–In W. BAUKNECHT, W. BRAUER, TH. MÜCK (eds.): *Informatik 2001. Tagungsband der GI/OCG Jahrestagung 2001*. Bd. II, 736-742–

Das algorithmische Zeichen

Frieder Nake
Universität Bremen
nake@informatik.uni-bremen.de

Zusammenfassung: *Nach gut dreißig Jahren akademischer Disziplin Informatik ist es an der Zeit, sich über Kern und Eigenart der Disziplin zu verständigen. Hier wird vorgeschlagen, sie als eine Technische Semiotik zu begreifen. Dafür ist es ein zentrales Anliegen, den neuen Begriff des algorithmischen Zeichens zu umreißen und zu definieren. Die geschieht in Abstützung auf die Semiotik von Peirce. Eingangs geben wir einen kurzen Einblick in bisherige Versuche, die Semiotik als Grundlage einer Theorie der Informatik zu reklamieren. Das Unterfangen ist vielversprechend, notwendig und lohnend.*

Schlüsselwörter: *Semiotik, Technische Semiotik, Theorie der Informatik, Zeichen*

1. Einleitend

Wenn eine wissenschaftliche Disziplin durch Stellen, Studiengänge, Institute, Konferenzen, Zeitschriften, Lehrbücher, durch Namen und Berufsbezeichnungen etabliert ist, wenn es (unter heutigen Umständen) ihren Propagandisten gelungen ist, Finanzmittel zu binden und fließen zu lassen, dann ist es an der Zeit, sich nach den üblicherweise von einer akademischen Disziplin zu erwartenden Merkmalen zu fragen. Was sind Deine kennzeichnenden Fragen? Welche Methoden erlaubst Du und welche Begriffe bildest Du zu ihrer Beantwortung? Welche Ergebnisse hast Du vorzuweisen?

An ihren Fragen, Methoden, Begriffen und Ergebnissen messen wir eine Disziplin und wollen dabei ihre Eigenständigkeit und Besonderheit in Trennschärfe erwarten dürfen. So borniert die Frage nach der trennenden Identität der Disziplin sein mag, so wollen wir sie doch stellen, da wir anders geneigt sein mögen, der Disziplin ihre Berechtigung abzusprechen und ihre Finanzmittel anders widmen zu wollen. Unser Fall ist die Informatik. Für sie hat Peter Denning 1991 deutlich die disziplinäre Frage gestellt [9]. Er hat die Beantwortung in einem beunruhigenden Zustand gelassen. Abstraktion und Design sind seither in der curricularen Ausrichtung der Informatik in den USA wichtige Begriffe geworden, eine Trennung in *Computing Science* und *Computer Engineering* wird häufiger.

Wir vertreten seit geraumer Zeit die Auffassung, es handele sich bei der Informatik um eine *Technische Semiotik*. Das wäre eine ins Technische gewendete Semiotik, eine technisch-konstruktiv orientierte Zeichentheorie. Das Verhältnis einer solchen Technik, einer Semiotechnik, zu ihrer theoretischen Herkunft, der Semiotik, wäre in Analogie z.B. zum Verhältnis der Elektrotechnik zur Physik zu sehen.

Andere Bezeichner sind für den gleichen Umstand vorgeschlagen worden: Semiotronics (Maranda [15]), Semiotic Engineering (Jorna, de Souza), Computational Semiotics. Ich sehe hier eine gewisse, wenn auch oberflächliche Bestätigung für die Auffassung, daß die Informatik es zwar mit Software auf ähnliche Weise zu tun habe wie andere Ingenieurdisziplinen mit ihren Gegenständen, daß diese Software-Befassung aber eine technische Befassung mit Zeichen und ihren Prozessen sei – mit Immateriellem also.

Der vorliegende Beitrag ist der Frage gewidmet, um welche besonderen Zeichen es sich auf dem Computer und in der Softwaretechnik handle. Ich bilde mir nicht ein, damit einen Beitrag zu leisten, aus dem sich eine effizientere, ökonomischere oder sonstwie geänderte praktische Entwicklung von Software herleiten ließe.

Einen Beitrag soll diese Notiz vielmehr zur disziplinären Begründung der Informatik leisten. Für die Softwaretechnik in enger Sicht bleibt, was hier folgt, folgenlos. Für die akademische Disziplin Informatik in weiterer Sicht hingegen behaupte ich, eine grundlegende Differenzierung leisten zu können. Zu ihr aber besteht eine Notwendigkeit gerade auch dann, wenn junge Menschen in hellen Scharen an die Tore klopfen, um an Segnungen teilhaben zu dürfen, die drinnen vergeben werden sollen. Es dürfte dabei doch wenigstens nicht begriffslos zugehen, will ich meinen.

