Prof.Dr.–Ing.Siegfried Wendt

(Vortrag im Göttinger Informatik–Kolloquium am 12.1.1995)

Formalismus und Einsicht

Kommunikationsprobleme der Softwaremacher

An Ihrer Universität einen Vortrag halten zu dürfen, ist für mich eine große Ehre, und ich danke deshalb allen, denen ich meine Einladung zu verdanken habe. Es hat schon einen besonderen Reiz, an einer Universität vorzutragen, an der solche Größen wie Gauß und Hilbert gelehrt haben. Daß ich dennoch einigermaßen angstfrei hierher gekommen bin, ist in meiner Vermutung begründet, daß wohl auch hier in Göttingen die durchschnittliche Professorenqualität beträchtlich unter dem Niveau von Gauß und Hilbert liegen wird. Auf Hilbert werde ich übrigens später noch einmal zurückkommen.

Ich habe meinen Vortrag wie folgt gegliedert: Zur Einführung möchte ich das Problem als ein solches charakterisieren, welches man nur vollständig erfassen kann, wenn man auch seine politischen Hintergründe kennt. Ich werde mir dabei die Freiheit nehmen, meine Argumentation mit etwas Polemik zu würzen. Diese Polemik wird auch in den Schlußbemerkungen wieder auftauchen, wo ich über die betrüblichen Folgen einer bestimmten Einäugigkeit berichten werde. Als Problemlösung werde ich im Hauptteil meines Vortrags die sogenannte Technische Semiotik vorstellen. Diese Technische Semiotik ist das Leitthema, welches uns durch weite Strecken dieses Vortrags begleiten wird. So werde ich insbesondere über semiotische Vor– und Nachteile von Formalismen sprechen und werde die Software und das interpretierte dynamische System als semiotische Gegenstände behandeln.

Bei der Vorbereitung meines heutigen Vortrags fiel mir zufällig auf, daß ich meine Ausführungen durch die Titel zweier Bücher meines akademischen Lehrers Karl Steinbuch charakterisieren könne. 1968 erschien sein Buch mit dem Titel "Falsch programmiert", welches den ersten Platz auf den Bestsellerlisten erreichte. Darin ging es nicht um Fehler in der Software, sondern um zurückliegende sozialpolitische und kulturpolitische Entscheidungen und ihre negativen Langzeitwirkungen. Im Jahre 1973 setzte er seine Argumentation fort mit dem Buch "Kurskorrektur".

Auch in meinem Vortrag geht es um negative Langzeitwirkungen einer zurückliegenden Entscheidung, so daß man auch hier von einer falschen Programmierung sprechen kann, die eine Kurskorrektur erforderlich macht. Die Entscheidung, die ich hier meine, war eine wissenschaftspolitische Entscheidung im Jahre 1969, und sie betraf die Gründung der Informatik–Fachbereiche in der damaligen Bundesrepublik. Die Älteren unter Ihnen werden es wohl noch wissen, die Jüngeren vermutlich nicht, daß es damals einen heftigen Streit zwischen Mathematikern und Elektrotechnik–Ingenieuren gab über die Frage der wissenschaftlichen Schwerpunktsetzung in der Informatik. Nach außen war es ein Streit zwischen zwei Verbänden, nämlich der GAMM, also der Gesellschaft für angewandte Mathematik und Mechanik auf der einen Seite

und der NTG, also der Nachrichtentechnischen Gesellschaft auf der anderen. Intern aber waren es politische Gefechte zwischen Personen. Sie brauchen nur die Strukturen der Prüfungsordnungen der Fakultäten Informatik, Mathematik und Elektrotechnik zu vergleichen, um sofort zu erkennen, wer damals gewonnen hat: Sieger blieben die Mathematiker unter der Führung von Friedrich Ludwig Bauer.

Ich bin überzeugt, daß dieser Sieg die Voraussetzung dafür war, daß das Problem, über welches ich heute rede, überhaupt entstehen konnte. Andererseits weiß ich natürlich nicht, welche anderen, vielleicht noch schwerwiegenderen Probleme entstanden wären, wenn die Partei der Ingenieure gewonnen hätte.

Eine zahlenmäßige Dominanz der Mathematiker in den Informatik–Fakultäten hat es nie gegeben. Aber trotz ihrer zahlenmäßigen Unterlegenheit in den Fakultäten stellten die Mathematiker aufgrund des errungenen Sieges in allen entscheidungsrelevanten überregionalen Gremien den Vorsitzenden, und sie brachten ein solches Selbstbewußtsein mit, daß sie im Ringen um das Selbstverständnis der Informatik als wissenschaftliche Disziplin tonangebend waren und teilweise immer noch sind.

Die Informatik konnte selbstverständlich anfangs noch kein stabiles eigenes Selbstverständnis haben, sondern mußte ein solches erst im Laufe der Zeit entwickeln. Eine wissenschaftliche Disziplin definiert sich durch die Probleme, die sie mit wissenschaftlicher Methodik lösen will.

Und da muß man nun feststellen, daß Mathematiker und Ingenieure völlig unterschiedliche Problemauffassungen haben. Für einen Mathematiker ist ein Problem primär dann interessant, wenn er durch eine Lösung des Problems den Erkenntnisfundus der Mathematik bereichern würde. In dieser Hinsicht ist der Mathematiker wie ein Philosoph: Er löst seine Probleme für sich und die Leser seiner Schriften.

Völlig anders sieht die Sache bei den Ingenieuren aus. Das primäre Problem eines Ingenieurs besteht darin, technische Produkte zu schaffen, mit denen die Kunden zufrieden sind. Dazu ist ihm jedes Mittel recht. Wenn er mathematische Erkenntnisse dazu braucht, zieht er diese selbstverständlich heran, ebenso wie andere wissenschaftliche Erkenntnisse, die ihm nützen können. Er darf sich aber nicht zu fein sein, in den Fällen, wo es der Sache dient, ganz auf die Wissenschaft zu verzichten und sich auf schlichtes sogenanntes *Know how* und den gesunden Menschenverstand zu verlassen.

Die Wissenschaftlichkeit der Mathematiker und der Ingenieure hat also zwei völlig unterschiedliche Quellen: Der Mathematiker fühlt sich zum Wissenschaftler berufen, wogegen für den Ingenieur die Wissenschaft nur Mittel zum Zweck ist und neben nichtwissenschaftlichen Mitteln steht, die er in gleicher Weise benötigt.

Vor diesem Hintergrund werden bestimmte Vorwürfe, die ich in der Anfangszeit der Informatik häufig und an verschiedenen Orten gehört habe, leicht verständlich: Die sogenannten Theoretiker warfen den sogenannten Praktikern vor, sie würden unwissenschaftliche Bastelei betreiben, und die Praktiker warfen umgekehrt den Theoretikern vor, sie würden Probleme lösen, die mit der Praxis nichts zu tun hätten. Ich war damals schon überzeugt, daß sie beide im Grunde recht hatten.

