„Mathematische Logik und Informatik“*

Professor Konrad Zuse

Die ersten Konstrukteure von Rechenmaschinen wie Schickard, Pascal und Leibniz sahen ihre Aufgabe nur darin, mechanische Hilfsmittel zur Lösung arithmetischer Rechenaufgaben zu entwickeln. Leibniz befaßte sich außerdem mit der Ars combinatoria, einem Vorläufer der heutigen mathematischen Logik. Auch entwickelte er bereits die Möglichkeiten des binären Zahlensystems das er Dyadik nannte.

Die Lösung der arithmetischen Operationen mit Hilfe von Rechenmaschinen stand dann über 2 Jahrhunderte im Vordergrund. Auch die Idee der Programmsteuerung, wie sie von Babbage konzipiert wurde, diente im wesentlichen noch dieser Aufgabe, wobei allerdings bereits die Grundkonzeption der heutigen Computer vorweg genommen wurde. Aber Babbage muß sich wohl schon weitere Gedanken gemacht haben; denn der bedingte Befehl war ihm bereits bekannt, wenn er ihn auch in seinen konstruktiven Plänen noch nicht benutzte.

Die Lochkartentechnik führte eine Reihe von Rechenmethoden ein, die über den Rahmen der reinen Zahlenrechnung hinausgehen. Auswähl- und Sortierprozesse bereicherten die Möglichkeiten der Rechenmaschinen, wobei man sich allerdings über die Analogien solcher Operationen zur mathematischen Logik noch keine Gedanken machte.

Unabhängig davon wurde die mathematische Logik von Mathematikern weiter entwickelt, womit - beginnend mit Leibniz - eine Reihe berühmter Namen verknüpft ist, wie z.B. Bool, Frege, Schröder, Carnap, Hilbert und andere.

Der Brückenschlag zwischen den Problemen des Rechnens und der mathematischen Logik erfolgte etwa gleichzeitig in den dreißiger Jahren auf verschiedenen Wege.

Vom Standpunkt der Mathematik aus entwickelte Turing seine inzwischen berühmt gewordene Idee der berechenbaren Funktionen. Wesentlich dabei ist, daß er das gedankliche Modell einer Rechenmaschine nur benutzte, um vom logischen Standpunkt aus den Bereich der beweisbaren Sätze auf die Berechenbarkeit zurückzuführen. ER dachte, jedenfalls zunächst, wohl kaum daran, damit auch die Konstruktion von Rechenmaschinen zu befruchten. Später erkannte man, daß die

*ZIA 0614. ZuP 035/054. Version 1. Durchgesehen von R. Rojas, G. Wagner, L. Scharf

Idee der Turing Maschine eine gute Basis für theoretische Untersuchungen der Informatik darstellt; jedoch gilt auch heute noch, daß die direkten Auswirkungen dieser Idee auf die konstruktive Entwicklung von Rechenmaschinen unbedeutend sind.

Unabhängig davon bot die inzwischen weiter fortschreitende Entwicklung programmgesteuerter Rechenmaschinen die Möglichkeit, von der anderen Seite, nämlich der praktischen Konstruktion von Rechenmaschinen her, eine Brücke zur mathematischen Logik zu suchen. Programmgesteuerte Rechengeräte führen tausende von aufeinanderfolgenden Rechenoperationen aus, ohne daß dabei der menschliche Benutzer die zahlreichen Zwischenwerte zu Gesicht bekommt. Damit war es möglich, die grundsätzliche Frage nach dem für eine Rechenmaschine günstigsten Zahlensystem zu stellen.

Die traditionelle Rechenmaschinentechnik arbeitete, in Anpassung an den menschlichen Benutzer, mit Ziffernrädchen für 10 Positionen. Die erste große elektronische Rechenanlage, der ENIAC, arbeitete noch mit direkter Simulation solcher aus einer Reihe von Ziffernrädchen aufgebauten Register.

