Z i f f e r n - R e c h e n a u t o m a t
m i t P r o g r a m m i e r u n g n a c h
m a t h e m a t i s c h e m F o r m e l b i l d

Habilitationsschrift
zur Erlangung des akademischen Grades
eines Dr. phil. habil. in der
Mathematisch-Naturwissenschaftlichen Fakultät
der Friedrich-Schiller-Universität Jena

vorgelegt von

W i l h e l m K ä m m e r e r

aus B ü d i n g e n (Oberhessen)

Dekan: Prof. Dr. Gerisch

Gutachter: Prof. Dr. Weinel

Zweitgutachter: Prof. Dr. Maier

Tag der Habilitation: 11. 12. 1958

Ziffern-Rechenautomat mit Programmierung nach mathematischem Formelbild

I. Teil

§ 1 Einleitung

Programmgesteuerte Rechenautomaten in der z. Z. üblichen Bauart betrachten als Programm-Elemente die einzelnen arithmetischen, logistischen oder organisatorischen Grundoperationen jeweils in Verbindung mit Adressen. Die solcherart arbeitenden Automaten im Typ der Ein- oder Mehr-Adress-Maschinen machen es notwendig, bei der Programmierung den in Form eines Algorithmus vorliegenden Plan dadurch maschinengerecht umzubilden, daß dieser den Programmelementen entsprechend aufgeteilt und gegebenenfalls umgestellt wird. Dadurch entfernt man sich aber derart weit von dem international eingebürgerten mathematischen Formelbild, daß man viele Vorteile verliert, die eine über Jahrhunderte verlaufene Entwicklung eben in diesem Formelbild gewonnen hat. Während eine algorithmische Niederschrift des Programms in Form von Indexvorschriften, Plangleichungen und Entscheidungsoperationen allgemein verständlich, übersichtlich, konzentriert und leicht hinschreibbar wie kontrollierbar ist, erfordert die maschinengerechte Programmierung erheblichen Aufwand an Zeit und routinemäßiger Konzentration und führt zu einer unübersichtlichen, weitschweifigen, schwer kontrollierbaren und derartig individuellen Form, daß Verständigungen über angewandte Verfahren zwischen verschiedenen Rechenzentren nur schwer möglich sind.

Dazu kommt, daß die Notwendigkeit einer direkten Adressenangabe an Stelle der Symbole der mathematischen Formeln von Seiten des mathematischen Problems aus als unerwünschte, das Problem selbst nicht berührende Maßnahme gesehen wird.

War man zunächst froh, überhaupt Rechenautomaten zu besitzen, so mußte man bald erkennen, daß man ihre Vorteile mit einer zeitraubenden, Unsicherheit tragenden Programmierungsarbeit bezahlte.

Es hat daher nicht an zahlreichen Versuchen gefehlt, diese Schwierigkeiten dadurch zu überwinden, daß man den Automaten selbst für die Übernahme auch dieser Routinearbeit heranzuziehen versuchte. Dabei wurde von verschiedenen Stellen erfolgreich an der Schaffung einer auf Bibliotheksprogramme aufbauenden Programmierungstechnik gearbeitet, teils in interpretierendem teils in kompilierendem Verfahren.
Auf der anderen Seite stehen Arbeiten, die ein automatisches Programmieren im engeren Sinn anstreben und den Übergang von den dem Problem zugrundeliegenden Formeln zu dem Programm völlig zu mechanisieren gestatten, wobei der Automat selbst den Rechenplan bildet.

Bei diesen Methoden nimmt man die in der Struktur der derzeitigen Automaten liegenden Unzulänglichkeiten hin, bemüht sich aber, ihre unangenehmen Auswirkungen von dem Programmierer fernzuhalten.
Man hat auch schon versucht, in der Befehlsliste stärker das mathematische Formelbild zu berücksichtigen und beispielsweise das Gleichheitszeichen sowie neben dem üblichen Multiplikationsbefehl einen zweiten vorgesehen, der einen dem Multiplikationszeichen vorangehenden Klammerschluß einbezieht. (1

In der vorliegenden Abhandlung werden in konsequenter Weiterführung der abgelaufenen Entwicklungstendenz auf einem neuen Weg Prinzipien für die Struktur eines Automaten erarbeitet, der in enger Anlehnung an das mathematische Formelbild eine bequeme, übersichtliche und adressenfreie Programmierung gestattet. Einbegriffen ist dabei das Auftreten von Indizes in der mathematisch üblichen Form, sowie das Auftreten von Funktionen auch mehrerer Parameter und in Verschachtelung.

- -

(1 N. J. Lehmann Bericht über den Entwurf eines kleinen Rechenautomaten an der Technischen Hochschule Dresden.
Berichte der Math. Tagung Berlin 53/Febr.

§ 2 Das Problem der Adressenänderungen

Zur Aufbereitung mathematisch-logistischer Probleme für die numerische Durchrechnung sowie zur Beschreibung numerischer Methoden eignet sich als Hilfsmittel das Fluß- oder Strukturdiagramm sowie eine sich daraus entwickelnde algorithmische Schreibweise.(2
Als Elemente treten Indexvorschriften, Ablaufentscheidungen - ev. jene mit diesen kombiniert - und Plangleichungen auf. Dabei trägt eine Plangleichung ($\Longrightarrow$) im Gegensatz zu einer Gleichung ($=$), die im Rahmen eines Algorithmus nur eine Feststellung ist, einen Richtungssinn entsprechend dem dynamischen Prozeßablauf und gibt linksseitig mit den klassischen Formelsymbolen der Mathematik ein Rechenverfahren an, das an bekannten Größen angreift und zu einer neuen Größe führt, die unter dem rechtsstehenden Symbol eingeführt wird.

Rationalisierungsbestrebungen führten dazu, eine Technik zu entwickeln, die es gestattet, öfter gebrauchte Programme bzw. Teile solcher in selbständiger und allgemein verwendbarer Weise vorzubereiten und als Bibliotheksprogramme aufzubewahren. Die beim Zusammenfügen solcher Teil- und Unterprogramme auftretenden Probleme führten zu Maßnahmen, die unter Adressenänderungen I., II. und III. Art gekennzeichnet sind. (3 (4

(2 H. Rutishauser: Maßnahmen zur Vereinfachung des Programmierens.
Nachrichtentechnische Fachberichte der NTZ, Band 4 - 1956
Elektronische Rechenmaschinen u. Informationsverarbeitung
S. 26-30

(3 Golstine, H.H. und von Neumann, I.: Planning and coding of problems for an electronic computing instrument.
Part II Vol. 1,2,3 - Institute of Advanced Study.
Princeton 1947/48

(4 Sauer, R. und Bauer, F.L.: Einführung in die numerische Verfahrenstechnik für Rechenautomaten.
MTW-Mitteilungen TH Wien, Mathematisches Labor
III. Jahrgang Nr. 1, S. 8-14 - Nr. 2, S. 42-47

Die Adressenänderungen I. Art werden dabei der Tatsache gerecht, Adressen in einem Bibliotheksprogramm zunächst nur relativ etwa zu dem Anfang angeben zu können, nach dem Einbau in spezieller Lage aber für den Rechenablauf als absolute Adressen zu benötigen. Zweckmäßig wird diese Übersetzung während der Eingabe durchgeführt, indem die Rufadresse des Blocks, d. i. die absolute Adresse der Zelle, in die die erste Zeile des Teilplans (relative Adresse 0) in spezieller Lage eingebaut wird, festgehalten und allen auf dem Eingabemittel (Lochkarte) als relativ angemerkten Adressen additiv zugefügt wird(4. Eine davon abweichende Methode wird in § 3 angeführt.

Adressenänderungen II. Art dienen dazu, den Verkehr zwischen verschiedenen aus eingebauten Bibliotheksprogrammen entstandenen Teilen des Gesamtprogramms zu ermöglichen, insbesondere für die Überführung von Parameter, die in einem Teil bereitstehen oder berechnet werden, in einen anderen Teil, der auf ihnen weiter aufbauen soll. In diesen Kreis gehört auch das Problem, beim Sprung von einem (relativen) Hauptprogramm zu einem Unterprogramm diesem die spätere Rückkehr in jenes übergeordnete Hauptprogramm zu ermöglichen. (4 (5

Adressenänderungen III. Art schließlich dienen dazu, bei Schleifendurchläufen die nötigen Indexänderungen durchzuführen.

Es sind, bedingt durch den unterschiedlichen Bau der Rechenautomaten, von einander sehr abweichende Maßnahmen maschinentechnisch und programmseitig erdacht worden, um diesem Problemkreis der Adressenänderungen gewachsen zu sein.

(5 Schecher, H.: Maßnahmen zur Vereinfachung von Rechenplänen bei elektronischen Rechenanlagen.
ZAMM - ingenieurwissenschaftliche Forschungsarbeiten
Band 36, Heft 9/10, Sept./Okt. 1956, S. 377-395

Hervorgehoben seien hier das Verfahren, mittels Indexregister einen Einfluß auf die Adressen zu ermöglichen (5(6, das Verfahren der "Adresse von Adresse" (4 und das der Adressensubstitution. (4

Während die beiden letzten stärker dem inneren Maschinenablauf dienen, bietet sich das Prinzip der Indexregister auch, um programmseitig eine differenzierte Charakterisierung einer mathematischen Größe zu erreichen, wie es der Verwendung des Index im mathematischen Formelbild entspricht.

Im Zusammenhang mit Ablaufentscheidungen sei auf das Anhaken (7 einer Zahl verwiesen, als ein Mittel, das, aus dem Rechenbetrieb mit menschlichen Rechnern erwachsen, sich auch bei Automatenbetrieb bewährt, um diesem "ohne viele Worte", d. h. aufwandsparsam, auf das Eintreten eines besonderen Ereignisses, etwa auf das Ende eines Zahlenblocks, aufmerksam zu machen.

§ 3 Symbol und Adresse

Wenn man das Gesamtprogramm aus Teilplänen aufbaut und dabei auch solche allgemeinen Charakters, sogenannte Bibliothekspläne, einbezieht, werden die zu suchenden Größen in unterschiedlichster Art gekennzeichnet sein können. Diese Kennzeichnung kann einmal durch Adressen geschehen, andererseits aber auch durch Symbole, wobei diese wieder reine Symbole sein können oder auch solche mit Adressencharakter.

(6 Bookate, Kl.: Adressenmodifikation mit Indexregistern bei der Type IBM 704
Nachrichtentechnische Fachberichte der NTZ
Band 4-1956 Elektronische Rechenmaschinen u. Informationsverarbeitung, S. 150-153

(7 Lehmann, N. J.: Bemerkungen zur Automatisierung der Programmierung für Rechenautomaten. Nachrichtentechnische Fachberichte der NTZ Band 4 - 1956 Elektronische Rechenmasch. u. Informationstheorie S. 143, s. auch S. 166

Reine Symbole schließen sich am engsten an das mathematische Formelbild an; sie sind mit den darin auftretenden Buchstaben identisch oder aus ihnen abgewandelt, um sie für den Automaten lesbar zu machen. (Beispielsweise kann π eine solche Abwandlung etwa in pi oder auch nur p nahe legen). Ihre Behandlung von Automatenseite her ist am schwierigsten und erfordert wohl stets eine Umrechnungstechnik zwischen Eingabe und Rechnung. (Kompilierendes Verfahren).

Symbole mit Adressencharakter entstehen aus den Symbolen des mathematischen Formelbildes, wenn der Programmierer sie etwa durch fortlaufende Nummern ersetzt oder sie durch solche ersetzbar macht.

Um den Unterschied zwischen einem reinen Symbol und einem Symbol mit Adressencharakter an einem aus dem täglichen Leben entnommenen Vorgang zu erläutern, sei auf Anzeige unter Chiffre in einer Zeitung hingewiesen. Diese Chiffre kann Adressencharakter haben, wenn etwa die Redaktion die Adresse der ein Inserat aufgebenden Personen unter laufender Nummer notiert und diese Nummer als Chiffre wählt. Sie kann aber reines Symbol bleiben. Hier muß die Redaktion zwei zugeordnete Informationen, die Chiffre und die Adresse des Auftraggebers, notieren.
In einem vom Mathematiker zur Eingabe fertig gemachten Programm muß ein und dasselbe Symbol auf zwei verschiedene Weisen auftreten. Zunächst muß es im Plan überall dort stehen, wo im Verlauf der Rechnung nach dieser Größe gefragt wird, da sie dann Objekt des Programmablaufs ist. Als solches wird das Symbol auch nach der Eingabe in den entsprechenden Zellen des Speichers stehen. Erst ein nachfolgendes kompilierendes Vorgehen wird es an diesen Stellen durch absolute Adressen ersetzen. Das Symbol muß aber noch einmal in dem einzugebenden Plan in anderer Form vorhanden sein, um bei der Eingabe des Plans solche Maßnahmen zu veranlassen, daß die bei der späteren Rechnung auftretende

Frage nach einem durch das Symbol gekennzeichneten Objekt zu der Größe selbst führen kann. Es muß im Zusammenhang mit dem Vorgang, daß die Maschine bei der Eingabe die Größe in eine Zelle einspeichert oder ihr für spätere während der Rechnung erfolgende Einspeicherung eine Zelle zuteilt, die absolute Adresse dieser Zelle zusammen mit dem Symbol festgehalten werden. Es wird somit bei der Eingabe ein Symbol, das in dieser Funktion im Eingabeplan enthalten ist, nicht im Plan mitgespeichert, vielmehr von der zugehörigen Größe getrennt, aber mit deren absoluter Adresse zusammen "anderweitig" festgehalten. Dieses "anderweitig" hängt nur von dem Charakter des Symbols ab: Hat das Symbol Adressencharakter, so wird die absolute Adresse, unter der die Größe gespeichert oder zur Speicherung vorgesehen wurde, in der Zelle notiert, deren Adresse gleich dem als Adresse aufgefaßten Symbol ist oder sich aus diesem einfach bilden läßt, wenn dieses etwa relativ zur Rufadresse des Vermittlungsblocks angesetzt ist. Fehlt dagegen ein solcher Adressencharakter des Symbols, so wird das "anderswo" mit der nächst freien Zelle eines als Vermittlungsblock vorgesehenen Speicherteils beginnen.

Einen solchen Vermittlungsblock hat man in Anlehnung an die analoge Funktion als Telefonbuch bezeichnet. Allerdings hat er mehr Notizblock-Eigenschaft: Bei Symbolen mit Adressencharakter liegt das Fernsprech-Verzeichnis eingangseitig gedruckt vor, die Informationen ausgabeseitig müssen während des Gebrauchs nachgetragen werden, bei reinen Symbolen wird das Verzeichnis auch eingabeseitig erst bei Bedarf angelegt.

Betrachten wir andererseits die Kennzeichnung durch eine Adresse, so finden wir neben der absoluten Adresse die relative und die freie Adresse.
Die Verwendung einer absoluten Adresse entfernt den Programmierer am stärksten von seiner eigentlichen Aufgabe als Mathematiker.

Sie zwingt ihn zur Vornahme der speziellen Speicherzuweisung. Man wird sie daher nur in Ausnahmefällen anwenden.

Die relative Adresse wird besonders in Teilprogrammen mit Bibliothekscharakter dem Ablauf im Teilprogramm selbst dienen. Bei der Eingabe wird sie in eine absolute Adresse umgewandelt. Dazu ist nötig, daß der Automat die Rufadresse kennt, das ist die absolute Adresse, unter der die erste Zeile des Unterprogramms im Rahmen des Gesamtprogramms gespeichert wird. Es muß somit bei Beginn der Einspeicherung eines Blocks (Bibliotheksprogramms) der Zählerstand festgehalten werden, um ihn zu Daten späterer Zeilen des Blocks, die als relativ gekennzeichnet sind, zu addieren. Eine davon abweichende Methode bezieht eine relative Adresse auf die Adresse der Zeile, in der sie auftritt. Damit entfällt die Notwendigkeit einer Speicherung des Zählerstandes, da bei der Eingabe der gerade anliegende Zählerstand additiv der relativen Adresse zuzufügen ist.

Nun treten gerade in Bibliotheksprogrammen Größen auf, die man im voraus noch nicht einmal durch Symbole kennzeichnen kann, da ja der Zusammenhang, in dem das Bibliotheksprogramm in dem vorliegenden speziellen Fall eingesetzt werden soll, nicht bei der auf allgemeine Verwendung abzielenden Schaffung des Bibliotheksprogramms bekannt sein konnte.
Bei der Schaffung eines Bibliotheksprogramms weiß man ja weder, wo es zu stehen kommen wird, noch, wie die zur Verarbeitung aufzunehmenden oder erarbeiteten Größen benannt sein werden. Jenes führte zur Verwendung relativer Adressen, dieses bringt den Begriff der freien Adresse.

Es erscheint zweckmäßig, auch hier die Gedanken an ein konkretes Beispiel zu fixieren:

In einem Problem trete die Größe a auf und werde auch im Plan durch dieses Symbol gekennzeichnet.

Im weiteren Verlauf werde nun eine gewisse Funktion f (a) benötigt, wobei die Art dieser Funktion so speziell sei, daß man nicht die Absicht hat, das ihrer Bildung dienende Teilprogramm in die Bibliothek aufzunehmen.
In diesem Fall wird man einen Teilplan schaffen, der diese Funktion errechnen kann und wird dabei an den Stellen, an denen nach dem Argument gefragt wird, eben das Symbol a verwenden.