Diese gegenseitigen Vorwürfe hatten zweierlei Effekt, einen guten und einen schlechten. Der gute Effekt war, daß die Theoretiker bei ihrer Themenwahl praxisorientierter wurden, und der schlechte Effekt war, daß sich die Praktiker, um ihre Wissenschaftlichkeit zu demonstrieren, dazu verleiten ließen, ihre Probleme voreilig und unangemessen zu formalisieren, wodurch sie praxisferner wurden. Die Praktiker sahen sich also genötigt, den Satz zu akzeptieren, daß in einem Wissensbereich nur so viel Wissenschaft sei, wie darin Mathematik zu finden ist. Auf diese Weise sind leider sehr viele schwerwiegende Probleme, die nicht durch die Entwicklung von Formalismen zu lösen sind, aus dem Blickfeld der akademischen Informatik geraten, obwohl diese eigentlich dafür zuständig wäre.

Das ingenieurwissenschaftliche Erobern einer neuen Technologie setzt voraus, daß die damit befaßten Wissenschaftler das Recht haben, unbehelligt von Vorwürfen der mangelnden Wissenschaftlichkeit erst einmal unter Verwendung der neuen technologischen Möglichkeiten Systeme zusammenzubasteln, damit sie überhaupt erkennen können, welche wesentlichen Probleme es denn sind, an die man mit wissenschaftlicher Methodik herangehen muß. Die Informatik ist entstanden durch das Aufkommen einer neuen Technologie, die darin besteht, daß man informationsverarbeitende Systeme dadurch realisiert, daß man Software mit einer universellen Hardware zusammenbringt. In dieser Technologie gibt es einen Problembereich, der garantiert nicht mit mathematischen Methoden zu lösen ist. Es handelt sich um das Problem der Beherrschung der Systemkomplexität.

Erkenntnisse, die uns der Lösung des Komplexitätsproblems näher bringen, kann man nur gewinnen, indem man sich jahrelang auf "Großbaustellen der Softwarekonstruktion" herumtreibt. Ich habe hierbei Systeme vor Augen, die mit einem Entwicklungsaufwand von über hundert Mannjahren realisiert wurden. Ein klassisches Beispiel einer derart komplexen Software ist das System R/3 der Firma SAP, das rund zehn Millionen Zeilen Quellcode umfaßt. Ich kenne dieses System recht gut, weil ich aktuell an seiner Weiterentwicklung mitwirken darf.

So wie der Maschinenbau als wissenschaftliche Ingenieurdisziplin nicht entstanden wäre, wenn sich nicht in der Zeit des Aufkommens der Dampfmaschine die Ingenieure auf völlig unwissenschaftliche Weise die Finger schmutzig gemacht hätten, so kann auch die Informatik keine reife Ingenieurwissenschaft werden, solange die Informatikprofessoren glauben, es lohne sich nicht, den Herstellungsprozeß derjenigen Software näher anzuschauen, mit der heute das Geld verdient wird.

Es zeugt von extremer Realitätsferne, wenn ein Autor im GI–Spektrum schreibt, die Probleme, mit denen die Softwareindustrie kämpft, gäbe es nicht, wenn die dortigen Praktiker doch nur bereit wären, die an den Universitäten längst erfundenen Formalismen einzusetzen.

Ein Indiz dafür, daß die Informatik noch keine reife Ingenieurwissenschaft ist, sehe ich auch in der Existenz eines Informatikwettbewerbs für Schüler und in den Aufgaben, die dort gestellt werden. Die Informatiker sollten sich einmal fragen, wieso es eigentlich keinen Maschinenbauwettbewerb oder Elektrotechnikwettbewerb für Schüler gibt. Im Informatikwettbewerb wird ein viel zu enges Bild von dem vermittelt, was die zentralen Problemstellungen der Informatik sein sollten. Überspitzt gesagt geht es in diesem Informatikwettbewerb immer um das Erfinden oder

das Analysieren von Algorithmen und somit letztlich nur um das mehr oder weniger methodische Lösen von Denksportaufgaben.

Es wäre schlimm, wenn ich mit meiner Analyse der Situation alleine stünde, denn dann müßte ich mich doch fragen, ob ich nicht einem Verfolgungswahn erlegen sei. Glücklicherweise aber bin ich nicht der einzige Prediger in der Wüste. Der vielleicht renommierteste Prediger ist Peter Naur aus Kopenhagen, den Sie vermutlich alle aus der Backus–Naur–Form kennen. 1992 ist sein Buch "Computing: A human activity" erschienen, welches eine systematische Zusammenstellung seiner Aufsätze aus drei Jahrzehnten enthält. Ich habe dieses Buch erst vor wenigen Wochen in die Hände bekommen, und ich war hell begeistert über die Entdeckung, daß Peter Naur die Lage sehr ähnlich einschätzt wie ich. Aus der Tatsache, daß ich hier Peter Naur erwähne, der bekanntermaßen kein Gegner von Formalismen ist, dürfen Sie schließen, daß ich auch keiner bin. Es geht mir keineswegs darum, die Formalismen zu verbannen, sondern lediglich darum, ihren Globalanspruch abzuwehren und ihnen die Rolle zuzuweisen, in der sie den größten Nutzen bringen.

Damit bin ich am Ende des einführenden Abschnitts meines Vortrags angekommen, den ich mit dem Hinweis auf den Steinbuchtitel "Falsch programmiert" eingeleitet habe. Ich trete nun in den Hauptteil des Vortrags ein, den ich mit dem Hinweis auf den Steinbuchtitel "Kurskorrektur" eröffnen will. Ich möchte Ihnen darin meine Vorstellungen von einer geänderten Schwerpunktsetzung in der Informatik präsentieren, die ich für unbedingt erforderlich halte.

Ich habe hier ein Bild, anhand dessen ich den Themenbereich lokalisieren möchte, über den ich im folgenden reden werde. Das Bild zeigt, welchen Unterbau die Ingenieurwissenschaften von den informationsverarbeitenden Systemen benötigen. Ganz unten sitzt die Philosophie; unmittelbar darüber liegt die Mathematik, und darüber sitzen gleichberechtigt nebeneinander zwei Fundamentblöcke. Der eine Block ist die Physik, und den anderen habe ich hineingeschrieben: "Nichtphysikalische Grundlagen der Informationstechnik". Diese Bezeichung findet sich auch wieder als Titel meines letzten Buches und als Bezeichnung einer Vorlesung, die ich in Kaiserslautern halte.

Die Notwendigkeit zweier Erkenntnisfundamente ergibt sich selbstverständlich aus der Tatsache, daß jedes informationstechnische System ein mit physikalischen Effekten realisierter Formalismus ist. Es ist klar, daß die Formalismen in den linken Fundamentblock gehören; die Frage ist lediglich, welchen Raum sie darin einnehmen sollen.