Andere, wie Stibitz und Aiken, benutzten bei ihren ersten Geräten bereits die Relaistechnik, welche konstruktiv das Arbeiten mit Werten, die nur zwei Möglichkeiten zulassen, also Bits, wie wir heute sagen, bedeutet. Stibitz ging dabei konsequent den Weg, das binäre Zahlensystem zu benutzen. Aiken arbeitete mit durch Bits verschlüsselten Dezimalziffern.

Bei meinen eigenen Entwicklungen in Berlin, die etwa gleichzeitig und unabhängig von diesen Arbeiten in den USA liefen, ging ich, ähnlich wie Stibitz, auch konsequent zum vollen binären System über.

Damit wurde ein neues Gebiet aktuell, das wir heute mit Schaltalgebra bezeichnen. Sie wurde ebenfalls unabhängig an verschiedenen Stellen mit verschiedenartigen Ansätzen entwickelt. Die Idee, die Gesetze für Relaisschaltungen mathematisch zu formulieren, ist allerdings nicht unbedingt an die Entwicklung der Rechenmaschinen gebunden. Hansi Piesch entwickelte Ansätze, die allgemeinen Aufgaben dienten. Shannon dachte zwar schon an die Lösung rechnerischer Aufgaben, aber nur als typisches Anwendungsbeispiel von vielen. Auch Aiken entwickelte einen Formalismus, um Schaltungen und ihre Gesetze zu beschreiben.

Es ist vielleicht bezeichnend, daß all diese Versuche zunächst noch darauf hinausliefen, für Schaltungsprobleme eine eigene formale Sprache zu schaffen.

Auch bei meinen eigenen Bemühungen ging ich ähnliche Wege. Das konsequente Arbeiten mit Relaisschaltungen forderte geradezu dazu heraus, leistungsfähige mathematische Werkzeuge zu schaffen, um damit komplizierte Formulierungen elegant durchführen zu können. Ich entwickelte eine „Bedingungskombinatorik“ und erprobte sie in einer Reihe von Schaltungsaufgaben.

Ich schickte diesen meiner Meinung nach neuen Kalkül meinem ehemaligen Mathematiklehrer und hatte das große Glück, daß er mich auf die Analogien dieser Bedingungskombinatori zum Aussagenkalkül aufmerksam machte. Damit war der Weg frei, einen bis dahin sehr abstrakten und für rein theoretische Ziele entwickelten Zweig der Mathematik für praktische Aufgaben des Ingenieurs als Werkzeug zu benutzen. Auch an anderer Stelle erkannte man später, daß die Übernahme eines fertig entwickelten Kalküls, nämlich des Aussagenkalküls, große Vorteile bietet. Heute ist uns das völlig selbstverständlich. Die Auflösung von Schaltungen in „Und-“ bzw. „Oder-“ Glieder ist uns völlig in Fleisch und Blut übergegangen. Nur wenige wissen, daß das ursprünglich nicht selbstverständlich war und z.T. erst auf Umwegen erreicht wurde. Noch Aiken scheute sich, diese einfachen Operationen direkt zu benutzen, und arbeitete anstelle dessen mit - nach unseren heutigen Vorstellungen - umständlichen Formulierungen mit Hilfe der dem Ingenieur geläufigen Operationen der Addition und Multiplikation.

Die Analogien zum Aussagenkalkül erlaubten es nun, die dort geltenden Gesetze sinngemäß auf Schaltungen zu übertragen. So gab mir das Dualitätsprinzip sofort die Möglichkeit, meine Additionsschaltungen gewissermaßen „invers“ aufzubauen, was z.T. zu wichtigen Vereinfachungen führte.

