Bauer et al 1958a

Formelübersetzungsprojekt Zürich-München-Darmstadt

Projektstufe 1: Festlegung der Formelsprache

Interner Bericht Nr. 2c

19.2.1958

21 Seiten

23 Blatt Zeichnungen

F.L. Bauer - H. Bottenbruch - P. Graeff - P. Läuchli -
M. Paul - F. Penzlin - H. Rutishauser - K. Samelson

Durchgeführt mit Unterstützung

der Deutschen Forschungsgemeinschaft,
des Instituts für angewandte Mathematik der ETH Zürich,
des Mathematischen Instituts und des Rechenzentrums der TH München und
des Instituts für Praktische Mathematik der TH Darmstadt.

Sent by Samelson to Naur, 1977 Dec
P. 7, 8 also from Läuchli
Zeichnungen from Läuchli

Nachstehend wird die erste Stufe einer algorithmischen Sprache entwickelt, die die Basis des Formelübersetzungsprojekts Darmstadt-München-Zürich darstellt. Sie soll die vollständige Beschreibung beliebiger Rechnungsabläufe in einer übersichtlichen, leicht kontrollierbaren Form gestatten, mit der wichtigen Einschränkung, daß zunächst nur Rechnungen mit reellen Zahlen zugelassen sind. Hyperkomplexe Größen aller Art (komplexe und mehrfach genaue Zahlen, Vektoren, Matrizen ....) sind in dem System vorläufig nicht vorgesehen. Rechnungen mit solchen Größen sind also komponentenweise anzugeben.

A. Elemente

I. Zeichen

Die Grundlage der Formelsprache bildet eine Reihe von Symbolen mit bestimmten Bedeutungsinhalten. Diese Bedeutungsinhalte sind selbstverständlich unabhängig von der speziellen Darstellung der Symbole durch Schreibzeichen. Die Zuordnung ist jedoch durch die Verwendung in Texten der Umgangssprache weitgehend festgelegt.

Folgende Gruppen von Zeichen sind vorgesehen:

			Sammelbezeichnung
1)	Buchstaben	a b ..... z A B ..... Z und andere Alphabete	$\mathfrak{L}$
2)	Ziffern	0 1 9	$\mathfrak{Z}$
3)	Zahlkennzeichen	. , Dezimalpunkt 10 Dezimalbasis	
4)	Trennzeichen	; | : ... , ( ) [ ] ⟨ ⟩ 1)	$\mathfrak{T}$
5)	arithmetische Operationszeichen	· / + − 2)	$\mathfrak{O}$
~~6)~~	~~logische Operationszeichen~~	~~¬ ∧ ∨~~	~~$\mathfrak{O}$~~
7)	Belegungszeichen	⇒ :=	

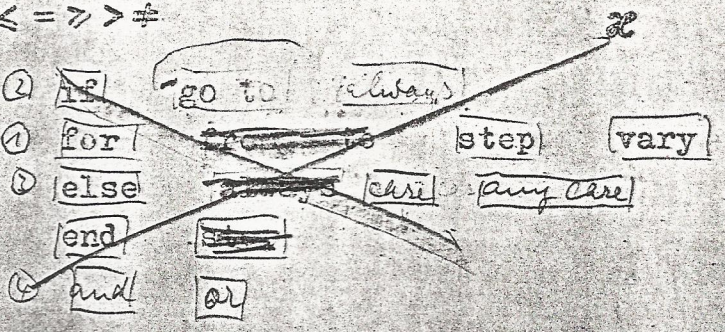
---

1) Die Einrahmung bedeutet, daß das Zeichen nicht zum Abdruck kommt, sondern in anderer Weise (hier Tief- bezw. Hochsetzen der Schreibzeile) bemerkbar ist.

2) Der hochgestellte Kal-Punkt kann, wenn er zur Unterscheidung von Dezimalpunkt nötig ist, durch ein Malzeichen × ersetzt werden

8) Vergleichszeichen $< \leq = \geq > \neq$

9) universelle Textworte



for	step	vary
if	always	
case	else	any case
and	or	
goto	stop	
end		

II. Arithmetik

Die Formelsprache dient vornehmlich der Beschreibung arithmetischer Prozesse. In diesen treten auf

1) Zahlen Z

Form: $\pm\zeta_1 \ldots \zeta_k \cdot \zeta_{k+1} \ldots \zeta_n \cdot 10^{\pm\zeta_{n+1} \ldots \zeta_m}$

Der Skalenfaktor $\cdot 10^{\pm\zeta_{n+1} \ldots \zeta_m}$ ist fakultativ, ebenso das positive Vorzeichen für die gesamte Zahl und für den Skalenexponenten. Bei ganzen Zahlen dürfen der Dezimalpunkt und folgende Nullen weggelassen werden, dagegen ist bei echten Dezimalbrüchen eine Null vor dem Punkt erforderlich.