Meine Erfahrungen mit den Problemen aus der Praxis geben mir die Überzeugung, daß in diesem Fundamentblock eine Theorie der interpretierten Systeme enthalten sein muß, und für eine solche Theorie ist die Begriffswelt der Formalismen lediglich eine Grundlage. Deshalb sieht die von mir propagierte Strukturierung dieses Fundamentblocks so aus, wie es das nächste Bild zeigt.

Die Theorie der interpretierten Systeme benötigt als Grundlagen einerseits die Theorie der nichtinterpretierten Systeme und andererseits eine Erkenntniswelt, die ich eigenmächtig als "Technische Semiotik" bezeichnet habe.

Im Rahmen dieses Vortrags will ich mich auf diesen Themenbereich konzentrieren. Ich habe bereits erwähnt, daß es sich bei dem Begriff der Technischen Semiotik um eine eigenmächtig von mir gewählte Bezeichnung handelt. Ich muß hinzufügen, daß ich heute zwar nicht mit den Inhalten, aber mit der Bezeichnung "Technische Semiotik" zum ersten Mal an die Öffentlichkeit trete. Während der Begriff der Semantik in der Theorie der formalen Sprachen für die Informatik bereits ein selbstverständlicher Begriff ist, ist meines Wissens der Begriff der Semiotik in der Informatik bisher noch nirgendwo ernsthaft aufgetaucht. In der Vortragsankündigung habe ich weder im Titel noch in der Kurzfassung das Wort *Semiotik* benutzt, weil ich vermutete, daß dies eher abschreckend als einladend wirken würde.

Wenn man Literatur zum Thema Semiotik sucht, wird man heute wohl am ehesten auf das Buch "Einführung in die Semiotik" von Umberto Ecco stoßen. Umberto Ecco ist vielen ja nur bekannt als Autor von Bestsellerromanen wie *Der Name der Rose* und *Das Foucaultsche Pendel.* Auf der Rückseite des Umschlags von Eccos Buch über Semiotik steht folgender Text: "Die Semiotik, d.h. die Erforschung der Kultur als Kommunikation bedeutet Grundlagenwissen für den gesamten Bereich der philosophischen Fakultät. Umberto Ecco, der zu den Pionieren auf diesem Gebiet zählt, ist wie kein anderer berufen, in die allgemeine Lehre von den Zeichen in gründlicher Gesamtschau einzuführen. Für die Literatur– und Sprachwissenschaften für Soziologie, Philosophie, Rhetorik, Kunst einschließlich solcher Probleme wie Filmästhetik, Reklame, politische Propaganda, Trivialliteratur und Subkultur werden in gut faßlicher und fesselnder Weise neue Wege des Verständnisses aufgewiesen. " (Ende des Zitats).

Aus diesem Zitat wird deutlich, daß es bisher keine Technische Semiotik gibt. Es ist aber durchaus angebracht, sich um die Schaffung einer solchen zu bemühen.

Die abstrakteste Form dessen, worum es in der Semiotik geht, zeigt das folgende Bild. Die Semiotik untersucht die Assoziationsvorgänge, die das Informationserleben der Menschen ausmachen. Deshalb gibt es hier einen sogenannten Assoziationsapparat, von dem aber nicht gefragt wird, wie er aufgebaut sei, sondern nur, welche Informationen an seinen Schnittstellen vorkommen. Die Vereinigung dieser drei Schnittstelleninformationen wird in der Semiotik als *Zeichen* bezeichnet. Ein semiotisches Zeichen besteht also aus Information aus der Wahrnehmung, aus Wissen um vereinbarte Assoziationen zum Wahrnehmbaren und aus der vereinbarungsgemäß assoziierten Information.

Die Information aus der Wahrnehmung stammt aus dem Wahrnehmungsapparat, der sie gewinnt durch Überlagerung dessen, was über die Sinnesorgane hereinkommt, mit dem, was hier über diesen internen Rückkopplungsweg kommt. Es gibt auch noch einen äußeren Rückkopplungsweg, der über den Aktionsapparat läuft, mit dem der Mensch das über die Sinnesorgane Wahrnehmbare verändern kann.

Als Beispiele für wahrnehmbare Formen wird man in diesem Zusammenhang primär an Schriftzeichen oder gesprochene Texte denken, aber die semiotische Betrachtung ist nicht auf solche reinen Kommunikationsformen beschränkt. Alles Wahrnehmbare kann zu einem semiotischen Zeichen werden. So kann man beispielsweise zu einem wahrgenommenen Hund das Wort *Hund,* das Wort *Treue* oder das Gefühl *Angst* assoziieren.

Was unbedingt aus der allgemeinen Semiotik in die Technische Semiotik übernommen werden muß, ist die Konzentration auf den Menschen. Das bedeutet, daß dieser Assoziationsapparat grundsätzlich der Assoziationsapparat des Menschen ist und kein technisches Gebilde. Wir sind zwar gewohnt, unsere Computer als Assoziationsapparate zu sehen, aber dies ist selbstverständlich selbst eine menschliche Assoziation. Der Computer sitzt nicht in diesem Viereck, sondern äußert sich grundsätzlich zuerst einmal als Information aus der Wahrnehmung, zu der wir dann das Analogon der menschlichen Assoziationsfähigkeit assoziieren.

Grundsätzlich bleibt auch in der Technischen Semiotik der Mensch das Maß aller Dinge in zweierlei Hinsicht: Zum einen werden in unseren technischen Systemen keine Strukturen auftreten, die nicht als Analogien zu menschlichen Assoziationsmustern geplant worden wären, und zum anderen müssen wir die Dokumente, die wir bei der Entwicklung unserer technischen Systeme erstellen, hinsichtlich der semiotischen Bedürfnisse des Menschen optimieren.

Ich zeige Ihnen hier eine Auflistung von Themen, die m.E. unbedingt in die Technische Semiotik gehören. Diese Liste ist mit Sicherheit unvollständig. Auf die ersten drei dieser Themenbereiche werde ich hier eingehen können, auf die letzten beiden nicht.

Bei den Alternativen der Identifikation geht es um die Frage nach den Möglichkeiten, die uns zur Verfügung stehen, um unsere eigene Aufmerksamkeit oder die eines anderen auf einen bestimmten Gegenstand oder einen Begriff zu lenken. Man denke an die Identifikation eines Bankräubers durch einen Zeugen. Da kann es sein, daß der Zeuge vor eine Reihe mehrerer Personen gestellt wird und auf den Täter zeigen soll, falls dieser unter den anwesenden Personen ist. Neben dem Zeigen gibt es noch zwei weitere Möglichkeiten der Identifikation Es handelt sich zum einen um die Präsentation eines früher vereinbarten stellvertretenden Symbols und zum anderen um die Präsentation einer sogenannten Umschreibung. Wenn der Zeuge beispielsweise weiß, wie der Täter heißt, dann kann er ihn auch identifizieren, wenn er nicht aktuell im Bereich der Sinnesorgane liegt, indem er für die Sinnesorgane wahrnehmbar den Namen ausspricht.