Damit war nun auch der Weg vorgegeben, um Schaltungen verschiedener Technologien auf eine gemeinsame Darstellungsform zurückzuführen. Ich entwickelte eine „Abstrakte Schaltglied-Technik“, welche wahlweise auf mechanische, elektromechanische und elektronische Schaltungstechnologien übertragen werden konnte. So gelang es auch verhältnismäßig leicht, die gesamte Gleitkomma-Logik im binären Zahlensystem mit all ihren Übersetzungsaufgaben vom Dezimalsystem ins binäre System und umgekehrt nur mit Hilfe von Relais, also logisch gesehen, mit Ja-Nein-Werten aufzubauen. Die z.T. sehr komplizierten Bedingungen waren mit Hilfe aussagenlogischer Formeln leicht und elegant darstellbar. Dabei zeigte sich, daß die Relaistechnik die Möglichkeit bietet, fast direkt von der formalen Darstellung zur konkreten Schaltung überzugehen.

Damit war auch der Übergang zur elektronischen Technologie erheblich erleichtert. Als mein Freund Schreyer mitleidig lächelnd meine mechanischen und elektromechanischen Schaltungen betrachtete, meinte er, das müsse man mit Röhren machen. Durch die Verwendung der mathematischen Logik war nun dieser Übergang zu einer grundsätzlich anderen Technologie sehr erleichtert. Schreyer brauchte nur noch für die logischen Grundschaltungen die Bauelemente zu entwickeln, die dann entsprechend den bereits in anderer Technologie bewährten Modellen zusammengeschaltet zu werden brauchten. Damit ging unsere deutsche Entwicklung elektronischer Rechengeräte auch grundsätzlich andere Wege, als die etwas später einsetzenden in den USA, wo man noch mechanische Zählwerke möglichst direkt simulierte.

Ein solcher Erfolg der mathematischen Logik führte nun zu der Frage, ob nicht

auch weitere Kalküle auf dem Gebiete der Computerentwicklung nützlich eingesetzt werden könnten.

Es zeigte sich dann bald, daß der Prädikatenkalkül sehr gut zur Formulierung rechnerischer Probleme geeignet war. So entstand die Idee des allgemeinen Rechnens, welches wesentlich über das reine Zahlenrechnens hinausgeht.

Damit war der Weg frei für zunächst rein theoretische Untersuchungen. Zwar kannten wir damals noch nicht die Begriffe „Hardware“ und „Software“. Allerdings konnten nicht alle theoretisch gewonnenen Erkenntnisse sofort praktisch angewandt werden. Wir entwickelten im Kriege Anfang der vierziger Jahre einige Computer im Auftrage der deutschen Luftfahrtindustire, die jedoch auf klar erreichbare Ziele abgestellt waren.

Diese Arbeiten erfolgten unter sehr beschwerlichen Bedingungen des Krieges, zuletzt während der Bombenangriffe auf Berlin.

Es war im Kriege kaum möglich, Fachkräfte zu bekommen, geschweige denn einen Vollspur-Mathematiker. Da geschah ein Wunder. Die Luftwaffe, die ja an meinen Entwicklungen interessiert war, stellte Fachkräfte von der Front ab, und auf diese Weise erschien eines Tages bei mir Herr Lohmeyer, ein Mathematiker aus der Schule des bekannten Logikers Scholz in Münster. Er traute sich zunächst kaum zuzugeben, daß sein eigentliches Fachgebiet die mathematische Logik wäre und war dann sehr erstaunt, zu hören, daß er genau der Mann wäre, den ich gebrauchen könnte.

Mit Begeisterung gingen wir dann daran, die Beziehungen zwischen mathematischer Logik und Rechenmaschinen auszubauen. Die Technologie der Relaisschaltungen begriff er sehr schnell. Wir machten uns meistens nicht erst die Mühe, die erforderlichen Änderungen in den Geräten erst über das Konstruktionsbüro, die Arbeitsvorbereitung usw. laufen zu lassen, sondern nahmen selbst den Lötkolben in die Hand und in wenigen Minuten waren die aussagenlogischen Ansätze vom Papier in die Wirklichkeit übertragen.

Trotz all der Unannehmlichkeiten der letzten Kriegsjahre in Berlin, hatte diese Zeit doch auch ihre guten Seiten. Das Zusammenspiel zwischen Mathematiker und Ingenieur war sehr harmonisch und wir wuchsen so in das Aufgabengebiet hinein, daß jeder beide Seiten bis zu einem gewissen Grad beherrschte.