Bei einem anderen Problem werde auch eine Funktion, etwa die Sinusfunktion, mehrfach benötigt, und sogar mit verschiedenen Argumenten a, b, usw.
Selbst wenn für die Sinusfunktion noch kein Bibliotheksprogramm vorhanden wäre und man auch nicht die Absicht hätte, den anliegenden Fall dazu auszuweiten, so läge hier folgende Alternative vor: Man könnte wie im vorigen Beispiel ein Teilprogramm für sin a schaffen und es an die passende Stelle einschieben, um dann mit sin b ebenso zu verfahren. Man könnte aber auch, und das ist der ökonomische Weg, den Sinus-Teilplan nur einmal einbauen.

Führt man jetzt für das Argument im Sinne des Teilplans ein neues Symbol, etwa z, ein, so wär vor den jeweiligen Sprungstellen (Aufrufstellen aus dem Hauptprogramm) zum Unterplan im Hauptprogramm entsprechend eine Plangleichung $a \Longrightarrow z$ bzw. $b \Longrightarrow z$ usw. einzusetzen. Dieses neue Symbol z stellt auch an den Vermittlungsblock entsprechende Raumforderung.
Es ist zu beachten, daß im Fall eines Bibliotheksplans dieses Symbol z ein für alle Mal, wenn dieser Plan zur Verwendung kommt, für andere Verwendung blockiert ist.
Es spielen aber sehr oft eine Reihe Parameter in ein Bibliotheksprogramm ein, so daß sich die Schwierigkeiten dementsprechend noch steigern.

Nun kann man die Größen, die einem Unterplan von außen zugeführt werden müssen, um ihn arbeitsfähig zu machen, und die man als freie Parameter bezeichnet, eben diesem Unterplan in bestimmter Reihenfolge zuordnen, so daß beispielsweise ein Unterplan für Bildung einer Potenz mit ganzem Exponenten die Basis als 1. Parameter, den Exponenten als 2., das Resultat als 3. und die Rückkehradresse als 4. Parameter ansieht.
Damit kann das Unterprogramm selbst eine Struktur annehmen, in der wieder gewisse relative Adressen, hier 1 bis 4, auftreten. Dem Programmierer wird, wenn er die Absicht hat, auf ein Bibliotheksprogramm zurückzugreifen, durch ein Deckblatt von dieser "Verabredung" Kenntnis gegeben. Ordnet er jetzt in seinem Hauptplan die jeweiligen speziellen Symbole für die Parameter anschließend an die Aufrufstelle ein, so genügt für das Unterprogramm die Kenntnis einer einzigen absoluten Adresse, nämlich die der Aufrufstelle, um über sie und die relativen Adressen zu allen Parametern zu kommen. Diese Kenntnis läßt sich aber mit einer Indexregister-Technik sehr bequem vermitteln, wenn etwa bei einem Sprung mit "Rückblickabsichten" der momentane Zählerstand durch die Kundgabe dieser Absicht in einem speziellen Indexregister automatisch festgehalten werden kann und dieses dann seinen Inhalt additiv den als relativ zur Aufrufstelle gekennzeichneten Adressen zufügt.

Von Samelson wird das von Wilkes stammende Verfahren der "floating address" als Verfahren der absoluten Leitzellen benannt, das letztere demgegenüber als Verfahren der relativen Leitzellen (Versorgungszellen) bezeichnet.

Es erscheint angebracht, diese Verhältnisse noch von einer anderen Seite her zu beleuchten.
An gewissen Rechenstellen werde nach gewissen Größen gefragt. Die Größen selbst sind an irgendwelchen Stellen gespeichert. (Abb. 1)

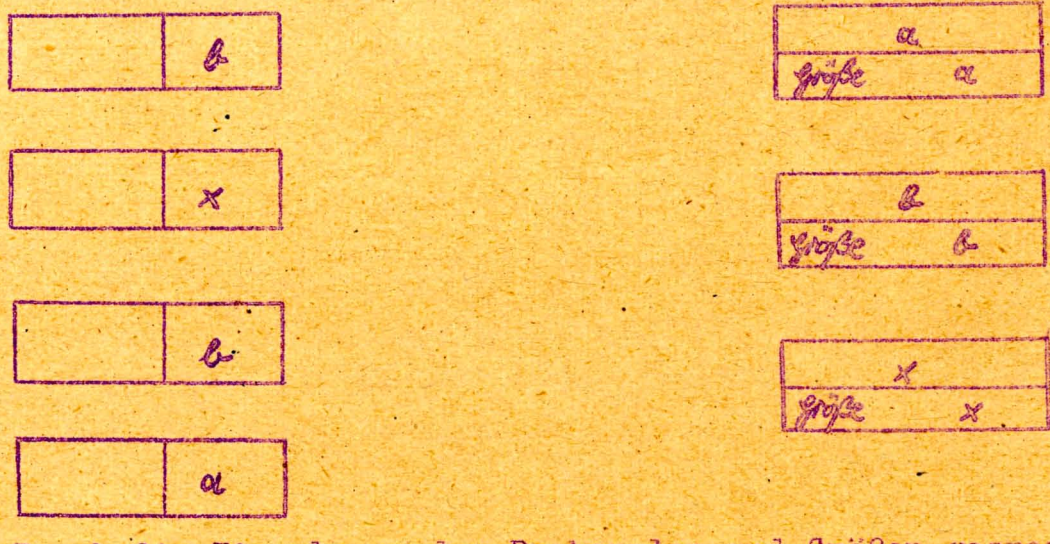
Rechenstellen
Frage nach den Größen

Gespeicherte Größen:

(Abb.1)

Der Automat kann die Rechnung nur durchführen, wenn ihm über das Symbol (z. B. b) die Adresse (β) ermöglicht wird, unter der die Größe (b) gespeichert ist.

Damit werden schon für den einzugebenden Plan bestimmte Maßnahmen notwendig (Abb. 2)



(Abb.2)

Durch die Eingabe werden Rechenplan und Größen gespeichert (Abb. 1) sowie der Vermittlungsblock geschaffen. Dieser hat unterschiedliche Form, je nachdem das Symbol Adressencharakter hat oder reines Symbol ist. (Abb. 3/4)

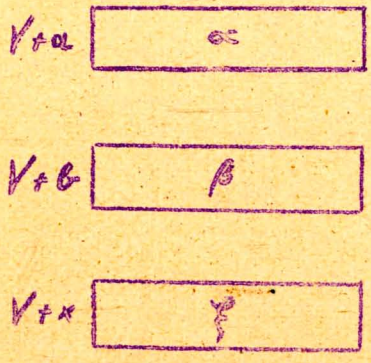


Abb. 3

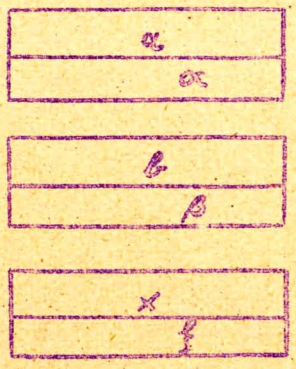


Abb. 4

Diese damit charakterisierten Verfahren entsprechen dem als "floating address" oder Verfahren der absoluten Leitzelle bezeichneten Verfahren.
Jeder gespeicherten Größe ist eine Leitzelle (Leitzellenpaar) zugeordnet, um bei der Frage nach einer Größe zu dieser selbst kommen zu können (Abb. 5).

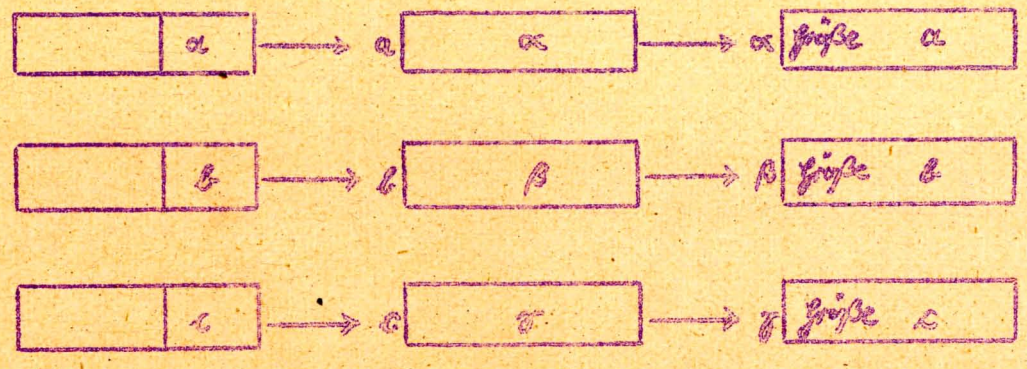


Abb. 5

Faßt man die zu speichernden Größen nach inneren Gesichtspunkten zusammen, so läßt sich die Zahl der Leitzellen einschränken, wenn eine Möglichkeit besteht, die Größen durch Symbole und relative Adressen zu kennzeichnen, d. h. letzthin durch zwei Daten (ev. verschieden umfangreich), wobei das eine Datum vor der Eingabe ein Symbol ist, das auf einen Block verweist, das zweite Datum die relative Adresse in diesem Block. Dieser feste Index gibt mit dem Symbol zusammen eine sehr bequeme Kennzeichnung von Größen, wenn ein Gesamtprogramm der logischen Struktur nach aus kleinen, leicht übersehbaren Teilproblemen aufgebaut wird. (Abb. 6)

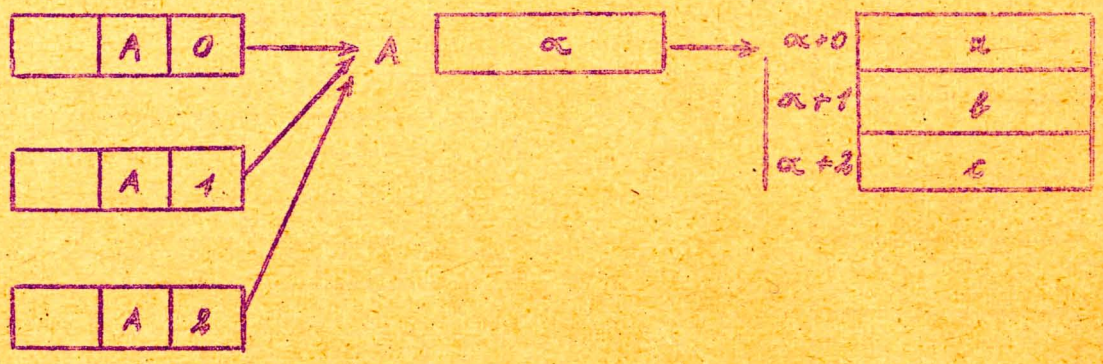


Abb. 6

Bei freien Adressen in dem obigen Sinn liegen die Verhältnisse noch schwieriger, da in diesem Fall der Inhalt der Leitzelle während der Rechnung geändert werden muß und von der Vorgeschichte abhängt. (Abb. 7)

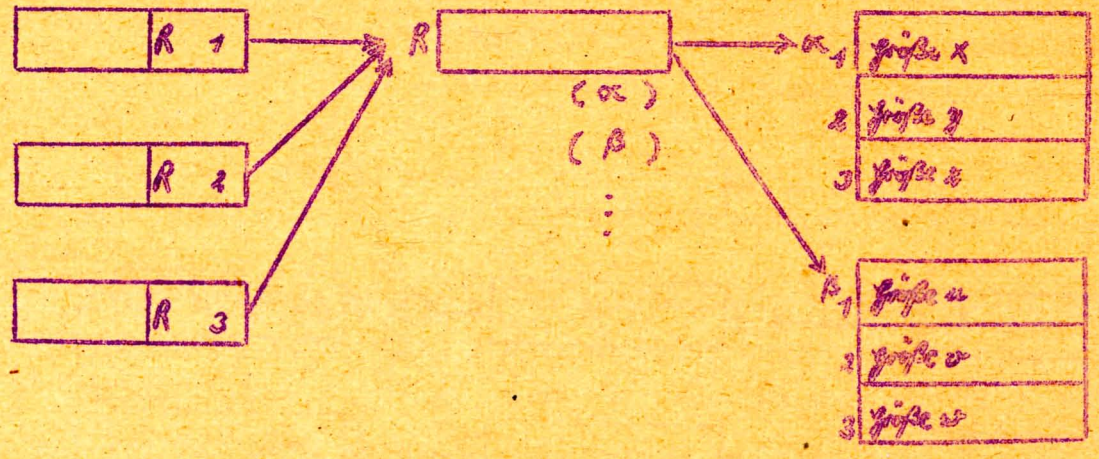


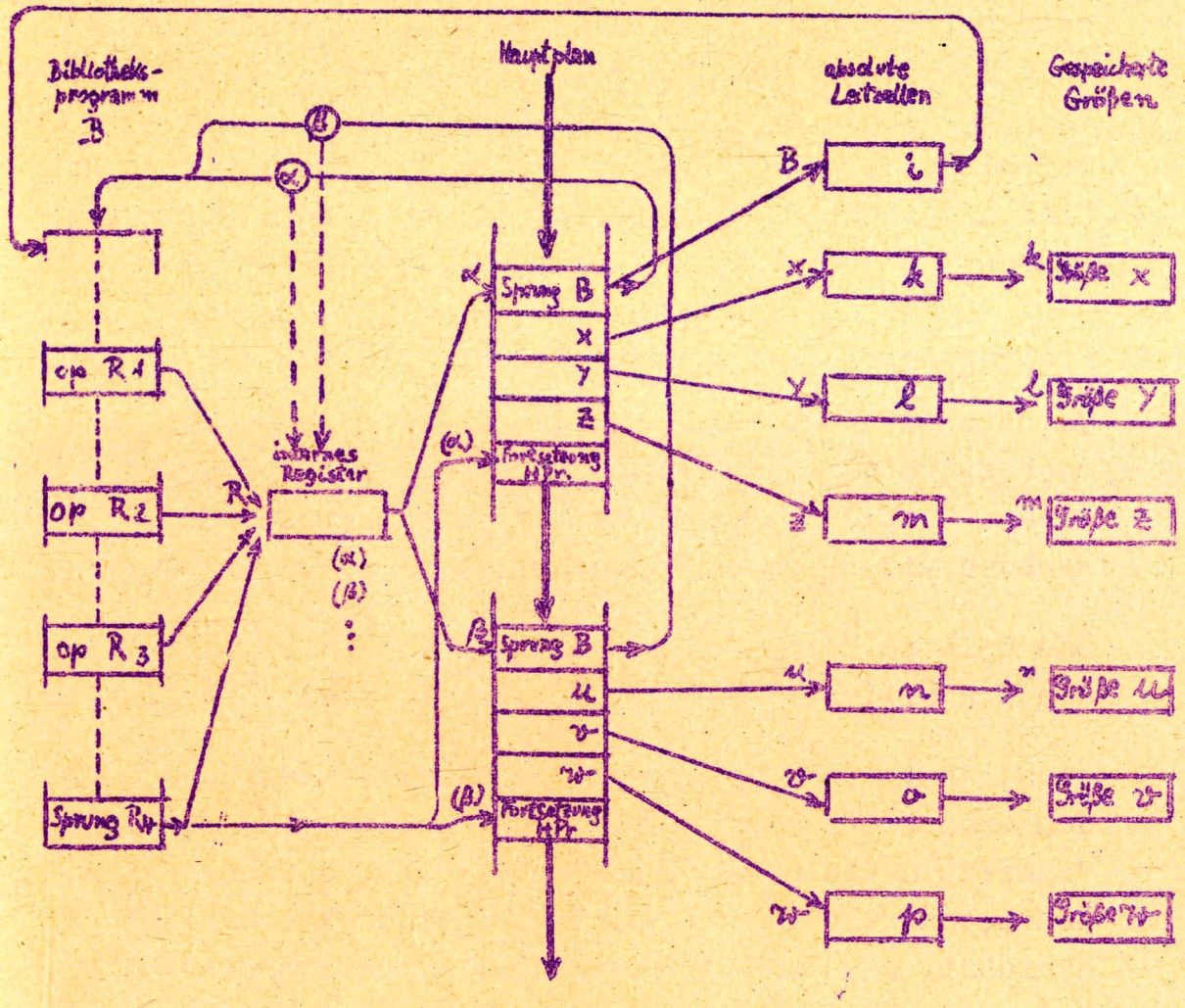
Abb. 7

Hier werden als Parameter eines Bibliotheksplans R_1, R_2, R_3 einmal die Größen x, y, z, das andere Mal u, v, w usw. benötigt. Hier kann die Leitzelle (Adresse R) nicht bei der Eingabe einen festen Inhalt erhalten, dieser muß ihr vielmehr aus dem Ablauf der Rechnung geboten werden und steht mit den jeweiligen Parametern x, y, z bzw. u, v, w usw. in innerem Zusammenhang.

Dieser wichtige Sonderfall der Leitzelle mit Inhalt auf Zeit entspricht dem von Samelson als Verfahren der relativen Leitzellen (Versorgungszellen) bezeichneten Vorgehen, wenn noch hinzugenommen wird, daß die Größen x, y, z bzw. u, v, w usw. selbst wieder zunächst nur auf dem Umweg über Symbole zur Verfügung stehen und diese erst durch wahre Adressen ersetzt werden müssen.

Damit führt das Problem, einem Bibliotheksprogramm die nötigen freien Parameter zur Verfügung zu stellen, zu einer Methode, die sowohl das Verfahren der absoluten Leitzellen als auch das der relativen Leitzellen (Versorgungszellen) benötigt. (Abb. 8)

Abb. 8: Freie Adressen eines Bibliotheksplans



Es liegen zwei unterschiedliche Arten von Adressenpfaden vor, wenn unter einem Adressenpfad der Weg verstanden wird, der von einer Frage nach einer Größe zu dieser führt. Die einen, die über die absoluten Leitzellen führen, können einem kompilierenden Verfahren unterworfen werden, so daß danach im Hauptprogramm an Stelle der Symbole B x y z, B u v w die Adressen i k l m, i n o p stehen. Die anderen Pfade, die über das interne Register R führen, können erst während der Rechnung durchlaufen werden. Beim Sprung aus dem HPr nach B wird jeweils der Befehlszählerstand nach R übertragen, also α bzw. β, so daß beim anschließenden Durchlauf durch das Unterprogramm die freien Adressen über das interne Register zu den jeweiligen Parametern führen.

