

Über den Plankalkül*

Von K. Zuse, Bad Hersfeld

Elektron. Rechenanl. 1 (1959) 2
Manuskripteingang: 18.3.1959

Der Plankalkül entstand im Jahre 1945 mit dem Ziel, eine einheitliche Formelsprache für alle rechnerischen Problemstellungen zu schaffen.

Die verwendeten Angaben oder Informationseinheiten werden nach logistischen Gesichtspunkten streng aus Elementareinheiten aufgebaut. Eine möglichst weitgehende Anlehnung an Formalismen der Logistik wird dabei angestrebt.

Besondere Formelzeichen wie das Ergibt-Zeichen und das Zeichen für bedingte Planteile usw. werden angeführt und erläutert. Verschiedene Anwendungsbeispiele werden erwähnt und aus dem Gebiet der Formulierung von Aufgaben des Schachspieles wird ein besonderes Beispiel herausgegriffen und im einzelnen besprochen.

The „Plankalkül“ (program calculus) originated in 1945 with the aim to provide a uniform language of formulae adapted to all kinds of calculating problems.

The data or information units used are assembled from elementary units according to strictly logistic view points with the intention to refer to the logistic formalisms as far as possible.

Special signs of formulae as for instance the „results-in“-sign and the sign for conditioned sub-routines and so on, have been introduced and will be discussed. Several examples of application are mentioned. A special example is selected out of the field of the formulation of problems of chess and discussed in detail.

*ZIA 0084. ZuP 040/004. Version 1, Abbildungen fehlen. Durchgesehen von R. Rojas, G. Wagner, L. Scharf

Einleitung

Durch theoretische Untersuchungen und praktische Arbeiten mit programmgesteuerten Rechengerten wurde der Verfasser etwa in den Jahren 1937 bis 1945 zu der Erkenntnis geföhrt, daß der Begriff „Rechnen“ wesentlich über das Zahlenrechnen hinaus erweiterungsfähig ist. Aus der damaligen Perspektive heraus erschien es sinnvoll, verschiedene Typen von programmgesteuerten Rechengerten zu bauen, z.B. vorwiegend numerische, vorwiegend logistische und solche, die vorwiegend der Programmfertigung dienen sollten. Wir wissen heute, daß eine solche Einteilung nicht unbedingt zweckmäßig ist. Moderne elektronische Geräte sind in der Lage, alle derartigen Aufgaben zu bewältigen. So kann z.B. das Gerät Z22 sowohl Programme rein numerischer Art mit verschiedenen Variationen der arithmetischen Grundoperationen als auch logistische Abteilungen im weitesten Sinne ausführen und schließlich die Aufgabe der Programmfertigung übernehmen, die heute unter dem Namen „Compiler“ oder „Superprogramm“ oder dergleichen läuft. Das Arbeiten mit derartigen Geräten erfordert „rechenmaschinengerechte“ Formulierung der verschiedenartigsten Probleme. Die seinerzeit zur Verfügung stehenden Formalismen reichten hierzu nicht aus. Der Verfasser hielt es daher für erforderlich, zunächst einmal eine Formelsprache zu schaffen, welche geeignet ist, derartige Zusammenhänge allgemeinsten Art bis in alle Einzelheiten exakt auszudrücken. Die spätere Entwicklung ist einen etwas anderen Weg gegangen. Man baute zunächst verschiedenartige Geräte mit speziellen Formelsprachen (Programmcode).

Mit der Zeit zeigten sich dann aus der Praxis heraus ähnliche Tendenzen der Formulierung einer allgemeinen Formelsprache wie sie z.B. heute in Form des „ALGOL“ vorgeschlagen wird.

Dagegen entstand der Plankalkül als eine theoretische Arbeit auf dem Papier zu einer Zeit, als noch wenig Erfahrungen vorlagen. Das ergab zwar allgemeingültige und exakte Lösungen, mit denen grundsätzlich jedes Problem formulierbar ist, jedoch lag diesem ersten Versuch noch eine gewisse Schwerfälligkeit zugrunde, die das praktische Arbeiten mit dem Kalkül erschwert.

Ziel dieses Aufsatzes ist es nicht, den Plankalkül in der damals geschaffenen Form zu propagieren, sondern einen Diskussionsbeitrag zur Verfeinerung und Ergänzung heute bereits benutzter Formelsprachen zu geben. Gewisse Zeichen, wie z.B. das Ergibt-Zeichen, konnten sich bereits weitgehend durchsetzen.