Ich hole in Abschnitt 2 noch ein wenig aus, um den Ort dieser Anregung zu kennzeichnen. Danach referiere ich knapp den Begriff vom Zeichen, der verwendet wird. Abschnitt 4 legt dar, was ich unter einem algorithmischen Zeichen verstehen will. Dazu wird es vor allem wichtig sein, die Interpretationsleistung des Computers zu würdigen. Der Ausblick beschließt die Notiz.

Vorab muß ich einer angenehmen Pflicht nachkommen. Die hier formulierten Gedanken sind in vielen Stunden anregender Diskussion mit Peter Bøgh Andersen während meiner Tätigkeit an der Aarhus Universitet im Jahr 2000 entstanden. Er hatte mir nicht nur die Gelegenheit zu diesem fruchtbaren Aufenthalt verschafft, sondern darüber hinaus in beispielloser Großzügigkeit Fragen aufgeworfen und Vermutungen erörtert. Wir arbeiten seither an einem umfangreichen Manuskript, zu dem er mittlerweile viel mehr geliefert hat als ich.

2. Geschichtlich ausholend

Bevor es die Informatik als eigenständige akademische Disziplin gab, gab es den Computer. Er ist das allgemeine Mittel der Maschinisierung von Kopfarbeit [18]. Soll eine konkrete geistige Tätigkeit dem Prozeß der Maschinisierung unterworfen werden, so muß ihr ein Korrelat in Form spezieller Software konstruiert werden.

Dies hatte ich gelegentlich so zu kennzeichnen versucht, daß der Wirklichkeit eine Zeichenhaut zuwachsen müsse. Schon immer ist klar, daß Software als *Beschreibung* aufzufassen ist. Solch eine Beschreibung weist jedoch eine merkwürdige Eigenheit auf: Software ist einerseits *Text*, andererseits *Maschine*. Sie *ist* Maschine nur als Text, als Text also, der wirken kann, als wäre er selbst Maschine. Software ist Maschine zunächst nicht in höherem Maße als ein Blatt beschriebenen Papieres das ist. Software aber *ist* Text als Maschine, Maschine also, die gelesen werden kann, als wäre sie Schrift.

Es soll hiermit gesagt werden, daß eine Redeweise wie ”Software ist Maschine” falsch, zumindest problematisch ist. Software weist Merkmale von Maschinen auf und weist sie nicht auf. Nur in Funktion weist sie sie auf; in Ruhe ist sie beschreibender Text. Jeder weiß, daß das Geheimnis darin liegt, daß dieser Text exekutierbar ist und daß eben darin seine Maschinenhaftigkeit besteht.

Software *ist* naturgemäß weder das eine (Text) noch das andere (Maschine). Denn wenn ich ein Stück Software nehme und auf den Tisch lege und den Einschaltknopf für die Maschine suche, schon dann scheitere ich. Wenn ich deswegen anfange, die Schwärzungen auf Papier zu lesen und vorzutragen, wie ich das mit Texten zu tun gewohnt bin, so scheitere ich ein zweites Mal. Software *ist* eben Software, zunächst und vor allem. Erst wenn wir ihr diese ihre Eigenständigkeit und Besonderheit zugestehen, kommen wir in die Lage, die Funktionen genauer zu beschreiben, die mit ihr aufgekommen sind. Dazu aber müssen wir auf andere, uns bekannte Erfahrungsbereiche zurückgreifen. Vermutlich werden wir Metaphern benutzen müssen.

Der völlig ungewohnte und befremdend immaterielle Charakter der Produkte der Softwaretechnik gab deswegen schon zu Anfang der siebziger Jahre all jenen ein ontologisches und erkenntnistheoretisches, ein ökonomisches oder ein psychologisches Rätsel auf, die damit nicht zufrieden waren, daß die Informatik nun als eigenständige universitäre Disziplin existierte, sondern die eben wegen der Disziplinbildung hinter den Schleier blicken und das Geheimnis lüften wollten.

Von der epochemachenden Anstrengung um die präzise Definition einer umfassenden Programmiersprache herkommend, nämlich der Definition von Algol60, hatte es einen Schimmer von Wittgenstein-Rezeption gegeben. Die Semiotik blitzte kurz auf, und Heinz Zemanek [30] und Saul Gorn [12] gaben die ersten Hinweise darauf, daß die Zeichenthematik der Informatik gut anstünde. Dies war in der Zeit, als in der BRD die Gründung der Informatik-Studiengänge diskutiert wurde. In ihrer verborgenen Weitsicht und gewiß eher technischen Prägung wirken die Texte heute fast rührend. Wurden sie aufgegriffen? Naheliegenderweise enthält ein Buch zur Ästhetik als Informationsverarbeitung zum ersten Mal das Zeichenthema als roten Faden [17].