Beispiele

273
$2.73 \cdot 10^2$
3.14159
-0.07
$+0.478 \cdot 10^{-2}$

2) Variable V

als Decknamen für Zahlen wie in der elementaren Arithmetik.

Form: Ein Buchstabe λ gefolgt von beliebigen Buchstaben und/oder Ziffern

$\lambda \quad \lambda,\zeta$

Beispiele

a
x1
alpha

Eine Unterscheidung zwischen Festkomma- und Gleitkommavariablen gibt es nicht.

Variablennamen dürfen selbstverständlich nicht mit einem der universellen Textworte zusammenfallen.

5) Indizierte Variable 𝒱
als Deckmamen für Komponenten mehrdimensionaler Zahlsätze.

Form:

	Variablenname V
gefolgt von	Indexbeginnzeichen ⇩
k durch	Trennzeichen ;
getrennten Indexstellen u.	Indexschlußzeichen ⇧

Die Indexstellen sind besetzt mit arithmetischen Ausdrücken 𝒜 (vgl.6), im einfachsten Falle mit Zahlen Z oder Variablen V.

Beispiele

a ⇩ i ⇧

zeta ⇩ 7 ; i·(i+1)/2 ⇧

b ⇩ i-j ; k+j ⇧

inverse ⇩ i ; k ⇧

Die Zahl der Indexstellen ist nicht grundsätzlich beschränkt. Ebenso bestehen keine Beschränkungen hinsichtlich der auf den Indexstellen stehenden Ausdrücke. Jedoch sind indizierte Variable grundsätzlich nur für ganzzahlige Indexwerte definiert.

4) Funktionen $\mathfrak{f}$

als Abkürzungen für kompliziertere Rechenvorschriften, insbesondere für die Standardfunktionen der Analysis.

Form:

	Funktionsname (identisch mit Variabler ohne oder mit Index)	$\mathfrak{V}$
gefolgt von	Argumentbeginnzeichen	[
k durch	Trennzeichen	;
getrennten Argumentstellen und	Argumentende-Zeichen	]

Die Argumentstellen sind besetzt mit arithmetischen Ausdrükken $\mathfrak{A}$ (vgl.6),
im einfachsten Falle mit Zahlen $\mathfrak{Z}$ oder Variablen $\mathfrak{V}$.

Beispiele

sqr [x]
sin [w·t + f]
P[1 ; m] [v;w]

5) Funktionale $\mathfrak{F}$

als Abkürzungen für Rechenvorschriften, die noch willkürlich vorgebbare Funktionen oder Variablensätze (= Gesamtheit aller Komponenten einer indizierten Variablen) enthalten.

Form:

	Funktionalname (wie Funktionsname)	$\mathfrak{V}$
gefolgt von	Argumentbeginnzeichen	[
k_1 durch	Trennzeichen	;
getrennten Argumentfunktions-Stellen		
	starkem Trennzeichen	!
k_2 durch	Trennzeichen	;
getrennten Argumentstellen und	Argumentende-Zeichen	]

Die Argumentstellen sind wie bei Funktionen besetzt mit beliebigen arithmetischen Ausdrücken $\mathfrak{A}$ (6)

Die Argumentfunktions-Stellen dagegen sind besetzt (ausschließlich)
entweder mit einer Funktion mit vorgeschriebener Argumentanzahl
oder mit einer indizierten Variablen mit vorgeschriebener Indexstellenanzahl,

deren Argument- bzw. Indexstellen mit einfachen Variablen als Dummies ausgefüllt sind, die gegebenenfalls auch zur Identifizierung von Argumenten dienen. Diese Dummy-Variablen haben also nur Bedeutung für das betreffende Funktional und sind ohne Beziehung zu irgendwann an anderer Stelle eingeführten gleichlautenden Variablen.

Beispiele:

phi [r [a;b]; s[a;b] !]
simpson [f[x] ! a;b]
runge kutta[g[u;v]! x;y;h]
skalarprodukt [a$_{i}$; b$_{i}$! n]

6) arithmetische Ausdrücke $\mathfrak{A}$

Zusammengesetzt aus		
	Operationszeichen	ω
	Klammern	( )
	Zahlen	Z
	Variablen	$V, \mathfrak{V}$
	Funktionen	$\mathfrak{F}$
	Funktionalen	$\mathfrak{G}$

nach den üblichen Regeln der Arithmetik.

Insbesondere gilt · / geht vor + − ,
d.h. zum Beispiel a·b + c/d ≡ (a·b) + (c/d)
Nicht zulässig, da mißverständlich, ist a/b·c, a/b/c .
Im Sinne der Bindungsordnung überzählige Klammern sind zulässig. Sie sind numerisch nicht bedeutungslos, da die in Klammern gesetzten Ausdrücke für sich ausgewertet werden.