Die Darstellungsweise des Plankalküls

Der Plankalkül geht von folgender Definition des Rechnens aus:

„Rechnen heißt, aus gegebenen Angaben¹ nach einer Vorschrift neue Angaben bilden“ [1].

Die einfachste Angabe ist der Ja-Nein-Wert. Alle anderen Informationseinheiten lassen sich durch Ja-Nein-Werte in mehr oder weniger komplizierter Form aufbauen. Ebenso sind sämtliche Rechenoperationen zerlegbar in Elementaroperationen zwischen Elementarangaben.

Dementsprechend enthält der Plankalkül strenge Formulierungen zunächst der Angabenarten und ihrer Strukturen.

Es wird unterschieden:

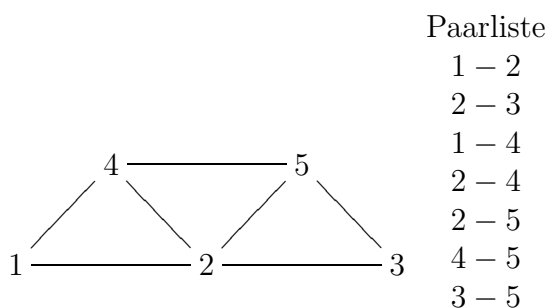
1. Die Angabenstruktur

Das ist die Art des Aufbaues der betreffenden Angaben aus elementaren Angaben. Die Angabenstrukturen werden durch streng aufgebaute Strukturformeln aus den einfachsten Angaben zusammengesetzt, z.B.:

S_0	Ja-Nein-Wert
$S_1 \cdot n = n \times S_0$	n -stellige Folge von Ja-Nein-Werten
$S_1 \cdot 4$	Tetrade
σ	allgemeines Zeichen für Angabe beliebiger Struktur
2σ	Paar von Werten
$m \times 2\sigma$	Paarliste

Auf diese Weise lassen sich z.B. beliebige technische Strukturen, wie Netzwerke, Stabwerke, Konstruktionen usw., darstellen.

Beispiel: Stabwerk



2. Angabenbeschränkungen

Die Kennzeichnung der Strukturen ist strenggenommen nicht immer ausreichend, da in gewissen Fällen für eine bestimmte Angabenart nicht sämtliche Struktu-

¹Anstelle des Ausdruckes „Angabe“ hat sich heute der Ausdruck „Information“ eingebürgert.

ren definiert sind (z.B. bei der Darstellung der Dezimalziffern durch Tetraden: von 16 Möglichkeiten sind nur 10 definiert). Es sind dann Beschränkungsformeln erforderlich.

3. Angabentyp

Angaben gleicher Struktur können verschiedene Bedeutung haben (etwa X- und Y-Koordinaten). Hierfür wird der Begriff des Angabentyps eingeführt.

4. Angabenart

Die verschiedenen Kennzeichen für eine Angabe (Struktur, Beschränkung, Typ) können unter einem „Angabenart-Zeichen“ zusammengefaßt werden.

Das Angabenart-Zeichen kann je nach Situation mehr oder weniger als das Struktur-Zeichen aussagen. So lassen sich z.B. Zahlen in Rechenmaschinen durch verschiedene Strukturen darstellen (rein dual, durch Tetraden, gleitendes Komma, verschiedene Vorzeichendarstellung usw.).

Dies interessiert jedoch im Rahmen einer algebraischen Formel nicht. Man kann daher ein allgemeines Angabenart-Zeichen für „Zahl“ überhaupt einführen.

Diesem Zeichen können dann mehrere Strukturen zugeordnet werden. Umgekehrt können die gleichen Strukturen verschiedenen Angabentypen und somit auch Angabenarten zugeordnet sein, z.B. in bezug auf ihre Bedeutung (physikalische Dimensionen usw.).

Im Rahmen einer Formel werden die Angaben durch Indizes in verschiedenen Zeilen gekennzeichnet.