§ 4 Rechenplanberechnung nach Rutishauser(8

Um durch den Automaten aus einer gegebenen Formel den Rechenplan anfertigen zu können, muß dieser in numerische Gestalt gebracht werden, die Grundlage für ein Rechnen mit Befehlen bietet.

Das weitere Verfahren muß dem Vorgehen entsprechen, das ein menschlicher Rechner in einem solchen Fall zeigt:

Dieser untersucht zunächst den Klammerausdruck auf den Grad und die Art der Verschachtelung, um so einen "innersten Klammerausdruck" zu finden, der herausgeschrieben wird und in seinem Resultat eine Hilfsgröße einführt, die dann den innersten Klammerausdruck ersetzt und so die Verschachtelung reduziert.

(8 Rutishauser,H.: Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen.
Mitteilungen aus dem Institut für angewandte Mathematik TH Zürich - 1952 Nr. 3

Durch Fortsetzung dieses Verfahrens zunächst in gleicher Höhe des Verschachtelungsgrades ergeben sich für den Rechner u. U. eine Reihe von einander unabhängiger Gleichungen (Plangleichungen), bis der dann folgende Abbau der dann noch verbliebenen, absolut niedrigeren Verschachtelung Plangleichungen liefert, in die auch die vordem eingeführten Hilfsgrößen einspielen und selbst wieder neue Hilfsgrößen definieren, bis schließlich eine letzte unverschachtelte Gleichung zum Resultat der Gesamtformel führt.

Dem Automaten sind somit drei Aufgaben gestellt: Er muß die erste Stelle höchster Verschachtelung erkennen, er muß aus ihr eine Plangleichung mit einer neuen Hilfsgröße als Resultat bilden und schließlich mittels dieser Hilfsgröße eine Reduktion der Verschachtelung erreichen.

Elemente einer Formel sind Formelanfang und öffnende Klammer, schließende Klammer und Ergibtzeichen, Operationszeichen, Operand einschließlich Resultat und aus organisatorischen Gründen noch das Formelende.

Diesen Elementen müssen von Seiten der Eingabe losgelöst aus ihrem Zusammenhang Schlüsselzahlen (Code) zugeordnet werden. Die Funktion in Hinblick auf die Verschachtelung dagegen hängt von der Stellung des Elementes innerhalb der Formel ab.
Um dieser Tatsache gerecht zu werden, ordnet Rutishauser jedem Element E_k ein Zahlenpaar a_k b_k zu.
Dabei sollen die a_k als Maß der Verschachtelung dienen, die b_k dagegen die Eingabe der Formel ermöglichen und Grundlage für die herauszuziehenden unverwickelten Plangleichungen bieten, die in ihrer Gesamtheit dann den Rechenplan bilden.

Die Maschinenbefehle werden in der Form vorausgesetzt, daß der eine (vordere) Teil die Operationsangabe, der andere (hintere) Teil die Adresse aufnimmt.

Je nach Natur des Elementes E_k wird diesem ein b_k zugeteilt, das im vorderen oder hinteren Teil eine wesentliche Information trägt, im anderen dagegen leer ist. (Keine Operation bzw. Adresse der Zelle 0).

So haben nur die Elemente, die einem Operanden einschließlich Resultat entsprechen, einen wesentlichen Adressenteil aber einen leeren Operationsteil im zugeordneten b_k. Alle übrigen Elemente haben dagegen Operationscharakter und besitzen in ihrem b_k eine leere Adresse, d. h. sind Operationen mit Zelle 0 als Adresse. Anfang und öffnende Klammer stellen dabei in b_k "Lesen", Ergibt-Zeichen und schließende Klammer "Schreiben" (Speichern) dar, während die eigentlichen Operationen in b_k ihr maschinenmäßig zukommendes Operationszeichen tragen.
Formelende in Form eines Q-Zeichens in b_k dient der Eingabe als Markierung und wird selbst nicht in die Maschine eingespeichert.

Durch die Art der Codierung ist es dem Automaten ermöglicht, aus der Größe von b_k unmittelbar eine Klassifikation der Elemente in die zwei Gruppen vorzusehen, die unterschiedlich hinsichtlich ihrer Verschachtelungsfunktion sind: Öffnende Klammer und Operand bilden die eine Gruppe, Operationszeichen und schließende Klammer die andere. Ein Element der ersten Gruppe wirkt in Richtung auf eine Steigerung des Verschachtelungsgrades, während ein Element der zweiten Gruppe eine absteigende Tendenz bringt. Dementsprechend sind die Definitionsgleichungen der a_k so gewählt, daß diese von $a_o = 0$ ausgehend, ganze positive aufeinanderfolgende Zahlenwerte in auf- und absteigender Tendenz annehmen: Bei einem Element der ersten Gruppe steigt der a-Wert um eine Einheit, bei einem Element der zweiten Gruppe fällt er um eine Einheit.

Eine graphische Darstellung der a_k über der Elementnummer k ergibt einen Streckenzug, dessen Strecken mit gleicher positiver oder negativer Steilheit verlaufen. Dieses "Gebirge" spiegelt die Struktur der Formel wieder: Die "Bergspitzen" entsprechen Elementen, die einem Operanden zugeordnet sind, die "Täler" liegen über Operationszeichen, die "Hänge" über Klammern. Die Operanden einer innersten Klammer machen sich durch die höchsten Spitzen bemerkbar. Zwischen ihnen liegen um eine Einheit gesenkt die Täler. Der innerste Klammerausdruck gleicht so einem Höhenzug, der im regelmäßigen Gang um eine Einheit steigt und fällt. Sein Anfang und sein Ende werden durch Hänge über zwei Einheiten erkenntlich.

Der Automat empfängt zunächst von der Eingabe her die Folge der b_k, errechnet gleichlaufend die Folge der a_k und speichert sie in gemischter Folge

$$a_o \quad b_o \quad a_1 \quad b_1 \quad \ldots\ldots\ldots \quad a_N \quad b_N \quad Q$$

Nach Speicherung des gesamten Programms nimmt der Automat die Berechnung der Rechenpläne für eine Formel nach der anderen vor, wobei die Q-Zeichen die Markierungen geben.

Durchmusterung der gemischten Folge hinsichtlich der a_k bis zum nächsten Q-Zeichen läßt $a_{max.}$ finden. Damit ist der Automat weiter in der Lage, durch erneute Durchmusterung die erste Nummer k festzustellen, für die ein a_k den Wert $a_{max.}$ annimmt. Damit ist das erste Operandenelement gefunden, das zu einem innersten Klammerausdruck gehört. Das Element davor steht an einem Hang und entspricht einer öffnenden Klammer; (Anm.)

- -

Anm.: Dies setzt voraus, daß die im Distributivgesetz enthaltene Bevorzugung der Multiplikation gegenüber Addition und Subtraktion aufgegeben werden muß. Produkte, die ausgerechnet werden müssen, bevor eine vorhergehende Summation ausgeführt werden kann, sind demzufolge wenigstens bei der ursprünglichen Methode von Rutishauser in Klammer zu setzen. (s. auch 9)

das zugehörige b_{k-1} ist der Befehl "Lies" mit leerem Adressenteil, während b_k der Spitze neben einem leeren Operationsteil die Operandenadresse trägt.
Addition beider b liefert den vollen Befehl "Lies Inhalt gemäß Operandenadresse" d.h. "Lies den Operanden".
In entsprechender Weise läßt sich aus dem b des nächsten Tals und dem b der dann folgenden Spitze der zweite Befehl für den Plan der innersten Klammer aufbauen. Dieses Verfahren läßt sich fortsetzen, bis die abfallende Flanke die schließende Klammer zu erkennen gibt und im zugehörigen b den Befehl "Speichere" mit leerem Adressenteil bietet. Aus dem freien Speicherraum bestimmt nun der Automat eine Adresse für die Hilfsgröße und bildet damit den abschließenden Befehl für den Rechenplan dieser innersten Klammer. In die Lücke des Streckenzugs, die durch Entfernung des ausgewerteten Höhenzugs entstanden ist, wird ein Ersatzgipfel der Hilfsgröße zugehörig eingeschoben, dessen Höhe um eine Einheit unter dem seitherigen Maximum liegt, und der Rest des Streckenzugs entsprechend dicht herangezogen. Dieses reduzierte Gebirge wird weiter auf Stellen durchmustert, die das seitherige Maximum erreichen; diese werden in gleicher Weise abgebaut und die zugehörigen Plangleichungen nach außen abgegeben. Wenn keine weitere Stelle in Höhe des seitherigen Maximums anzutreffen ist, wird das Verfahren der Durchmusterung mit einem um eine Einheit niedriger angesetzten Maximum fortgesetzt. Nach völliger Reduktion des Gebirges liegt der Rechenplan für diese Formel vor. Der Automat greift sodann über das Q-Zeichen zur nächsten Formelgruppe.

Leichte Modifikationen gestatten, auch Rechenoperationen, die nur einen Operanden benötigen, in den Kreis einzubeziehen, sowie Operationen, die nicht in der Maschine eingebaut sind und mittels Unterplänen ausgeführt werden.

Diese für Probleme mit linearer Struktur erläuterte Methode läßt sich auch auf zyklische Probleme anwenden, wobei die berechneten Pläne dabei auch ganz oder teilweise gestreckt werden können. Dabei ist zu bemerken, daß ein gestreckter Plan Rechenzeit einspart, da er frei von organisatorischen Befehlen ist, die im anderen Fall den Zyklus steuern, dafür aber einen erhöhten Speicherbedarf fordert.

II. Teil

§ 5 Prinzip eines Automaten mit Programmierung nach mathematischem Formelbild

Während das von Rutishauser entwickelte Verfahren aus der nach mathematischem Formelbild durchgeführten Eingabe von Automaten den Rechenplan herstellen läßt, der dann anschliessend zur Durchführung der Rechnung dient, soll hier durch besondere Ausbildung des Automaten erreicht werden, daß das nach mathematischem Formelbild eingegebene Programm schon der Plan ist, den der Automat direkt verarbeiten kann. Während seither die Automaten so gebaut werden, daß sie gewisse logische Verknüpfungen durchführen können, aus denen sich Abläufe aufbauen lassen, die dem in einer mathematischen Formel gewünschten Zusammenhang entsprechen, wird hier versucht, den Automaten so zu organisieren, daß er die mathematische Formel selbst zu lesen und zu verarbeiten versteht.

Mit Erreichung dieses Ziels würde dem Mathematiker ein Hilfsmittel zur Durchführung seiner Rechenprobleme gegeben sein, das von ihm selbst nur Maßnahmen fordert, die in den tatsächlichen Arbeitsbereich eines Mathematikers gehören: Das Programmieren würde mit der Aufstellung des algorithmischen Plans und der strukturellen Klärung der auftretenden Größen sein Ende finden.

Die folgenden grundsätzlichen Betrachtungen lassen bewußt technische Details außer acht und gehören der logisch-organisatorischen Ebene an. Anderseits liegen der Studie bestimmte Vorstellungen von der Art einer Realisierung zu Grunde, die auf der Erfahrung beruht, daß die Anwendung neuerer Bauelemente eine Leitkettentechnik gestattet und damit auch stärker komplexe logische Verknüpfungen in den inneren Ablauf der Maschine übernehmen kann, die seither einer äußeren Unterprogrammtechnik übertragen waren. Dabei dürfte der Umfang und der Grad der Kompliziertheit nicht wesentlich anwachsen.

In einem Rechenprogramm sind die Operationen im allgemeinen die aktiven Elemente, den Zahlen kommt dabei eine passive Rolle zu, da sie den Operationen als Objekte dienen. Nur gelegentlich greifen auch Zahlen aktiv ein, wenn sie bei Entscheidungen den weiteren Fortgang beeinflussen. (Anm. 1 oder wenn sie beim Rechnen mit Befehlen ihrer Natur nach zwar Operationssymbole sind, die aber, um sie Änderungen unterwerfen zu können, formal als Objekte von Operationen, also als Zahlen behandelt werden und als Resultatzahlen dann wieder ihren Charakter als Operationssymbole gewinnen. (Anm. 2

Jede Operation benötigt eine oder mehrere Objekte, die sie verarbeitet, und eines das sie erarbeitet. Da Zahlen für die Maschine im allgemeinen nur durch die Ortsangabe der sie speichernden Zelle zugänglich sind, wird sich an jede Operation ein Kreis von Ortsangaben d. h. Adressen anschließen.
So entsteht zunächst der Eindruck, daß in einem maschinengerechten Programm die Adressen in der Häufigkeit die Operationssymbole übertreffen.

Anm. 1: Z. B. bedingter Sprung abhängig von Erreichung einer vorgegebenen Genauigkeitsschranke.

Anm. 2: Z. B. Operationsschlüssel wird zur Adressentransformation benutzt, um für das gegebene Programm mit Hilfe eines interpretierenden Programms eine andere Behandlungsweise zu erreichen.

Nun gibt es zwei stark unterschiedliche Anwendungsgebiete der programmgesteuerten Rechenautomaten: Der Einsatz für kommerzielle Zwecke und für Planungsfragen bildet die eine Gruppe, die man als Datenverarbeitung kennzeichnet; die andere Anwendung liegt in der Durchführung wissenschaftlicher und ingenieurmäßiger Rechnungen.
Während bei einem Problem der ersten Art eine sehr große Anzahl von Eingangswerten mit relativ geringem Rechenaufwand verarbeitet wird und wieder zu einer großen Anzahl von Ausgangswerten führt, liegt das Charakteristische eines mathematischen und ingenieurmäßigen Rechenproblems darin, daß relativ kleine Zahlenmengen sehr umfangreichen und komplizierten Rechenvorgängen unterworfen werden.
Da aber der Ablauf eines Programmes nur in einzelnen Schritten vor sich gehen kann, wird der sehr weit gespannte Bogen der Gesamtrechnungen mittels Bereitstellung entsprechender Speicherzellen für die zahlreichen Hilfsgrößen vielfach abgestützt. Diese Bereitstellung von Speicherzellen für Hilfsgrößen läßt sich bei einer universellen Maschine nicht vermeiden, ebensowenig wie das Auseinanderbrechen lang gespannter Plangleichungen in maschinengerechte kleinere Teilaufgaben und deren zielstrebiges Aneinanderreihen, wobei nichts geschieht, was im Gesamtablauf nicht schon in dem Formelbild der geschlossenen Plangleichung gegeben ist. Aber man muß nicht notwendigerweise diese Arbeit dem Programmierer überlassen.

Im maschinenmäßigen Ablauf eines Rechenprogramms gehören zu jeder Operation Objekte, die durch Adressen zugänglich werden. Bei wissenschaftlich-technischen Problemen sind nur wenige dieser Adressen eng an das Problem angelehnt, die Mehrzahl hat nur von Seiten der Maschine her für die Durchführung Interesse. Die problemnahen Adressen entsprechen den Buchstabensymbolen und ev. explizit auftretenden Zahlen des Algorithmus.

Die übrigen Adressen sind ausführungsbedingt und sollten dem Blickfeld des Programmierers ferngehalten werden.

Wenn durch eine solche Maßnahme die meisten Adressen als nur ausführungsbedingt nicht mehr im Programm erscheinen, werden entsprechend Operationssymbole adressenfrei als selbständige Programmelemente auftreten. Innere Aufgabe der Maschine muß es sein, ihnen die ausführungsbedingten Adressen zur Verfügung zu stellen. Es liegt nahe, diese Maßnahme der Verselbständigung der Operationssymbole vollkommen durchzuführen und damit auswirkend in der anderen Richtung die problemnahen Adressen von den seither zugehörigen Operationssymbolen zu trennen und ebenfalls zu selbständigen Programmelementen zu machen.
Dieser formale Schritt gibt die Möglichkeit, den dadurch in einer Befehlszeile frei werdenden Speicherraum voll der örtlichen und ablauftechnischen Charakterisierung einer problemnahen Größe zur Verfügung zu stellen.
Auch hier zeigt die Erfahrung, daß die Angabe einer einzigen Adresse allein zur Erfassung einer Größe nur schleppenden Programmablauf ermöglicht und dem mathematischen Formelbild weit unterlegen ist, das neben dem Buchstabensymbol den festen und variablen Index unter Umständen sogar mehrfach verwendet. Im praktischen Rechenbetrieb tritt hinzu noch das einfache oder auch farbige Anhaken einzelner Zahlen als Hinweis auf besondere einzuleitende Maßnahmen.

Die Reduktion der in einem Programm auftretenden Adressen auf die problemnahen Größen läßt die Operationssymbole vielfach adressenfrei werden. In konsequenter Durchführung wird damit jedes Operationssymbol ein selbständiges Planelement. Dieses wird dadurch relativ kurz. Ein Wort kann daher u. U. mehrere Planelemente aufnehmen.

Durch die Verselbständigung der Operationssymbole wird anderseits auch die Charakterisierung einer problemnahen Größe zu einem selbständigen Planelement. Der im Befehlswort durch Herausnahme der Operationsangabe frei gewordene Raum kann einer differenzierten Kenn zeichnung der Größe durch Angabe fester oder variabler Indizes sowie durch verschiedenartiges Anhaken dienen und bietet so ein bewährtes, dem mathematischen Formelbild nahe kommendes Mittel der Programmgestaltung.