Es war auch ein großes Erlebnis für mich, als Herr Lohmeyer eines Tages seinen ehemaligen Lehrer, Herrn Professor Scholz, aus Münster nach Berlin in unsere Werkstatt holte. Ich war zunächst sehr skeptisch und glaubte nicht, bei einem ausgesprochenen Tehoretiker Verständnis für derartig „profane“ Aufgaben wie die Konstruktion von Rechenmaschinen erwarten zu dürfen. Zu meiner Überraschung zeigte er jedoch sofort lebhaftes Interesse an den Problemen, was uns ein Ansporn war, unseren einmal eingeschlagenen Weg intensiv weiter zu verfolgen.

Das Jahr 1945 stellte uns dann allerdings zunächst vor sehr reelle Aufgaben. Unsere Diskussionen fanden zunehmend im Luftschutzkeller statt. Zuletzt mußten wir - mitten im Bombenkrieg - das Gerät Z 4 aus Berlin retten. Es wurde schließlich nach abenteuerlicher Reise in einem kleinen Alpendorf, Hinterstein im Allgäu, in einem Schuppen versteckt.

Etwa zur gleichen Zeit, als wir notgedrungenermaßen unsere „Hardware-“ Entwicklung von Computern unterbrechen mußten, wurden die Ergebnisse der amerikanischen Entwicklungen weltbekannt. Es lag eine Tragik darin, daß wir nun zusehen mußten, wie an anderer Stelle ähnliche Ideen, mit großem materiellem Aufwand verwirklicht werden konnten, während für uns in Deutschland nur rein theoretische Arbeit übrig blieb. Dadurch wird aber auch mancher Unterschied erklärt.

Die bei uns während des Krieges entwickelten Geräte waren auf klar formulierte Ziele abgestellt. Die mathematische Logik wurde dabei voll eingesetzt, um die Schaltungen zu entwickeln. Für die Programmierung bedurften diese Geräte jedoch noch keines besonderen Aufwandes. Für die damals sowohl bei uns als auch in den USA verfügbaren Computer brauchte man noch keine hochentwickelten algorithmischen Sprachen.

Die Einsamkeit des bayrischen Alpendorfes gab mir nun Gelegenheit, meine theoretischen Gedanken zu ordnen. Schon während des Krieges hatte ich an „logischen“ Rechengeräten zur „Rechenplan-Fertigung“ und anderen Aufgaben gearbeitet. Dabei war es selbstverständlich, daß sämtliche Informationen, also nicht nur Zahlen sondern auch Programme, komplizierte Datenstrukturen usw. in Bitmuster aufgelöst und somit auch gespeichert und der Informationsverarbeitung zugänglich gemacht werden können.

Als einen wesentlichen Punkt bei solchen logistischen Rechenmaschinen erkannte ich die Rückwirkungsmöglichkeit von errechneten Resultaten auf den Ablauf der Rechnung, also das, was wir heute als bedingte Befehle, Umrechnung von Adressen, bedingte Ablaufanweisungen usw. bezeichnen. Man kann diese Möglichkeiten konstruktiv durch einen einzigen Draht symbolisieren, der vom Rechenwerk zum Programmwerk führt. Ich hatte eine ausgesprochene Scheu, diesen Draht konkret zu legen; denn es war mir klar, daß er zu unübersehbaren Konsequenzen in Richtung der Möglichkeiten des Rechnens überhaupt führen würde.

All diese Überlegungen konzentrierte ich nun im ersten Nachkriegsjahr auf die Entwicklung einer algorithmischen Sprache, den Plankalkül. Ich stellte mir die Aufgabe, zunächst einmal einen Formalismus zu entwickeln, mit dem sämtliche überhaupt denkbare Informationsverarbeitungsprozesse ohne die Wortsprache formuliert werden können. Damit wurden die wesentlichen Charakteristike späterer Entwicklungen algorithmischer Sprachen vorweg genommen.