Die semiotische Sichtweise auf die Software wird zu Beginn der neunziger Jahre durch die Debatte um eine Theorie der Informatik wieder belebt [8]. Um diese Zeit erscheinen etliche Versuche, Aspekte der Informatik semiotisch zu dekonstruieren. Mit ein wenig Übertreibung können wir behaupten, daß sich die tieferen Schichten der Wirklichkeit nach langem Schlummer Bahn brechen und die Sphäre der Erkenntnis erreichen. An den Rändern der Disziplin dringen Begriffe in das Schaffen der Softwaretechniker ein: Medium, Spiel, Design, Zeichen [19].

Ausser allgemeinen Überlegungen, den Charakter von Software zwischen Maschine und Text betreffend bzw. die Theorie der Programmiersprachen fundierend, gibt es bis in die frühen siebziger Jahre hinein keine semiotischen Ansätze. Allerdings begründet Ronald Stamper schon 1973 mit seinem Buch [28] eine Tradition der semiotischen Auffassung von Information in Organisation und Management, die sich bis heute zu einem eigenen Weg praktischer Relevanz entwickelt hat.

Mihai Nadin ist wohl der Semiotiker überhaupt, der die Bedeutung der Zeichenthematik für die Software erkannt hat und am frühesten und klarsten darauf aufmerksam gemacht hat, daß Software sowohl Ingenieurprodukt wie Zeichenarrangement ist [16]. Eher aus Sicht der Anwendung von Software ist der noch frühere, nicht leicht zugängliche Aufsatz [7] zu sehen.

Schon 1988 gibt Baron im Rahmen einer allgemeineren semiotischen Aufsatzsammlung einen Hinweis auf die semiotische Sichtweise der Berechnung [5]. In einer vom Philosophen Max Bense und Informatiker Walter Knödel betreuten Dissertation nimmt Jorge Bogarin 1989 eine semiotische Analyse der drei klassischen Informatik-Gegenstände – Algorithmen, Automaten und formale Sprachen – vor [6]. Von Fragen der Kognitionsforschung und Künstlichen Intelligenz herkommend publiziert René Jorna ab 1990 mehrere Aufsätze und Bücher zur semiotischen Betrachtung [13], [14]. Er prägt die Bezeichnung ”Semiotic Engineering”, auf die unabhängig von ihm

auch Clarisse Sieckenius de Souza stößt. Sie nimmt das semiotische Thema während eines Aufenthaltes an der Stanford University bei Terry Winograd auf [27] und führt es mittlerweile in einer Forschungsgruppe an der PUC in Rio de Janeiro in hoher Konzentration fort.

Eine physikalisch abgestützte semiotische Auffassung des Umgangs mit Computern kommt von Figge in Bochum [11]. Seit mehreren Jahren befaßt sich eine Gruppe im Rahmen der IFIP mit diffizilen Fragen der Begrifflichkeit von Informations-Systemen. Fast bin ich geneigt zu sagen, daß sie bei ihrem Bemühen um Normen zwangsläufig auf die zentrale Rolle der Semiotik stoßen mußten [10].

Die erste internationale Tagung zum Thema, *Informatics and Semiotics*, ein Dagstuhl-Seminar, stößt der Autor 1996 an [3]. Auf einer weiteren Arbeitstagung, die der Theorie der Informatik gewidmet war, war im Frühjahr 2001 in Heppenheim der semiotische Aspekt einer von dreien, unter denen die Frage nach der Disziplin Informatik gestellt wurde [22]. Dem generell semiotischen Charakter der Computerdinge wird auf allgemeiner Ebene in [21] nachgegangen.

Die bisher herausragendste Einzelarbeit aber ist die Dissertation von Peter Bøgh Andersen von 1990 [1]. In ihr unternimmt er den Versuch, die Besonderheiten der Zeichen auf dem Computer zu identifizieren. Er stützt sich dabei auf den dänischen Semiotiker Louis Hjelmslev (1899-1965), der eine eigene Begrifflichkeit für die immer wieder neu beschriebene Dialektik von Form und Inhalt entwickelt hat. Es ist bemerkenswert, wenn auch nicht überraschend, zu beobachten, daß sich auch Andersen mittlerweile eher an Peirce orientiert. Seine Arbeiten zur semiotischen Auffassung der Informatik beginnen Mitte der achtziger Jahre. Sie werden vom Rande der Informatik, von den Informationswissenschaften her, angestellt und sind die eines Sprachwissenschaftlers.