Im übrigen entspricht die Reihenfolge der Ausführung der Operationen der Reihenfolge der Aufschreibung.

Die "intended multiplication" ab für a·b ist verboten.

Beispiele:

(a+b)·(c+d)
a/(c+d) − b·e/f
x·sin[a·y] + b·cos[c·y]
a$_{i;j}$ · b$_{j;k}$

7) Ergibt-Formeln ⇐

als die eigentlichen Rechenanweisungen.
Sie haben die Form von nach der Unbekannten ausgelösten Gleichungen.

arithmetischer Ausdruck Ergibt-Zeichen Variable Trennzeichen

$$a \Rightarrow b \;;$$

Variablenbezeichnungen werden frei in dem Augenblick, in dem die durch sie bezeichneten Zahl (oder Zahlen) nicht mehr gebraucht werden. Die Wiederverwendung einer Variablenbezeichnung darf sogar innerhalb ein und derselben Ergibt-Formel erfolgen.

Beispiel: $s + a_j \Rightarrow s;$

Soll (lediglich im Druck test!)

See Faüchli's copy

7) Ergibt-Formeln €

als die eigentlichen Rechenanweisungen.
Sie haben die Form von nach der Unbekannten ausgelösten Gleichungen.

arithmetischer Ausdruck Ergibt-Zeichen Variable Trennzeichen

$$\mathfrak{A} \Rightarrow \mathfrak{B} \quad ;$$

Variablenbenennungen werden frei in dem Augenblick, in dem die durch sie bezeichnete Zahl (oder Zahlen) nicht mehr gebraucht werden. Eine "Wiederverwendung" einer Variablenbezeichnung darf sogar innerhalb ein und derselben Ergibt-Formel erfolgen.

Beispiel: $s + a_j \Rightarrow s \quad ;$

Lediglich im Drucktext soll bei Wiederverwendung einer Variablenbezeichnung durch Beifügung hochgestellter eingeklammerter Index eine Unterscheidung der nacheinander mit der gleichen Variablenbezeichnung belegten Zahlen getroffen werden. Dies dient insbesondere der Hervorhebung des rekurrenten Charakters einer solchen Wiederverwendung.

Beispiel: $s^{(j)} + a_j \Rightarrow s^{(j+1)} \quad ;$

Bei Einhaltung dieser Vorschrift sind eingeklammerte Indizes für die Formelübersetzung belanglos. Auf die Möglichkeiten, durch geschickte Transformation von Indexläufen die Einklammerung von Indizes zu ermöglichen (Beispiel: ZAMP, Bd. 3, S. 312 (1952)), sei besonders hingewiesen.

This page copied from Läuchli's version

III. ~~Logik~~ Bedingungen und Fallunterscheidungen.

1) ~~Arithmetische~~ Kriterien K
bestimmen jeweils eine einfache "ja-nein"-Alternative (Kriterium "erfüllt" oder "nicht erfüllt").

Form:
arithm. Ausdruck Vergleichszeich. arithm.Ausdr. Trennzeichen

$\mathfrak{A}_1$ $\mathfrak{R}$ $\mathfrak{A}_2$;

Beispiele: abs [(x-y)/x] > e ;

$x \neq 0$;

Nichtarithmetische Kriterien sind

2) Fallbezeichnungen B_k
dienen zur abkürzenden Kennzeichnung der verschiedenen Zweige mehrfacher Alternativen.

Form: ein Buchstabe, gefolgt von einer

Folge beliebiger ~~Buchstaben und/oder~~ Ziffern, abgeschlossen durch das Zeichen) ~~→, ≤~~ $\lambda(\xi_n - \xi_\kappa)$

der . bedeutet eine Fallgruppe, die durch die Ziffern bestimmt dabei eine Fall-
~~jede Buchstaben-Ziffern-Verteilung~~ ~~Fallklasse~~ B_k. in Sinne der Definition klassifiziert wird.

~~Je einer Klasse~~ gehören also ~~z.B.~~ an 1) 2) 3) a1)
ebenso a) b) c) a2)
oder a1) a5) b3)
oder cond) cont)

Variable sind

3) logische ~~Ausdrücke~~ $\mathfrak{L}$
~~werden nach den Regeln des Aussagenkalküls gebildet aus den logischen Operationszeichen $\mathfrak{S}$, den Klammern ( ) und den logischen Variablen. Als solche stehen zur Verfügung:~~ (s. III.1)
a) arithmetische Kriterien, ~~die einzuklammern sind,~~
b) nichtarithmetische Kriterien, z.B. Wahlschalter,
c) Fallbezeichnungen B_k

Beispiele: ~~a)∧(x>0)~~
~~a)∧(x>0)∨b)∧(x<0)~~
~~(x>0)∧(y>0)~~

III. Bedingungen und Fallunterscheidungen

1) Kriterien K

bestimmen jeweils eine einfache "ja-nein"-Alternative (Kriterium "erfüllt" oder "nicht erfüllt").

