W. EHRENBERGER

Laboratorium für Reaktorregelung
und Anlagensicherung an der
Technischen Universität München

Nachweis der richtigen Arbeitsweise von Prozeßrechnerprogrammen

1 Zusammenfassung

Es werden zwei Möglichkeiten vorgestellt um die Sicherheit von Prozeßrechnerprogrammen nachzuweisen: der Statistische Test und die Programmanalyse. Für den Test muß das zu testende Programm in einen Rechner geladen werden, der mit einem anderen Rechner gekoppelt ist. Der zweite Rechner beliefert den zu Testenden mit einer sehr großen Anzahl von Eingangsdaten. Anhand eines Modells des zu testenden Programms stellt er fest, ob der Prüfling richtig reagiert. Die Analyse geht von den Abbildungen aus, die ein Programm ausführt. Sie beginnt beim Speicherabzug: zunächst wird der Gegenstand der Analyse in seine rein sequenziell zu durchlaufenden Teile zerlegt, dann festgehalten mit welchen Kernspeicherbereichen die einzelnen verkehren. Anhand der Verknüpfungen der Bereiche und des Unterscheidungsvermögens der einzelnen Befehle bezüglich der jeweiligen Eingangsdaten läßt sich eine Anzahl von Tests spezifizieren durch die das betreffende Programm dann vollständig auszutesten ist.

2 Einleitung

Prozeßrechner übernehmen immer wichtigere Aufgaben. In vielen Fällen darf ein Rechnereinsatz nur dann gestattet werden, wenn nachgewiesen ist, daß die verwendeten Programme extrem fehlerfrei sind. Dies gilt zum Beispiel für den Einsatz in Reaktorschutzsystemen. Ein solcher Einsatz von Rechnern wird zur Zeit für die Kernkraftwerke Brunsbüttel und Philippsburg geplant (Bild 1)

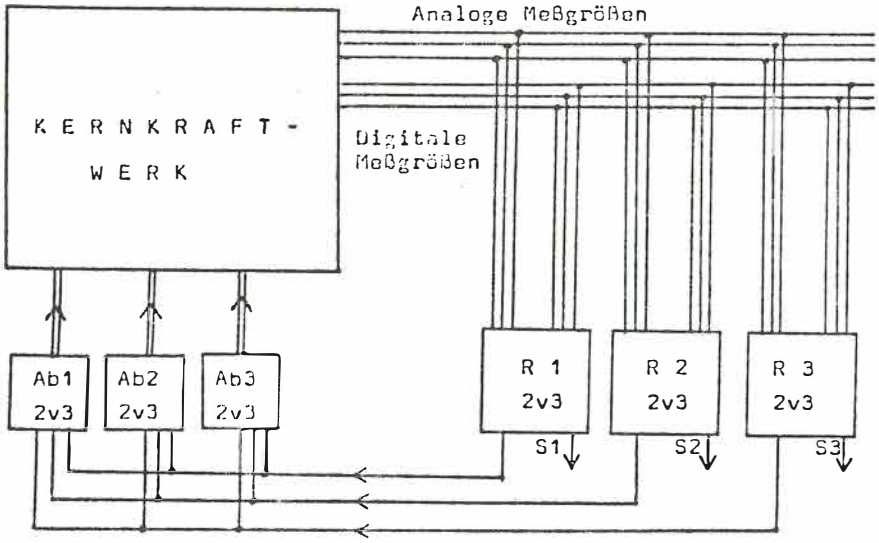


Bild 1 Übersicht über ein rechnerbestücktes Reaktorschutzsystem

R 1,2,3	Prozeßrechner 1,2,3
S 1,2,3	Schreibmaschine 1,2,3
Ab1,2,3	Abschaltvorrichtung 1,2,3
2v3	2 von 3 Auswahl

Ein Rechner hat einem fest verdrahteten System gegenüber den Nachteil, daß die Einzelfunktionen nicht ohne weiteres räumlich trennbar sind. Im Kernspeicher sind eine astronomische Zahl von Verkopplungen denkbar. Diese Zahl liegt bei

$$N = (2^{i.n.}m^{k.n})$$

wenn der Rechner i binäre und k analoge Eingänge hat; von den Analogwerten sollen m Zustände unterschieden werden können; der Rechner möge n Überwachungszyklen lang seine gesamten Einlesewerte als relevant ansehen und die Reihenfolge der Einlesungen möge Bedeutung haben können.

$$N > 10^{60}$$

kann durchaus auftreten.