In dem von Andersen et al. herausgegebenen Band [2] findet sich eine Begründung der medialen Auffassung vom Computer, in der zentral semiotisch argumentiert wird.

Auffallend ist, dass alle Autoren, die eine semiotische Argumentation um algorithmische Prozesse herum aufbauen, sich auch einmal speziell mit der Interaktion von Mensch und Computer befassen. Keine Frage: hier, auf der Ebene der Benutzung, wo es der Erscheinung nach kommunikativ zugeht, *müssen* die Zeichen auftreten, wenn eine theoretische Fundierung gesucht wird [20], [27], [25].

3. Das ”Zeichen” referierend

Es sei nun der Begriff von ”Zeichen” umrissen, den ich zu Grunde lege. Im Rahmen dieses Beitrages kann es nicht darum gehen, den Begriff aus der Rezeption der einschlägigen Literatur heraus und in Abgrenzung gegenüber wichtigen anderen Begriffen vom Zeichen darzulegen. Das bleibt ausführlicheren Abhandlungen überlassen [4], muß letzten Endes aus der Semiotik selbst kommen.

Der glückliche Umstand, bei Max Bense Vorlesungen besucht zu haben, hat mir frühzeitig die Semiotik von Charles S. Peirce nahegebracht. Ich habe mich von ihr nicht lösen können oder wollen. Es hat allerdings auch den Anschein, als sei die dreistellig relationale, prozeßhafte und rekursive Auffassung vom Zeichen, die ich aus Peirce‘ Schriften herauslese, für die Belange der Informatik und Software besonders geeignet. Die folgende Kennzeichnung stützt sich auf [24]. Die bis vor etlichen Jahren noch relativ umständlich zugänglichen Schriften des amerikanischen Genius werden in letzter Zeit vielfältig aufbereitet und verlegt. Eine exzellente allgemeine Quelle zur Semiotik ist [23].

Zeichen ist zu allererst Relation und nicht Ding. Wir finden Zeichen nicht draußen in der Welt irgendwo vor, schon gar nicht in der Natur. Wir stellen Zeichen her. Ohne aktiven Eingriff des Menschen gibt es kein Zeichen.

Als Relation setzt das Zeichen mehrere Gegenstände ins Verhältnis zueinander. Seine semiotische Natur zeigt sich im Verhältnis der aufeinander bezogenen Gegenstände: im Bezogen-Sein und nicht im Nur-Sein. Im Falle der Peirceschen Semiotik werden drei Gegenstände im Zeichen ins Verhältnis zueinander gesetzt: ein Repräsentamen, ein Objekt und ein Interpretant. Wir erläutern diese Begriffe weiter unten, geben aber erst ein Beispiel.

In der Regel denken wir im Kontext der Informatik bei ”Zeichen” an Lichterscheinungen auf dem Bildschirm. Greifen wir deswegen eine solche als erstes Beispiel heraus! Denken wir an eine kleine Zeichnung (jetzt meist ”Icon” genannt) auf der Benutzungsoberfläche eines Betriebssystems. das Gebilde wird oft ”Folder” oder ”Ordner” genannt.

Diese Lichterscheinung selbst ist *kein* Zeichen. Sie ist *Teil* eines Zeichens, besser: sie wird von uns zum Teil eines Zeichens gemacht. Sie ist zunächst einmal sie selbst und nur sie selbst: Licht bestimmter Farbe, an bestimmtem Ort, in bestimmter Form. Der Ort kann gern wechseln, die Farbe können wir setzen, die Größe wählen. Es bleibt die Form.

Diese Lichterscheinung machen wir, wenn wir das Betriebssystem kennenlernen, zur sinnlich wahrnehmbaren Seite eines Zeichens. Wir setzen sie ins Verhältnis zu einem Gegenstand auf dem Computer, zu einem Stück Software. Dieser Gegenstand ist im Beispiel ein ”Verzeichnis” von anderen Computerdingen. Das Verzeichnis sagt uns in Form einer Namensliste (und einiger weiterer Angaben, die jetzt nichts zur Sache beitragen), was alles hier zu einer Einheit zusammengefaßt ist.

Die Lichterscheinung bezeichnet also das Verzeichnis. Mit ihm wollen wir umgehen. Wir gehen mit ihm um, indem wir die Lichterscheinung manipulieren. Hinter ihr aber wirken wir – wirkt die Software – auf ein Objekt (eine Datei) ein.