Eine nach diesen Prinzipien entwickelte Maschine wird über einen großen Teil des maschineneigenen Speicherraums selbst verfügen. Während der Durchführung der Rechnung wird sie die jeweiligen freien Adressen zur Hilfsspeicherung oder benötigte Adressen zur Orientierung über vordem gemachte Notizen bereitstellen und auch wieder als hinsichtlich des Inhalts nicht mehr interessierend freigeben. Über einen anderen beträchtlichen Teil des Speicherraums wird die Maschine in bekannter weise bei der Eingabe verfügen, wenn ihre Regie so verläuft, daß sie das meist aus Teilplänen bestehende Programm laufend hintereinander einspeichert.
Der Befehlsschlüssel für eine solche Maschine muß dem mathematischen Formelbild entnommen werden.
Ihm gehören somit im wesentlichen an: Symbole in Form von Buchstaben, die Zeichen für die arithmetischen Grundoperationen + − · : , die öffnende Klammer { wie auch die schließende Klammer } , der als öffnende Klammer anzusehende vordere Absolutstrich ⌈ , der schließende Absolutstrich ⌉ und das Ergibt-Zeichen ⟹. Dazu treten zur Bewältigung von Funktionen oder Verfahren Funktionssymbole, denen eine öffnende Argumenten-Klammer folgen muß und u.U. fest zugedacht werden kann, wie f 12⌈ , sin⌈ , log usw., das Paramenter-Trennzeichen; und der Funktionsabschluß ⌋.
Dabei muß es dem Automaten möglich sein, zu unterscheiden zwischen Operandensymbole, Funktionssymbole und den übrigen Zeichen.

Die letzteren sind in der Codierung nicht umfangreich, so daß sie u. U. zu mehreren ein Wort bilden können. Operanden- und Funktionssymbole dagegen benötigen als adressenähnlicher Natur die volle Wortlänge.

Charakteristisch ist somit einmal diese Trennung in Elemente, die den Automaten zu einer Funktion anregen, ohne ihm dabei eine Adresse anzugeben, und Symbole, die durch den Compiler durch Adressen ersetzt werden, bei denen dem Automaten nichts über die daran durchzuführende Operation vom Plan aus gesagt wird.

Weiter ist entscheidend, daß die Elemente, auch die operativer Natur, den Automaten nicht dazu anregen, zu erledigen, was in ihrer mathematischen Bedeutung liegt, sondern vielmehr ihn nur dazu bringen, das zu erledigen, was er im gegenwärtigen Zustand aus den "Rückständen" fertig machen kann, das "Neue" dagegen mehr oder weniger nur zur späteren Ausführung zur Kenntnis zu nehmen.

An einem einfachen Beispiel läßt sich die andersgeartete Arbeitsweise erkennen.

$$a = b + c \circ (d + e) - (f + g) \Longrightarrow h$$

Das Verfahren von Rutishauser packt das Problem ähnlich dem Vorgehen eines menschlichen Rechners an. Durch Abtasten von links nach rechts wird der höchste Grad der Verschachtelung festgestellt und durch erneutes Durchmustern von links beginnend das Auftreten einer ersten Stelle von diesem Verschachtelungsgrad erfaßt.
Zuvor muß aber das auftretende Produkt in eine Klammer gefaßt werden. (Anm.

$$a - b + (c \circ (d + e)) - (f + g) \Longrightarrow h$$

(Anm. Nach einer Fußnote in 8) S. 12 kann diese Umwandlung vermieden werden. Die zitierte Arbeit [9] stand nicht zur Verfügung:

9) Böhm C.: Calculatrices digitaler du dechriffrage des formules logico mathematiques par la machine même. Annali di mat. pura ed applica, Bologna Serie IV Vol1.37 (1954)

Einführung einer Hilfsgröße ermöglicht Abgabe des Rechenplans, der zur unverschachtelten Plangleichung

$$d + e \Longrightarrow x$$ gehört, sowie Reduktion der Formel in

$$a - b + (c \cdot x) - (f + g) \Longrightarrow h$$

Sodann wird auf nächst niedrigerem Niveau der Verschachtelung

$$c \cdot x \Longrightarrow y$$ behandelt und

die Formel in $a - b + y - (f + g) \Longrightarrow h$ reduziert.

Schließlich wird auf gleicher Höhe $f + g \Longrightarrow z$ behandelt und die Formel in

$$a - b + y - z \Longrightarrow h$$ reduziert.

Das Charakteristische dieses Verfahrens ist ein fortgesetztes Durchmustern mit einem immer wieder von links nach rechts wechselnden Augenspiel.

Das hier vorgeschlagene Verfahren dagegen beginnt links mit der Rechenarbeit ohne vorhergehende Umwandlung - es wird nur vorausgesetzt, daß eine Multiplikation durch das ausgeschriebene Zeichen gekennzeichnet ist - und führt diese durch, soweit die erfaßte Formelfolge schon eindeutige Aussage macht. Rückstände, die so verbleiben, werden aufgeholt, sowie die weiter erkannte Formelfolge eine vordem bestehende Unklarheit beseitigt. Dieses nötig werdende Rückgreifen steht im Gegensatz zu dem obigen Durchmustern.

Dabei wird unter Durchmustern ein Verfahren verstanden, das an eine Reihe von Objekten mit ein und derselben Frage herantritt. Eine Fragestellung an ein Objekt, die bei besonderem Ausfall der Antwort zu einer von der ersten verschiedenen Fragestellung an ein anderes Objekt führt, ist keine Durchmusterung.
Der Automat faßt zunächst die Zahl a auf. Mit ihrer Verarbeitung kann er aber noch nicht beginnen.

Auch die dann erfolgende Erfassung des Zeichens - läßt nicht die Durchführung einer Subtraktion zu. Sie gestattet aber die Verarbeitung des zuvor aufgenommenen a, da es sich geklärt hat, daß a als Summand einer algebraischen Summe auftritt, und damit a in einem zum Anfang gelöschten Register akkumuliert werden kann. a selbst hat damit seine individuelle Rolle ausgespielt. Der Automat kann weiterschreiten, nachdem er sich noch als Rückstand "-" notiert hat. Es folgt die Erfassung von b, ohne daß damit eine Verarbeitung möglich wäre, da ja beispielsweise damit gerechnet werden muß, daß die Fortsetzung b als Faktor erkennen läßt. Erst die dann erfolgende Einholung von "+" gibt Klarheit hinsichtlich der beiden Rückstände, die in der akkumulierenden Subtraktion von b aufgeholt werden. Das "+" selbst aber verbleibt als Rückstand, dem sich das dann gelesene c zugesellt. Aber auch das folgende Multiplikationszeichen kann nur Klarheit bringen, wenn es von der "Zukunft" wenigstens soviel weiß, ob es mit einer folgenden Zahl rechnen kann oder nicht. Im vorliegenden Fall, da eine öffnende Klammer folgt, müssen sowohl c wie auch das Multiplikationszeichen bei Seite gelegt werden, so daß diese auf die später zu erwartende schließende Klammer hin wieder aufgenommen werden können. Die öffnende Klammer ist das Zeichen, daß das Weitere auf einem neuen Niveau verläuft in einer den vorhergehenden Schritten analogen Weise. Die schließende Klammer muß den akkumulierten Wert der auf diesem Niveau erfolgten Rechnung - dies ist das Resultat des Klammerausdrucks - in eine solche Lage bringen, daß er nach endgültiger Aufgabe dieses Niveaus und Rückkehr zum früheren Niveau dort bereit steht, um nach Erinnerung an das rückständige Multiplikationszeichen mit dem ebenfalls aus den "Notizen" auftauchenden c zum Produkt verschmolzen zu werden, das nach Erfassung des folgenden "-" akkumuliert wird unter Beachtung des rückständigen "+".

Damit ist von der obigen Formel der Teil

$$a - b + c \circ (d + e)$$ berechnet.

In entsprechender Weise erfolgt die Bildung des nächsten Klammerausdrucks und seine Akkumulation.
Das Ergibt-Zeichen schließlich gibt Hinweis, daß das folgende Symbol das Ziel einer Abspeicherung ist.

Aus der vorhergehenden Darlegung ist zu erkennen, daß der Kern für die strukturelle Durchbildung einer Maschine nach diesem Prinzip in der Organisation des benötigten Notizblocks für die "Rückstände" liegt. Der Automat teilt bei der Eingabe dem Programm und den darin auftretenden Größen einen ihrem Umfang zukommenden Speicherraum zu. Die dafür vorgesehene Speicherkapazität werde als Eingabespeicher gekennzeichnet. Längere Programme und ein umfangreicheres Zahlenmaterial stellen an diesen Teil erhöhte Anforderungen.

Zusätzlich greift der Automat freien Speicherraum auf bei Durchführung der Rechenpläne selbst, eben für Notizen. Die Höhe dieser Anforderung hängt nicht von der Programmlänge und nicht von dem Umfang des Zahlenmaterials ab, sondern nur von dem Grad der höchsten Verschachtelung, die einmal erreicht wird. Damit bleibt dieses automatische Gedächtnis, wie es hier genannt werden soll, relativ klein und kann in Form eines Schnellspeichers (Matrixspeichers) ausgebaut werden. Während die Zellen des Eingabespeichers beim Ablauf des Plans zum Teil sprunghaft und außer der Reihe belangt werden, treten die Ansprüche an Zellen des automatischen Gedächtnisses nur in der Reihe benachbarter Nummern. Dabei übernimmt, beginnend mit der 1. Zelle, jeweils eine Zelle eine Funktion, die mit dem Ausdruck "Zelle vom Dienst" (ZvD) charakterisiert werden kann. Sie kann dieses Amt nur an eine nachbarliche Zelle abgeben, wobei die Tatsache, ob ein solcher Wechsel eintritt und in welcher Richtung er dann stattfindet, sich aus dem Planablauf ergibt.

Dabei sind jeweils alle Zellen mit Nummern über der der ZvD noch leer, oder ihr Inhalt ist schon wieder uninteressant geworden. Alle Zellen, deren Nummer unter der der ZvD liegen, haben dagegen wesentliche Informationen, die zur möglichen Zeit zur Auswirkung gelangen. Anfang und Ende einer Plangleichung liegen auf einem Niveau, haben dieselbe Zelle als ZvD.
Dieses Auf- und Abgleiten in der Lage der ZvD geht in relativ langsamer Folge vor. Bei vielen Planelementen tritt kein Wechsel in der ZvD ein. Elemente, die in Richtung auf eine stärkere Verschachtelung führen - dabei greift der Begriff Verschachtelung über die von klammerbedingten Rückstände hinaus auch auf die hinterlassenen Operationszeichen und nimmt den Charakter eines Zustandes "erhöhter Spannung"an, verschieben die ZvD in aufsteigender Richtung. Elemente, die angelaufene Unklarheiten aufhellen, wirken abbauend, verschieben die ZvD in absteigender Richtung. Ein Aufsteigen in der Lage des ZvD ist mit Niederschrift von z. Z. hinsichtlich der Ausführungsbestimmungen unklar bleibenden Plananordnungen verbunden. Ein Absteigen in der Lage des ZvD ist mit Lesen von Rückständen und entsprechenden Ausführungsmaßnahmen verbunden.

Da es sich erweist, daß die Lage der ZvD von einem Planelement zum nächsten sich um höchstens 2 Zellen nach oben oder 3 Zellen nach unten verschiebt, dabei die äußersten Zellen dieses Bereiches nur als Positionen des ZvD für das nächste Planelement beansprucht werden, nicht aber zur Übernahme eines Rückstandes bzw. zur Befragung, sind es maximal 4 Zellen, die ev. in direkte Aktion bei einem Planelement kommen. Es erscheint daher zweckmäßig, vier Register vorzusehen, die zum Rechenwerk gehören und vo m vorigen Planelement aus jeweils aus dem automatischen Gedächtnis gefüllt bzw. zu ihm entleert werden.

So wie eine Befehlszeile einer traditionellen Maschine in sich den Befehl für das Beschaffen der danach zu bearbeitenden Befehlszeile trägt, kommt ihr hier in der Form des Planelements noch eine Bereitstellungsaufgabe hinsichtlich der ZvD mit ihren Begleitzellen zu. Die Mittel einer modernen

Technik gestatten durchaus eine derartige parallele Durchführung.

§ 6 Die mathematischen Symbole als Planelemente und die durch sie ausgelösten Maschinenoperationen

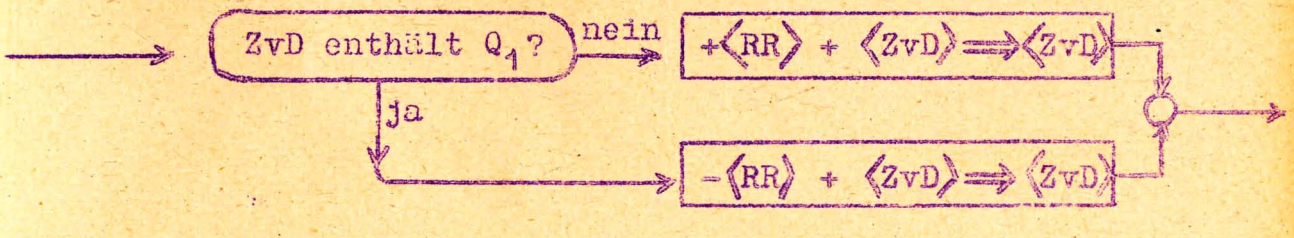
Im folgenden werden die einzelnen Symbole aus dem mathematischen Formelbild nach den entwickelten Prinzipien betrachtet und festgestellt, welche Operationen sie auslösen. Es wird begonnen mit Elementen, die den Grad der Spannung nicht ändern.

Anfang einer Plangleichung: Löscht die ZvD.
Löscht das Resultatsregister RR.

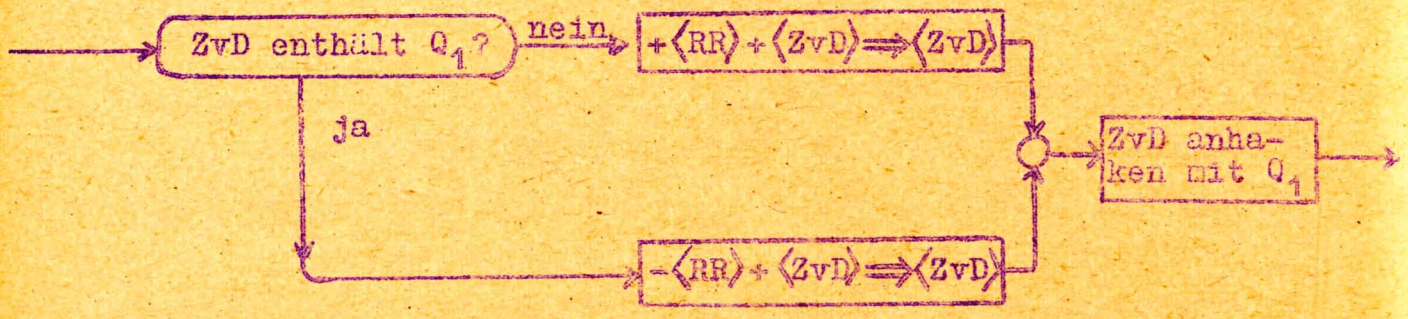
Symbol z. B. a

Vom Kompiler durch wahre Adresse ersetzt.
Auch mit Indizes. : Bringt den Zahlenwert der Größe nach RR.

+ Zeichen : Befragt die ZvD, ob eine Addition oder Subtraktion nachhängt, führt diese mit dem Inhalt des ZvD als 2. Operanden und dem Inhalt des RR als 1. Operanden aus und bringt das Resultat dann nach der ZvD. Sodann wird dem späteren Ablauf das Nachhängen dieses + Zeichens zu erkennen ermöglicht: ZvD verläßt diese Operation etwa ohne das Zeichen Q_1.



- Zeichen : Wie bei + Zeichen, nur verläßt ZvD diese Operation mit dem Zeichen Q_1.



$\Longrightarrow$ Ergibt-Zeichen: Befragt die ZvD, ob eine Addition oder Subtraktion nachhängt, führt diese mit dem Inhalt des ZvD als 2. Operanden und dem Inhalt des RR als 1. Operanden aus.

Im Gegensatz zum vorhergehenden (+ bzw. - Zeichen) bleibt das Resultat im RR.

Greift umdeutend in den Ablauf des folgenden Planelementes ein:
Dieses muß aus der mathematischen Struktur heraus ein Symbol der Form a sein, wird jetzt aber als Speicherbefehl interpretiert oder in einen solchen abgeändert.

$$\langle RR\rangle \longrightarrow \rangle \text{ Symbol } \langle$$

Der Unterschied in den beiden angegebenen Möglichkeiten liegt in Folgendem:

Bei einer Durchführung in abgeänderter Form behält das folgende Befehlselement grundsätzlich seinen Charakter als Träger der dann auszuführenden Operation, die aber alternativ ausgebildet wird. Während normalerweise die Operation in einem Einholen der Größe nach RR besteht, führt ein vorsteuernder Eingriff des Ergibt-Zeichens zur Ausspeicherung des Inhaltes von RR unter dem Symbol. Dabei wird bei der Ausführung die durch den Eingriff abgeänderte Lage beseitigt und die normale Bereitschaft wieder hergestellt.

Bei einer interpretierenden Durchführung dagegen verliert das im Plan folgende Element, nämlich das Symbol a, seinen Charakter als Befehlselement. Es wird in der Reihe der Befehlselemente übersprungen. In der Befehlskette tritt seine Adresse dafür als Adresse von Adresse im Zusammenhang mit dem Abspeicherungsvorgang des Inhaltes des RR auf.

Bei den bis dahin betrachteten Elementen tritt kein Wechsel hinsichtlich der ZvD ein.
Es folgen die Elemente, die den Grad der Spannung bedingt erhöhen.