Ein vollständiger Test aller Programme ist also unmöglich, falls man den Rechner als schwarzen Kasten betrachtet. Es gibt 2 Möglichkeiten, die geforderte Sicherheit nachzuweisen: den statistischen Test und den Kurztest nach stattgefundener Analyse, wobei die Analyse auch genau vorschreibt, wie zu testen ist.

3 Statistischer Test

In dem angesprochenen Fall des Einsatzes von Rechnern in Reaktorschutzsystemen ist eine Fehlerfreiheit der Programme von $1-10^{-6}$ zu fordern. Daraus folgt, daß, falls über die Programme weiter nichts bekannt ist, zwischen 10^6 und 10^7 Tests mit zufällig verteilten Werten gemacht werden müssen. Diese Testanzahl ist nur zu realisieren, wenn man weiß, daß das zu testende Programm sehr rasch auf seine Eingangsdaten reagiert, da die Testzeit sonst zu lang wird. Eine Reaktionszeit des Testobjekts von unter einer Sekunde dürfte unabdingbar sein. Zum zweiten muß für den Test ein weiterer Rechner zur Verfügung stehen. Bild 2 zeigt im groben den Testaufbau am LRA der TU München. Der Vorteil des Hybridrechners liegt darin, daß man den Prüfling leicht anschließen kann und daß man wegen der Übergabemöglichkeit von Analogwerten auch die dem eigentlichen Programm vorgeschaltete Hardware des Prozeßrechners und seiner Peripherie automatisch mit testet. Dies gilt auch für das Betriebssystem des Testobjekts.

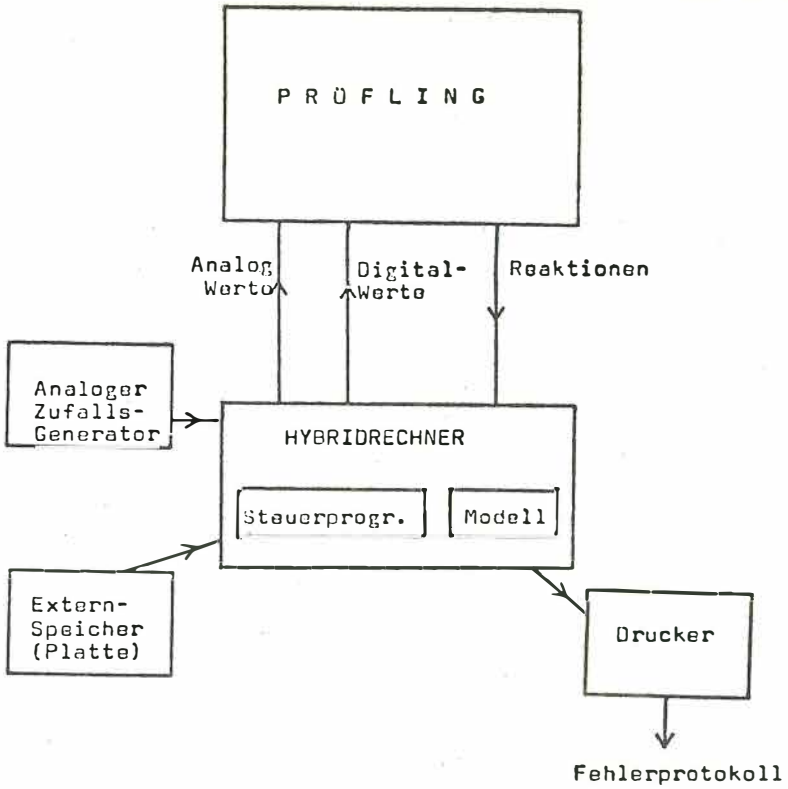


Bild 2 Übersichtsbild des Testaufbaus für den Test eines Prozeßrechners mit Hilfe eines Hybridrechners

Die Kopplung beider Rechner ist im vorliegenden Fall aus Zeit- und Verschleißgründen ohne mechanische Teile ausgeführt. Der Hybridrechner übergibt dem Prüfling jeweils verschiedene Abschaltfälle und kontrolliert anhand eines Modells der Programme des Prüflings dessen Reaktion (Bild 3). Die zu übergebenden Daten werden teilweise durch einen Zufallzahlengenerator ausgewählt, teils über Adressen festgelegt, die auf einem Externspeicher bereit liegen. Der Externspeicher wird vorher mit den festen Testdaten beschrieben. Diese festen Werte werden zur Auswahl von systematischen Anregungen für den Prüfling benützt. Daher erfordert jeder Testfall mindestens eine ganz bestimmte Reaktion des Schutzrechners. Die Veränderungen beziehen sich auf die analogen und digitalen Eingänge des Schutzrechners. Zur Aufbringung kleiner Schwankungen auf die Analogwerte dient ein Analoger Zufallszahlengenerator.