Die Bezeichnungsfunktion macht das Zeichen zu einer zweistelligen Relation. In ihr wird das sinnlich wahrnehmbare Zeichensubstrat für ein Anderes, ein Abwesendes genommen. Das Repräsentamen (die Lichterscheinung) steht für das Objekt (das Verzeichnis). Oft begnügt man sich mit solch einer dyadischen Relationalität des Zeichens. Wir tun dies mit Peirce ausdrücklich *nicht* und weisen hier schon vorsorglich darauf hin: das Repräsentamen ist *nicht* das Zeichen. Es ist nur die handgreifliche Seite des Zeichens, das, woran wir uns halten können.

Zeichen wird gesetzt. Es wird gesetzt, indem ein materielles Substrat, das *Repräsentamen*, gewählt wird. Dieses wird in eine Bezeichnungsrelation zu einem (abwesenden, gedachten, vergangenen, ...) *Objekt* gebracht. Die Relation von Repräsentamen und Objekt wird in eine weitere Relation, die Bedeutungsrelation, zu einem *Interpretanten* gebracht. Wenn das Zeichen im Repräsentamen abgekürzt handgreiflich erscheint, so wird es im Interpretanten als Sinn, Absicht, Bedeutung zusammengefaßt. Das Repräsentamen sehen wir, den Interpretanten denken wir.

Noch einmal sei ein Blick auf das Beispiel geworfen. Die Lichterscheinung des Icon steht für das Verzeichnis der zusammengefassten Dateien. Sie beide bedeuten mir etwas in der Anwendung: hier, im "Ordner", habe ich alles zusammengefaßt, was ich für diesen Aufsatz benötige, vielleicht. Der pragmatische Zusammenhang meines Umgangs mit den Dingen also stiftet die Bedeutung, führt zum Interpretanten. Auch umgekehrt und häufiger gilt: meine Absicht, meine Arbeit auf bestimmte Weise zu organisieren, führt mich zuerst zu einem Interpretanten. Zu ihm passend schaffe ich mir dann ein Paar aus Repräsentamen und Objekt.

Es sei sogleich angemerkt, daß die Welt nicht in Repräsentamina, Objekte und Interpretanten zerfällt. Was jetzt als Objekt in Erscheinung tritt, kann dann Repräsentamen werden oder auch Interpretant.

Die dreistellige Relationalität, die Peirce Zeichen nennt, ist nun darüber hinaus rekursiv und dynamisch. Sie ist rekursiv in folgendem Sinn. Fragen wir genauer nach dem Interpretanten, wollen wir ihn explizit haben, so müssen wir selbst wieder zum Zeichen greifen. Der Interpretant ist ein Zeichen, das vom Paar (R, O) hervorgerufen wird oder das zu diesem Paar Anlaß gibt. Im Zeichen steht ein Erstes für ein Zweites vermittels eines Dritten. Dieses Dritte aber ist von der gleichen Art wie das Ganze, als dessen Teil wir es denken.

Mit diesem in sich rekursiven Begriff von Zeichen haben wir einen sehr allgemeinen, hochabstrakten, aber auch sehr mächtigen Zugang zu den informatischen Gegenständen. Er scheint diesen geradezu auf den Leib geschneidert zu sein. Denn die informatischen Gegenstände sind die Gegenstände der praktischen Rekursivität.

Im rekursiven Charakter des Zeichens haben wir gleichzeitig die Wurzel seiner Dynamik. Zeichen treten nie einzeln, vereinzelt und isoliert auf. Wir sind immer schon von ihnen umgeben. Das Zeichen ist Zeichen nur im Zeichenprozeß. Ein Zeichen trennt eine Zeichensituation von einer anderen. Wir können nicht anders, als im Zeichen zu denken. Jeder Gedanke ist für Peirce Zeichen. Das Zeichen ist dem Denken nicht äußerlich, ist nicht Träger von etwas anderem

Vieles mehr wäre zu sagen – doch nicht hier. Wenden wir uns der Ergänzung im Zeichenbegriff zu, die wir vornehmen, um der Informatik ein allgemeines begriffliches Fundament zu schaffen.

4. Das "algorithmische Zeichen" einführend

Die Entwicklung von Unix und die davon völlig unabhängige Verschiebung der Anwendungen von Software auf den Sektor von Büro und Verwaltung hatten Software in hohem Maße unter die Metapher vom *Werkzeug* gestellt. Die weitere Verschiebung des Brennpunktes hin auf Spiel, Unterhaltung, Lernprozesse und kulturelle Phänomene ganz allgemein hat die Auffassung vom Computer als Medium bestärkt. Wir kennzeichnen die digitalen Medien genauer *als instrumentale Medien* [26] und erblicken darin eine Kategorie, die dem schillernden Da- und So-sein von Software auf angemessene Weise nahekommt.