° Zeichen : Liest das folgende Befehlselement und befragt es, ob es ein Symbol a ist. (Nach früheren Forderungen sind die Elemente für die Maschine gekennzeichnet nach drei Gruppen:
Symbole a, Funktionssymbole, restliche Elemente). Wenn ein Symbol vorliegt, wird die entsprechende Zahl eingeholt und mit dem Inhalt des RR multipliziert. Das Produkt steht im RR.
Es wird ohne Wechsel hinsichtlich der ZvD zum nächsten Planelement übergegangen.

Liegt kein Symbol vor, so wird der Inhalt des RR in der der ZvD folgenden Zelle höherer Nummer gespeichert. Diese übernimmt damit den Charakter der ZvD und wird zusätzlich mit Q_2 gekennzeichnet.

: Zeichen : Wie bei ° Zeichen mit der sinngemäßen Änderung "... wird die entsprechende Zahl eingeholt, und der Inhalt des RR durch sie dividiert. Der Quotient steht im RR".

Die Kennzeichnung nach ev. ZvD Wechsel erfolgt durch gleichzeitigen Zusatz von Q_2 und Q_1.

In diesem Zusammenhang muß darauf hingewiesen werden, daß es noch zu überlegen ist, ob man nicht die Kraft zu einer Inkonsequenz aufbringt, indem man unterschiedliche Zeichen für Multiplikation bzw. Division in den Befehlscode aufnimmt, je nachdem ein Größensymbol folgt oder nicht, da damit eine Vereinfachung in der Struktur der Maschine erreicht wird und man dem naheliegenden Vorwurf einer Automation um jeden Preis entgeht.
Es folgen die Elemente, die eine Erhöhung des Spannungsgrades um eine Stufe mit sich bringen.

{ öffnende Klammer : Setzt die der ZvD folgende Zelle höherer Nummer als ZvD ein, löscht diese sowie das RR.

⌈ Absolutstrich
als öffnend zu kennzeichnen: Setzt die der ZvD folgende Zelle höherer Nummer als ZvD ein, löscht diese sowie das RR.

Auftreten von Funktionen, wie etwa sin, eine Wurzel oder eine Potenz, und die Forderung nach bestimmten Verfahren (im Sinn eines Algorithmus), wie beispielsweise Bildung äquidistanter Werte oder Auflösung einer Differentialgleichung nach der Methode von Runge-Kutta, bringt ebenfalls eine Erhöhung des Spannungsgrades um eine Stufe mit sich. In allen diesen Fällen setzt ein Funktionssymbol, das als solches von der Maschine zu erkennen ist, die Maschine davon in Kenntnis, daß ein spezieller Unterplan eingreift, der als Bibliotheksplan der gewünschten Funktion oder des angestrebten Verfahrens in relativer Form wie ein Zahlenblock eingegeben wurde, der dabei absolute Form angenommen hat und dessen Rufadresse (Adresse der ersten Zeile eines Blocks) durch die Arbeit des Kompilers an die Stelle des Symbols gebracht worden ist, ohne dabei die allgemeine Funktionskennzeichnung zu verwischen.

$\int$ Funktionssymbol
(steht verallgemeinernd für das spezielle Funktionssymbol) : Setzt die der ZvD folgende Zelle höherer Nummer als ZvD ein und speichert in ihr die Rufadresse mit besonderer Kennzeichnung Q_3.

Es ist zu beachten, daß jedem Funktionssymbol eine öffnende Klammer folgen muß, die einer später folgenden schließenden Klammer die Möglichkeit gibt, das Regieende des Funktionssymbols anzuzeigen. Man könnte auch daran denken, die Zeichenkombination $\int$ [als ein Element aufzufassen und entsprechend dem Automaten zur Ausführung übertragen. Dann läge hier die Erhöhung des Spannungsgrades um zwei Stufen vor.
Es folgen die Elemente, die in Richtung einer Lösung der Spannung wirken.

} schließende Klammer: Prüft die ZvD hinsichtlich Q_1, ob Addition oder Subtraktion als Abschluß nachhängt und führt diese im gleichen Sinn aus, wie es auf das Element $\Longrightarrow$ geschieht:
Der akkumulierte Wert bleibt in RR.
(Dieser akkumulierende Schritt ist auch dann nötig, wenn die schließende Klammer das Ende der Regie eines Funktionselementes ansagt; in diesem Fall wird durch den akkumulierenden Schritt der letzte Parameter fertiggestellt.)

Greift sodann auf die vorhergehende ZvD zurück und prüft, ob ein Q_3 die Klammer als Funktionsende deklariert.

Liegt kein Q_3 vor, so wird nach Q_2 ev. in Kombination mit Q_1 gefragt

um eine nachhängende Multiplikation oder Division zu erkennen und gemäß der Plangleichung

$$\langle ZvD \rangle \;.\; \langle RR \rangle \Longrightarrow \langle RR \rangle \quad \text{bzw.}$$
$$\langle ZvD \rangle \;:\; \langle RR \rangle \Longrightarrow \langle RR \rangle$$

auszuführen. (Man beachte, daß hier eine Division vorliegt, die der üblichen Quotientenbildung reziprok ist und im Aufbau der Maschine vorgesehen sein muß.) Ein nochmaliges Zurückgreifen hinsichtlich der ZvD beendet die Operation.

Liegt aber weder Q_3 noch Q_2 vor, so ist mit dieser Feststellung schon die Operation beendet.

Liegt dagegen Q_3 vor, so gibt die ZvD die Adresse, unter der der letzte Parameter - er steht im RR - abgespeichert wird. Erhöhung dieser Adresse um 1 gibt die Adresse für die Fortsetzung des Programms in Form des Sprungs in das Unterprogramm. Der ZvD nimmt den um 1 erhöhten Zählerstand auf und trägt damit die später benötigte Rückkehradresse für den Sprung aus dem Unterprogramm in das Hauptprogramm. Nach Übergang zu nächsten höheren Zelle als ZvD und Löschung des RR ist die Operation beendet. Es folgt das Unterprogramm.

Die schließende Klammer wirkt in Richtung einer Lösung der Spannung u. U. nur vorübergehend. Hat sie nicht den Charakter eines Funktionsendes, so erniedrigt sich das Niveau um eine Stufe, ist dann noch eine Multiplikation oder Division nachhängend, so tritt nochmals eine Erniedrigung um eine Stufe ein. Gibt sie dagegen den Abschluß eines Funktionssymbols an, so ist die erste Erniedrigung nur vorübergehender Natur.

| Absolutstrich als schließend zu kennzeichnen: Prüft die ZvD hinsichtlich Q_1, ob Addition oder Subtraktion nachhängt und führt diese aus; der absolute Wert dieser Akkumulation bleibt in RR.
Greift sodann auf die vorhergehende ZvD zurück und prüft, ob ein Q_2 ev. in Kombination mit Q_1 auf eine nachhängende Multiplikation bzw. Division aufmerksam macht. Im bejahenden Fall wird die Multiplikation bzw. Division durchgeführt.

$\langle ZvD \rangle \cdot \langle RR \rangle \Longrightarrow \langle RR \rangle$ bzw.
$\langle ZvD \rangle : \langle RR \rangle \Longrightarrow \langle RR \rangle$

Ein nochmaliges Zurückgreifen hinsichtlich der ZvD beendet die Operation. Hängt dagegen keine Multiplikation bzw. Division nach, so unterbleibt dieses nochmalige Zurückgreifen.

Das folgende Parameter-Trennzeichen zeigt ebenfalls eine vorübergehende Senkung des Niveau um eine Stufe.
Es wird erforderlich, um bei Funktionen (Unterprogrammen) mit mehreren Parametern deren Übertragung in das Unterprogramm nacheinander vor sich gehen zu lassen.

; Parameter-Trennzeichen: Prüft am ZvD über Q_1, ob Addition oder Subtraktion nachhängt, erledigt diese wie beim $\Longrightarrow$ Zeichen und hält damit den Wert des vor dem Trennzeichen stehenden Parameters - dieser kann ein zusammengesetz-

ter Ausdruck oder selbst eine Funktion sein- im RR.
Rückgriff auf die vorhergehende ZvD gibt die Adresse der Unterprogrammzeile, in die der Inhalt des RR abgespeichert wird. Der Inhalt der ZvD wird danach um 1 erhöht, wobei das stets vorhandene Q_3-Zeichen beibehalten wird.
Danach Übergang zur nächst höheren Zelle als ZvD, wobei diese und RR gelöscht werden.

Unterprogramm-Ende: mit Rückkehr zum H.Pr.

Greift hinsichtlich der ZvD um eine Stufe zurück und erhält dort die Rückkehradresse. Nochmaliges Zurückgreifen um eine Stufe gibt die Möglichkeit, eine ev. nachhängende Multiplikation oder Division zu erfassen und durchzuführen, woran sich ein weiteres Rückgreifen hinsichtlich der ZvD anschließt.
Es folgt die Fortsetzung im Hauptprogramm.

§ 7 Das automatische Gedächtnis

Die Angaben des § 6 stellen eine Möglichkeit dar, das vordem entwickelte Prinzip zu verwirklichen. Es sind eine Reihe von Varianten möglich, die im wesentlichen davon beeinflußt sind, wie weit man die Automatisierung treiben, welche Maßnahmen man dagegen dem Programmierer zuweisen will. Auf die eventuelle Verwendung zweier Multiplikations- (Divisions-) Zeichen wurde schon hingewiesen.

Eine weitere Möglichkeit bietet sich in dem Einsatz einer besonderen Funktionsklammer, da sich dann die Frage nach Q_3 erübrigt.

In welcher Weise diese Einzelheiten nun auch behandelt werden mögen, wesentlich an der Methode ist, daß dem Automaten ein Gedächtnis eigen ist, über das er selbst verfügt. Die Zellen dieses Gedächtnisses werden in drei unterschiedlichen Richtungen verwendet: Einmal übernehmen sie die Funktion von Akkumulationsregistern, so daß ungestört Akkumulationen auf verschiedenem Niveau nebeneinander durchgeführt werden können. In dieser Richtung sind sie dem Rechenwerk zugehörig. Anderseits ergänzen sie den Operationsteil, das Befehlsregister, da sie durch die Anmerkungen (Q_1, Q_2, Q_3), die sie tragen können, aus den verschiedenen Möglichkeiten, die in einem isolierten Befehlselement an sich liegen, die eine auswählen, die dem Befehlselement im Zusammenhang mit den umgebenden, insbesondere vorausgegangenen Elementen zukommt. Schließlich liefern sie Beitrag zu dem Datenteil des Befehlsregisters, wenn sie eine Übergangs- oder Rückkehradresse zwischen Haupt- und Unterprogramm vermitteln.

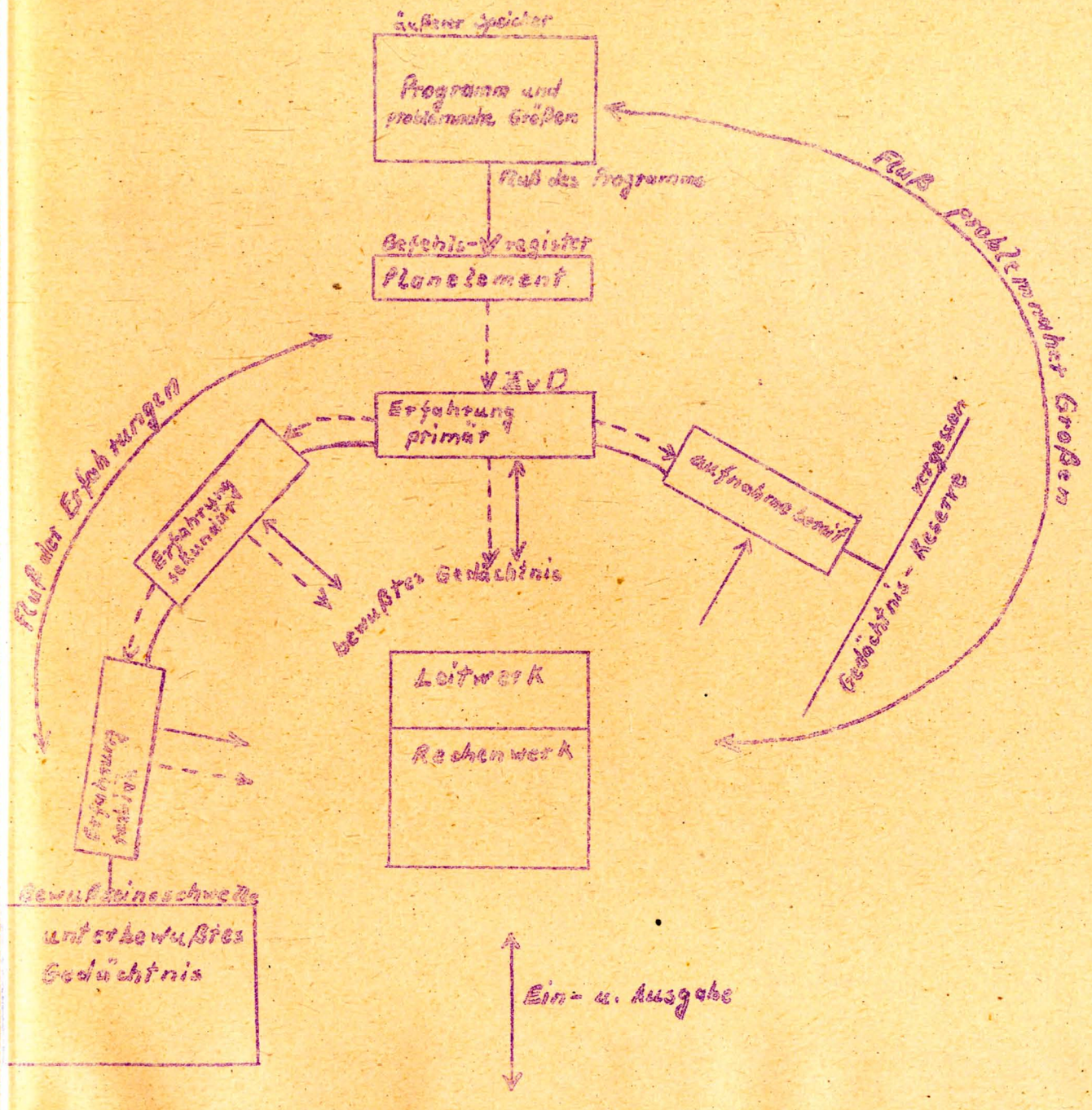
Während des Anstehens eines Befehls im Befehlsregister verbleibt der größte Teil dieser Gedächtniszellen im "Unterbewußtsein" des Automaten. Nur vier Zellen treten mehr oder weniger "bewußt" auf. Der Schwerpunkt, die Stelle höchsten Bewußtseins, liegt dabei in der ZvD. Die Zelle nächst höherer Nummer ist bereit, Informationen aufzunehmen, wenn das Befehlselement in Richtung auf eine Erhöhung des Spannungsgrades zielt, die Zelle niederer Nummer dagegen bietet ihre Informationen bei Lösung der Spannung, nach ihr tritt in der gleichen Art eventuell noch die nächst niedere Zelle in Aktion. Im Ablauf der Operation klärt sich mit dieser eventuellen nach links oder rechts verlaufenden Aktivierung, ob, in welcher Richtung und wie weit sich der Schwerpunkt verschiebt, so daß gegebenenfalls nachbarliche Zellen über die Bewußtseinsschwelle gehoben werden, andere unter sie hinabsinken, und gerade wieder vier Zellen für die nächste Operation bereit stehen.

Auch diese Methode hat ihre Grenzen. Allerdings liegen sie nicht in Richtung der Bedenken, daß bei Steuerungsfehler im Ablauf des Bereitschaftsspeichers die Zuordnung zwischen Befehlselement und auswählendem Zusatzbefehl nicht erhalten bleibt und der Automat damit in die Lage eines Menschen gerät, an den die Umwelt (mit einer Aufgabe) herantritt - Folge der Befehlselemente im äußerem Speicher -, der aber diese äußeren Reize unrichtig ausdeutet, da diese "Erfahrungen" auslösen, die ihnen nicht zugehören, sei es, daß völlig fremde Komplexe aus dem Unbewußten hervor geholt werden oder nur in dem allzulangen Bogen der Zuordnung eine Störung aufgetreten ist. So wie der Mensch sich bemühen wird, durch geeignete Querverbindungen einen eventuell eingetretenen Zustand der Verwirrung rechtzeitig zu erkennen, kann man auch den Automaten zur bedingten Selbstkontrolle dieses Ablaufs befähigen.
Dem Anfang einer Plangleichung muß nach mehr oder weniger weitgespanntem Bogen das Ergibtzeichen folgen. Beide liegen auf gleichem Niveau hinsichtlich des Gedächtnisses, bedienen sich derselben ZvD. Ein vom Element "Anfang einer Plangleichung" eingebrachtes besonderes Kennzeichen kann vom Element "Ergibt" als Zeichen des noch bewahrten logischen Synchronismus ausgewertet werden.

Was dieser Methode versagt bleibt, ist das Erfassen von Rechenvorteilen, wie es einem menschlichen Rechner eigen ist. Dieser erkennt bei entsprechender Reife beispielsweise das Auftreten gleicher Rechenausdrücke an verschiedenen Stellen. Er wird ihn, einmal berechnet, aufbewahren und auf ihn an der späteren Stelle zurückgreifen.
Der Automat dagegen wird infolge seines engen, eindimensional geführten Gedächtnisses an diesen Möglichkeiten vorbeigehen und in jedem Fall den Wert im neuen Anlauf beschaffen, ohne zu "merken", daß er diese Arbeit vor kurzem schon einmal geleistet hat.