Der prüfende Rechner wird in einer anwenderorientierten Sprache programmiert (FORTRAN 300). Damit sinkt die Wahrscheinlichkeit gemeinsamer Fehler mit dem Schutzrechner. Zur Verkürzung der Rechenzeit verwenden die Programme des Prüfrechners besondere Routinen zur Übergabe von Parametern in Unterprogramme und zum Adressieren in Feldern.

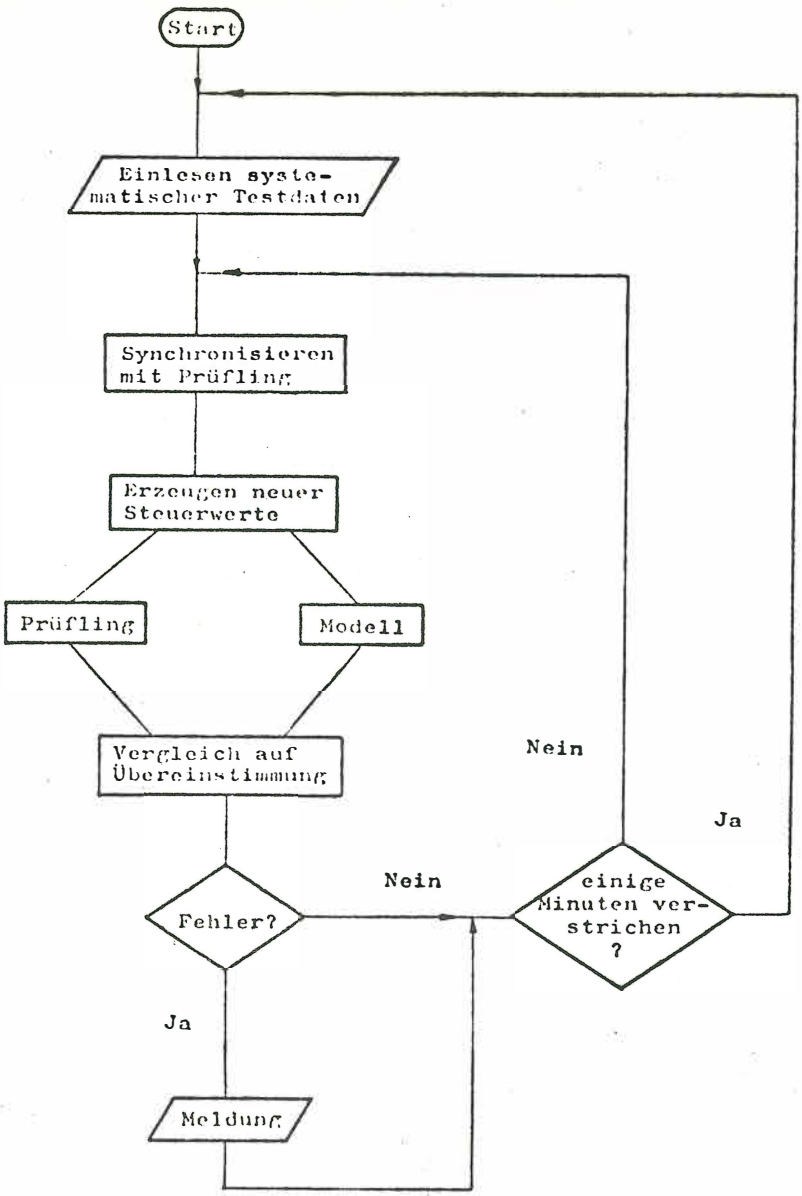


Bild 3

T E S T A B L A U F

Am Beispiel der Auswahl der Analogwerte für den Prüfling soll der Vorgang des Werteheraussuchens näher erläutert werden. Im vorliegenden Testfall sind 39 Analogmeßgrößen vorgesehen. Aus der Spezifikation des Programms im Prüfling geht hervor, daß 5 verschiedene Einzelwerte einer jeden Meßgröße eine Bedeutung haben können:

- ein Wert im Sollwertbereich
- je ein Wert nach unten bzw. oben leicht abweichend, sozusagen jenseits der Meldeschwellen
- je ein Wert jenseits des oberen bzw. unteren Grenzwerts.