Am kompliziertesten und deshalb auch am interessantesten ist die Identifikation durch das Präsentieren einer Umschreibung. Der Zeuge könnte in diesem Fall beispielsweise berichten, daß der Täter ein glatzköpfiger Mann sei mit einer Körpergröße von ungefähr 1,75 m und einem leichten Bauchansatz, der blaue Jeans und einen grünen Pulli trug. Interessant ist hier insbesondere die Unterscheidung zwischen direkten und indirekten Umschreibungen. In einer direkten Umschreibung wird ein Weg angegeben, den man gehen muß, um garantiert zum identifizierten Objekt zu gelangen.

Wenn ich Ihnen beispielsweise sage: Nehmen Sie die Zahl 20, ziehen Sie davon 4 ab, aus dem, was Sie nun erhalten haben, ziehen Sie die Wurzel, und auf dieses Ergebnis addieren Sie nun eine 1. Damit wissen Sie das Alter der kleinen Annemarie. In diesem Fall habe ich zwar das Alter der Annemarie nicht durch Präsentation des Wortes 5 identifiziert, aber Sie hatten trotzdem keine Mühe, am Ende genau zu wissen, welche Zahl gemeint war. Eine mögliche indirekte Umschreibung hätte lauten können: Wenn Annemarie doppelt so alt sein wird, wie sie heute ist, wird sie fünfmal so alt sein, wie sie heute vor 3 Jahren war. Wer nicht gelernt hat, eine lineare Gleichung mit einer Unbekannten hinzuschreiben und aufzulösen, wird mit dieser Umschreibung von An-

nemaries Alter wenig anfangen können, obwohl doch auch diese Umschreibung das Alter eindeutig identifiziert. Um direkte und indirekte Umschreibungen geht es selbstverständlich auch bei ergebnisorientierten Programmen, die prozedural, funktional oder deklarativ formuliert sein können.

Wie Sie sehen, nehmen in meiner Liste die Strukturen den größten Raum ein. Strukturen sind im allgemeinsten Sinne Gebilde aus Individien, zwischen denen bestimmte Beziehungen bestehen oder nicht bestehen. Es wird Sie vermutlich etwas erstaunen, daß ich hier eine Unterteilung gemacht habe nach einem Kriterium, welches aus den Urgründen der Philosophie stammt. Ich habe zwischen seienden und werdenden Strukturen unterschieden. Die seienden Strukturen sind diejenigen, bei deren Behandlung der Begriff *Zeit* nicht vorkommen darf. Deshalb sind auch bei der Behandlung dieser Strukturen die Wörter *Bewegung* oder *Veränderung* ausgeschlossen. Wenn man diese Strukturen beschreibt, beschreibt man also etwas Unveränderliches.

Wenn die Philosophen sich mit solchen Strukturen befassen, dann ist der Ausschluß der Zeitbetrachtung absolut, d.h. ihre seienden Strukturen müssen von Anbeginn der Zeit bis in alle Ewigkeit unverändert bleiben. Wir Techniker sind da schon etwas großzügiger: Wir verlangen nur, daß diese Strukturen für die Dauer unseres Interesses konstant bleiben. So ist beispielsweise ein Gebäude in der Philosophie selbstverständlich keine seiende Struktur, denn das Gebäude wird irgend einmal gebaut werden, eine bestimmte Zeit, möglicherweise sogar ein paar Jahrhunderte lang unverändert stehen bleiben, und dann zerfallen oder abgerissen werden. Wenn ich dagegen nur eine bestimmte Zeit in diesem Haus wohne und das Haus in dieser Zeit nicht umgebaut wird, dann kann ich das Haus für das Zeitintervall meines Darinwohnens als seiende Struktur betrachten.

Strukturen im Raum sind solche, die wir sehen können. Ich habe sie unterteilt in Strukturen von Nichtbeschreibungen und Strukturen von Beschreibungen. Eine Nichtbeschreibung ist etwas sichtbares, was zwar beschrieben werden kann, was aber nicht selbst etwas beschreibt. Man denke an eine Stadt, die durch einen Stadtplan beschrieben wird. Der Stadtplan ist eine Beschreibung, die genauso sichtbar ist wie die Stadt selbst, und die auch eine Struktur ist aus Individuen, die zueinander in bestimmten Beziehungen stehen. Normalerweise wird durch eine Beschreibung eine Nichtbeschreibung beschrieben, aber es gibt auch den Fall der Beschreibung einer Beschreibung. Im Zusammenhang mit der Informatik ist besonders die Frage interessant, wie die Programme, die ja selbst Beschreibungen sind, beschrieben werden können, und was sie eigentlich beschreiben.

Axiomatische Strukturen sind Strukturen, die aus Individien aufgebaut sind, die das Abstrakteste des Abstrakten darstellen. Auch die Beziehungen zwischen diesen Individuen sind so abstrakt, daß es abstrakter nicht mehr geht. Ich will hierzu ein Beispiel bringen, welches vielen von Ihnen zumindest zum Teil bekannt sein wird.

Es handelt sich um die durch die berühmten fünf Peanoschen Axiome definierte Struktur. Links im Bild sehen Sie die Axiome, allerdings nicht in der ursprünglichen Symbolik, wie sie Peano verwendet hat, sondern in einer Symbolik, wie sie heute üblicher ist.

Man kann solche Formeln nur dann mit Verständnis lesen, wenn man den Unterschied kennt zwischen dem, was man über die verwendete Formelsprache a priori wissen muß, und dem Wis-

sen über die durch die Formeln definierte Struktur, welches man durch das Studium der Formeln erst gewinnen kann. Bevor man mit dem Lesen dieser Formeln beginnt, muß man über die verwendete Formelsprache schon folgendes wissen:

Man muß die Junktorensymbole der Aussagenlogik und ihre Bedeutung kennen – hier also den Punkt für die UND–Verknüpfung und den Pfeil für die Implikation – und man muß die Quantorensymbole der Prädikatenlogik kennen – hier also das auf dem Kopf stehende A für die Allquantifikation. Außerdem muß man wissen, daß Funktionen und Prädikate immer so geschrieben werden, daß ihre Argumente in runden Klammern stehen und vor dieser Klammerung der Funktions– bzw. Prädikatsbezeichner steht. Ferner muß man wissen, daß kursiv gesetzte Buchstaben die Variablen für die Quantifizierung sind, während steil stehende Buchstaben die Bezeichner der abstrakten Objekte sind, die durch die Axiome definiert werden sollen. Großbuchstaben stehen für Prädikate, kleine griechische Buchstaben für Individuen, und die kleinen lateinischen Buchstaben stehen für Individuen oder Funktionen, wobei aus der jeweiligen Position relativ zu den runden Klammern stets eindeutig hervorgeht, ob ein Individuum oder eine Funktion gemeint ist.