Neben der Hauptzeile, welche die Formel im wesentlichen in der traditionellen Form enthält, wird eine zweite Zeile (V) für den Variablen-Index, eine dritte für den Komponenten-Index (K) und eine vierte für den Struktur-Index (S) eingeführt. Die letztere braucht, strenggenommen, nicht immer ausgefüllt zu werden, dient aber wesentlich zur Erleichterung des Verständnisses einer Formel. Die Zeilen werden Durch Vorsetzen der zugeordneten Buchstaben (V, K, S) gekennzeichnet.

Beispiele:

$\begin{array}{l} V \\ V \\ K \\ S \end{array} \left \begin{array}{l} V \\ 3 \\ m \times 2 \times 1 \cdot n \end{array} \right.$	<p>Die Variable V_3 ist eine Paarliste von m Paaren der Struktur $2 \cdot 1 \cdot n$ und soll als Ganzes in die Rechnung ein- gehen.</p>
$\begin{array}{l} V \\ K \\ S \end{array} \left \begin{array}{l} V \\ 3 \\ i \\ 2 \times 1 \cdot n \end{array} \right.$	<p>Von der Paarliste V_3 soll das i.Paar genommen werden (Struktur $2 \cdot 1 \cdot n$). (i kann dabei ein laufender Index sein.)</p>
$\begin{array}{l} V \\ K \\ S \end{array} \left \begin{array}{l} V \\ 3 \\ i \cdot 0 \\ 1 \cdot 0 \end{array} \right.$	<p>Von dem i.Paar der Paarliste V_3 soll das Vorderglied (er-stes Element des Paares) genommen werden (Struktur $1 \cdot n$).</p>
$\begin{array}{l} V \\ K \\ S \end{array} \left \begin{array}{l} V \\ 3 \\ i \cdot 0 \cdot 7 \\ 0 \end{array} \right.$	<p>Von dem Vorderglied des i.Paares der Paarliste V_3 soll der Ja-Nein-Wert Nr. 7 genommen werden (Struktur $S_0 = \text{Ja-Nein-Wert}$).</p>

Beim Beispiel des Stabwerkes bedeutet für $i = 4$:

V	3	die gesamte Paarliste des Stabwerkes
V	3	die Kennzeichnung des Stabes 2 – 4(4. Paar der gegebenen Liste)
V	4	
V	3	den 1. Knotenpunkt des Stabes 2 – 4, also Knotenpunkt 2
$i \cdot 0$		
V	3	den 7. Ja-Nein-Wert aus der Angabe, welche den Knotenpunkt 2
$i \cdot 0 \cdot 7$		kennzeichnet

Umgekehrt zu dem Prozeß der Komponentenbildung können Angaben zusam-
mengefaßt werden:

$\begin{array}{l} V \\ K \\ S \end{array} \left \begin{array}{l} (V, V) \Rightarrow V \\ 5 \quad 6 \quad 7 \\ \sigma \quad \sigma \quad 2\sigma \end{array} \right.$	<p>2 Angaben V_5 und V_6 von der allgemeinen Struktur σ werden zu einem Paar der Struktur 2σ zusammengefaßt.</p>
---	--

Der Struktur-Formalismus gibt lediglich Auskunft über den komponentenmäßi-
gen Aufbau der Informationseinheiten. Er kann, wie bereits oben erwähnt, durch
Beschränkungsformeln und Zusammenfassung verschiedener Strukturen zu Infor-
mationsarten gleicher Bedeutung erweitert werden.

Die Rechenpläne selbst erhalten als ganze ebenfalls eine strenge Kennzeichnung durch Indizes, die eventuell in mehrere Komponenten zerfallen; so ist z.B. P3.7 das Programm 7 der Programmgruppe oder des Gebietes 3.

Es werden weiterhin die innerhalb eines Rechenplanes auftretenden Informationsarten in ihrem Verhältnis zu dem Rechenplan selbst unterschieden:

1. Eingangswerte (Variable) V mit Index,
2. Zwischenwerte Z mit Index,
3. Resultatwerte R mit Index,
4. Konstanten C mit Index.

Sämtliche auftretenden Informationen können von verschiedenster Struktur sein. Die Eingangs- und Resultatwerte gehören zu den „Randwerten“, welche die Verbindung des betreffenden Rechenplanes mit anderen Teilen der gesamten Rechenvorschrift bilden.