Somit hat man insgesamt einhundertfünfundneunzig verschiedene Werte. Diese Werte werden vom Externspeicher in ein Feld eingelesen. Diesem großen Wertefeld gegenüber steht ein nur neununddreißig Werte umfassendes Feld in das an den Prüfling auszugebende Werte eingetragen werden. Dieses zweite Feld erhält in jedem Zwischen-Zyklus die neununddreißig Sollwerte des ersten Feldes.

Vor dem Arbeitszyklus wählt der Zufallszahlengenerator aus den einhundertfünfundneunzig Werten eine vorgehbare aber während jedes Testlaufs konstante Anzahl von Werten aus und überträgt sie in das kleinere Feld. Der Übertrag geschieht so, daß der ausgewählte Wert jeweils den Ruhewert ersetzt, der zu "seiner" Analoggröße gehört. Anschließend kommt die Auswahl der systematischen Anregungen. Hier werden, maximal 6, Ruhewerte ersetzt. Dann geht das Neununddreißig-Werte-Feld über den Analogrechner an den Prüfling. Am Analogrechner wird noch eine zufällige Schwankung dazu addiert, deren Amplitude jedoch die Hälfte des jeweiligen Bereichs der Einzel-Werte nicht überschreitet. Durch Verschieben der Grenzwerte im gesamten Analogenrechenbereich und durch Verändern der Bereichsgrößen ist es möglich, dem Prüfling, zufallsgesteuert, völlig beliebige Werte aus seinem möglichen Eingangswertebereich anzubieten.

Nach einer gewissen Zeit - eben dann, wenn der Prüfling mit Sicherheit reagiert haben muß - liest der Hybridrechner die Ausgänge des zu prüfenden Prozeßrechners. Er vergleicht die dort angebotenen Werte mit den Bitmustern die das Modell als die richtigen Ergebnisse errechnet hat. Ergibt sich eine Diskrepanz, so wird eine Fehlermeldung über den Drucker ausgegeben. Stimmen beide Ergebnisse überein, so wird der Zufallszahlengenerator erneut angestoßen und der Prüfling mit neuen Eingangswerten versorgt. Etwa in Minutenabstand holt der Hybridrechner von der Platte neue systematische Anregungswerte für den Prüfling. Wegen ihrer relativ langen Dauer dürfen die Plattenzugriffe nicht zu schnell aufeinander folgen.

4 Programmanalyse

Nachdem der Kernspeicherplatz in einem Rechner begrenzt ist, kann auch nur eine begrenzte Zahl von Algorithmen untergebracht werden. Es gilt herauszufinden, was getan werden muß, das heißt, welche Eingangsdaten dem Rechner angeboten werden müssen, damit alle möglichen Variationen und Kombinationen dieser Algorithmen ausprobiert werden.