Die Frage nach einer stabilen und soliden theoretischen Begründung von Informatik, Software und vielleicht auch Softwaretechnik läßt sich semiotisch befriedigend beantworten. Die Antwort ist überraschend schlicht. Wir haben in Software eine *besondere Art von Zeichen* vor uns. Diese Zeichen werden stets und ständig und unausweichlich auf doppelte Weise interpretiert, vom Menschen einerseits, vom Computer andererseits, gleichzeitig und konkurrierend. Dies gilt es zu erklären.

Bevor ich das jedoch tue, belege ich die Ergebnisse der beiden Interpretationsversuche mit Namen. Wir hatten gesehen, daß im Zeichen ein Repräsentamen für ein Objekt vermittels eines Interpretanten steht. Diesen Zusammenhang stellt eine interpretierende Instanz her: der Mensch. Das Geschehen am Computer führt uns dazu, ein wenig gegen die eigenen Vorurteile den Computer als eine andere solche Instanz anzunehmen. Wir werden gleich sehen, daß dies nur in einem sehr eingeschränkten Sinne gelten kann (und wollen uns auf keinen Fall auch nur in die Nähe jener Auffassung rücken lassen, die eine wie immer geartete Gleichheit von Mensch und technischem System behauptet).

Wenn wir Menschen und Computer auf einer bestimmten Betrachtungsebene als zwei interpretierende Instanzen für Zeichenprozesse zulassen, so schlägt sich ihre Aktivität, semiotisch gesehen, in Zeichenketten nieder. Wir sind an solchen Vorgängen interessiert, bei denen in einer Art von Arbeitsteilung manche Operationen im Rahmen einer Handlung vom Computer, andere aber vom Menschen ausgeführt werden. Für das Gelingen solcher arbeitsteiliger Prozessketten wird der Computer

heute standardmäßig interaktiv benutzt. Die interaktive Benutzung aber weist an der Oberfläche Merkmale kommunikativer Art auf. Sie ist selbstredend Kommunikation nur in einem sehr eingeschränkten Sinn, den wir hier nicht zu problematisieren haben. Es ist jedoch nützlich geworden, die allgemeine Verbreitung des Computers, seine damit einhergehende Medialität und die Verschiebungen im Alltagsbewußtsein der Menschen in dem Sinne aufzunehmen, daß wir die Interaktion als eine Quasi-Kommunikation betrachten. Es handelt sich um Zeichenprozesse – um Zeichenprozesse, die durch einen ständigen, unmerklichen und glatt funktionierenden Wandel von Zeichen in Signale und wieder in Zeichen usf. gekennzeichnet sind. Was eben, "draußen", beim Menschen noch Zeichen (dreistellig) ist, ist sofort nach Durchgang durch die Schnittstelle, "drinnen" im Computer nur noch Signal (einstellig) (vgl. genauer [20]).

Signale sind eine besondere Zeichenart. In ihnen fallen Interpretant und Objekt zusammen. Die Bedeutungsrelation geht auf in der Bezeichnungsrelation. Damit geht die interpretierende Vielfalt verloren.

Existiert nun auf der Seite der zeichenstiftenden Instanz darüber hinaus kein Bewußtsein, d.h. werden dort Zeichen nicht aus freien Stücken geschaffen, sondern in Form von Befehlen nur empfangen, so besteht nicht einmal die Möglichkeit, ein und dasselbe Repräsentamen zu unterschiedlichen Objekten ins Verhältnis zu setzen – oder, wo dies doch der Fall sein sollte (bei einem *Overloading*), kann dies nur auf festgelegt-eingeschränkte Weise geschehen.

Mit einer solchen Quasi-Interpretationsinstanz haben wir es beim Computer zu tun. Wir können ihm keinerlei Intelligenz zumessen, müssen aber feststellen, daß auf ihn einzelne geistige Tätigkeiten in formalisierter und dann maschinisierter Form durchaus übertragen werden können. Das dann stattfindende Geschehen läßt sich semiotisch gut fassen, wenn wir die gemachten Einschränkungen im Auge behalten. Dies gelingt, weil die Peircesche Semiotik eine *formale* Theorie ist, die uns Beschreibungsmittel, aber keine Erklärungen liefert.