Der Randauszug

Anzahl und Struktur dieser Randwerte werden in einem „Randauszug“ wiedergegeben, welcher im übrigen über den Inhalt des Rechenplanes nichts aussagt.

$$\begin{array}{l|l} & R(V_0, V_1) \Rightarrow (R_0, R_1, R_2) \\ V & \begin{array}{ccc} 0 & 0 & 0 \\ & & 1 & 2 \end{array} \\ S & \begin{array}{ccc} m \cdot \sigma & n \cdot \sigma & (m+n)\sigma \\ & & 1 \cdot n & 0 \end{array} \end{array}$$

Dieser Randauszug besagt:

Es gehen 2 Variable V_0 und V_1 von der Struktur je einer Liste ($m \times \sigma$ bzw. $n \times \sigma$) in die Rechnung ein. Aus diesen werden 3 Resultatwerte gebildet:

1. eine neue Liste R_0 , Struktur $(m+n) \times \sigma$ (z.B. die Mischung der beiden gegebenen Listen nach einer bestimmten Vorschrift),
2. eine Dualzahl R_1 (z.B. Summe sämtlicher Einzelwerte),
3. eine Aussage R_2 , welche irgendeine Eigenschaft der gegebenen Liste oder der Gesamtliste kennzeichnet (z.B. „Die Liste enthält keine sich wiederholenden Glieder“).

Resultatwerte eines Rechenplanes, welche in einem anderen Planteil weiterverwendet werden sollen, können mit Hilfe des Randauszuges eindeutig gekennzeichnet werden. Ist z.B. der oben erwähnte Randauszug dem Rechenplan P3.7 zugeordnet, so hat der Ausdruck 4 folgende Bedeutung:

$$\begin{array}{l|l} & R3.7 (Z_{9}, Z_{11}) \Rightarrow Z \\ V & 1 \quad 9 \quad 11 \quad 12 \quad \text{d.h.:} \\ S & 1.n \quad m.\sigma \quad n.\sigma \quad 1.n \end{array}$$

„Wende auf die Zwischenwerte (Listen) Z_9 und Z_{11} des gerade behandelten Oberplanes die Rechenvorschrift P3.7 an. Das Resultat R_1 dieses Planes ergibt den Zwischenwert Z_{12} , mit welchem weiter operiert werden kann.“

Jeder in sich abgeschlossene Rechenplan wird durch ein Schlußzeichen FIN beendet.

Die Einführung von Schlußzeichen verschiedenen Grades (Fin^2) bedeutet das Zurückschalten auf höhere Ablaufstufen des Gesamtplanes; jedoch ist die Benutzung von Konnektoren im allgemeinen übersichtlicher.

Das Ergibt-Zeichen

Besondere Beachtung verdient die Verwendung des Ergibt-Zeichens. Rechenpläne werden durch einzelne explizite Rechenplangleichungen gebildet. In diesen steht links vom Ergibt-Zeichen ein Ausdruck, in dem die Eingangswerte oder bereits definierte Zwischenwerte enthalten sind, und rechts der neu zu bestimmende Zwischenwert bzw. Resultatwert. Das Ergibt-Zeichen kann in speziellen Fällen gleichbedeutend sein mit dem Gleichheitszeichen der Mathematik oder dem Äquivalenzzeichen des Aussagenkalküls, jedoch gelten folgende Regeln:

1. Das Ergibt-Zeichen deutet stets an, daß der rechts davon stehende Wert errechnet werden soll. Es ist also niemals selbst eine Rechenoperation. Gleichheitszeichen im Rahmen einer Plangleichung bedeuten die Rechenoperation „Vergleiche“ mit dem Ergebnis einer logistischen Aussage (Ja-Nein-Wert).
2. Treten rechts und links des Ergibt-Zeichens die gleichen Zeichen auf, so sind diese nicht gleichen Werten zugeordnet.
Beispiel: $Z + 1 \Rightarrow Z$.
3. Tritt das gleiche Zeichen in mehreren Plangleichungen rechts des Ergibt-Zeichens auf, so gilt stets die letztere Bestimmung des Wertes.

Bedingte Planteile werden durch das Symbol $\dot{\rightarrow}$ gekennzeichnet. Zur Unterscheidung vom Zeichen \rightarrow für die Implikation im Aussagenkalkül wird der Bedingungs Pfeil durch einen Punkt ergänzt. Das Zeichen wird zwischen den Ansatz für die zu erfüllende Bedingung und den im Falle des Zutreffens der Bedingung durchzurechnenden Planteil gesetzt.