Form:

arithm. Ausdruck Vergleichbezeich. arithm.Ausdr. Trennzeichen

$\mathfrak{A}_1$ $\mathfrak{R}$ $\mathfrak{A}_2$;

Beispiele: abs $|(x-y)\ x| > e$;

$x \neq 0$;

Nicht-arithmetische Kriterien, auch Bedienungseingriffe, wie Wahlschalter, sind entsprechend den Möglichkeiten spezieller Anlagen durch Textworte zu formulieren.

2) Fallbezeichnungen B

dienen zur abkürzenden Kennzeichnung der verschiedenen Zweige mehrfacher Alternativen

Form: $\mathfrak{z}$) { ··· {)

mit einer oder mehreren Ziffern im Sinne der Dezimalklassifikation

Beispiel: a7)
a2o1)
a2o2)
⋮
a217)
a3

Hinsichtlich der Wiederverwendung von Fallbezeichnungen gilt das für Variablen unter II 7 gesagte.

3) Bedingungen

dienen zur Kennzeichnung derjenigen Anweisungen, die lediglich bedingt auszuführen sind.

3a) explizite Bedingungen $\mathfrak{x}$

sind mit einem der mehreren Kriterien in Konjunktion gebildet

Form: if K and K .... and K :

Beispiel: if x>y and y≠0

3b) fallweise Bedingungen $\mathfrak{R}$

sind mit einer oder mehreren Fallbezeichnungen in Disjunktion gebildet

Form: case B or B ... or B

eine Fallbezeichnung, gleichgültig ob sie selbst definiert ist oder nicht, vertritt dabei alle ihr im Sinne der Dezi-

This page from Läuchli

4) Falldefinitionen $\mathfrak{D}$
bestimmen die Zweige mehrfacher Alternativen.

Form: Fallbezeichnung logischer ~~Ausdruck~~ Variable Schlußzeichen
B_k $\mathcal{L}$ (eine, mehrere durch und verknüpfte) ;

Die Falldefinition bedeutet die Belegung der Fallbezeichnung B_k als logischer Variabler mit dem Wahrheitswert des zugehörigen logischen Ausdrucks.

Der Fall B_k "liegt vor", wenn $\mathcal{L}$ den Wert "ja" hat, sonst nicht.

Beispiele:
1) $x > 0$;
a) $(x > a) \wedge (y > b)$;
c3) $a) \wedge (x \neq y)$;

5) Fallaufrufe $\mathfrak{R}$
dienen zur Kennzeichnung derjenigen Prozesse, die beim Vorliegen eines oder mehrerer vorher definierter Fälle beliebiger Klassen auszuführen sind.

Form: if $B_{k(1)}^{(1)} \vee \ldots \vee B_{k(\nu)}^{(\nu)}$

Beispiele:
if a) $\vee$ b) $\vee$ c1):
if 1) $\vee$ a):

Wenn der Aufruf eines Falles mittelbar auf die zugehörige Falldefinition folgt, kann die Angabe des Fallaufrufes entfallen, wenn die Falldefinition statt durch ";" durch ":" abgeschlossen wird

Beispiele:

anstatt 1) $x > 0$; if 1): also 1) $x > 0$:
anstatt a) $(x > a) \wedge (y > b)$; if a): also a) $(x > a) \wedge (y > b)$:

Einen totalen Fallaufruf, der sämtliche vorher (im Sinne der Aufschreibung) definierten Fälle beliebiger Klassen aufruft, stellt das universelle Textwort "always:" dar.

B. Anweisungen

Die einfachsten abgeschlossenen und für sich verständlichen Bausteine der Formelsprache sollen als Anweisungen bezeichnet werden. Es gibt folgende Klassen:

I. Arithmetische Anweisungen

1) Ergibt-Formeln $\mathfrak{E}$

Form: $\mathfrak{A} \Rightarrow \mathfrak{V}$; (s.A.II.7)

2) Pseudoformeln

Sie dienen technisch gesehen zum Aufruf von Bibliotheksprogrammen, d.h. von vorprogrammierten umfangreichen Algorithmen.

Diese haben im allgemeinen den Charakter von Funktionalen $\mathfrak{F}$. Sie liefern aber häufig mehrere unabhängige Resultate $R_1, \ldots, R_n$, die z.T. Variablensätze sein können.

Darüber hinaus besitzen sie gewöhnlich auf Grund interner Fallunterscheidungen eine Reihe von Nebenausgängen $Y_1, \ldots, Y_n$.

Aus diesen Gründen können sie nicht in arithmetische Ausdrücke eingebaut werden, sondern sind nur in Form von Pseudoformeln aufrufbar.