Ich interpretiere nun die Axiome von oben nach unten. Das erste Axiom besagt, daß das Individuum α von der Art N ist. Weil die Interpretation von N bereits vorab festgelegt wurde, bedeutet dieses Axiom, daß α ein Individuum in der Struktur ist. Ich habe rechts willkürlich ein Beispiel einer beziehungslosen Struktur aus drei Individuen hingezeichnet, unter denen das Individuum α vorkommt.

Das zweite Axiom sagt, daß auf allen Individuen der Struktur eine Funktion r definiert ist, deren Ergebnis wieder ein Individuum der Struktur ist. Zur Veranschaulichung habe ich rechts den Funktionszusammenhang durch Pfeile ausgedrückt. Wenn man also beispielsweise das Individuum α als Argument in die Funktion r einsetzt, dann erhält man als Ergebnis dieses Individuum, auf welches der von α ausgehende Pfeil zeigt. Man kann also dieses Axiom auch in der Form lesen, daß man sagt, von jedem Individuum in der Struktur geht genau ein Pfeil aus, der wieder auf einem Individuum der Struktur landet.

Das dritte Axiom sagt, daß es in der Struktur kein Individuum gibt, auf dem zwei Pfeile landen. Damit wird also diese Struktur, die mit den ersten beiden Axiomen noch verträglich war, ausgeschlossen, weil auf diesem Individuum hier zwei Pfeile landen.

Das vierte Axiom sagt, daß auf dem Individuum α kein Pfeil landet. Damit ist diese Struktur, die noch mit den ersten drei Axiomen verträglich war, nicht mehr zugelassen. Aber diese neben dem vierten Axiom stehende Struktur, die aus zwei isolierten Teilstrukturen besteht, ist noch mit den ersten vier Axiomen verträglich ist. Da auch diese Struktur nicht die gemeinte ist, bedarf es noch eines fünften Axioms, welches nur noch die gemeinte mit α beginnende unendliche Kette übrig läßt. Dieses Axiom besagt auf gut deutsch, daß es in der Struktur keine Individuen gibt, die nicht in der von α ausgehenden Kette hängen.

Sie sehen, welcher Aufwand getrieben werden muß, um diese einfache Struktur axiomatisch zu definieren, die man doch hier so schön graphisch hinzeichnen kann. Dennoch kann man nicht auf

diese Axiome verzichten, weil man mit dieser schönen graphischen Struktur nicht formal operieren kann. Das Problem ist bekanntermaßen die Unendlichkeit, die mit diesen Punkten hier für uns zwar einleuchtend, aber für eine Maschine unbrauchbar symbolisiert ist.

Ein wesentlicher Nutzen axiomatischer Formeln liegt also darin, daß man sie mit Maschinen manipulieren kann. Es gibt aber noch einen zweiten wesentlichen Nutzen; diesen will ich an einem weiteren Beispiel zeigen. Ich hatte ja bereits zu Beginn meines Vortrags angekündigt, daß ich später noch einmal auf Hilbert zurückkommen würde, und dieser Zeitpunkt ist nun gekommen. Ich zeige Ihnen hier zwei Axiome, die in natürlicher Sprache formuliert ganz am Anfang des Buches GRUNDLAGEN DER GEOMETRIE von David Hilbert vorkommen. Das erste Axiom lautet: *Zwei voneinander verschiedene Punkte A, B bestimmen stets eine Gerade a.* Das zweite Axiom lautet: *Irgend zwei voneinander verschiedene Punkte einer Geraden bestimmen diese Gerade.*

Wenn man diese beiden Sätze miteinander vergleicht, dann drängen sich unmittelbar zwei Fragen auf: Erstens: Wieso kommt hier unten das Wort *irgend* vor und hier oben nicht? Hat dies eine besondere Bedeutung, oder ist dies irrelevant? Zweite Frage: Wieso werden hier oben für die Punkte die Buchstaben A und B und für die Gerade der Buchstabe a verwendet, während im zweiten Satz überhaupt keine Buchstaben vorkommen? Ist dies irrelevant oder gibt es dafür einen bestimmten Grund? Eine weitere Frage drängt sich auf, wenn man an anderer Stelle feststellt, daß Hilbert dort die mathematische Redewendung *eine und nur eine Gerade* verwendet. Nachdem man dies gefunden hat, blättert man wieder zu diesen Axiomen zurück und fragt: Soll dieses *eine* hier *mindestens eine* oder *eine und nur eine* heißen?

Sie sehen, daß die Vielfalt der Möglichkeiten, einen Sachverhalt in natürlicher Sprache zu formulieren, die große Gefahr mit sich bringt, daß Mißverständnisse zwischen dem Schreiber und dem Leser auftreten. Dies wird ausgeschlossen, wenn man die Axiome in einer genormten Formelsprache formuliert. Ich habe diese beiden Axiome in die gleiche Formelsprache übersetzt, in der ich Ihnen auch schon die Peanoschen Axiome präsentierte. Hier habe ich mir allerdings nicht die Mühe gemacht, genau nachzuprüfen, ob ich mit dieser Übersetzung den Hilbertschen Absichten gerecht geworden bin.

Es geht mir ja hier nicht um exakte Geometrie, sondern um den Zusammenhang zwischen Formalismen und Einsicht. Dieses Axiom liest man nun wie folgt: Für jedes Paar von Individuen x und y, die von der Art P sind und die verschieden sind, gibt es mindestens ein Individuum a, welches von der Sorte G ist und welches mit dem Individuen–Paar (x, y) in der Beziehung V steht.

Ich habe die Buchstaben so gewählt, daß man die Standardinterpretation dafür nehmen kann: Individuen der Sorte P sollen Punkte sein, Individuen der Sorte G sollen Geraden sein, und das Prädikat V soll besagen, daß die Gerade a die beiden Punkte x und y verbindet. In entsprechender Weise ergibt sich dann für das zweite Axiom diese Formel. Für alle Tripel von Individuen (x, y, a), von denen x und y die von der Sorte P sind und a von der Sorte G ist, wobei x und y verschieden sind, und wenn diese noch die Bedingung erfüllen, daß x mit a in der Relation E steht und y mit a in der Relation E steht, dann ist sichergestellt, daß auch x und y mit a in der Relation V stehen. In

der Standardinterpretation ist das Prädikat E als Enthaltensein zu deuten und besagt, daß der Punkt *x* auf der Geraden *a* liegt.