Links vom Bedingt-Zeichen muß also ein Ja-Nein-Wert bzw. ein aussagenlogischer Ausdruck stehen, rechts eine Rechenvorschrift.

Beispiel: $V = V \dot{\rightarrow} (V + V \Rightarrow V)$
 $\quad \quad \quad 3 \quad 5 \quad \quad 6 \quad 8 \quad 9 \quad \cdot$

Die Benutzung des Ergibt-Zeichens und des Bedingungs-Zeichens hat sich inzwischen weitgehend eingebürgert. Allerdings werden die Regeln der Anwendung nicht immer so streng eingehalten, wie es der Plankalkül vorschreibt. Mit Hilfe des Bedingungs-Zeichens ist die Darstellung zyklischer Programme leicht möglich, wobei sich die inzwischen allgemein eingeführten Regeln mit Abzählen von Indizes usw. im Plankalkül leicht formulieren lassen.

Logistische Begriffe im Plankalkül

Der Plankalkül lehnt sich in formaler Hinsicht möglichst weitgehend an Begriffe an, die sich in der Logistik als brauchbar erwiesen haben:

„Alle“, „Es gibt“, „Derjenige welcher“, „Diejenigen welche“, „Das Nächste“, „Anzahl von“ mit den Symbolen

$$\begin{aligned} (x)(x \in V_0 \rightarrow R(x)) \\ (Ex)(x \in V_0 \wedge R(x)) \\ \acute{x}(x \in V_0 \wedge R(x)) \\ \hat{x}(x \in V_0 \wedge R(x)) \\ \hat{\hat{x}}(x \in V_0 \wedge R(x)) \\ \mu x(x \in V_0 \wedge R(x)) \\ N(x)(x \in V_0 \wedge R(x)) \end{aligned}$$

Hierin bedeutet der Ausdruck $x \in V_0$, daß die laufende Variable x der durch die Liste V_0 gekennzeichneten Menge von Gliedern angehören soll und daß durch $R(x)$ eine bestimmte Eigenschaft R , die im einzelnen Falle näher zu kennzeichnen wäre, gefordert wird.

Die Anwendung des Operators \acute{x} bedingt, daß es genau ein Glied mit der Eigenschaft $R(x)$ in der Menge V_0 gibt.

Der Operator \hat{x} erfordert im Plankalkül eine Variation $\hat{\hat{x}}$, je nachdem, ob aus der gegebenen Menge ein Auszug gemacht werden soll, der die Elemente mit der Eigenschaft $R(x)$ je nur einmal enthält, oder in der gleichen Zahl von Wiederholungen wie in der gegebenen Menge V_0 .

Der Operator μx hat große Vorteile bei der systematischen Untersuchung einer sich evtl. in ihrem Umfang laufend ändernden Liste auf Glieder einer bestimmten Eigenschaft und Verarbeitung derselben. Entgegen der Regel der Logistik (Hilbert) gilt jedoch die Regel: Suche das nächste Glied der Eigenschaft R . Gibt es kein solches, so gehe zum nächsten Planteil über.

Es können ferner mit Vorteil verschiedene Begriffe des Relationenkalküls benutzt werden: z.B. „Feld einer Relation“, „Vorbereich“, „Nachbereich“.

Ausgearbeitete Programme des Plankalküls

Es seien jedoch noch kurz die Gebiete erwähnt, für welche seinerzeit Programme im Plankalkül ausgearbeitet wurden:

1. Allgemeiner Rechenplan betreffend das Rechnen mit Strukturen allgemeiner Art, wie Folgen von Ja-Nein-Werte, Listen, Paarlisten, Relationen usw.
2. Rechenpläne der Zahlenrechnung, Aufstellung von streng detaillierten Programmen für verschiedene arithmetische Operationen mit und ohne gleitendem Komma.
3. Operationen mit algebraischen Ausdrücken. Hierunter wird die Behandlung von Formalismen und das Bearbeiten, Umformen usw. von formelmäßigen Ausdrücken verstanden.
4. Probleme der Schachtheorie. Die Bearbeitung dieses Problems erwies sich als besonders geeignet, um den entwickelten Kalkül in bezug auf seine Vielseitigkeit zu erproben. Es wurde das vollständige Programm der Spielkontrolle entwickelt.