Solche Vorteile zu erkennen, dürfte aber noch durchaus dem Aufgabenbereich des Programmierers zugehören, der durch Einführung einer Hilfsgröße und Voranstellung einer entsprechenden Plangleichung den Automaten in die gewünschte Richtung lenkt. Zu der Hilfsgröße, die damit quasi problemnah geworden ist, wird man meist dann auch eine reale Bedeutung in dem physikalisch-technischen Problem vor dessen mathematischer Erfassung finden können.

Abb.: Schema des inneren Gedächtnisses und äußeren Speichers



Die folgende Zusammenstellung der Verhältnisse am "bewußten Gedächtnis" und am Resultatregister bezieht sich auf die in § 6 entwickelte Methode. Die Niveaulinien geben durch Heben und Senken die jeweilige Verschiebung hinsichtlich der ZvD an. Befragungen sind durch ein Fragezeichen angedeutet, ein senkrechter Strich weist auf Neubeschreibung der Zelle hin. Unter Akk. wird die gemäß der Q_1-Frage durchgeführte Akkumulation der Inhalte der ZvD und des RR verstanden.

Um in jedem Fall ein eindeutiges Bild zu erhalten, werden die einzelnen Planelemente eventuell mehrfach aufgeführt, um sie in jedem Planzusammenhang zu charakterisieren.

Anfang einer Plangleichung ____ | 0 K 0 $\Longrightarrow$ ⟨RR⟩

K ein eventuelles Kontrollzeichen, etwa in Form der sonst nicht auftretenden Kombination $Q_3\ Q_2\ Q_1$, wird vom Ergibtzeichen erwartet, um eine gewisse Sicherung des logischen Gleichlaufs zu ermöglichen.

\+ Q_1? | Akk.

\- Q_1? | Akk. Q_1

Größensymbol etwa a ____ a $\Longrightarrow$ ⟨RR⟩ nach Ergibtzeichen umgedeutet in ⟨RR⟩ $\Longrightarrow$ a

∘ wenn eine Größe a folgt ____ ⟨RR⟩ ∘ a $\Longrightarrow$ ⟨RR⟩

: wenn eine Größe a folgt ____ ⟨RR⟩ : a $\Longrightarrow$ ⟨RR⟩

∘ vor { und vor $\int$ ____ ⟨RR⟩ Q_2

: vor { und vor $\int$ ____ ⟨RR⟩ $Q_2 Q_1$

⟹ K? Q_1? — Akk. ⟹ ⟨RR⟩

Kontrolliert ev. das Vorhandensein des Kontrollzeichens K. Vergl. Bemerkung zu Anfang einer Plangleichung.

{ 0 — 0 ⟹ ⟨RR⟩

f f Q_3

Es muß eine öffnende Klammer folgen.

; Q_1? — f+K Q_3 | f+K+1 Q_3 — 0

Übertragung des Kr1 Parameters

Akk. ⟹ ⟨RR⟩
⟨RR⟩ ⟹ ⟨f+ K⟩
0 ⟹ ⟨RR⟩

} als Ende einer Funktion

Q_1? — f+K Q_3? ja | f+K+1 Rückkehradr. — 0

Übertragung des letzten Parameters

Sprung ins übr.

Akk. ⟹ ⟨RR⟩
⟨RR⟩ ⟹ ⟨f+ K⟩
Sprung → f+K+1
0 ⟹ ⟨RR⟩

} nicht als Funktionsende; keine nachhängende Mult. od. Div.

Q_1? — Q_3? nein Q_2? nein — Akk. ⟹ ⟨RR⟩

} nicht als Funktionsende Mult. od. Div. hängt nach.

Q_1? — Q_3? nein Q_2? ja Q_1? — Multiplikand Dividend

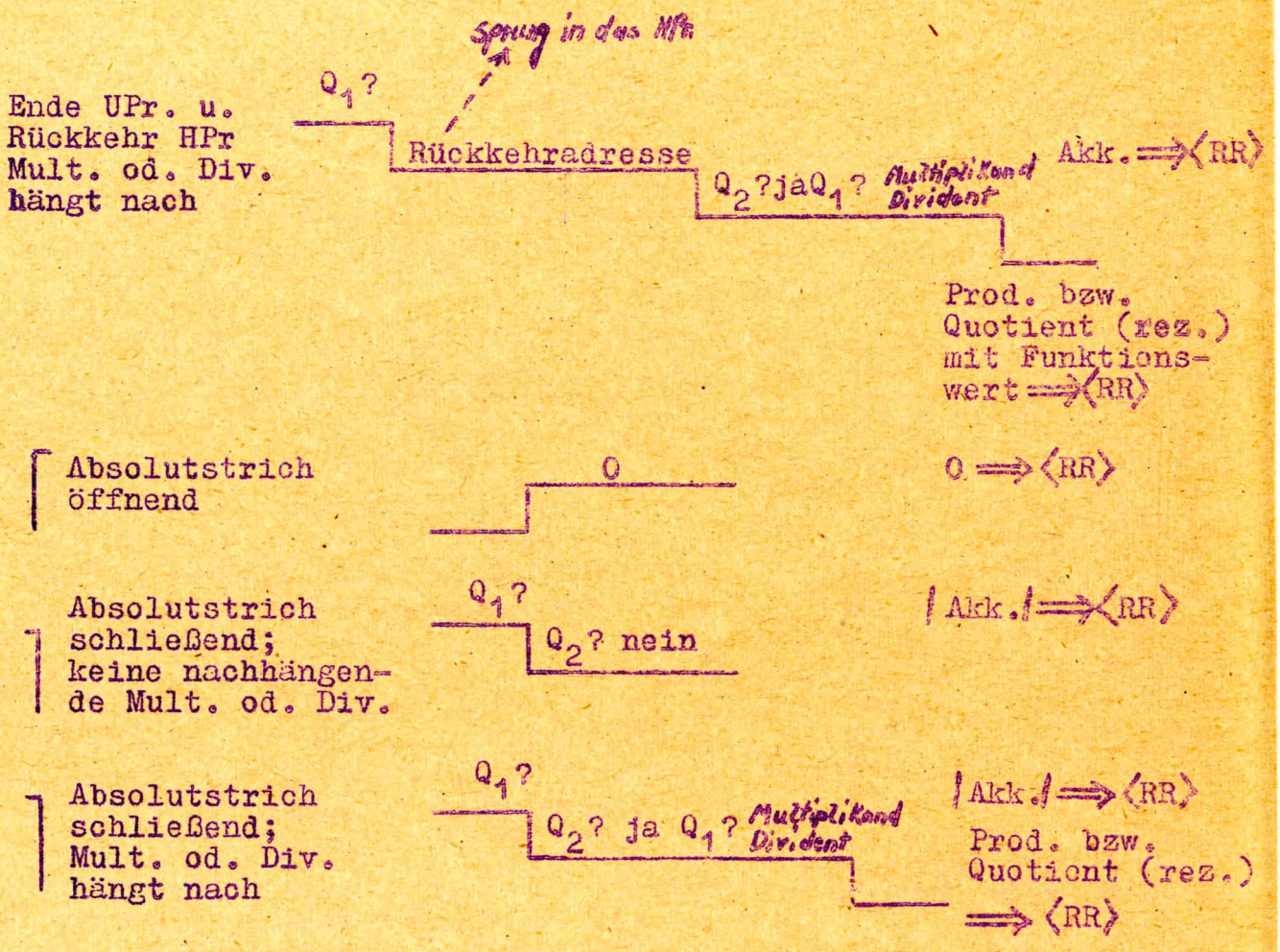
Akk ⟹ ⟨RR⟩
Prod. bzw. Quotient (rez.) ⟹ ⟨RR⟩

Sprung in das HPr.

Ende UPr und Rückkehr HPr. ohne nachhängende Mult. od. Div.

Q_1? — Rückkehradresse — Q_2? nein

Akk. ⟹ ⟨RR⟩ enthält den Funktionswert



§ 8 Beispiele und Vergleich

Von Rutishauser wird eine Abschätzung der für Programmierung aufzuwendenden Zeit im Verhältnis zur Rechenzeit des Automaten auf Grund einer vierjährigen Instituts-Erfahrung am Relaisautomaten Z 4 gegeben, die mit den Erfahrungen am Doppelautomaten Oprema in dessen zweijährigem industriellem Einsatz in drei Schichten in Übereinklang stehen. Bei diesen relativ langsamen Relaisautomaten sind sich beide Zeiten etwa gleich, der Ablauf ist ausgeglichen, ein eigentliches Programmierungsproblem tritt nicht auf. Mit dem Übergang zu schnellen elektronischen Anlagen dagegen wird die Programmierung der entscheidende Engpaß, da sich das Verhältnis der Zeiten zu etwa 200 : 1 verschiebt. Dort, wo im wesentlichen gleichbleibende Routinearbeiten anstehen, wird dieses Mißverhältnis nicht so zu Tage treten, wo dagegen immer wieder neue Probleme an den Automaten getragen werden, wird seine Auslastung unter gegenwärtigen Verhältnissen schwer zu erreichen sein.

Einer vereinfachten Programmierungstechnik, wie sie hier angestrebt wird, wird man an anderen Stellen zahlen müssen. Man wird unter den obigen Verhältnissen dazu bereit sein. Der Aufwand in der Struktur des Automaten wird etwas steigen, da die auszuführenden Operationen komplexer Natur sind. Es werden anderseits andere Feinheiten in den Befehlslisten fallen können, die eine derzeitige Programmierungstechnik vereinfachen sollten. Die Herausnahme der Speicherkapazität für Hilfsgrößen aus dem allgemeinen Speicher in das innere Gedächtnis wird die benötigte Gesamtkapazität kaum berühren: Wenn auf der einen Seite eine spezialisierende Aufteilung infolge der dann nicht mehr bestehenden Ausgleichmöglichkeiten auf eine Steigerung der Speicherkapazität zielt, wird anderseits der systematische Einsatz des Gedächtnisses sparsamer arbeiten.

Das im folgenden herangezogene Beispiel kann in der Frage Speicherkapazität und Rechenzeit nur orientierende Aussagen geben. Es zeigt aber, daß die Verhältnisse keineswegs ungünstig liegen. Ausführlichere Angaben müssen ausgedehnten Programmstudien überlassen werden.
Die Plangleichung

$$a - b + c \circ (d + e) - (f + g) \Rightarrow h$$

führt nach der Methode von Rutishauser zu einem vom Automaten angefertigten Rechenplan, der sich auf folgende unverwickelte Plangleichungen stützt:

$d + e \Rightarrow x$	(3)
$c \circ x \Rightarrow y$	(3)
$f + g \Rightarrow z$	(3)
$a-b+y-z \Rightarrow h$	(5)

Die in Klammer hinter jeder Plangleichung stehenden Zahlen geben dabei die Anzahl der Befehle an, die in einer Einadreßmaschine üblicher Struktur zur Realisierung nötig sind.

Für die erste Plangleichung ist es die Folge der drei Maschinenbefehle

```
Lies        d
Addiere     e
Speichere   x
```

Der errechnete Rechenplan enthält danach 14 Befehle, jeweils aus Operations- und Adressenangabe bestehend. An Speicherraum werden außerdem 8 Zellen für die problemnahen Größen a, b,..., h sowie 3 Zellen für die Hilfsgrößen x, y, z gefordert, zusammen 25 Zellen.

Da bei jedem Befehl nicht nur dieser selbst eingeholt werden muß, sondern auch eine Größe entweder gelesen oder gespeichert wird, müssen insgesamt 28 äußere Transporte durchgeführt werden. Diese drei Zahlen - 14 Befehle, 25 Zellen, 28 Transporte - geben eine gewisse Charakterisierung der Belastung, die das anliegende Problem in dieser Art der Ausführung (automatische Planberechnung) an den Automaten stellt.

Wie würde demgegenüber ein Mathematiker den Plan optimal aufstellen?
In der Umstellung

$$(d + e) \cdot c + a - b = f - g \Rightarrow h$$

erweist sich das Problem als ohne Hilfsgrößen vom Automaten durchführbar. (Man erkennt daran, daß das Beispiel keineswegs pro domo gewählt wurde!)

Beginnend mit

```
Lies          d
Addiere       e
Multipliziere e
    usw.          und mit
Speichere     h          endend,
```

führt der optimale Rechenplan auf 8 Befehle, 16 Zellen und 16 äußere Transporte.

Und nun das hier vorgeschlagene Verfahren einer Programmierung nach mathematischem Formelbild.

Eine besondere Programmierungsarbeit entfällt hier vollkommen. (Anm. Die Zeichen der Plangleichung liefern direkt die Planelemente; es ist lediglich "Anfang Plangleichung" voranzustellen. Die Abzählung ergibt damit 20 Planelemente. Um zu einem Vergleich hinsichtlich Raumbedarf zu kommen, muß man beachten, daß ein Planelement entweder nur Operationssymbol oder nur Größensymbol trägt. Daß man die gewonnene Halbzelle pro Befehl ev. für Erringung anderer Vorteile (siehe indexbehaftete Größen) wieder drangibt, darf hier nicht gewertet werden. Hier wird man zwei Planelemente im Speicherbedarf einem Operations-Adressen-Befehl gleichsetzen müssen.

Für die problemnahen Größen werden auch hier 8 Zellen beansprucht. Benötigte Zellen des inneren Gedächtnisses beeinflussen nicht die Rechenzeit, da ihr Aufruf parallel zu den der Befehle vor sich geht; sie bleiben daher auch bei Aufstellung der Anzahl der äußeren Transporte unberücksichtigt. Ruft man jedes Planelement einzeln auf, so erhält man mit den problemnahen Größen 28 Transporte, wie bei der Methode des automatisch errechneten Rechenplans. Geschieht der Aufruf dagegen paarweise, so reduziert sich die Zahl auf 18.

Wie ein Blick auf die Plangleichung zeigt, tritt maximal eine Steigerung des Spannungsgrades um zwei Stufen ein, hervorgerufen durch das Auftreten des Multiplikationszeichens mit folgender öffnenden Klammer. Somit treten drei Gedächtniszellen in Aktion. Ihnen steht der Bedarf für die Hilfsgrößen im Fall des errechneten Rechenplans mit ebenfalls drei Zellen gegenüber.

- -

(Anm. Das Problem der Adressen wird hier nicht mehr berührt. Es ist in vorhergehenden Abschnitten behandelt und liegt den drei im Vergleich stehenden Verfahren in gleicher Weise an.

Damit ergibt sich für die Methode einer Programmierung nach math. Formelbild die Bilanz

$$20 \text{ Planelemente}, \quad \frac{20}{2} + 8 + 3 = 21 \text{ Zellen},$$

$$\frac{20}{2} + 8 = 18 \text{ Transporte}$$

Zusammenstellung der Automatenbelastung durch die obige Plangleichung nach den drei Methoden:

	Umfang des Plans	Speicher-bedarf	Aufruf des äußeren Speichers
optimaler Rechenplan:	8 Befehle	16 Zellen	16 Transporte
errechneter Rechenplan	14 Befehle	25 Zellen	28 Transporte
Rechenplan nach math. Formelbild	20 Plan-elemente	21 Zellen x)	18 Transporte

x) Zwei Planelementen entspricht 1 Zelle und 1 Transport.

Im folgenden werden Ablauf des Rechenplans nach math. Formelbild und die sich dabei im Gedächtnis abspielenden Vorgänge für das obige Beispiel gegeben.

Rechen-plan	Abspeicherung in RR	Abspeicherung in ZvD		Lage der ZvD im Gedächtnis	Antwort auf er. Befragung nach Kennzeichnung in ZvD Q_1?	Q_2?	Q_3?
Anfang	0	0		*			
a	a						
−		0+a	Q_1	*	nein		
b	b						
+		a−b		*	ja		
c	c						
∘		c	Q_2	*			
(	0	0		*			
d	d						
+		0+d		*	nein		
e	e						
)	d+e c∘(d+e)				nein nein	ja	nein
−		a−b+c∘(d+e)	Q_1	*	nein		
(	0	0		*			
f	f						
+	0+f			*	nein		
g	g						
)	f+g				nein nein	nein	nein
⟹	a−b+c∘(d+e)−(f+g)						
h	Umdeutung: ⟨RR⟩ ⟹ h						

Plan und Ablauf bei Programmierung nach math. Formelbild

$$a-b+c\circ(d+e)-(f+g) \Longrightarrow h$$

Anm.: * Neubeschreibung der ZvD

An einem zweiten Beispiel werde die Verarbeitung von Funktionen in Plangleichungen gezeigt.

$$e^{-x^2} \circ \sin cx \Longrightarrow z$$

Als vorhanden werden dabei Bibliothekspläne für

$$\exp(x) = e^x$$
$$\sin(x) = \sin x$$

sowie für pot (x; n) = x^n (n ganz zahlig positiv) angenommen. Das letzte geschieht hier, um eine Demonstration einer Funktion mit zwei Parametern und die einer Funktion von einer Funktion zu geben.