Form: $\mathfrak{F}\ \ R_1 \Rightarrow \mathfrak{V}_1; \ldots; R_n \Rightarrow \mathfrak{V}_n !\ Y_1 := N_1; \ldots; Y_k := N_k !$

Hierin ist $\mathfrak{F}$ ein normales Funktionalzeichen mit entsprechender Argumentbesetzung, die R_1 bis R_n haben die Form von Variablennamen, die Y_1 bis Y_k die Form von Abschnittsbezeichnungen. $\mathfrak{V}_1$ bis $\mathfrak{V}_n$ sind Variable, gegebenenfalls indizierte Variable mit Dummy-Indizes, N_1 bis N_k sind ~~Satznummern~~ Anweisungsnummern oder Abschnittsaufrufe

Im übrigen ist hinsichtlich der Verwendung von Pseudoformeln auf die Bibliotheksprogramm-Beschreibungen zu verweisen.

Beispiel:

```
gauss jordan[a⊞1;k⊞ ; b⊞1;m⊞ !n; s; w-s]
inverse ⇒c⊞1;k⊞; loesung ⇒y⊞1;m⊞; det ⇒z !
sing := 15; summenpr := 20!
```

II. Logische Anweisungen (Fallunterscheidungen)

1) Falldefinitionen $\mathfrak{D}$

Form: $B_k \mathfrak{L}$; (s.A.III.4)

Ergibt die Auswertung des logischen Ausdruckes $\mathfrak{L}$ den Wert "ja", so daß der Fall B_k "vorliegt", so wird der nächste (im Sinne der Aufschreibung) Aufruf des Falles B_k aufgesucht und es werden die darauf folgenden Anweisungen ausgeführt.

Liegt dagegen der Fall B_k nicht vor, so wird die nächstaufgeschriebene Falldefinition der gleichen Klasse B_k in derselben Weise ausgewertet.

Hat die Definition die Form "$B_k \mathfrak{L}$:", so ist sie gleichzeitig Aufruf (s.B.II.2) und die auszuführenden Anweisungen folgen unmittelbar.

2) Fallaufrufe $\mathfrak{R}$

Form if $B_k^{(1)} \vee \ldots \vee B_k^{(m)}$ (s.A.III.5)

Sie stehen unmittelbar vor den bei Vorliegen eines der aufgerufenen Fälle auszuführenden Anweisungen.

Jeder Aufruf hat Gültigkeit bis zum nächsten Fallaufruf oder bis zur nächsten Definition eines Falles, der einer der im Aufruf aufgeführten oder diesen übergeordneten Klassen angehört.

Wiederholte Aufrufe eines Falles sind möglich und werden durch vorher erfolgte Zusammenführungen mehrerer Fälle, selbst durch die totale Fallzusammenführung durch das Textwort "always:" nicht verhindert.

3) Fallunterscheidungen $\mathfrak{A}$

Die Gesamtheit aller aufeinanderfolgenden Falldefinitionen einer Klasse mit den zugehörigen Fallaufrufen und fallabhängigen Anweisungen bildet eine vollständige Fallunterscheidung. Dies ist eine zusammengesetzte Anweisung, die von der ersten Falldefinition der betreffenden Klasse bis zur nächsten vollständigen Zusammenführung aller Fälle der Klasse durch expli-

II. Logische Anweisungen

1. Falldefinitionen ϑ

Form: B_k $\mathfrak{L}$; (s.A.III.4)

2. bedingte Anweisungen

Form: if $\mathfrak{L}$: Anweisungen

Eine bedingte Anweisung ist eine zusammengesetzte Anweisung. Sie reicht bis zum nächsten "if", das eine neue bedingte Anweisung einleitet, oder bis zum Textwort "any case:". Andere zusammengesetzte Anweisungen dürfen dadurch nicht zerschnitten werden.
Hat der logische Ausdruck $\mathfrak{L}$ den Wahrheitswert "ja", so werden die zur bedingten Anweisung gehörigen Anweisungen ausgeführt. Andernfalls werden diese Anweisungen übergangen.

Beispiel:

a) $x > 0$;
b) $x = 0$;
c) $x < 0$;

if a) Anweisungen
if b) Anweisungen
1) $y < 0$
2) $y = 0$
if b)$\wedge$ 1) Anweisungen
if b)$\wedge$ 2) Anweisungen
if c) Anweisungen
any case Anweisungen
if 1)

a) $x > 0$
b) $x = 0$
c) $x < 0$
1) $y < 0$
2) $y = 0$
if a) Anweisungen
if b) Anweisungen
if b)$\wedge$ 1) Anweisungen
if b)$\wedge$ 2) Anweisungen
if c) Anweisungen
any case Anweisungen
if 1) Anweisungen

if $x > 0$ Anweisungen
if $x = 0$ Anweisungen
if $(x = 0) \wedge (y < 0)$ Anweisungen
if $(x = 0) \wedge (y = 0)$ Anweisungen
if $x < 0$ Anweisungen
any case Anweisungen
if $x > 0$ Anweisungen