Es ist also ganz offensichtlich, daß die Formelschreibweise nicht nur die maschinelle Manipulation ermöglicht, sondern auch das Mißverständlichkeitsproblem löst. Dadurch sind wir aber vor ein neues Problem gestellt, und das ist das Unverständlichkeitsproblem.

Nehmen Sie einmal an, ich hätte Ihnen diese beiden Formeln präsentiert, ohne Ihnen zu sagen, daß es sich hier um die formale Darstellung der ersten beiden Hilbertschen Geometrie–Axiome handelt. Hätten Sie dann eine Chance gehabt zu erkennen, daß es hier um Geometrie geht? Ich habe hierzu schon verhältnismäßig oft folgendes Experiment gemacht: Ich habe eine formale Definition eines Stacks angeschrieben, auf dem die klassischen Operationen *PUSH*, *POP* und *CLEAR* sowie die *Abfrage des aktuellen Füllstandes* definiert waren. Ich habe aber weder gesagt, daß es sich um einen Stack handelt, noch habe ich irgend welche anschaulichen Bezeichner verwendet, die auf den Stack hingedeutet hätten. So habe ich beispielsweise anstelle von PUSH und POP die Bezeichner HOSS und LIEB genommen. Jedesmal haben meine Studenten ziemlich verständnislos auf meine Hieroglyphen geschaut, und die Überraschung war stets recht groß, wenn ich ihnen die Übersetzung der Bezeichner verriet. Erst danach konnten sie die Korrektheit der einzelnen Formeln einsehen. Was will ich damit sagen? Ich will sagen, daß man eine Struktur oder ein System schon intuitiv verstanden haben muß, bevor man aus einer formalen Darstellung dieser Struktur oder des Systems einen Nutzen ziehen kann.

An dieser Stelle muß ich nun kurz den Begriff der formalen Semantik charakterisieren. Dieser Begriff hat überhaupt nichts mit dem Problem zu tun, formale Beschreibungen zu verstehen. Es geht bei der formalen Semantik überhaupt nicht um Verständnis, also im Grunde auch nicht um Semiotik, sondern nur um eine semiotische Analogie. Ich greife hierzu noch einmal auf das Bild zur Charakterisierung der Semiotik zurück. Wenn ein Mensch eine Formel ansieht, dann liegt hier das Wissen über das Aussehen der Formel, hier oben liegt das Wissen, welches sich der Mensch durch das Lernen der Formelsprache erworben hat, und hier hinten liegt dann also das, was sich der Mensch zu der aktuellen Formel denkt.

Der Begriff der formalen Semantik entsteht nun dadurch, daß man sich von der Vorstellung löst, dieser Assoziationsapparat befände sich in einem Menschen. Man nimmt vielmehr an, es handle sich hier um eine Maschine, der man die aktuelle Formel eingibt und die dazu eine weitere Formel erzeugen soll, die dann hinten aus dieser Maschine herauskommt. Die Funktion, die allen möglichen Eingabeformeln jeweils eine Ausgabeformel zuordnet, ist hier oben durch eine formale direkte oder indirekte Umschreibung identifiziert, d.h. prozedural, funktional oder deklarativ bestimmt.

Wir wollen hierzu ein einfaches Beispiel betrachten. Diese lineare Gleichung mit einer unbekannten x ist die mathematische Formel für die bereits früher als Beispiel einer indirekten Umschreibung gebrachte Aussage: "Wenn Annemarie doppelt so alt sein wird, wie sie heute ist, dann wird sie 5 mal so alt sein, wie sie heute vor 3 Jahren war."

Wir nehmen nun an, diese Formel werde in unsere Maschine eingegeben, und wir können uns überlegen, welche Formel wir denn gerne am Ausgang dieser Maschine entnehmen wollten. Ich

kann beispielsweise festlegen, die Maschine solle den hier darübergezeichneten Ableitungsbaum ausdrucken, der sich ergibt, wenn man auf diese Formel hier unten einen Algorithmus anwendet, der auf einer attributierten Grammatik beruht. Sie können alle diese Mengenausdrücke, die hier in den Knoten des Baumes stehen, als Beschreibungen von Punktemengen interpretieren, wobei jede dieser unendlichen Punktemengen eine Gerade in einem kartesischen Koordinatensystem bestimmt. Dieses hier sind Punkte einer Geraden, die durch den Ursprung geht und die Steigung 2 hat, während dieses hier Punkte einer Geraden sind, welche die Abszisse an der Stelle 3 schneidet und die Steigung 5 hat. Als Durchschnitt dieser beiden Punktemengen steht hier ganz oben der Schnittpunkt dieser beiden Geraden, und dieser Schnittpunkt besagt, daß die linke Seite nur dann gleich der rechten Seite ist, wenn x den Wert 5 hat.

Da eine formale Semantik also nur eine Funktion ist, die einer Eingabeformel eine Ausgabeformel zuordnet, erfährt man daraus selbstverständlich überhaupt nichts bezüglich der Frage, was man sich beim Anschauen der Eingabeformel oder beim Anschauen der Ausgabeformel denken soll. Vermutlich hatten Sie keine Mühe in diesem Beispielsfalle, sich etwas sinnvolles zu der Eingabeformel zu denken; ich nehme aber an, daß es für Sie ganz hilfreich war, daß ich Ihnen erklärt habe, was Sie sich beim Betrachten dieses Ableitungsbaumes denken sollten. Andererseits sind Sie mit der Interpretation dieser Mengensymbolik durchaus so vertraut, daß Sie nach einem längeren Studium dieser Inschriften auch von alleine darauf gekommen wären, was dies alles zu bedeuten hat.

Es ist selbstverständlich, daß unsere Computer bis zum Kragen mit formaler Semantik angefüllt sind. Man könnte salopp sagen, die formale Semantik sei in Strukturen aus Silizium und Kupfer gegossen. Deshalb braucht niemand zu befürchten, daß ich die formale Semantik abschaffen wolle. Ich kämpfe nur gegen diejenigen, die meinen, die formale Semantik müsse allmählich die Semiotik verdrängen.

Zwei sehr bekannte Informatiker, die ich aufgrund ihrer Schriften als Verfechter der Dominanz von Formalismen einstufen muß, sind Hoare und Dijkstra, die ich ansonsten wegen ihrer großartigen Beiträge zum Fortschritt der Informatik sehr hoch schätze. Hoare schrieb:"Computer programs are mathematical expressions." Und nur als solche will er sie behandelt wissen. Und Dijkstra schrieb:"Reduce the use of the brain and calculate!"