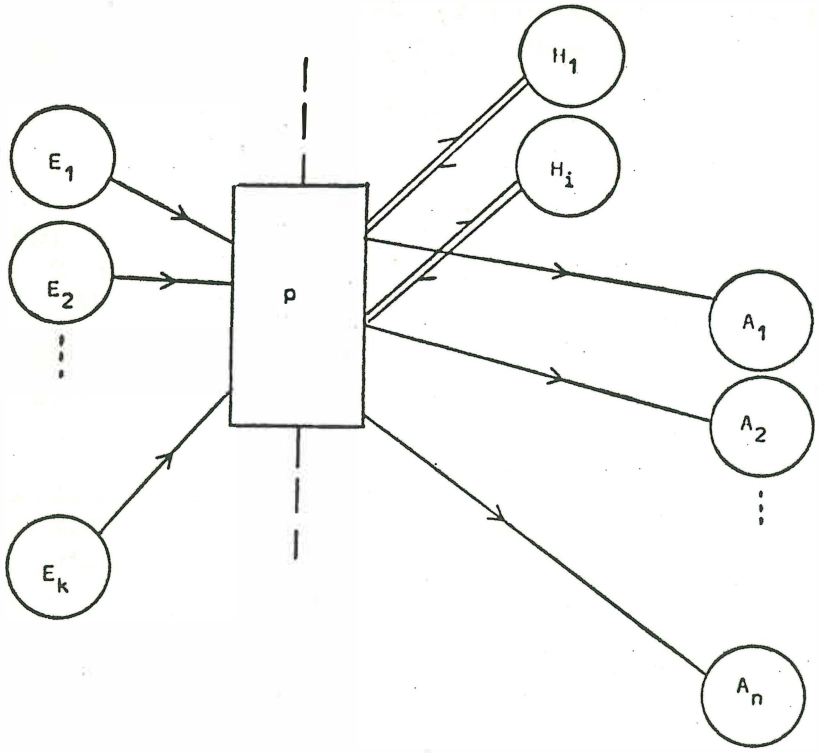


Bild 4 Standardprogrammteil p

Unter Zuhilfenahme der Hilfsbereiche H bildet p die Eingangsbereiche E auf die Ausgangsbereiche A ab. Es wird angenommen, p arbeite richtig, wenn $\{A_1, A_2, \ldots A_n\} = p(E_1, E_2, \ldots E_k)$ richtig ausgeführt wird.

Die Programmanalyse beginnt beim Speicherabzug oder beim Assemblerprotokoll. Zunächst ist ein Ablaufplan zu zeichnen. In diesem Plan ist jeder Baustein in Stücke aufgeteilt. Ein Stück ist eine Befehlsfolge ohne Sprungziele und Verzweigungen. Eingangsdatenabhängige Stücke werden von eingangsdatenunabhängigen unterschieden. Bild 5 zeigt ein Beispiel. Es handelt sich um ein Unterprogramm des alten FORTRAN Compilers. Im Ablaufplan erscheint auch, welche Stücke mit welchen Datenbereichen verkehren. Ein Datenbereich faßt mindestens ein Bit, meist mindestens ein Wort. Man unterscheidet Eingangsbereiche E_i, Ausgangsbereiche A_i und Hilfsbereiche H_i.