zeiten Aufruf oder durch das Textwort "always" reicht. Ist eine Fallunterscheidung in einem Abschnitt A (s.III 3) oder einer Schleife S (s.III 5c) enthalten, so ersetzt das entsprechende Endzeichen "End A!" bzw. "vary V " das die Fallunterscheidung abschließende Textwort "always:". Eine Fallunterscheidung enthält hinter jedem Fallaufruf eine "Leerstelle", die durch beliebige (auch zusammengesetzte) Anweisungen besetzt werden kann. Befindet sich unter diesen an das Vorliegen eines der aufgerufenen Fälle geknüpften Anweisungen wieder eine Fallunterscheidung, so sind die zugehörigen Fallbezeichnungen einer Klasse zu entnehmen, die verschieden ist von den Klassen aller derjenigen Fälle, bei deren Vorliegen der betreffende Fallaufruf durchlaufen werden kann, d.h. den Klassen aller im Aufruf aufgeführten und diesen übergeordneten Fällen.

Die zu einer Fallunterscheidung gehörenden aufeinanderfolgenden Fälle einer Klasse müssen einander ausschließen***) und müssen alle Möglichkeiten überdecken. Letzteres wird automatisch erreicht durch Benutzung der Restfalldefinition $\triangleright_K$ else;

Folgt ein erneuter Fallaufruf auf das Ende einer Fallunterscheidung, d.h. auf die Zusammenführung aller Fälle der betreffenden Klasse, so bildet er mit allen folgenden Fallaufrufen ebenfalls eine Fallunterscheidung, die erst mit der vollständigen Zusammenführung endet.

Beispiele:

a) $x \geqslant 0$;	a) $x \geqslant 0$;	Anweisg.
b) $x \leqslant 0$;	b) $x \leqslant 0$:	Anweisg.
c) $x = 0$;	c) else:	Anweisg.
if a):	always:	
Anweisg.	Anweisg.	
if b):	if a)$\vee$ b):	Anweisg.
Anweisg.	always:	
if c):		
Anweisg.		
if a)$\vee$ b)$\vee$c):		

***) Ist dies nicht der Fall, überdecken sich also zwei Falldefinitionen teilweise, so wird (technisch) der Durchschnitt stets dem zu erst niedergeschriebenen Fall zugeschlagen und aus dem an zweiter Stelle niedergeschriebenen Fall gestrichen.

III. Gliederungs- und Ablauf-Anweisungen

1) Anweisungsnummern N :
dienen zur Kennzeichnung irgendwelcher Anweisungen.

Form: $\zeta_1 \ldots \zeta_k$:

Beispiele: 1:
15:

2) Sprünge
geben eine von der Reihenfolge der Niederschrift abweichende Fortsetzung der Rechnung an.

Form: go to N ;

N ist die Anweisungsnummer, bei der fortzufahren ist.

Beispiel: go to 1;

3a) Abschnittsbezeichnungen A
dienen der Kennzeichnung von Programmabschnitten zum Zwecke der Gliederung und der mehrfachen Verwendung an verschiedenen Stellen des Formelprogramms.

Form: ein Buchstabe, gefolgt von beliebigen Buchstaben und/oder Ziffern — Schlußzeichen :

λ λ/ζ

Beispiele: a:
pivotsuche:
konv 1:

3b) Abschnittsende
kennzeichnet das Ende eines Abschnitts

Form: end A !

Beispiele: end A !
end pivotsuche!

3c) Abschnitte A

Die Gesamtheit der Anweisungen von einer Abschnittsbezeichnung bis zum zugehörigen Abschnittsende bildet einen Abschnitt. Abschnittsbezeichnung und Abschnittsende stellen also eine zusammengesetzte Anweisung mit einer zwischen den beiden liegenden Leerstelle dar, über die beliebige weitere Anweisungen, darunter auch zusammengesetzte Anweisungen wie Fallunterscheidungen, Abschnitte und Schleifen (s. III 5c) eingesetzt werden können.

Bei der Bildung von Abschnitten besteht völlige Freiheit, nur dürfen dadurch nicht andere zusammengesetzte Anweisungen zerrissen werden.

4a) Abschnittsaufrufe
gestatten die vollständige Einfügung eines einmal niedergeschriebenen Abschnitts an beliebiger Stelle. Die Einfügung ist dabei logisch als Ausführung und nicht als Abschrift des aufgerufenen Abschnitts zu betrachten, so daß insbesondere der Aufruf eines Abschnitts aus einem Zweig einer in ihm selbst enthaltenen Fallunterscheidung möglich ist.

Form: $\mathbb{A}$;

Beispiele: $\mathbb{A}$;
konvergenz;
pivotsuche;

Abschnittsaufrufen können vorangehen

4b) Substitutionsanweisungen,
die es gestatten, in einem oder mehreren aufgerufenen Abschnitten beliebige Größen durch andere zu ersetzen.