Die maschinengerechte Schreibweise der Plangleichung lautet

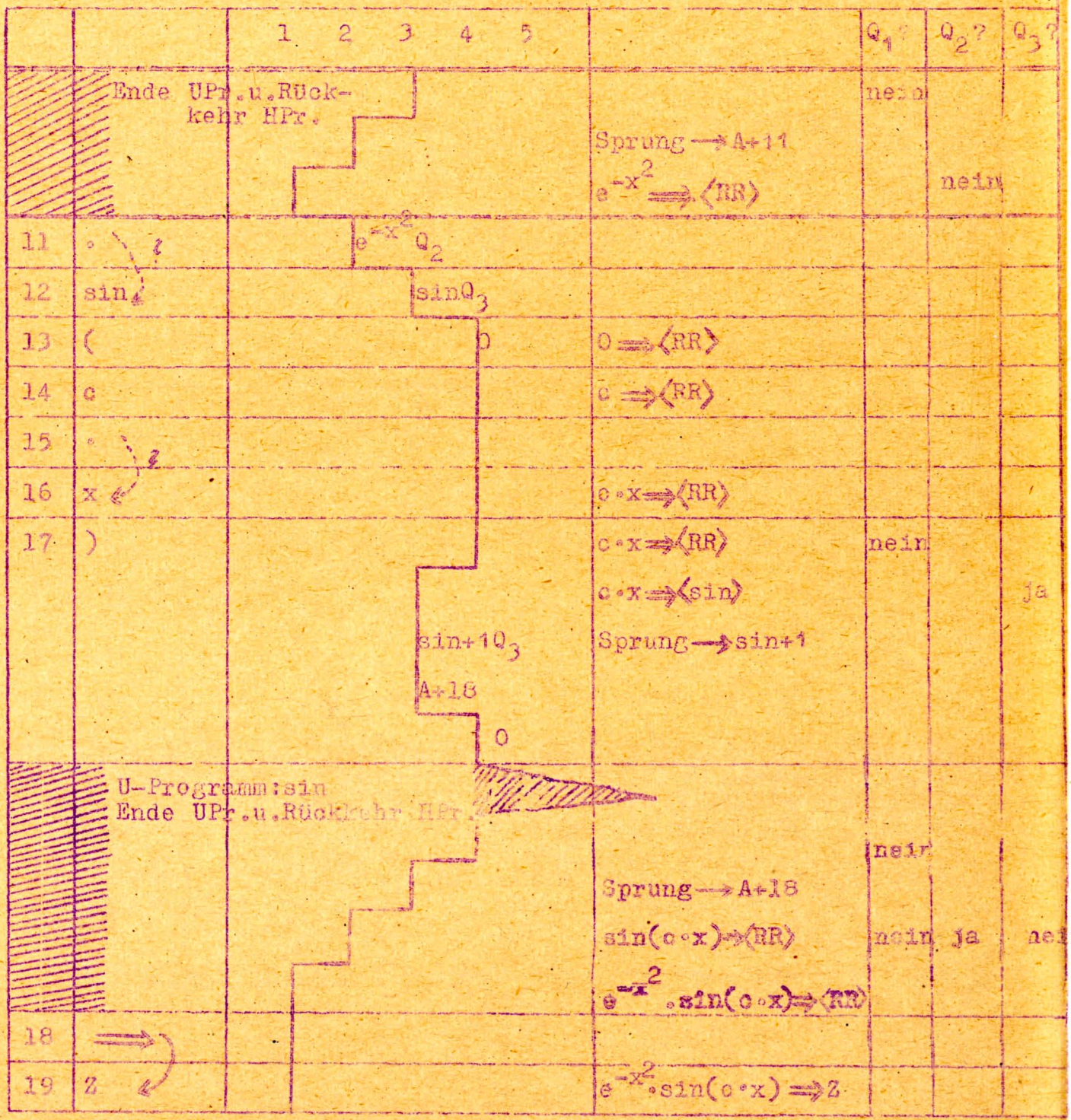
Anfang exp(- pot(x; 2)) ∘ sin (c ∘ x) ⟹ z und enthält

20 Planelemente. Als problemnahe Größen treten x, c, z und 2 auf. Da jedes Funktionssymbol, jede öffnende Klammer und jedes Multiplikations-(Divisions-) zeichen, auf das keine Zahl folgt, das Niveau der ZvD um eine Stufe heben, ergibt sich für den vorderen Teil der Plangleichung, der die Bildung von e^{-x^2} anstrebt, eine Steigerung um 4 Stufen mit entsprechendem Wiederabstieg, während der zweite Teil eine Steigerung von nur 3 Stufen bringt.
Somit werden von H. Pr. maximal 5 Gedächtniszellen beansprucht.

Der folgende Planablauf läßt die interessierenden Einzelheiten erkennen.
Man beachte, daß es in den Rechenplan des UPr. gehört, den Abstieg zum alten Niveau, bei nachhängender Multiplikation (Division) sogar noch eine Stufe unter das alte Niveau zu veranlassen, während der Anstieg vordem bei den Vorbereitungsarbeiten zum Aufruf des UPr. vom H. Pr. erregt wurde.

$$e^{-x^2} \cdot \sin(c \cdot x) \Longrightarrow z$$

		1 2 3 4 5		Q_1?	Q_2?	Q_3?
A+0	Anfang	0	$0 \Longrightarrow \langle RR \rangle$			
1	exp	exp Q_3				
2	(	0	$0 \Longrightarrow \langle RR \rangle$			
3	-	0 Q_1		nein		
4	pot	pot Q_3				
5	(	0	$0 \Longrightarrow \langle RR \rangle$			
6	x		$x \Longrightarrow \langle RR \rangle$			
7	;	pot+1 Q_3 0	$x \Longrightarrow \langle pot \rangle$	nein		
8	2		$2 \Longrightarrow \langle RR \rangle$			
9	)	pot+2 A+10 0	$2 \Longrightarrow \langle RR \rangle$ $2 \Longrightarrow \langle pot+1 \rangle$ Sprung → pot+2	nein		ja
	U-Programm: pot. Ende UPr. u. Rückkehr HPr.		Sprung → A+10 $x^2 \Longrightarrow \langle RR \rangle$	nein	nein	
10	)	exp+1 Q_3 A+11 0	$-x^2 \Longrightarrow \langle RR \rangle$ $-x^2 \Longrightarrow \langle exp' \rangle$ Sprung → exp+1	ja		ja
	U-Programm: exp					



Anm.: A, pot, exp, sin als Symbole werden vom Kompiler durch absolute Adressen ersetzt, nämlich durch die Rufadressen der entsprechenden Programmteile (Hauptprogramm, Unterprogramm für die Potenz-, Exponential- bzw. Sinus-Funktion).

Zum Abschluß dieses Beispiels werde die Bildung eines UPr. gezeigt. Die Funktion exp (x) = e^x werde in "Gerade-aus-Form" durch die auf 11 Glieder beschränkte Teilsumme der Exponentialreihe berechnet.

$$e^x = \left(\left(\ldots\left(\left(\frac{1}{10!}\cdot x + \frac{1}{9!}\right)\cdot x + \frac{1}{8!}\right)\cdot x + \ldots + \frac{1}{2!}\right)\cdot x + 1\right)\cdot x + 1$$

Kennzeichnet man in einem Rechenplan stehende Elemente, die eine Größe selbst und nicht deren Symbol tragen, so, daß der Automat dies erfassen kann, so ist damit eine sich sehr ökonomisch auswirkende Möglichkeit gegeben, Zahlenangaben direkt in den Plan aufzunehmen. (Anm. Während das Auftreten einer Adresse a (entstanden aus dem Symbol a) die Operation "Lies a" auslöst und damit die unter der Adresse a gefundene Zahlengröße in das RR bringt, wird das Auftreten einer Zahl selbst die Operation "Übertrage Inhalt des Befehlsregistersin das RR" auslösen. Man beachte, daß bei den Maschinen klassischer Bauart diese Möglichkeit nicht vorhanden sein kann, da bei ihnen eine Adresse stets mit einem Operationssymbol verbunden das Befehlsregister füllt.

Das Bibliotheksprogramm für e^x muß an Stelle von x eine freie Adresse erhalten. Bei dem im vorhergehenden gewählten Verfahren der Parameterübertragung ist die Anfangszeile des UPr. (relative Adresse 0) zur Aufnahme des ersten Parameters bestimmt. Somit entsteht das Bibliotheksprogramm aus der obigen Formel durch Ersatz des Symbols x durch die Adresse 0, die als relativ gekennzeichnet wird.
Der Bibliotheksplan enthält in dieser Form 61 Planelemente. Die hintereinander öffnenden Klammern ergeben eine Hebung des Niveau der ZvD um maximal 9 Stufen. Anfang und Ende sind im folgenden wiedergegeben.

Anm.: Siehe § 9 - 2 b

rel. Adresse		
0	....	leer, empfängt 1. Parameter
1	(	
2	(	
3	(	
4	(	
:	:	
9	(	
10	Z 1/10!	Zahl
11	.	
12	r 0	rel. Adr.
13	+	
14	Z 1/9!	Zahl
15	)	
16	.	
:	:	
50	)	
51	.	
52	r 0	rel. Adr.
53	+	
54	Z 1	Zahl
55	)	
56	.	
57	r 0	rel. Adr.
58	+	
59	Z 1	Zahl
	Ende UPr.	
60	Rückkehr HPr.	

§ 9 Weiterer Ausbau und Ausblick

1. In die Befehlsliste müssen nun noch Planelemente aufgenommen werden, die den im Strukturplan (Flußdiagramm) eines Rechenproblems u. U. in mehr oder weniger verwickelter Form auftretenden Verzweigungen alternativen Charakters und Vereinigungen gerecht werden.

Nun liegt eine besondere Schwierigkeit darin, daß für derartige strukturelle Fragen in der Schreibweise noch kein Analogon zu dem mathematischen Formelbild erwachsen ist. Daher ist man hier z. Zt. noch stark auf individuelle Prägungen angewiesen.

Zunächst wird ein Planelement nötig, das durch die sprachliche Fassung

"Es folgt Fortsetzung unter"

und etwa durch das Zeichen ⟶ gekennzeichnet werden kann. Es entspricht dem unbedingten Sprungbefehl der üblichen Maschinen, nur daß das Ziel in einem folgenden selbständigen Planelement gegeben wird.

Andere Planelemente, die eine Möglichkeit zur Alternative geben müssen, tragen bedingten Charakter.
Die Tatsache, daß gerade die Behandlung der Planstellen, die eine Alternative einbringen, besondere Programmierungsschwierigkeiten bereitet, dürfte darin begründet sein, daß man den Automaten zur Erledigung solcher Aufgaben zu bringen versucht, ohne ihm ausreichende Kraft dazu gegeben zu haben. Ausgehend von dem vorhandenen Rechenwerk fand man sich bereit, zur Behandlung der Adressen ein eigenes Adressenwerk zu schaffen; die Behandlung der strukturellen Probleme durch den Automaten verblieb in Nebenrolle.
Sprachliche Fassungen wie

"Wenn A, dann folgt Fortsetzung unter...
sonst unter ..."
oder "Für...., folgt ...." u. dgl.

haben gemeinsam, daß eine Operation hinsichtlich der Ausführung an das Vorhandensein eines bestimmten Zustandes gebunden wird.
Der Komplex läßt drei funktionelle Aufgaben erkennen:

a) Dem Automaten müssen über die Natur des in Frage kommenden Zustandes Angaben gemacht werden. Ein "Zustand" bedeutet dabei, daß gewissen Trägergrößen eine gewisse Eigenschaft zukommt, und kann somit als Aussage formuliert werden.

b) Der Automat muß feststellen können, ob der Zustand vorhanden ist; er muß die Aussage auf ihren Wahrheitsgehalt prüfen und das Ergebnis der Prüfung fixieren.

c) Einer als "bedingt auszuführend" gekennzeichneten Operation muß die "Bedingung" genannt werden; es muß ihr durch Angabe der speichernden Zelle Zugang zum vorher entstandenen Prüfungsergebnis geschaffen werden.

a) und b) lassen eine Durchführung zu, die zu einer Parallele zu den arithmetischen Plangleichungen führt. Die Frage nach dem Vorhandensein einer gewissen Eigenschaft kann als operatives Planelement betrachtet werden, das die Größen der Testgleichungen zu einem Resultat verarbeitet. Dabei wird das Resultat eine einstellige Dualzahl bilden, dem "ja" oder "nein" als Antwort auf die Frage nach dem Wahrheitsgehalt entsprechend. Schafft man nun in der Maschine eine Reihe einzelliger Speicher, die den Indexregistern auf Wortebene entsprechen, so ist damit dem Leitwerk jederzeit die Möglichkeit gegeben, bei bedingten Operationen diese dem jeweiligen Zustand entsprechend zu interpretieren.
Ein Beispiel diene der Erläuterung:

$$| x_{n+1} - x_n | > \varepsilon \Longrightarrow P$$

Dem testenden Operationselement $>$ kommt hier zu, festzustellen, ob der Inhalt der RR, hier also $|x_{n+1} - x_n|$ größer als die folgende Größe ε ist, und das "ja" oder "nein" als Resultat nach der Zustand-Zelle P abzugeben.

Neben $>$ und ev. $<$ dürfte "enthält" als Testoperationen nötig sein.
Als bedingte Operationselemente kommt etwa der bedingte Sprung $\xrightarrow{\text{bedingt}}$ in Frage. Das Auftreten eines Zustand-Symboles mit dem bedingten Sprung gibt diesem die Ausführungsbestimmung.
Als Beispiel diene die Testgleichung:

$$\text{Test} \quad a > b \Longrightarrow P$$

und eine dadurch bedingte Fortsetzung:

$$\xrightarrow{\text{bedingt}} \quad P \; A \; B$$

im Sinn "Wenn a $>$ b wahr ist, folgt Fortsetzung unter A, sonst unter "B".

Von besonderem Interesse sind Schleifen, d. h. Planteile, die mehrfach durchlaufen werden. Es gibt Schleifen, die ihrer Natur nach Index-gebunden sind. Sie liegen vor, wenn in ihnen Plangleichungen auftreten, die auf Index-behaftete Größen Bezug nehmen und für verschiedene Werte der Indizes behandelt werden sollen. Sie lassen sich besonders bequem mittels einer auf Indexregistern aufbauenden Technik bewältigen. Dabei wird jeweils ein Indexregister die Führung der Schleife übernehmen.

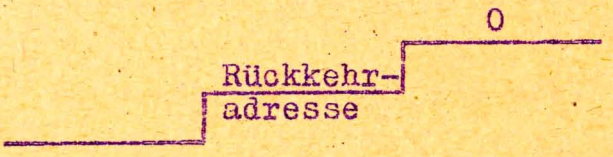
Andere Schleifen, wie sie etwa bei iterativen Näherungsverfahren auftreten, können auch ohne Bindung an einen Index durchgeführt werden.

Der mehrfach zu durchlaufende Planabschnitt wird durch ein öffnendes und ein schließendes Symbol abgegrenzt werden können.

|: als öffnendes

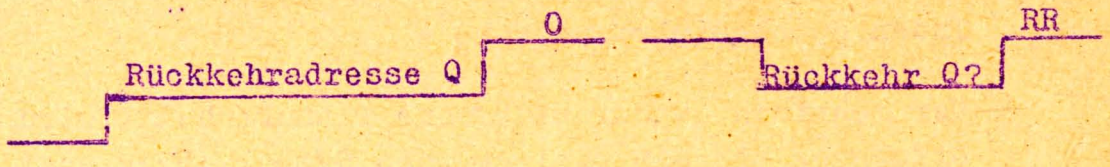
:| als schließendes Symbol.

Dabei muß das schließende Symbol (Repetitionszeichen) ein "bedingtes" Element im obigen Sinn sein. Mit dem öffnenden Zeichen (Iteration) muß die Speicherung der Rückkehradresse in einer um eine Stufe angehobenen ZvD und die Löschung der nochmals um eine Stufe angehobenen ZvD verbunden sein.



Das Repetitionszeichen wird alternativen Charakter haben: Es wird nach Absenkung um eine Stufe in der ZvD die Rückkehradresse finden lassen, die nächst höhere Stufe wieder löschen und mit der Repetition am Planelement nach dem Iterationszeichen beginnen.

Nun gibt es aber im mathematischen Formelbild ein Iterationszeichen besonderer Prägung, das Summenzeichen Σ. Es tritt an Stelle des reinen Iterationszeichen |: und macht sich durch Niederschrift im ZvD an der angehakten Rückkehradresse dem Repetitionszeichen gegenüber kenntlich. Das Repetitionszeichen wird daher durch entsprechende Frage feststellen, ob es eine reine Iteration zu veranlassen hat oder eine mit akkumulierendem Charakter. Im letzteren Fall wird nach Erfassung der Rückkehradresse in die nächst höhere Stufe der Inhalt des RR abgelegt.



Wenn dagegen das Repetitionszeichen nicht mehr die Bedingung zur Repetition vorfindet, muß ein Gang ähnlich dem der schließenden Klammer folgen.

Um eine Iteration - sei sie rein oder von akkumulierendem Charakter - beginnen, fortsetzen und schließlich beenden zu können, müssen dem Flußdiagramm entsprechend zunächst gewisse Indexvorschriften vor Beginn und zwischen den Zeichen eingefügt werden. Dies läßt sich in Form von Plangleichungen durchführen. Daneben ist noch eine Testung zwischen den Zeichen nötig, um zu erkennen, ob die Repetition zu beenden ist.
Das folgende Schema

$$1 \Longrightarrow i \sum \ldots\ldots\ldots i = n \Longrightarrow P \ldots i + 1 \Longrightarrow i \ldots :| P$$

gibt die Struktur für eine von i = 1, 2 bis n laufende Summierung. Um solche Plangleichungen in Form von Zwischenbemerkungen mitten in einen beliebigen Rechenablauf einwerfen zu können, erscheint es zweckmäßig, Parenthese-Zeichen aufzunehmen. Das öffnende Parenthese-Zeichen $\Longrightarrow$ speichert den Inhalt des RR in der nächst höheren Stufe und löscht die nächstfolgende Stufe; das schließende Parenthese-Zeichen $\Longrightarrow$ greift den abgespeicherten Wert wieder auf und kehrt zur Ausgangsstufe zurück.
Ein Schleifenende kann

1. von vornherein festliegen
2. von einem anderen Index abhängen
3. durch Unterschreitung einer Prüfgröße entschieden werden
4. durch Auftreten einer Markierung erkenntlich werden

Dabei können hinsichtlich der Repetition drei Fälle auftreten:

a) Das Ende ist noch nicht erreicht.
b) Das Ende ist gerade erreicht.
c) Das Ende ist schon überschritten.