Einige Jahre später wurde ich dann allerdings wieder durch die auch in Deutsch-

land wieder mögliche Hardware-Entwicklung in Anspruch genommen. Es kam zu der sehr angenehmen Zusammenarbeit mit der Eidgenössischen Technischen Hochschule in Zürich, wo das aus Berlin gerettete Gerät Z 4 für einige Jahre aufgestellt werden konnte.

Auch in Zürich setzte sich zunächst die gute Zusammenarbeit zwischen Mathematiker und Ingenieur fort. Das Zusammenspiel mit den Herren Stiefel, Speiser und Rutishauser war sehr fruchtbar und der Unterschied zwischen Hardware und Software-Entwicklung begann sich erst allmählich herauszubilden.

Auch die Analogien zwischen Schaltungen und Aussagenlogik wurden uns allen bald zur Selbstverständlichkeit.

Schwieriger war es allerdings, um diese Zeit Verständnis für meine weiteren theoretischen Arbeiten, insbesondere für den Plankalkül, zu finden. Anfang der fünfziger Jahre begannen die elektronischen Rechengeräte sich Schritt für Schritt in Richtung der heutigen Computer zu entwickeln. An anderer Stelle hatte man keinerlei Hemmungen, den entscheidenden Draht der Rückwirkung vom Rechenwerk zum Steuerungswerk zu legen. John v. Neumann konnte als erster ein Gerät mit Programmspeicherung bauen, bei dem die Möglichkeit der Beeinflussung des Programms selbst durch die Ergebnisse der Rechnung gegeben war. Sein Konzept war sehr allgemein und wurde erst später in seinen Möglichkeiten voll ausgebaut.

Trotzdem bereits damals die Möglichkeit des universellen Rechnens gegeben war, erstreckten sich die Bemühungen der Programmformulierung doch noch auf die gerade vorliegenden Tagesprobleme. Numerische Rechnungen standen noch absolut im Vordergrund. Logische Operationen wurden im wesentlichen nur im Zusammenhang mit bedingten Befehlen eingesetzt.

Dementsprechend waren die damals an verschiedenen Stellen entwickelten algorithmischen Sprachen zunächst auf derartige Probleme beschränkt.

Ich glaubte damals, daß nunmehr die Stunde gekommen sei, meinen Plankalkül ins Spiel zu bingen. Die Entwicklung des Computers war nunmehr auf breiter Basis in Gang gekommen. Was lag da näher, als von vornherein die Weichen richtig zu stellen, um alle Möglichkeiten des Einsetzens der mathematischen Logik schon in den Ansätzen zu erfassen.

Es war eine der größten Enttäuschungen meines Lebens, daß es bei dieser meiner zweiten Begegnung mit Mathematikern nicht zu einer so harmonischen Zusammenarbeit kam, wie seinerzeit während des Krieges. Hinzu kam, daß ich persönlich durch die Hardware-Entwicklungen und den Aufbau einer Firma voll in Anspruch genommen war und mich daher der Aufgabe der algorithmischen Sprachen nicht mehr widmen konnte.

Rückblickend frage ich mich allerdings, warum es damals nicht zu einer fruchtbaren Zusammenarbeit gekommen ist. Abgesehen davon, daß der Einfluß der ame-

rikanischen Entwicklungen übermäßig stark war, mögen folgende 4 Mißverständnisse dazu beigetragen haben, daß meine Gesprächspartner und ich in den wesentlichen Punkten aneinander vorbei redeten.

1. Da im Plankalkül von vornherein die volle Variationsbreite aller möglichen Datenstrukturen erfaßt werden sollte, entwickelte ich eine Darstellungsform, die in verschiedenen Zeilen die zu den einzelnen Daten gehörenden Elemente, wie Indizes, Komponentenkennzeichnungen und Datenstrukturen enthalten. Diese Disziplin des preußischen Kasernenhofes gibt zwar ein Maximum an Klarheit, ist aber für den Mathematiker ungewohnt. Es ist nun leider so, daß solche Gewohnheitseffekte für die Einführung neuer Ideen von großer Bedeutung sind.