Form: for $\mathfrak{B}^{(1)} := \mathfrak{A}^{(1)}; \ldots; \mathfrak{B}^{(n)} := \mathfrak{A}^{(n)}$! $\mathfrak{f}^{(1)} := \bar{\mathfrak{A}}^{(1)}; \ldots; \mathfrak{f}^{(m)} := \bar{\mathfrak{A}}^{(m)}$!
$\mathbb{A}^{(1)} := \bar{\mathbb{A}}^{(1)}; \ldots; \mathbb{A}^{(k)} := \bar{\mathbb{A}}^{(k)}$! .... usw..... :

unmittelbar anschließend Abschnittsaufrufe $\mathbb{A}^{(1)}; \ldots; \mathbb{A}^{(r)}$!

Die jeweils links von dem Definitionszeichen ":=" stehenden Symbole (Variable, Funktionen, Abschnittsbezeichnungen, Fallbezeichnungen) werden in den Anweisungen der aufgerufenen Abschnitte durch die rechtsstehenden Symbole ersetzt.

Diese Ersetzung ist nur für die betreffenden Aufrufe gültig. Sie bedeutet insbesondere für die Variablen nur eine Symbolauswechslung und nicht eine Eintragung der aus den Ausdrücken rechts sich ergebenden Zahlenwerte unter den links stehenden Variablennamen.

Die Auswechslung von Abschnittsbezeichnungen beschränkt sich selbstverständlich auf in den aufgerufenen Abschnitten vorkommende Aufrufe anderer Abschnitte.

Die Auswechslungen sind innerhalb der einzelnen Gruppen als simultan zu betrachten, so daß die Reihenfolge der Aufschreibung belanglos ist.

Beispiel: for x:=y; y:=x! P[x]:=(a·x+b)·x+c! konv:=konv 2:

5a) Variablenlauf-Anweisungen
weisen einer Variablen die Werte einer einfachen arithmetischen Progression zu und gestatten damit die Niederschrift von Rekursionen.

Form: for $\mathfrak{v}$:= $\underline{\mathfrak{a}}$ ... $\overline{\mathfrak{a}}$; step $\mathfrak{a}'$:

Die Angabe "step 1:" kann wegfallen, dann ist das Zeichen ";" hinter $\overline{\mathfrak{a}}$ durch ":" zu ersetzen.

Die Ausdrücke $\underline{\mathfrak{a}}$ für den Anfangswert, $\overline{\mathfrak{a}}$ für den Endwert und $\mathfrak{a}'$ für die Schrittweite von $\mathfrak{v}$ dürfen die Variable $\mathfrak{v}$ selbst nicht enthalten.

Die Laufanweisung weist der Variablen $\mathfrak{v}$ für die nachfolgenden Anweisungen der Reihe nach die Werte $\underline{\mathfrak{a}}$, $\mathfrak{a}+\mathfrak{a}'$, ..., $\mathfrak{a}+n\cdot\mathfrak{a}'$ mit $n=\left[\frac{\overline{\mathfrak{a}}-\underline{\mathfrak{a}}}{\mathfrak{a}'}\right]$ zu. Die Progression bricht also ab mit dem letzten Wert von $\mathfrak{v}$, der nicht größer als $\overline{\mathfrak{a}}$ ist.

Beispiele:
for i := 1...n:
for j := k...n·k; step k:

Die Folge der Anweisungen, für die $\mathfrak{v}$ die von der Laufanweisung vorgeschriebenen Werte annimmt, wird beendet durch die

5b) Fortschaltungsanweisung.
Sie veranlaßt die Erhöhung des laufenden Wertes von $\mathfrak{v}$, die Prüfung auf Überschreitung des Endwertes und bei negativem Ausfall

die Wiederholung der auf die Laufanweisung folgenden Anweisungen. Ist der Endwert von ⓥ überschritten, so wird die auf die Fortschaltung folgende Anweisung ausgeführt.

Form: vary ⓥ ;

Laufanweisung und Fortschaltung sind Anfang und Ende einer

5c) Schleife,
die eine weitere zusammengesetzte Anweisung mit einer Leerstelle darstellt. Die Leerstelle kann wie bei den Abschnitten mit beliebigen Anweisungen besetzt sein.

Ein Sprung in das Innere einer Schleife unter Umgehung der Laufanweisung ist jedoch unsinnig, wenn in diesem Falle der Wert der Variablen ⓥ nicht definiert ist. Sprünge und Satznummern im Inneren einer Schleife dürfen also nur von Punkten im Inneren der gleichen Schleife ausgehen und können nur die Auslassung bzw. Ersetzung eines Teils der Anweisungen der Schleife bewirken.