Wir stehen also vor einer Situation folgender Art: Ein Mensch, ein Computer (Software), ein Zeichen (z.B. unser Dateiorganisations-Beispiel von oben). Offensichtlich machen sich sowohl Mensch wie Computer an dem Zeichen zu schaffen. Das Angenehme an der Situation ist, daß die Daten im Computer gespeichert sind und auf vielfache Weise manipuliert werden können. Soll dies geschehen, greift der Mensch zum Zeichen. Über Zeichen teilt er der Software mit, was zu geschehen hat. Es gibt also an der Oberfläche der Benutzung einen Zeichenprozeß zwischen Mensch und Computer. Insofern muß es, der formalen Theorie folgend, je einen Interpretanten geben. Wir nehmen für den Zweck einer ersten erweiterten Begriffsbildung Repräsentamen und Objekt des Zeichens als fest an. Im Interpretanten aber unterscheiden wir seine *intentionale* Seite (beim Menschen) von seiner *kausalen* Seite (beim Computer).

Wir beabsichtigen damit folgendes. Die Interpretation, die der Mensch einem Software-Element oder -Ereignis zukommen läßt, ist von vollständig anderer Art als jene Interpretation, die der Computer zum gleichen Zeitpunkt und aus gleichem Anlaß leistet. Ihm geht es so wie jeder anderen Maschine auch: er kann gar nicht interpetieren, wenn wir unter "Interpretation" die Zuschreibung einer handlungsrelevanten Bedeutung verstehen wollen, die der Situation, dem Kontext und dem Interesse eines lebendigen körperlichen Wesens eigen ist.

Die Interpretationsleistung des Computers ist der Grenzfall einer Interpretation: die Entscheidung für eine Zuschreibung aus einer Menge möglicher Zuschreibungen (intentional) schrumpft zusammen auf die Bestimmung der im allgemeinen Schema vorgesehenen und vorher bestimmten Zuschreibung (kausal). Wir nennen diesen Grenzfall *Determination*.

Interpretation findet durch Herstellen und Auswählen von Kontext statt. Determination findet im Rahmen eines gesetzten und unverrückbaren Kontextes statt, des Kontextes der Berechenbarkeit nämlich. Die Interpretation des Computers ist die präzise und wiederholbare Ausführung einer berechenbaren Funktion. Wir wären unzufrieden, wenn es anders wäre.

Das *algorithmische Zeichen* ist also ein Zeichen, das durch einen gleichzeitigen Vorgang der Interpretation und Determination bestimmt wird. Dieser doppelte Vorgang führt zu einer Aufspaltung in mindestens einer der Dimensionen des triadischen Zeichens. Wir haben hier andeutungsweise eine erste solche Aufspaltung im Interpretanten betrachtet: in einen intentionalen und einen kausalen Interpretanten. Andere Differenzierungen (im Objekt oder Repräsentamen oder in mehreren gleichzeitig) bleiben vorbehalten.

Was gewinnen wir begrifflich durch solche einen Schritt? Wir können damit jedes Geschehen zwischen Menschen in ihren verschiedenen Rollen und dem Computer differenziert, jedoch auf einheitlichem Grund beschreiben. Dieser Grund ist nicht willkürlich ein semiotischer, sondern als solcher naheliegend: es geht um quasi-kommunikative Vorgänge. Wir fassen z.B. Software im Zustand ihrer Entwicklung als Herstellung und Gestaltung algorithmischer Zeichen auf, also von Zeichen, deren künftige Doppelinterpretation es zu gestalten gilt. Das Phänomen der Software-Ergonomie hat hier seine Wurzel. Software im Zustand ihrer Benutzung dagegen ist die Verwendung algorithmischer Zeichen, also die Aufspaltung in den Doppelprozeß, was zu Irritation Anlaß geben kann.

5. Ausblickend

In dem hier vorgeschlagenen Begriff des algorithmischen Zeichens meine ich, eine theoretische Grundlage zu sehen, die zusammen mit der ökonomischen Betrachtung

(hier ausgespart: Maschinisierung von Kopfarbeit) ein Fundament der Informatik als einer wissenschaftlichen Disziplin abgibt. Die Probleme erscheinen damit – arbeiten wir den Ansatz im einzelnen aus – als im Wesentlichen gelöst. Mit Wittgenstein, der so im Vorwort zum Tractatus spricht, sehen wir aber auch weiter, "wie wenig damit getan ist, daß die Probleme gelöst sind" [29].

Literatur

[1] Peter Bøgh Andersen: *A theory of computer semiotics. Semiotic approaches to construction and assessment of computer systems.* Cambridge: University Press 1990

[2] P. B. Andersen, B. Holmqvist, J. F. Jensen (eds.): *The computer as medium*. Cambridge: University Press 1993

[3] Peter Bøgh Andersen, Mihai Nadin, Frieder Nake (eds.): Informatics and Semiotics. *Dagstuhl-Seminar-Report* 135, Wadern: IBFI Schloß Dagstuhl 1996

[4] Peter Bøgh Andersen, Frieder Nake: *Informatics and Semiotics*. In Vorber.