2. Der Plankalkül arbeitet bewußt mit dem Bis als elementarem Datenelement und baut alle weiteren Strukturen darauf auf. Damit ist u.a. auch die Möglichkeit gegeben, die arithmetischen Operationen in jedem beliebigem Zahlensystem, sei es in Festkomma- oder in Gleitkommadarstellung bis in die letzten Bits zu zerlegen.

   Das hat zu dem Eindruck geführt, daß man im Plankalkül diese Zerlegung durchführen müsse. Das ist jedoch keineswegs der Fall; denn der Plankalkül erlaubt es, entsprechend den heutigen Sprachen ohne weiteres Datenarten wie Int, Real usw. einzuführen.

3. Gerade die damals in den Jahren um 1955 herum interessierenden Programme für numerische Rechnungen wurden im Plankalkül nicht behandelt, weil sie zu trivial erschienen und keine besonderen Schwierigkeiten bei der Entwicklung einer algorithmischen Sprache boten.

   Das führte wieder zu dem Eindruck einer auf abstrakte Probleme bezogenen Sprache.

4. Im Plankalkül wurde ein verhältnismäßig breiter Raum den logischen Operationen des Prädikatenkalküls überlassen. Insbesondere das Schachspiel zeigte sich als ein ehrvorragendes Modell, um die Flexibilität einer algorithmischen Sprache zu testen. Dadurch entstand wiederum der Eindruck einer auf logische Operationen spezialisierten Sprache, die man, aus der Sicht von etwa 1955 gesehen, nicht für erforderlich erachtete.

Diese 4 Gründe zeigen, daß die Ablehnung bzw. Ignorierung des Plankalküls in erster Linie psychologische Gründe hatte. Zu wirklich sachlichen Diskussionen ist es damals leider gar nicht erst gekommen.

Die weitere Entwicklung führte bedauerlicherweise zu einer immer weiteren Entfremdung zwischen Mathematiker, Ingenieur und Anwender. Die verschiedenen

Aufgaben erforderten bald die Bildung von Spezialgebieten, die nur von Spezialisten beherrscht werden konnten.

Der von John v. Neumann gezeigte Weg wurde in verschiedenen Richtungen konsequent ausgebaut. Van der Poel, Fromme und andere entwickelten sehr flexible Geräte, mit denen sehr komplizierte Informationsverarbeitungsprozesse durchgeführt werden konnten.

An der Entwicklung algorithmsicher Sprachen wurde intensiv gearbeitet. Sprachen wie Cobol, Fortran und Algol waren den damals vorliegenden Aufgaben gut angepaßt und konnten eine weite Verbreitung finden.

Das alles war jedoch noch stark anwendungsbezogen. Was jedoch nicht ohne weiteres voraus zu sehen war, war die Ausbildung eines sehr theoretischen Zweiges der Informatik, wie wir heute sagen. Die Mathematiker sahen ihre Aufgabe zunächst weniger darin, ihren logischen Scharfsinn auf die Entwicklung einer wirklich universellen algorithmischen Sprache nach Art des Plankalküls zu konzentrieren, sondern eher darin, eine Art Metatheorie zu schaffen, um das Problem der formalen Sprachen, und insbesonders der algorithmischen und Programmiersprachen zu untersuchen.

Vielleicht hatten manche Mathematiker zunächst einige Minderwertigkeitskomplexe, als sie sich vor die Aufgabe gestellt sahen, sich mit so profanen Dingen, wie Rechenmaschinen zu befassen. Für den „reinen“ Mathematiker war das sicher ein Stein des Anstoßes. Nun, sie haben in den letzten 20 Jahren gezeigt, daß sie die Kunst, weltabgewandte Theorien zu schaffen, mindestens ebenso beherrschen, wie die sogenannten reinen Mathematiker. Dabei werden verschiedene Teile neuer mathematischer Lehrgebäude mit herangezogen bzw. ausgebaut, wie z.B. die Informationstheorie und die Automatentheorie. Ihre Bedeutung für die Computerentwicklung wird z.T. stark überschätzt.

