des Codes bewältigt werden können. Die digitale Revolution geht demnach mit einer Deanthropomorphisierung des Digitalen einher.

Tilo Reifenstein unterzieht in seinem Beitrag *(Elektrographisches) Schreiben als Praxis in Bild, Schrift und Material* das Schrift-Bildlichkeitsparadigma anhand der digitalen Schriftlichkeit einer kritischen Revision. Dabei betont er eine nicht nur medien-, sondern auch eine materialspezifische Betrachtungsweise der digitalen Schriftlichkeit. Im Rückgriff auf zeitgenössische kulturwissenschaftliche Theorien zur Schrift im Digitalen verortet Reifenstein diese gerade in der Bildmaterialität: ASCII, Unicode, und Optical Character Recognition (OCR)-Verfahren sind damit – ebenso wie digitale Schriftlichkeit – nicht ohne die Materialität einer Bild-Schriftlichkeit zu denken. Dabei adaptiert der Beitrag nicht einfach die Humboldt'schen Begriffe, sondern stellt gerade ihre anthropozentrische Perspektive heraus, um dagegen einen prozessontologischen Begriff digitaler Schriftlichkeit zu gewinnen.

Wilhelm von Humboldt steht im Fokus von Lucas Falkenhains Beitrag *Maschinenschreiben zwischen Energeia und Ergon. Eine sprachphilosophische Untersuchung von Large Language Models*. Falkenhain bezieht dabei Humboldts Unterscheidung von Energeia (Tätigkeit) und Ergon (Werk) auf die Mensch-Computer-Kommunikation, um die Agentialität maschinellen Schreibens zu erfassen. Ferner zeigt der Beitrag, dass mit Humboldts Konzept der Weltlichkeit algorithmische Sprachmodelle auf die Probe gestellt werden können. Das hat Konsequenzen nicht nur für das Maschinensprechen, sondern auch für die Kognitionswissenschaft und die Frage nach einer künstlichen Intelligenz.

Im Zentrum von Julia Nantkes Beitrag *Schreiben und Lesen als Mensch-Maschine-Kommunikation* steht eben jene titelgebende Relation von Mensch und Maschine. Ausgehend von Umberto Ecos Kommunikationsmodell, das auf die Passgenauigkeit und Kenntnis des verwendeten Codes verweist, basiert eine funktionierende Mensch-Maschine-Kommunikation auch auf kontinuierlichen Übersetzungen. Nantke differen-

ziert ihrerseits diese Konstellation aus: In Bezug auf Optical Character Recognition (OCR) lassen sich so Lesen und Schreiben als Decodierung begreifen. Ferner muss auch die Frage nach der Vorgabe des Zielcodes und seinem Autor:innensubjekt gestellt werden. KI-Trainingsprozesse sind dahingehend von den Trainer:innen und ihren ethischen Normen abhängig. Aber auch umgekehrt kann eine nicht-geleitete KI Muster erkennen, um ihre ›Botschaft‹ zu adressieren. Generative Algorithmen und maschinelles Lernen sind aber drittens nicht nur auf plausible Alltagskommunikation beschränkt. Vielmehr lässt sich mit Eco auch, wie Nantke argumentiert, die Maschine als ästhetische Codespenderin begreifen.

In seinem Beitrag *Romantische Maschinen oder: Ein Bericht für ein Literaturhaus* geht Philipp Schönthaler auf das Spannungsverhältnis von Computerliteratur und Romantik ein. Er zeigt, dass sich in der Gegenwartsliteratur – zum Beispiel in Daniel Kehlmanns *Mein Algorithmus und ich* – ein neues Verhältnis zu schreibenden Maschinen abzeichnet, das gar nicht so neu ist: Computer sind nicht mehr rationale, sondern romantische Schreibmaschinen. In der Gegenüberstellung des zeitgenössischen Diskurses mit der Romantik und der Computerliteratur der 1960er Jahre um Max Bense zeigt sich so eine Re-Romantisierung von Poesie, Werk und Autorschaft. Die Rationalität der Maschine wird gegen die Kreativität des Menschen ausgespielt, um eine Wieder-Verzauberung des Kunstwerks als unergründlich zu gewährleisten. Dabei ist die Maschine dort romantisch, wo sie gemäß der unendlichen Deutbarkeit und der Unergründbarkeit als KI zum Emblem dieses (Selbst-)Verständnisses wird.

Joachim Harst widmet sich in seinem Beitrag der postdigitalen Literatur Gregor Weichbrodts. In seinem Aufsatz *»What the Heck is a Book?« Code and Codex in Postdigital Literature* nimmt er Weichbrodts Text *I don't know* von 2014 zum Ausgangspunkt, die poetischen Verfahren algorithmischer Textproduktion zu untersuchen. Im Rückgriff auf den Begriff des »Postdigitalen« verbindet Harst das Algorithmische der Textproduktion mit dem Verhältnis von Code, Codex und Gesetz. Weichbrodts Text erscheint so als enzyklopädisch, insofern es Wissen und Nicht-Wissen ausstellt, das nur durch die algorithmische Verarbeitung der Wikipedia-

Datenbank erzeugt wird. Damit einher gehen eine subjektlose Autorschaft und eine Perspektive auf die Erhabenheit des Digitalen, die das Konzept ›Buch‹ auf die Probe stellt.