Beispiel: for i:= 1...n:

p.x ÷ a ⇩ⁱ ⇧ⁱ ⇒ p

vary i

IV. Definitionsanweisungen

1a) Funktions-(Funktional-) Definitionsanweisung
Sie kennzeichnet die einer bestimmten Funktion (Funktional) entsprechende Rechenvorschrift.

Form: Funktions-
Funktional- Bezeichnung, gefolgt von einem Doppelpunkt

f: g:

Beispiele: P[x]:

gamma[t]:

simpson[f[x]; a;b]:

Durch die Definitionsanweisung wird festgelegt:

a) welche der in der Rechenvorschrift auftretenden Variablen als Argumente der Funktion gelten,

b) welche der in der Rechenvorschrift auftretenden Funktionen bzw. indizierten Variablen als Argumentfunktionen bzw. Argument-Variablensätze des Funktionals gelten und wie demnach die Argumentfunktionsstellen des Funktionals besetzt werden dürfen.

Die Argumentstellen sind daher in der Definitionsanweisung mit einfachen Variablen zu besetzen, die Argumentfunktions-Stellen entweder mit einer Funktionsbezeichnung oder einer indizierten Variablen mit den in der Rechenvorschrift hierfür angegebenen Argument- bzw. Indexvariablen als Dummy-Argumenten bzw. Indizes.

Der Funktions-(Funktional-) Begriff ist hier so gefaßt, daß jede Funktion (Funktional) für feste Argumente als Funktionswert eine reelle Zahl definiert. Die Rechenvorschrift enthält daher eine Variable V als Deckname für das Resultat. Diese Variable wird als Funktionswert gekennzeichnet durch die

1b) Funktionswert-(Funktionalwert-) Zuweisung.

Form: $f := V;$ $g := V;$

Beispiel: $P[x] := p;$

1c) Funktionsdefinition

Die der Funktion (Funktional) entsprechende Rechenvorschrift folgt unmittelbar auf die Definitionsanweisung und wird abgeschlossen durch die Wertzuweisung.

Definitionsanweisung und Wertzuweisung stellen also wieder eine zusammengesetzte Anweisung mit einer Leerstelle dar, die durch beliebige Anweisungen besetzt werden kann.

Besteht die Rechenvorschrift aus einem einzigen Ausdruck, so können Definitionsanweisung, Rechenvorschrift und Wertzuweisung zusammengezogen werden zur

Funktionsdefinitionsformel

Form: $f := a;$

Die Definitionen unterscheiden sich aber insofern von allen anderen Anweisungen, als sie stets als außerhalb des eigentlichen Formelprogramms liegende Hilfsdefinitionen behandelt werden. Sie werden nur an Stelle der entsprechenden Funktionsbezeichnungen in arithmetischen Ausdrücken eingefügt.

Sie können an beliebiger Stelle im Formelprogramm niedergeschrieben werden und werden im Durchgang stets ignoriert.

Sprünge aus dem eigentlichen Formelprogramm in Funktionsdefinitionen und in umgekehrter Richtung sind daher sinnlos.

Beispiel:

$P[x]: a \boxed{\downarrow}_{0} \boxed{\uparrow} \Rightarrow p$

for i:= 1...n:

$p \cdot x + a \boxed{\downarrow}_{i} \boxed{\uparrow} \Rightarrow p$

vary i

$P[x] := p;$

2) Definitionsanweisungen für Bibliotheksprogramme brauchen im Augenblick nicht behandelt zu werden. Sie sollen jedenfalls in enger Anlehnung an IV 1. formuliert werden.

V. Stops und Signale

1) Die Anweisung "stop" bewirkt ein Anhalten der Rechnung. Sinngemäß sollen andere Signalabgaben nach außen formuliert werden.

VI. Daten-Ein- und -Ausgabe-Anweisungen

Ein- und Ausgabe-Anweisungen für Daten sollen grundsätzlich in der Form von Funktionen, die sich auf Bibliotheksprogramme stützen, formuliert werden.

C. Technische Anweisungen

Um die Arbeit des Formelübersetzers technisch zu erleichtern und effektiv zu gestalten, müssen eine Reihe logisch an sich unnötiger Anweisungen gegeben werden. Solche sind

Dimensionsangaben für indizierte Größen

Angabe über Mehrfachbelegung von Speicherplätzen

Häufigkeitsangaben und/oder Segmentierungen für Variable und Programmteile im Hinblick auf Schnellspeicherbelegung.

D. Bedienungsmaßnahmen

Bedienungsmaßnahmen für die Formelübersetzung sollen bei glattem Verlauf nicht nötig sein. Jedoch soll eine Anzeige von Verstößen gegen die Formelsprache, soweit technisch möglich, erfolgen. Behebbare Irrtümer sollen unmittelbar korrigiert werden. Die Durchführung eines Probelaufs für ein erzeugtes Programm unter Spezifizierung von Probelaufsdaten soll ohne weiteres möglich sein.