Ein Beispiel aus der Schachtheorie

Als Beispiel sei kurz auf die Schachtheorie eingegangen. Zunächst ist der Aufbau der auftretenden Angabenarten interessant.

S_0

Ja-Nein-Wert

$S_1 \cdot n$

n -stellige Folge von Ja-Nein-Werten

$A1$	$S1 \cdot 3$	= Koordinate
$A2$	$2 \times A1$	= Punkt (z.B.: L00, 00L entspricht Punkt e2 in üblicher Darstellung)
$A3$	$\begin{pmatrix} S1 \cdot 4 \\ B3 \end{pmatrix}$	= Besetzt-Angabe (z.B.: 00L0, Weißer König)
$A4$	$(A2, A3)$	= Punkt-besetzt-Angabe (z.B.: L00, 00L; 00L0 „Punkt e2 mit weißem König besetzt“)
$A5$	$64 \times A3$	= Feldbesetzung: C5 Anfangslage (Aufzählung der Besetzung der 64 Punkte in fester Reihenfolge)
$A6$	$64 \times A4$	= Feldbesetzung mit Punktangabe, C6 Anfangslage
$A7$	$12 \times S1 \cdot 4$	= Anzahlliste der Steine; C7 Anfangslage (Gibt an, wieviel Steine von jeder Sorte auf dem Feld sind, z.B. für Berechnungen wichtig).
$A9$	$(A5, S0, S1 \cdot 4, A2)$	= Spielsituation; C9 Anfangssituation (Feldbesetzung [A5]; Angabe, ob Weiß oder Schwarz am Zuge [S0]; Angaben über Rochademöglichkeiten [4 Ja-Nein-Werte] Angabe der Punkte mit den Möglichkeiten, „en passant“ zu schlagen).
$A10$	$(A6, S0, S1 \cdot 4, A2)$	= Spielsituation mit Punktangabe; C10 Anfangslage
$A11$	$(A2, A2, S0)$	= Zugangabe (zwei Punktangaben, gesetzt von ... nach ... Ein Ja-Nein-Wert „Es wird geschlagen“).

Weitere Konstanten:

C 0.1 Wertigkeitstabelle der Steine

C 0.2 Steinaufzählung

A3 ist begrenzt auf 13 Möglichkeiten (12 Steinsorten und 0 für unbesetzt).

Der Aufbau der Programme ist wie folgt entwickelt:

1. Geometrie des Spielfeldes;
Einteilung in Zonen, Aussagen über Lage eines Punktes usw., Aussagen über Lage zweier Punkte zueinander (z.B. Springerrelation).
2. Programme unter Berücksichtigung der Besetzung der Punkte mit Steinen auf Grund der Feldbesetzung (A4).

3. Allgemeine Aussagen über Feldbesetzung, z.B. „Die Besetzung ist möglich“ (Bewertungsformeln); Aussagen über Zugmöglichkeiten für einzelne Steine unter Berücksichtigung der Aufdeckungen von Schach usw.
4. Schach-matt und Patt-Bedingungen.
5. Einführung der Spielsituation (A5); Ergänzung der Feldbesetzung durch Angaben über
 - (a) weiß oder schwarz am Zuge,
 - (b) angaben über Rochaden,
 - (c) Angaben über die Möglichkeiten „en passant“ zu schlagen.Aufstellung der Bedingungen für die Möglichkeit der Rochaden usw.
6. Einführung der Zugangabe und des Spielverlaufs; Bildung der neuen Situation aus der alten und Zugangabe.
7. Vollständige Spielkontrolle; Eingangswerte: Aufstellung der Züge eines Spielverlaufs, Resultat: „Spiel entspricht den Spielregeln.“

Der Aufbau der Programme für das Schachspiel wurde seinerzeit an dieser Stelle abgebrochen. An sich beginnen hier die eigentlich interessanten Schachprobleme (Vorschriften zur Ermittlung günstiger Züge). Jedoch ging es dem Verfasser nicht um eine spezielle „Schachtheorie“, sondern um den Aufbau eines Formalismus, der allen auftretenden Situationen gewachsen ist.