In ihrem Beitrag *Transitorische Literatur. Ebenen digitaler Schrift im Blog* Ze zurrealism itself begegnen Lore Knapp und Claus-Michael Schlesinger dem Problem des Transitorischen von Literatur auf Weblogs. Damit ist die Veränderbarkeit des einmal Geschriebenen gemeint, die das Schreiben im Internet erst ermöglicht. Sie gehen der Frage nach, wie die Textgenese eines fluiden digitalen Texts erfasst, beschrieben und analysiert werden kann. Es werden mehrere Ebenen digitaler Schriftlichkeit deutlich, die keineswegs einfach und vollständig erfasst werden können. Zwar lassen sich mithilfe des Internet Archives verschiedene Textstufen und -versionen rekonstruieren, aber ein ganzer Text bleibt eine Utopie. Vielmehr werden Praktiken des Schreibens im Internet sichtbar, die einen performativen Charakter besitzen.

Christian Schulz zeigt in seinem Beitrag *Vernakulärer Code oder die Geister, die der Algorithmus rief – digitale Schriftlichkeit im Kontext sozialer Medienplattformen*, dass sich digitale Schriftlichkeit nicht mehr nur im Backend finden lässt. Gerade beim Produzieren von Text auf sozialen Medienplattformen wird noch ein anderer Konnex deutlich: nämlich die Abhängigkeit von maschinellem Lernen, Algorithmen und Nutzer:innenverhalten. Schulz beschreibt das Interface, mit dem Nutzer:innen Text auf sozialen Netzwerken eingeben, als Schwelle gleich dreier Relationen, der Mensch-Maschine-, Maschine-Maschine- und einer Mensch-Mensch-Interaktion. Sichtbar werden diese Relationen und Interaktionen anhand von *Captions*. Unsichtbar dagegen sind diese dann, wenn tatsächlich Sichtbarkeit zum Thema wird und unter Umständen Postings und Kommentare nicht mehr im Feed erscheinen. Während Captions die Algorithmen ›füttern‹ und für Sichtbarkeit sorgen sollen, verweist das Nicht-Erscheinen von Posts im Feed auf die Interaktion von Algorithmen mit Nutzer:innen. Das Soziale ist von der Technik und damit von der digitalen Schriftlichkeit und *vice versa* abhängig.

Stefan Höltgen führt in seinem Beitrag *Humanities of the Digital. Philologische Perspektiven auf Source Codes als Beitrag einer computerarchäo-*

logischen Knowledge Preservation vor, wie eine Philologie der digitalen Schriftlichkeit aussehen kann. Programme und Programmiersprachen sind für Höltgen »Elaborate einer Textkultur«, die als kulturelle Artefakte erst philologisch als solche erfasst werden können. Anhand der Programmiersprache BASIC führt Höltgen vor, wie diese quellenhistorisch erfasst sowie sprach-, literatur- und bibliothekswissenschaftlich interpretiert werden kann. Erkennbar wird eine Programmierkultur, die mitnichten nur digital ist. Codes auf Papier, Notizen und Zeitschriften gehören genauso zum Aussagensystem BASIC wie die dazugehörigen Rechner, mit denen programmiert wurde und wird.

Als Begründer der Critical Code Studies führt Mark C. Marino in seinem Beitrag »*I hope you can read this*«. *Uncovering messages with Critical Code Studies* vor, wie eine solche Lektüre von Code eigentlich funktionieren kann. Er untersucht den Code der interaktiven digitalen Literatur The Gay Science von Capricorn van Knapp. Der Code Inform 7, der sich durch seine Nähe zur natürlichen Sprache Englisch auszeichnet, ist nicht nur der Hintergrund für die »interactive fiction«. Damit ein:e Leser:in das Spiel gewinnt, muss er/sie den Code lesen. Und weil Computercode eigentlich mehr für den Menschen als für den Computer geschrieben ist, wie Marino aufweist, zeigt sein Gegenstand, inwieweit der Code nicht nur Narrative, sondern auch Interaktionen bedingt. Dabei reicht es nicht, den Code wörtlich zu nehmen. Vielmehr bedarf es der literaturwissenschaftlichen Lese- und Interpretationstechniken, um den Code nicht nur in seiner Funktion für den Computer zu verstehen, sondern ihn auch in seinem sozio-kulturellen Kontext zu entschlüsseln. Daher plädiert Marino für einen interdisziplinären Ansatz, mit Informatiker:innen zusammenzuarbeiten und Code gemeinsam zu interpretieren. Denn jeder Code hat mehr an sich, als nur eine Maschine zum Ausführen eines Befehls zu bringen.

***

Der Band geht auf einen Online-Workshop zurück, der am 09.12. und 16.12.2022 an der Heinrich-Heine-Universität Düsseldorf stattfand. Die

Herausgeber danken der Universitäts- und Landesbibliothek Düsseldorf für die großzügige Förderung der Open-Access-Publikation. Wir danken ferner auch der Anton-Betz-Stiftung der Rheinischen Post e.V. für die Förderung der Drucklegung dieses Bandes.