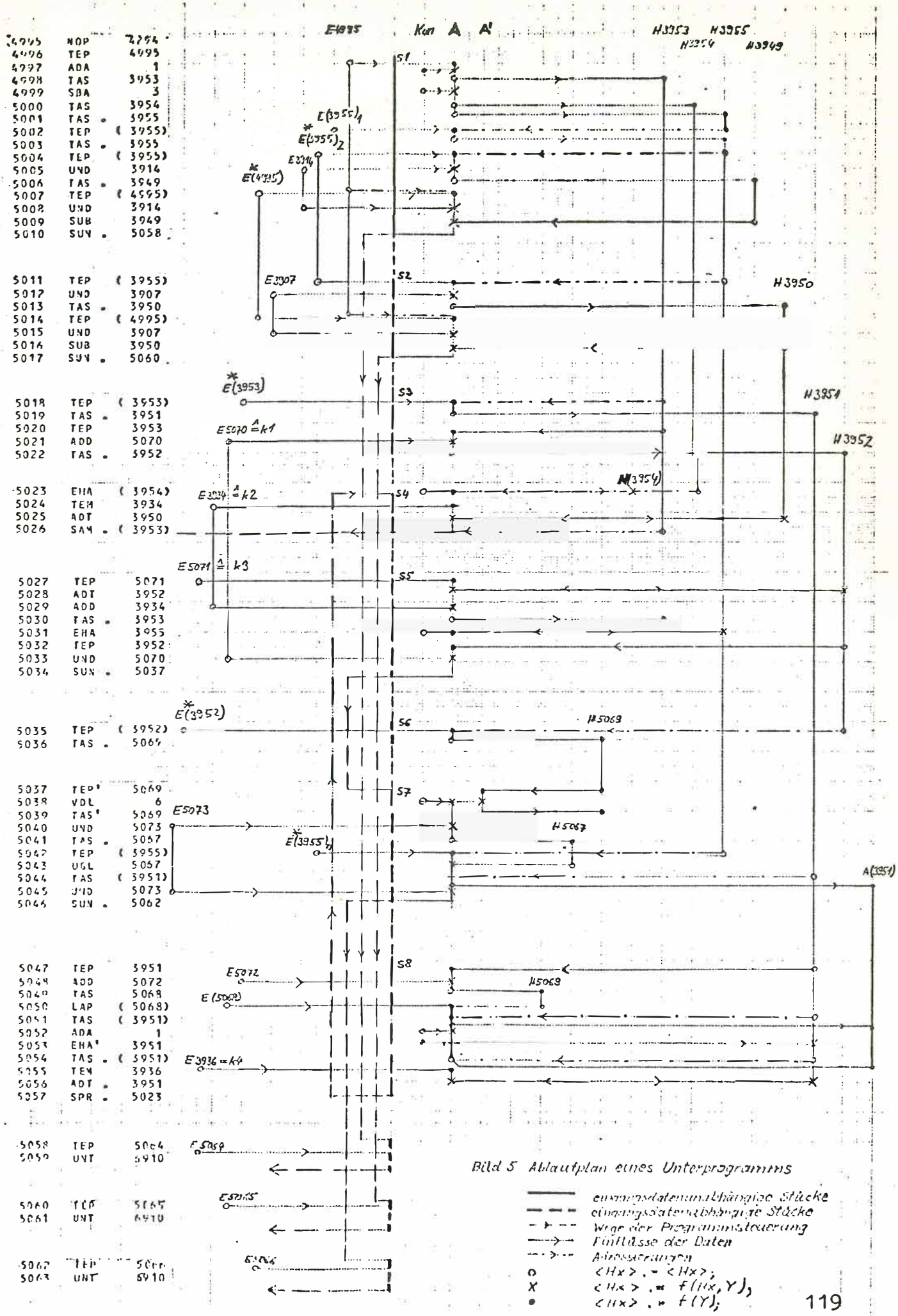
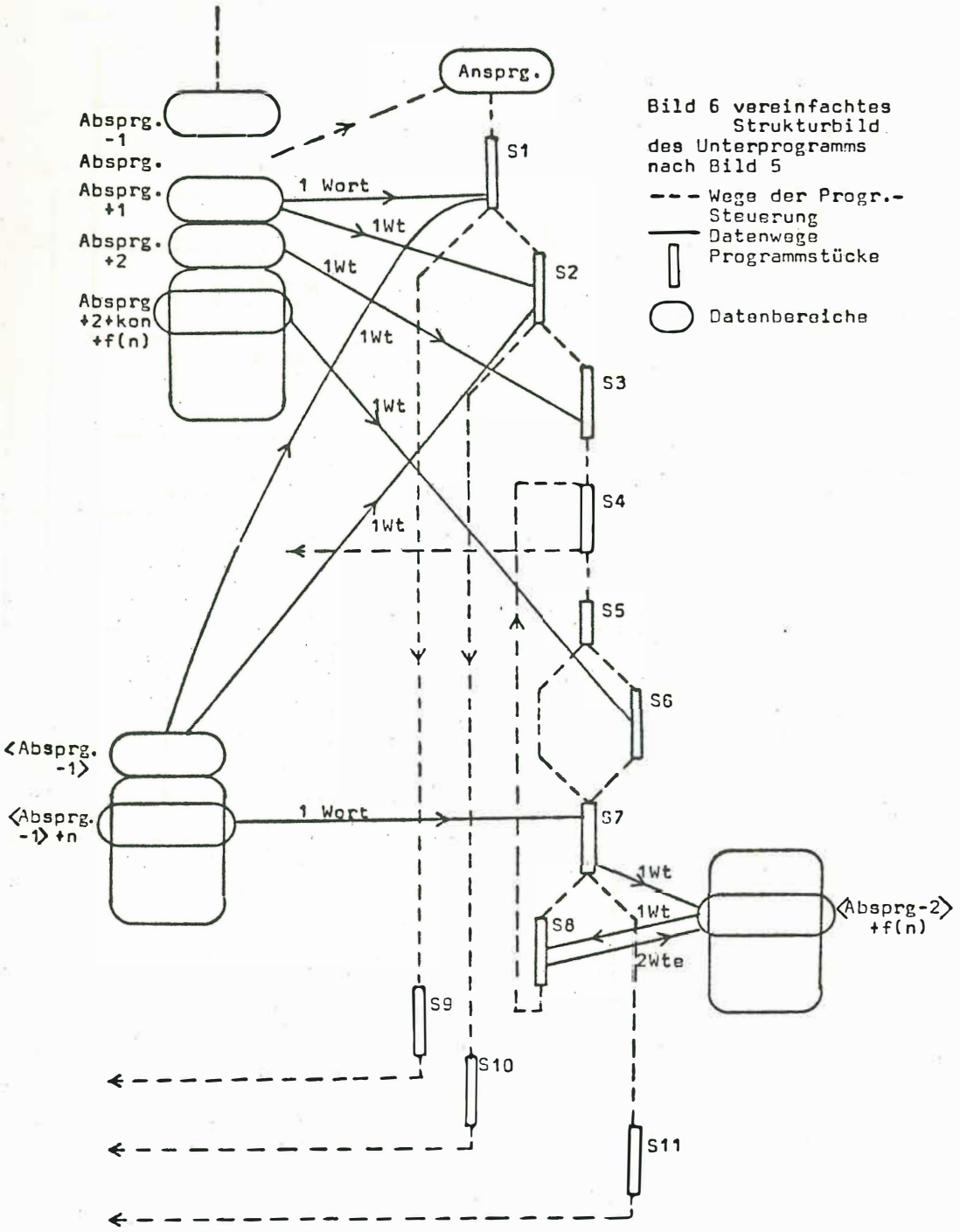