Erfreulicherweise sind solche Vorstellungen nicht unwidersprochen geblieben. Besonders pointiert hat sich Prof. van Emden von der University of Victoria in Kanada hierzu geäußert. Ich zitiere in deutscher Übersetzung: *In der Praxis sind die Welten der Mathematik und des Programmierens schlichtweg disjunkt.* Desweiteren sagt er: *Hilbert ist bekannt dafür, daß er die Mathematik als ein Spiel mit uninterpretierten Symbolen gesehen hat. Was er aber gemeint haben muß und wahrscheinlich auch sagte, ist, daß die Bedeutung für die Gültigkeit einer formalen Ableitung irrelevant ist. Aber das formale Spiel ist nicht Wert, gespielt zu werden, wenn es keine Bedeutung hat.* Und deswegen, so füge ich hinzu, gibt es in der Informatik keinen Nutzen der Formalismen ohne eine solide Technische Semiotik.

Kürzlich habe ich gelesen, Hilbert habe einmal gesagt, anstelle von Punkt, Gerade und Ebene müsse man auch Tisch, Stuhl und Bierseidel sagen können. Es ist sicher nicht von vorneherein

verboten, anstelle des Wortes *Ebene* das Wort *Bierseidel* zu setzen. Es macht aber nur einen Sinn, falls sich auf diese Weise tatsächlich eine schlüssige Interpretation der formalen Axiome der Geometrie ergibt. Eine solche Möglichkeit, die Geometrie–Axiome als schlüssige Aussagen über anschauliche Strukturen im Münchner Hofbräuhaus zu interpretieren, ist mir jedoch nicht bekannt. Ich kenne aber eine Interpretation in Form einer nicht Nichteuklidischen Geometrie, wo man anstelle von Punkt *Durchmesser einer Kugel* und anstelle von Gerade *Großkreis* sagen muß.

Bezüglich unserer Assoziation anschaulicher Inhalte zu den Elementen von Formalismen sind die sogenannten Anthropomorphismen meiner Erfahrung nach semiotisch besonders hilfreich, und ich kenne kein überzeugendes Argument für das Verbot, welches Dijkstra 1989 in einem Aufsatz aufgestellt hat: *"Never refer to parts of programs or pieces of equipment in an anthropomorphic terminology, nor allow your students to do so."*

Ich habe dieses Verbot heute bereits mehrfach mißachtet, indem ich beispielsweise gerade eben die formale Semantik als semiotische Analogie dargestellt habe. Hätte ich das nicht getan, wäre es völlig unmöglich gewesen, klar zu machen, was formale Semantik eigentlich sein soll.

Ich komme nun zum letzten Hauptpunkt meines heutigen Vortrags, bei dem ich mich aber aus Zeitgründen nicht mehr lange aufhalten kann. Ich möchte ihm aber doch zumindest ein paar Minuten widmen. In meiner Gliederung war dieser Punkt überschrieben mit "Software und das interpretierte dynamische System als semiotische Gegenstände". In meiner Liste der Themen aus der technischen Semiotik handelt es sich um den Punkt "Werdende Strukturen". Werden ist das Aktualisieren des Potentiellen. Unsere Beschreibungen sind natürlich immer seiende Strukturen, so daß wir fragen müssen, wie man in diesen Beschreibungen über werdende Strukturen reden kann.

Glücklicherweise geht es gar nicht darum, das Werden als solches darzustellen, sondern nur darum, einerseits das Gewordene und andererseits das Potentielle zu beschreiben. Sowohl das Gewordene als auch das Potentielle sind aber seiende Strukturen. Denn uns interessiert das Gewordene erst, nachdem es geworden ist, und danach kann es sich selbstverständlich nicht mehr ändern. Und das Potentielle interessiert uns nur solange, wie es noch Potentielles ist, und als solches verändert es sich selbstverständlich auch nicht.

Sie sehen hier u.a. das Thema *Abwicklung generativer Schemata*. Denken Sie sich hierzu als Beispiel ein Petrinetz, welches als Beschreibung von Potentiellem eine seiende Struktur ist, die wir aufs Papier gezeichnet haben. Wir können aber dieses Petrinetz abwickeln, indem wir die Schaltregel anwenden und dabei die Markierung des Netzes verändern. Man sitzt dazu beispielsweise am Tisch und verschiebt Pfennige über das Netz. Wenn man dabei aufschreibt, an welchen Transitionen man bei diesem Pfennigschieben jeweils vorbeikommt, dann entsteht wieder etwas Aufgeschriebenes, also eine seiende Struktur. Auf diese Weise wird aus der Potentialität die Aktualität.

Es mag manchem von Ihnen möglicherweise erscheinen, als ob ich hier Spitzfindigkeiten aus der Philosophie krampfhaft in die Welt der Informatik übertragen wollte. Es zeigt sich aber im-

mer wieder, daß man über die Strukturen in der Software und in den softwaregesteuerten Systemen nur dann mit der erforderlichen Präzision reden und schreiben kann, wenn man geschult ist, solche begrifflichen Feinheiten zu beachten.

Dazu gehört u.a. auch, daß man gelernt hat, die Programmierung als universelles Fertigungsprinzip zu sehen. Als Konsequenz daraus erkennt man dann, daß man die hier interessierenden Systeme derart abstrakt betrachten kann, daß man viel interessantes über sie aussagen kann, ohne dabei die Tatsache, daß sie programmiert sind, berücksichtigen zu müssen. Man kann dann nämlich auch über die Systeme so reden, als hätten sie ihr Verhalten dadurch bekommen, daß geeignet spezialisierte Hardwarekomponenten zu einem angemessenen Netz verbunden wurden.

Vor vielen Jahren hat mir einmal ein Softwareentwickler aus der Industrie, der inzwischen Informatikprofessor geworden ist, gesagt, meine Art, die programmierten Systeme so zu abstrahieren, daß ich sie in den Kategorien von Hardwaresystemen beschreiben könne, sei ein klarer Hinweis darauf, daß ich von Software keine Ahnung hätte. Ich weiß nicht mehr, was ich ihm damals geantwortet habe, aber ich werde wohl gedacht haben, wie schlimm es ist, wenn maßgebliche Informatiker ein derart beschränktes Abstraktionsvermögen haben. Allerdings könnte er sich inzwischen auch auf eine Aussage von Dijkstra berufen, die den Glauben fördert, Software sei etwas so grundsätzlich neues, daß man darüber nicht mehr in den Kategorien denken und reden könne, die sich in den Jahrtausenden der menschlichen Kulturgeschichte vor der Erfindung des Computers herausgebildet haben.

Dijkstra schrieb 1989: "Computers represent a radical novelty in our history. ... Our past experience is no longer relevant. ... One has to approach the radical novelty with a blank mind."

Ich zeige Ihnen hier einmal einen Plan als Beispiel dafür, wie wir softwaregesteuerte Systeme in den gleichen Kategorien wie die Hardware darstellen. Es handelt sich hierbei um den Architekturkern des R3–Systems der Firma SAP. Dieser Plan zeigt nicht die Hardware des Computers, sondern die Software, aber in Kategorien, die auch auf die Hardware passen. Der Plan ist eine Ingenieurszeichnung und sieht dementsprechend völlig anders aus als dieses farbige Bild hier.