Der Fall c), der eintreten kann, wenn das Ende von dem Wert eines anderen Index abhängt, bedarf einer besonderen Beachtung. In diesem Fall ist eine Rechnung durchgeführt worden, die sich nachträglich als zwecklos erweist. Das RR muß das Repetitionszeichen gelöscht verlassen. Um die drei Möglichkeiten erfassen zu können, sind zwei Testungen nötig. Das Repetitionszeichen muß doppelt bedingt sein.

---

2. Um einer Größe habhaft zu werden, benutzt der Automat im allgemeinen deren Adresse. Es treten aber Verhältnisse auf, in denen von diesem normalen Verfahren abgewichen wird.

2a) Es kann sein, daß eine Größe sich dem direkten Zugriff über die Adresse entzieht, da sie im Sinn der logischen Struktur nicht primär, sondern erst sekundär oder testiär erreicht werden kann. Hier setzt die schon anderweitig angewandte Technik der Adresse von Adresse ein, etwa in Form der Münchener Schule (Perm) in Kombination mit vor- und nachgeschalteter Addition. Um Größen, die Indizes tragen, im Plan dem mathematischen Formelbild entsprechend behandeln zu können, wird hier ein davon abweichender Weg gewählt, der als "Adresse von Adresse mit nachrückendem Index" gekennzeichnet werden kann.

Ein Planelement a ohne Index stellt die Größe a durch ihre Adresse zur Verfügung: a $\Longrightarrow$ Adresse; $\langle a \rangle \Longrightarrow$ Größe.

Ein durch "Adresse von Adresse" ($*$ Symbol der Perm) gekennzeichnetes Symbol benutzt den Inhalt der dadurch erreichten Zelle als Adresse der Größe a

$$\langle a \rangle \Longrightarrow \text{Adresse}$$
$$\langle\langle a \rangle\rangle \Longrightarrow \text{Größe}$$

Steht bei dem Symbol ein Index z. B. a i, so bildet der Automat die Adresse additiv aus dem Datum a und dem Inhalt des Indexregisters i

$$a + \langle i \rangle \Longrightarrow \text{Adresse}$$
$$\langle a + \langle i \rangle \rangle \Longrightarrow \text{Größe}$$

Ein mit einem Index versehenes Symbol kann ebenfalls als "Adresse von Adresse" gekennzeichnet werden.

Von besonderem Interesse ist aber dabei das Auftreten eines zweiten Index ev. auch weiterer Indizes. * a i k bildet zunächst aus a und i eine Adresse $a + \langle i \rangle$, die zu einem Inhalt leitet, der selbst wieder als Datum mit dem "nachrückenden" Index k zu einer weiteren Adresse führt:

$$\langle a + \langle i \rangle \rangle + \langle k \rangle \Longrightarrow \text{Adresse.}$$

Die natürliche Speicherung der Komponenten einer Matrix wäre die in einem Raum von n Dimensionen, wenn n die Zahl der charakterisierenden Indizes ist. Nun wird aber bei der Aufteilung des Speichers aus Gründen einer ökonomischen Ausnutzung dieser als eine einfach ausgedehnte Mannigfaltigkeit diskreter Elemente betrachtet, wenn auch der tatsächliche Aufruf mehrdimensional z. B. nach Spurengruppen, Spuren und Sektoren geschieht.

Das Umordnen der Elemente einer mehrdimensionalen Matrix in eine lineare Folge bei der Speicherung stellt die Adresse in linearen Zusammenhang mit den Indexwerten. Die darin auftretenden Koeffizienten sind adressenmäßig weitere Bestimmungsgrößen, so daß sich die Charakterisierung einer Matrix im Programm von der im mathematischen Formelbild vorliegenden Darstellung entfernt.

Man kann zwar formal die Zahl der Bestimmungsgrößen im Plan auf die im Formelbild nötige Zahl herabsetzen, wie es etwa Rutishauser durchführt, wenn er den Indizes i_k

das Gewicht 10^{-3k} zuordnet und das Produkt aus Gewicht und Koeffizient speichert. Es bleibt aber auch hier, daß dem Formelbild artfremde Bestandteile dem Plan eingefügt werden und damit den Programmierer belasten.

Daß die Adresse sodann erst durch Ausführung zusammengesetzter Operationen, nämlich Multiplikationen und Additionen gefunden wird, ist ein der Programmierungstechnik wie dem Ablauf unbequemes Faktum.

Wenn man sich demgegenüber die Technik der Adressenänderungen und ihre Realisierung als Änderungen I., II. und III. Art vor Augen hält und nach den Gründen sucht, die dabei diese elegante, d. h. hier dem inneren Wesen angepaßte Technik zustande kommen ließ, so findet man diese in der Tatsache, daß man die Durchführung dieser Maßnahmen dorthin verlegt hat, wo sie logisch hingehören:
Die Änderungen I. Art als Übergang von relativen Adressen der Bibliothekspläne zu absoluten Adressen sind durchführbar, sobald der Ort, den der Bibliotheksplan im Rahmen des gesamten Plans einnehmen wird, bekannt wird; das geschieht im Augenblick der Eingabe. Damit ist es sinnvoll, auch die Adressenänderung I. Art mit diesem Vorgang zu verknüpfen.

Die Änderungen III. Art gehören zum Schleifendurchlauf, also zum Rechenablauf. Sie lassen sich mit passenden Mitteln des Rechenwerks, nämlich Indexregistern, bestens erledigen.

Änderungen II. Art als Verknüpfungen von Teilen, die zu verschiedenen Teilprogrammen gehören, sind zweckmäßig an den Übergang (Sprung) zu binden, da hier im Augenblick des engsten Kontakts am leichtesten Informationen aus einem in den anderen Teil übertragen werden können.

Geht man mit einer entsprechenden Fragestellung an das Problem der Matrix-Adressen heran, so findet man, daß die Schwierigkeit von dem oben erwähnten Umordnen der Elemente einer mehrdimensionalen Mannigfaltigkeit in eine lineare Folge herrührt. Es ist ein Problem der Art und Weise der Speicherung, der Platzverteilung, und die Bewältigung dieser Schwierigkeit sollte daher an der Stelle der Speicherzuteilung gesucht werden, nicht aber im eigentlichen Plan. Legt man vor den zur Aufnahme der Matrixelemente vorgesehenen Speicherraum eine ebenfalls linear auseinandergezogene Leitpyramide, die sich mittels der obigen leicht modifizierten "Adresse von Adresse" - Technik programm - und ablaufmäßig bequem durchlaufen läßt, so ist das Programm völlig von den Bestandteilen befreit, die dem mathematischen Formelbild fremd sind.

Eine Größe etwa a_{ik} erscheint auch im Plan nur mit diesem Symbol a, begleitet von Zusätzen, die auf den Einsatz der Indexregister i und k hinweisen. Da man auch das Hilfsmittel des Anhakens auf Elemente der Leitpyramide anwenden kann, gelingt es, den jeder Schleife zugeordneten Index zur vollen Steuerung des Schleifenablaufs zu benutzen. Hinsichtlich des Speicherbedarfs wird zwar zusätzlicher Raum gefordert, um die Leitpyramide unterzubringen. Andererseits gestattet das Verfahren aber eine dicht gedrängte Niederschrift etwa, wenn eine Dreiecksmatrix vorliegt, so daß hier kein entschiedener Nachteil zu erkennen ist.
Im Plan selbst dagegen wird nur der Raum eines Wortes nötig, um die gesamte Charakterisierung der Größe mit Indizes durchzuführen.
Ein Beispiel runde diese Betrachtungen ab.

Für Speicherung der Matrixkomponenten a_{ik} $\begin{pmatrix} i = 0, 1, 2, 3 \\ k = 0, 1, 2, 3 \end{pmatrix}$
sei der mit 501 beginnende Block vorgesehen.

Dann bilden die vier ersten Zeilen die Leitpyramide und tragen bei der Eingabe als Inhalt die relativen Adressen 4, 8, 12, 16. Durch Adressenänderung I. Art bei der Eingabe werden sie in absolute Adressen umgewandelt. Das Symbol a wird im Vermittlungsblock der Rufadresse 501 gegenübergestellt, so daß das anschließende kompilierende Verfahren das Symbol a durch die absolute Adresse 501 ersetzt.
Dem Rechenablauf liegt damit ein Plan vor, der die Größe a durch das Planelement

501 i k sucht.

Der mit 501 beginnende Block hat das folgende Aussehen:

	Adresse	Zellen-Inhalt	
Rufadresse	501	505	Leitpyramide
	502	509	
	503	513	
	504	517	
	505	a00	Matrix-Komponenten
	506	a01	
	507	a02	
	508	a03	
	509	a10	
	510	a11	
	511	a12	
	512	a13	
	513	a20	
	514	a21	
	515	a22	
	516	a23	
	517	a30	
	518	a31	
	519	a32	
	520	a33	

Es werde der Schleifendurchlauf betrachtet, für den $2 \Longrightarrow \langle i \rangle$ und $1 \Longrightarrow \langle k \rangle$ vorliege. Dies bedeutet, daß die Größe a 21 gesucht wird. Aus dem Datum 501 und dem Inhalt des Indexregisters i wird als Scheinadresse gebildet 501 + 2 = 503.
Als Inhalt der Zelle unter Adresse 503 findet sich 513.

Dieses wird als Datum mit dem nachrückenden Index k kombiniert und liefert als Adresse 513 + 1 = 514. Die unter der Adresse 514 stehende Zelle enthält die gewünschte Größe a 21.

2b) Neben Fällen, in denen die Größe zu "weit" liegt und der Automat sie erst über das Hilfsmittel "Adresse von Adresse" erfassen kann, können andere auftreten, in denen die Größe zu "nah" gekommen ist: In das Befehlsregister ist nicht das Symbol der Größe a sondern diese selbst gelangt. Dieser Fall ist erst durch die Trennung von Operationszeichen und Operandenzeichen möglich geworden, bietet aber verlockende Möglichkeiten der Plangestaltung: Einzel auftretende Konstante können ohne Umweg über den Zahlenblock im Rechenplan selbst aufgenommen werden. Auch hier zeigt sich die stärkere Anlehnung an das mathematische Formelbild, das neben Buchstaben auch direkte Zahlen verwendet. So kann man im Beispiel 2 r $\pi \Rightarrow$ u auch bei der Planaufstellung dieser Geflogenheit folgen und die Konstante 2 direkt als Planelement erscheinen lassen. Sollte π nur einmal im Plan auftreten, so würde man auch hier die direkte Zahl 3, 14 15... einsetzen. Maschinentechnisch bedeutet dies die direkte Überführung des Inhalts des Befehlsregisters in das Rechenwerk.

§ 10 Zusammenfassung

Rechenzeit und Zeitaufwand für Programmierung stehen schon bei mittelschnellen elektronischen Rechenautomaten in einem starken Mißverhältnis. Damit wird das Problem der Programmierung eine zentrale Frage.

Es sind im wesentlichen vier Punkte, die das Programmieren bei den derzeitigen Maschinen zum zeitlichen Engpaß, zu einer beachtlichen Fehlerquelle und zu einer höchst unbequemen Arbeit werden lassen.

1. Die derzeitige Programmierung weicht stark von dem bewährten mathematischen Formelbild ab.

2. Die Angabe von Adressen ist maschinenbedingt, der Rechenaufgabe gegenüber aber problemfremd.

3. Die Behandlung Indizes-behafteter Größen ist schleppend.

4. Auf die strukturellen Probleme des Flußdiagramms sind die derzeitigen Automaten nur ungenügend vorbereitet.

Man hat versucht, diese Schwierigkeiten zu mildern, indem man besondere Programmierungstechniken entwickelte: Schaffung von Bibliotheksprogrammen, Verwendung von Pseudo-Codierung, Heranziehung des Automaten zur Berechnung der Rechenpläne usw.

Ziel dieser Abhandlung ist, diese Schwierigkeiten in der Programmierung dadurch zu überwinden, daß die Struktur des Automaten unter Berücksichtigung der heute gegebenen technischen Möglichkeiten den Erfordernissen enger angepaßt wird.

In einem ersten Teil (§ 1 - § 4) wird der gegenwärtige Stand in Richtung einer automatischen Programmierung gegeben, soweit dies für den weiteren Teil von Bedeutung erschien: Das Prinzip einer adressenfreien Programmierung, das Problem der Bibliotheksprogramme und die Rechenplanberechnung durch den Automaten.

Im zweiten Teil führt die Forderung, nur die problemnahen Größen in den Gesichtskreis des Programmierers aufzunehmen, zu einer Abkehr von der Verknüpfung eines Operationssymbols und eines Operandensymbols zu einem Befehl. Diese Verselbständigung der Operationssymbole auf der einen Seite und der eine Größe charakterisierenden Symbole auf der anderen Seite ist ein notwendiger Schritt, um Plangleichungen in mathematischem Formelbild direkt als Rechenpläne verwenden zu können;

damit tritt aber der Zwang auf, daß sich der Automat die ihm nun fehlenden Informationen selbst beschafft. Diese Entwicklungsrichtung trifft nun zusammen mit einer anderen Tendenz: Die Büro-Kurbel-Rechenmaschine wurde durch Verwendung elektrischer Bauelemente vollautomatisiert, erhielt zusätzlich Speichereinrichtung. Aber erst mit der technischen Möglichkeit der Groß-Speicher wurde der programmgesteuerte Automat realisierbar. Aber noch wird dem Automaten Schritt für Schritt der Speicher von außen d. h. über das Programm zugeteilt.
Er hat die Stufe eines primitiven Rechners erreicht, dem auch die Niederschrift und Einholung aller Zwischenergebnisse formular gebunden sind, während der qualifizierte Rechner nur in den problemnahen Größen an ein Formular gebunden ist und für Zwischenergebnisse selbständig seinen "Schmierzettel" anlegt.
Damit zeichnet sich eine klare Entwicklungsrichtung auch für den Rechenautomaten ab: Er muß über einen Teil seiner Speicherkapazität selbst verfügen, ihm muß ein "automatisches Gedächtnis" eigen sein.

Ein Automat mit einem solchen Gedächtnis wird nun die Planelemente, das sind die einzelnen Operationszeichen der Plangleichung und die Symbole, die Größen charakterisieren, aufnehmen; ausführen wird er dagegen jeweils nur, was sich als ausführungsmöglich bis dahin geklärt hat, das restliche wird er in seinem Gedächtnis zurückstellen, bis es nach Eingang noch weiterer Informationen als eindeutige Operation erkannt wird. Dabei durchläuft das Gedächtnis in einer kontinuierlichen auf- und absteigenden Folge ein relativ enges "Bewußtsein". Die übrigen gemachten "Erfahrungen" liegen außerhalb dieses Blickfeldes im "Unbewußten", aus dem sie zur gegebenen Zeit auftauchen.
Durch die Verselbständigung der Größensymbole ergibt sich die Möglichkeit, Indizes behaftete Größen in derselben Art zu behandeln, wie dies im mathematischen Formelbild geschieht,

wobei die sonst vorhandenen Schwierigkeiten einer besonderen Indexrechnung durch Speicherungsmaßnahmen auf die einfache Indexregister-Technik reduziert werden.
Eine entsprechend angepaßte Bibliotheksprogramm-Technik bezieht auch Funktionen in dieses Programmierungsverfahren nach math. Formelbild ein, und zwar Funktionen von einem und mehreren Parametern, wie Funktionen von Funktionen und in Verschachtelung, sowie mathematische Verfahren.

Die bei der Behandlung struktureller Probleme auftretenden Schwierigkeiten werden beleuchtet und führen zu besonderen Maßnahmen. Bedingte Operationen werden an "Zustände" des Automaten gebunden.
Zur Charakterisierung des jeweiligen Automaten-Zustandes dienen einige "Zustandsgrößen", die als einstellige Speicher in einer gewissen Parallele zu den Indexregistern von Wortlänge stehen.

So wie eine Zahl sich als Resultat von arithmetischen Verknüpfungen an Größen ergibt, stellt eine Zustandsgröße das Ergebnis einer Verknüpfung von Größen durch testende Operation dar.

Damit entsteht in der Test-Plangleichung ein gesuchtes Analogon zu der arithmetischen Plangleichung und gibt mit Summenzeichen, Iterationszeichen und Repetitionszeichen die Möglichkeit, eine direkt an das Flußdiagramm angelehnte Programmierung durchzuführen.

Zur Bereitstellung einer Größe wird im allgemeinen die Adresse der Größe (ev. auf dem Umweg über das Symbol) benutzt; in besonderen Fällen wird der Zugriff erst über "Adresse von Adresse" möglich, ein Verfahren, das hier passend weiterentwickelt wird. Die Verselbständigung der Operanden gibt aber nun obendrein eine bis dahin verschlossene Möglichkeit, eine Größe selbst ohne Umweg über ihre Adresse in den Plan einzubauen, entsprechend dem Vorgehen im mathematischen Formelbild,

das neben Buchstaben auch Zahlen direkt verwendet.
Ein orientierender Vergleich zwischen einem optimal aufgestellten Rechenplan, einem nach der Methode von Rutishauser errechneten Rechenplan und einem nach der hier vertretenen Methode der Programmierung nach math. Formelbild, stellt diese Methode zwischen jene hinsichtlich Automatenbelastung.
Diese Abhandlung strebt nicht eine konstruktive Bauvorschrift an, wenn sie auch gelegentlich geschehene Hinweise in dieser Richtung für nicht unangebracht hält.