Bild 5 Ablaufplan eines Unterprogramms

- ═══ eingangsdatenunabhängige Stücke
- ━ ━ ━ eingangsdatenabhängige Stücke
- - ▸ - - Wege der Programmsteuerung
- ——▸—— Einflüsse der Daten
- - - ▸ - - Adressierungen
- o <Hx> := <Hx>;
- X <Hx> := f(Hx, Y);
- • <Hx> := f(Y);

Der Ablaufplan läßt sich zum Strukturbild vereinfachen. Im Strukturbild erscheinen nicht mehr alle Einzel-Befehle, sondern die Stücke, dafür in übersichtlicher Anordnung (Bild 6).



Die Berücksichtigung weiterer im Ablaufplan nicht aufgeführter Speicherbereiche ergibt, daß alle im Bild 5 nicht mit Sternchen bezeichneten Eingangsbereiche aus Konstanten bestehen. Sie können daher für das folgende unberücksichtigt bleiben. Wir bezeichnen mit XXXX eine Adresse mit <XXXX> den Inhalt des der Adresse zugeordneten Speicherworts, mit ≪XXXX≫ den Inhalt des Wortes, dessen Adresse in XXXX steht usw. Dann zeigt die Tabelle 1 an, welche Inhalte die einzelnen Hilfszellen unseres Beispiels haben.

Bild 6 gibt ein vereinfachtes Strukturbild wieder. Es ist deutlich zu sehen, wo Verzweigungen entschieden werden, wo welche Eingangsdaten berücksichtigt werden und daß endlich in S7 und S8 richtige Abbildungen stattfinden.

Hieraus und aus dem Ablaufplan läßt sich entnehmen:

1. Aus den Eingangsbereichen wird immer nur ein Wort gelesen.

2. Die Entscheidungen der Verzweigungen hängen, wenn man die beteiligten Konstanten wegläßt, von folgenden Eingangsbereichen ab:

   (S1→S2 / S1→S9) = f (<Absprg. + 1>, ≪Absprg.-1≫) (1)

   (S2→S3 / S2→S10) = f (<Absprg. + 1>, ≪Absprg.-1≫) (2)

   (S4→S5 / S4→Rücksprg) = f (≪Absprg - 1≫) (3)

   Rücksprungziel = f (Absprg + 2, n) (4)

   (S7→S8 / S7→S11) = f (<Absprg + 2,n>, ≪Absprg -1>,n>) (5)

   n bezeichnet die Durchlaufzahl durch die Schleife.

   n = f (≪Absprg - 1≫) (6)

3. Es findet eine Abbildung statt

   ≪Absprg + 2>, n> = f (≪Absprg - 1> + n>), (7)

   die auf ihrer Ausgangsseite zwei nebeneinanderstehende Worte beschreibt.

   Zum vollständigen Test des Programms ist also wie folgt vorzugehen:

   - Herstellen aller relevanten Zustände von <Absprg + 1> und ≪Absprg - 1≫. Was als relevant anzusehen ist, ist unter Berücksichtigung der Konstanten die dem Programm als Eingangsdaten dienen, sowie dessen was das Programm von außen erhält, zu entscheiden. Dies testet (1) und (2).