Wenn derartige Bilder nur in Marketingunterlagen vorkämen, könnte man das ja noch tolerieren; aber leider findet man Bilder dieser Art sehr oft auch in Schriften, die den Anspruch erheben, Informationen auf Ingenieursniveau zu bieten. Dieses Bild suggeriert, man könne daraus etwas über die Systemstrukturen lernen. Dies ist aber ein Trugschluß; das Bild sagt einem kaum mehr als ein reiner Text, worin die einzelnen Wörter, die hier vorkommen, in irgend einer beliebigen Reihenfolge untereinander stehen mit dem Hinweis, dies seien alles Dinge, die im Zusammenhang mit dem R3–System irgend eine Bedeutung haben.

Mit diesem Bild bin ich nun bei den Schlußbemerkungen angekommen. In meiner Gliederung hatte ich diesen Schlußbemerkungen den Kommentar beigefügt "Betrübliche Folgen der Einäugigkeit".

Die Konzentration der Informatik auf das Formale hat leider dazu geführt, daß nur noch die Kommunikation zwischen dem Menschen und dem Computer einigermaßen akzeptabel abläuft.

Die Kommunikation zwischen den Menschen selbst aber ist so gestört, wie man sie sich gestörter kaum vorstellen kann.

Eigentlich müßten sich gerade die besten Informatiker der Aufgabe stellen, die Technische Semiotik informatikspezifisch voranzubringen. Denn die Inhalte, über die man in der Informatik reden und schreiben muß, sind sehr viel abstrakter als die Inhalte, worüber die klassischen Ingenieure kommunizieren müssen. Bei denen geht es nämlich vorrangig um Dinge, die man mit den Augen wahrnehmen kann, so daß nicht sehr viel Kreativität dazugehört, Darstellungsformen zu erfinden, bei denen die realen Dinge eins–zu–eins symbolisch aufs Papier gebracht werden.

In der Informatik dagegen ist der Abstand riesengroß zwischen dem, was das Auge primär zu sehen bekommt – also den Quelltexten, der Hardware und den Erscheinungen auf den Bildschirmen – und den Strukturen, die das geistige Auge aus semiotischen Gründen braucht. Erst wenn Begriffsgebäude und zugehörige Darstellungsformen Allgemeingut geworden sind, mit denen dieser Abstand angemessen überbrückt werden kann, wird es möglich sein, den Softwareentwicklungsprozeß so zu gestalten, daß er das Attribut "ingenieurmäßig" wirklich verdient.

Wenn ich es heute geschafft habe, Sie zu sensibilisieren, dann werden Sie in Zukunft vielleicht mehr als bisher beim Hören und Lesen von Informatiktexten und beim Betrachten von Informatikbildern erkennen, wie unbeholfen dort formuliert und gestaltet wird und mit welcher Beliebigkeit die unterschiedlichen Begriffskategorien durcheinandergewürfelt werden.

Der mögliche Vorwurf, es läge nur an mir und speziell daran, daß ich Ingenieur und kein Informatiker sei, daß ich hier Verständnisschwierigkeiten habe, geht ins Leere, denn in meiner Abteilung bei SAP sitzen lauter Diplominformatiker, die ihr Studium mit Auszeichnung abgeschlossen haben, und die dennoch meistens die gleichen Schwierigkeiten haben wie ich, das Informatikerkauderwelsch zu verstehen.

Zu Ihrer Belustigung habe ich für Sie einige charakteristische Textstellen aus Lehrbüchern und Produktmanualen ausgewählt.

Erstes Beispiel: "Rechenvorgänge sind aus Rechenschritten, den Operationen zusammengesetzt. Eine Operation ist eine Anweisung oder ein Klartext."

Zweites Beispiel: "Das Prädikat für die Eingabe ist read(x). Dieses Ziel liest einen Term vom aktuellen Eingabestrom."

Drittes Beispiel: "Durch die Auftrennung eines Objekts in einen Objekttyp und einem Objektnamen lassen sich bequem Systeme beschreiben."

Viertes Beispiel: "Das Ereignis beschreibt das Eingetretensein eines Zustands, der eine Folge bewirkt."

Das letzte Beispiel stammt aus einem Referenzhandbuch zu einem Softwarepaket für den rechnergestützten Systementwurf: "Each activity in the chart may contain a control activity whose role is to supervise the behavior of its siblings. The internal descriptions of control activities are

given by statecharts. ... A statechart starts activities through actions on transitions. ... When an activity with a control activity is started, the control activity and its associated statechart are started." Die hier gestiftete Verwirrung entspricht derjenigen, die entstünde, wenn in einem Text der Unterschied zwischen einem Sänger, seinem Gesang und seinen Noten verwischt würde.

Wenn die Technische Semiotik in der Informatik die gleiche Bedeutung bekäme, wie sie die Technische Mechanik im Maschinenbau hat, die dort über alle vier Semester des Grundstudiums hindurch dreistündig gelehrt wird, dann würde man vermutlich davon verschont bleiben, fast täglich solche semiotischen Fehlkonstruktionen lesen zu müssen.

Man kann die Sache natürlich auch von der amüsanten Seite betrachten und an die Szene in Goethes Faust denken, in der ein angehender Student sich vom Professor Faust beraten lassen will, dabei aber an Mephisto gerät.

Mephisto sagt ihm:	Im ganzen – haltet Euch an Worte! Dann geht ihr durch die sich're Pforte zum Tempel der Gewißheit ein.
Der Student gibt zu bedenken:	Doch ein Begriff muß bei dem Worte sein.
Doch Mephisto erwidert:	Schon gut! Nur muß man sich nicht allzu ängstlich quälen; denn eben wo Begriffe fehlen, da stellt ein Wort zur rechten Zeit sich ein.

Ich möchte meinen Vortrag nun beschließen mit einer Zitatenmischung aus Schopenhauers Abhandlung über die Universitätsphilosophie. Er schimpft dort u.a. über das angebliche Kauderwelsch von Hegel. Seine Schimpftiraden hätten aber auch Wort für Wort gegen das Informatikerkauderwelsch geschrieben sein können. Ich zitiere:

"Daher die jüngeren Gelehrten heut zu Tage meistens keines gesunden Gedankens mehr fähig sind – der wüste leere Wortkram hat ihre Denkkraft aufgelöst und verschwemmt. Da wird die arglose Jugend davongehen mit einem gelähmten Kopf, in welchem fortan bloße Worte für Gedanken gelten, also kastriert im Geiste, ein Gegenstand des Mitleids, ein bleibendes Thema der Vatertränen. Junges frisches Gehirn auf solche Weise zu desorganisieren ist wahrlich eine Sünde, die weder Verzeihung noch Schonung verdient."