[5] Naomi S. Baron: Low on RAM. The semiotics of computing. In Th. A. Sebeok, J. Umiker-Sebeok (eds.): *The semiotic web*. Berlin. New York: Mouton de Gruyter 1988. 369-386

[6] Jorge Bogarin: *Semiotik der Automaten, Algorithmen und Formalen Sprachen*. Diss. Universität Stuttgart 1989

[7] Barron Brainerd: Semiotics and the computer. *Recherches Sémiotiques* 4 (1984) 79-87

[8] W. Coy, F. Nake, J. Pflüger, A. Rolf, J. Seetzen, D. Siefkes, R. Stransfeld (Hrsg.): *Sichtweisen der Informatik*. Braunschweig: Vieweg 1992

[9] Peter Denning: Technology or management? Editorial *Comm. ACM* 34,3 (March 1991) 11-12

[10] E. Falkenberg, W. Hesse, P. Lindgreen, B.E. Nilsson, J.L.H. Oei, C. Rolland, R.K. Stamper, F.J.M. Van Assche, A.A. Verrijn-Stuart, K. Voss: FRISCO - A framework of information system concepts - *The FRISCO Report*. IFIP WG 8.1 Task Group FRISCO, Univ. of Leiden, Dec. 1998

[11] Udo L. Figge: Computersemiotik. *Z. f. Semiotik* 13, 3/4 (1991) 321-330

[12] Saul Gorn: The identification of the computer and information sciences: their fundamental semiotic concepts and relationships. *Foundations of Language* 4 (1968) 339-372

[13] René J. Jorna: Wissensrepräsentation in künstlichen Intelligenzen. Zeichentheorie und Kognitionsforschung. *Z. f. Semiotik* 12 (1990) 9-23

[14] R. J. Jorna, B. van Heusden, R. Posner (eds.): *Signs, search and communication. Semiotic aspects of artificial intelligence*. Berlin: de Gruyter 1993

[15] Pierre Maranda: Semiotics and computers: the advent of semiotronics? In T.A. Sebeok, J. Umiker-Sebeok (eds.): *The semiotic web*. Berlin: Mouton de Gruyter 1988, 507-533

[16] Mihai Nadin: Interface design and evaluation – semiotic implications. In H. Rex Hartson, Deborah Hix (eds.): *Advances in human-computer interaction*, Vol. II. Norwood, N.J.: Ablex 1988. 45-100

[17] Frieder Nake: *Ästhetik als Informationsverarbeitung*. Wien, New York: Springer 1974

[18] Frieder Nake: Informatik und die Maschinisierung von Kopfarbeit. In: W. Coy et al. (Hrsg.): *Sichtweisen der Informatik*. Braunschweig, Wiesbaden: Vieweg, 1992. 181-201

[19] Frieder Nake (Hrsg.): *Die erträgliche Leichtigkeit der Zeichen. Ästhetik, Semiotik, Informatik.* Baden-Baden: Agis-Verlag 1993

[20] Frieder Nake: Human-computer interaction. Signs and signals interfacing. *Languages of Design* 2 (1994) 193-205

[21] Frieder Nake: Der semiotische Charakter der informatischen Gegenstände. *Semiosis* 85-90 (1997) 24-35

[22] F. Nake, A. Rolf, D. Siefkes (Hrsg.): *informatik: aufregung zu einer disziplin.* Papiere zur Arbeitstagung in Heppenheim, 6.-8.4.2001 (unveröff.)

[23] Winfried Nöth: *Handbuch der Semiotik*. Stuttgart: Metzler (2. Aufl.) 2000

[24] Charles S. Peirce: *Phänomen und Logik der Zeichen*. Frankfurt: Suhrkamp 1983, 1993

[25] R. Oliveira Prates, C. Sieckenius de Souza, A. C. Bicharra Garcia: A semiotic framework for multi-user interfaces. *ACM SIGCHI Bulletin* 29,2 (April 1997) 28-39

[26] Heidi Schelhowe: *Das Medium aus der Maschine.* Frankfurt: Campus 1997

[27] Clarisse Sieckenius de Souza: The semiotic engineering of user interface languages. *Int. J. Man-Machine Studies* 39 (1993) 753-773

[28] Ronald Stamper: *Information in business and administrative systems*. London: B. T. Batsford 1973

[29] Ludwig Wittgenstein: *Tractatus logico-philosophicus*. Frankfurt: Suhrkamp 1963

[30] Heinz Zemanek: Semiotics and programming languages. *Comm. ACM* 9 (1966) 139-143