   - Zum Testen von (3) und (6) ist der ≪Absprg - 1≫ unter Berücksichtigung der Konstanten mit verschiedenen Inhalten zu versehen.

   - (4) überprüft man, ohne eigene Veränderungen vornehmen zu müssen.

- Für die Überprüfung von (5) sind die Inhalte der Worte nach Absprg + 2 und nach < Absprg - 1> jeweils geeignet zu verändern. Dies überprüft gleichzeitig auch (7).

Zum Durchführen des Tests ist es natürlich erforderlich die Spezifikation des Unterprogramms, also das was es leisten soll, genau zu kennen.

Um aus dem Beispiel nun noch auf einige allgemeine Grundsätze der Programmanalyse zu kommen: die Programmanalyse hat die Aufgabe zu zeigen

- welche Eingangsdatenbereiche unabhängig von anderen wirken,

- welche Eingangsdatenbereiche welchen Programmteilen zugeordnet sind,

- welche Wertebereiche von Eingangsvariablen überhaupt relevant sind,

- welche Eingangsdatenveränderungen durchgeführt werden müssen um das Programm vollständig zu testen,

- welche Tests nach einer gegebenenfalls nachträglichen Programmänderung wiederholt oder neu durchgeführt werden müssen.

Die Analyse eines Programms beansprucht etwa soviel Zeit, wie das Schreiben des betreffenden Programms ohne das Austesten. Eine Analyse ist vor allem dann sinnvoll, wenn das betreffende Programm nach bestem Wissen seines Programmierers fehlerfrei ist. Daher lohnt sich eine Analyse nur bei sehr wichtigen Programmen. Wie das vorangegangene Beispiel gezeigt hat, ist eine inteame Kenntnis der betreffenden Assemblersprache unabdingbar. Selbstverständlich wird der Aufwand für eine Analyse beträchtlich verringert, wenn ein einfach und übersichtlich aufgebautes Programm zu bearbeiten ist.

	H3949	H3950	H3951	H3952	H3953	H3954	H3955	<3954>	E4995	H5067	H5068	H5069
S1	<<Absprg-1>>				Absprg+2	Absprg-1	Absprg-1 <Absprg-1>	<Absprg-1>	Absprg+1			
S2		<<Absprg-1>>										
S3			<Absprg+2>	Absprg+2+ +k1								
S4		<<Absprg-1>> -k2·n						<Absprg-1> +n				
S5				Absprg+2+ +k1+n·k3	Absprg+2+ k1+n(k2+k3)		<Absprg-1> +n					
S6												<Absprg+2 +k1+n·k3>
S7										<Absprg+2 +k1+i·k3>		<Absprg+2 +k1+i·k3>
S8			<Absprg+2> +n-mk4 <Absprg.+2> +n(1-k4)								<Absprg+2> +Subst.Zei. +m(1-k4)	

Tabelle 1 Angaben der Eingangsbereiche und Durchlaufzahlen von denen die Inhalte der Hilfszellen abhängen

n Schleitendurchlaufzahl $= 1(1)\,k$

$m = 0\,(1)\,k-1$

$i \leq n$

5 Schlußbemerkung

Das vorliegende Papier konnte nur die wichtigsten Gedankengänge für den Test eines Rechners durch einen anderen Rechner und für die Programmanalyse streifen. Bezüglich der Analyse wurde absichtlich auf eine Darlegung der Theorie verzichtet und das Besprechen eines Beispiels vorgezogen.

6 Literatur

(1) W. Ehrenberger, E. Soklic
A Hybrid Method for testing Process Computer Performance
Enlarged Halden Program Group Meeting
Loen Norway May 28 - June 2, 1972

(2) W. Ehrenberger
Zur Theorie der Analyse von Prozeßrechnerprogrammen
Bericht MRR 118
Laboratorium für Reaktorregelung und Anlagensicherung der T.U. München

(3) Seymour Lipschutz
Theory and Problems of Set Theory and Related Topics
McGraw Hill Book Company New York 1964

(4) J.C. King
A program verifier
Ifip Congress 1971, Ljubljana

(5) E. Schrüfer
Vergleich zwischen einem festverdrahteten und einem frei programmierbaren Reaktorschutzsystem
Atom und Strom. Heft 9/10, 1972