Aus der Fülle der Programme des Schachspiels sei ein typisches Beispiel gebracht, und zwar das Programm für: „Der weiße König kann einen Zug machen, ohne dabei in Schach zu kommen.“

P 148

$$\begin{array}{l|l} & R(V) \Rightarrow R148 \\ V & 0 \quad 0 \\ A & 5 \quad 0 \end{array} \quad (1)$$

$$\begin{array}{l|l} & \acute{x} \left[(x \in V) \wedge (x = L0) \right] \Rightarrow Z \\ V & 0 \quad 0 \\ K & \quad \quad 1 \\ A & 4 \left[\begin{array}{cc} 5 & 3 \end{array} \right] \quad 4 \end{array} \quad (2)$$

$$\begin{array}{l|l} & (Ex) \left[\begin{array}{cccc} (x \in V) \wedge R17(Z, x) \wedge (x = 0) \vee x \\ 0 & 0 & & \\ & 0 & 0 & 1 & 1.3 \\ 4 & 4 & 5 & 2 & 2 & 3 & 0 \end{array} \right] \\ V & \\ K & \\ A & \end{array} \quad (3)$$

$$\begin{array}{l|l} & \wedge \overline{Ey} \left[\begin{array}{cccc} (y \in V) \wedge y \wedge R128(v, y, x) \\ & 0 & & 0 \\ & & 1.3 & 0 & 0 \\ 4 & 4 & 5 & 0 & 5 & 2 & 2 \end{array} \right] \\ V & \\ K & \\ A & \end{array} \quad (4)$$

Die hierbei benutzten Unterprogramme sind:

$$\begin{array}{l|l} & R17(V, V) \\ V & 0 \quad 1 \quad \text{„die Punkte } V_0 \text{ und } V_1 \text{ sind benachbart.“} \\ A & 2 \quad 2 \end{array}$$

$$\begin{array}{l|l} & R128(V, V, V) \quad \text{„Bei der gegebenen Feldbesetzung } V_0 \text{ ist} \\ V & 0 \quad 1 \quad 2 \quad \text{der Zug von Punkt } V_1 \text{ nach Punkt } V_2 \text{ er-} \\ A & 5 \quad 2 \quad 2 \quad \text{laubt.“} \end{array}$$

Das Programm R128 ist verhältnismäßig kompliziert, da untersucht werden muß, welcher Stein auf Punkt V_1 steht, ferner ob der Punkt V_2 zu V_1 in einer solchen geometrischen Relation steht, daß der auf V_1 stehende Stein dorthin setzen kann, und schließlich muß untersucht werden, ob dazwischen liegende Punkte vorhanden sind und ob diese frei sind.

Erklärung der Formel P148 in Worten:

- (1) ist der Randauszug, der besagt, daß über eine Feldbesetzung (A5) eine Aussage gemacht werden soll.
- (2) Diejenige Punkt-Besetzt-Angabe (x), welche in der Liste der Spielbesetzung (V_0) enthalten ist, deren Komponente Nr. 1 = L0 ist (Zeichen für König in der Numerierung der Steintypen), ergibt den Zwischenwert Z_0 .
- (3) Es gibt in der Liste der Spielbesetzung (V_0) einen Punkt (x) der zu Z_0 (Punkt, auf dem der König steht) benachbart ist und der unbesetzt (= 0)

oder mit einem schwarzen Stein besetzt ist ($x_{1,3}$) (das bedeutet Ja-Nein-Wert Nr. 3 der Besetzt-Angabe x_1 ; dieser charakterisiert schwarze Steine).

- (4) Es gibt keinen weiteren Punkt, der mit einem schwarzen Stein besetzt ist, welcher nach Punkt x gesetzt werden kann.

Literatur

- [1] K. Zuse, „Ansätze einer allgemeinen Theorie des Rechnens“. (1943). Unveröffentlicht.
- [2] K. Zuse, „Plankalkül“, Theorie der angewandten Logistik. (1945). Unveröffentlicht.
- [3] K. Zuse, „Über den allgemeinen Plankalkül als Mittel zur Formulierung schematisch-kombinativer Aufgaben“. Archiv der Math. 1 (1948/49), Heft 6, S. 441 - 449.
- [4] K. Zuse, „Die mathematischen Voraussetzungen für die Entwicklung logistisch-kombinativer Rechenmaschinen“. Zeitschrift f. angew. Mathematik und Mechanik (ZAMM). 29 (1949), Heft 1/2, S. 36 - 37.