Unter dem Motto, algorithmische Sprachen mathematisch klar und eindeutig zu formulieren, sind heute Theorien der Sprachbeschreibung entstanden, die an Abstraktheit den Vergleich mit mancher weltabgewandten mathematischen Theorie ohne weiteres vertragen. Die glückliche Kombination zwischen mathematischer Logik und der Arbeit des Ingenieurs, wie wir sie in den Kriegsjahren begonnen hatten, wurde leider nicht im damaligen Sinne fortgesetzt.

Inzwischen ist es längst klar geworden, daß wir zur Programmierung von Computern Sprachen brauchen, welche den vollen Umfang logischer flexibler Programme abdecken. Wir haben heute denn auch einige Sprachen, wie Algol 68 und PL/1, die diese Bedingung mehr oder weniger erfüllen, oder dies zumindestens anstreben.

Um trotz des babylonischen Sprachwirrwarrs zu brauchbaren Ergebnissen zu kommen, hat in letzter Zeit die Idee des strukturierten Programmierens Bedeutung gewonnen. Sie entspringt dem Wunsch des Praktikers nach Klarheit und Über-

sichtlichkeit. Aber vielleicht ist das nur der halbe Weg. Die Diskussionen über den Plankalkül, die ich in letzter Zeit führen konnte, haben mir gezeigt, daß dem Begriff der logischen algorithmischen Sprache, die jedoch implementationsnahe ist, besondere Bedeutung zukommt.

Eine logische algorithmische Sprache bietet die Mittel, um rechnerische Abläufe logisch klar zu formulieren, ohne auf Einzelheiten der Implementation wie Ein- und Ausgabe, Speicherreservierung usw. einzugehen. Sie muß jedoch implementationsnahe sein, um den Übergang zu maschinenfertigen Programmen zu erleichtern.

Eine algorithmische Sprache muß alle Möglichkeiten der mathematischen Logik ausschöpfen, jedoch so formuliert sein, daß sie auch der Praktiker versteht. Es wird heute sehr viel über Sprachtheorien, Dialogsprachen usw. geschrieben. Eine wichtige Aufgabe wird dabei jedoch vernachlässigt: der Dialog zwischen Theoretikern und Praktikern. Man hat manchmal den Eindruck, daß die Theoretiker ihre Ehre darin sehen, ihre Bücher so abstrakt wie möglich zu schreiben. Einen Zusammenhang einfach und klar zu beschreiben, gilt als unfein und unwissenschaftlich. Das führt dazu, daß diese Theoretiker ein Eigenleben führen und daß die Praktiker, hilflos und im Stich gelassen, zusehen müssen, wie sie trotzdem mit den Problemen fertig werden.

Nach rund 40 Jahren intensiver Computerentwicklung sollte das junge Gebiet der Informatik aber langsam aus den Kinderschuhen heraus kommen und seiner Reife entgegen wachsen. Die vor uns liegenden Aufgaben sind so groß, daß sie noch gar nicht voll überblickt werden können. Wir werden täglich mit den verschiedenen kritischen Situationen in Politik und Wirtschaft konfrontiert. Die Lösungen dieser Probleme sind nur mit intensivem Einsatz der Computer möglich. In Zukunft werden die Computerfachleute an den Schalthebeln der Weltwirtschaft stehen; und zwar unabhängig von dem jeweiligen Wirtschaftssystem. Fragestellungen, die noch unsere Urgroßväter bewegten, verlieren an Bedeutung. Sowohl kapitalistische als auch sozialistische Wirtschaftssysteme sind nur so gut, wie das System der Computer, auf welches sie sich abstützen.

Sind wir für diese gewaltigen Aufgaben gerüstet?

Die Beantwortung dieser Frage erfordert eine gründliche Besinnung. Wir brauchen jedenfalls einen neuen Typ von Informatiker, der seine Aufgabe darin sieht, Theorie und Praxis in harmonischer Form zu vereinigen.