LEXIKON DER KYBERNETIK
ERGÄNZUNGSBAND
A–Z

LEXIKON DER KYBERNETIK

A-Z

ERGÄNZUNGSBAND

Im Auftrag des Zentralinstituts für Kybernetik und Informationsprozesse
der Akademie der Wissenschaften der DDR

herausgegeben von
GÜNTER LAUX

Wissenschaftliches Redaktionskollektiv:

Klaus Fritzsch, Michael Gössel, Hans-Georg Lauenroth,
Günter Laux, Kurt Reinschke, Dieter Richter

AKADEMIE-VERLAG BERLIN
1985

Autoren und verantwortliche Bearbeiter:

Dipl.-Ing. Hans-Martin Adler, Dipl.-Math. Werner Aßmann,
Prof. Dr. sc. nat. Peter Bachmann, Dr. rer. nat. habil. Klaus Bellmann,
Dr.-Ing. Gerhard Billerbeck, Dipl.-Ing. Lutz Blencke,
Prof. Dr. sc. techn. Dieter Bochmann, Dr. sc. nat. Manfred Bonitz,
Dr. rer. nat. habil. Hans Günter Bothe, Prof. Dr. sc. techn. Georg Brack,
Dr.-Ing. Wolfgang Claus, Prof. Dr. sc. techn. Herbert Ehrlich,
Dr. rer. nat. Heinrich Endert, Dr. rer. nat. Peter Enskonatus,
Prof. Dr. sc. techn. Klaus Fritzsch, Dr. sc. nat. Michael Gössel,
Dr. rer. nat. Martin Grabow, Prof. Dr. sc. nat. Dieter Hammer,
Dr. sc. nat. Heinrich Herre, Dr.-Ing. Bernd Hofmann,
Dr.-Ing. Wolfgang Jansen, Dr. rer. nat. Jürgen Kaltwasser,
Dr. sc. nat. Dietrich Koch, Dr. rer. nat. Werner Kolbe,
Dr. oec. Eberhard Kummerow, Prof. Dr. sc. nat. Karl-Heinz Kutschke,
Doz. Dr. sc. oec. Wolfgang Lassmann,
Prof. Dr. rer. oec. habil. Hans-Georg Lauenroth, Dipl.-Ing. Günter Laux,
Dr. rer. nat. Eberhard Matthäus, Prof. Dr. sc. nat. Hermann Walter Meier,
Dipl.-Math. Andreas Meißner, Dr. rer. nat. Heinz D. Modrow,
Doz. Dr. sc. oec. Johann Adolf Müller, Prof. Dr. sc. techn. Rainer Müller,
Dipl.-Phys. Rüger Oßwald, Prof. Dr. sc. nat. Rudolf Arthur Pose,
Dr. sc. techn. Dietmar Reinert, Prof. Dr. sc. techn. Karl Reinisch,
Doz. Dr. rer. nat. Dr.-Ing. habil. Kurt Reinschke, Dr. rer. nat. Dieter Richter,
Dr.-Ing. Hans Rudolph, Dr.-Ing. Claus Sattler, Dr.-Ing. Hans-Jürgen Schiller,
Dr. phys.-math. Wiss. Wladimir Grigorjewitsch Sragowitsch,
Dr.-Ing. Bernd Straube, Dr. rer. nat. Bernhard Thalheim,
Dr. rer. nat. Wolfgang Uebel, Dr.-Ing. Hans-Michael Voigt,
Prof. Dr. rer. nat. habil. Horst Völz, Dr.-Ing. Ingo Wende,
Dr.-Ing. Dietrich Werner, Prof. Dr.-Ing. sc. techn. Wolfgang Wilhelmi,
Dipl.-Math. Achim Wittmüß, Dr.-Ing. Gottfried Wolf,
Prof. Dr.-Ing. habil. Gerhard Wunsch

Redaktionsschluß: Januar 1984

Erschienen im Akademie-Verlag Berlin, DDR - 1086 Berlin, Leipziger Straße 3–4

Lizenznummer: 202 · 100/420/85
Printed in the German Democratic Republic
Gesamtherstellung: VEB Druckerei „Gottfried Wilhelm Leibniz“,
4450 Gräfenhainichen
Lektor: Dipl.-Phys. Gisela Lagowitz
LSV 1097
Bestellnummer: 763407 8 (6297/E)
06000

Vorwort zum Ergänzungsband

Es sei noch einmal an das im Vorwort zum Lexikon der Kybernetik Gesagte erinnert, daß es weniger darauf ankommen sollte, eine allen Gesichtspunkten gerecht werdende Definition der Kybernetik zu finden als vielmehr das jeweilige Gebiet kybernetischer Forschungen genau zu bestimmen. Das große Gebiet der Kybernetik-Forschung in der DDR hat sich in den zurückliegenden vier Jahren seit Erscheinen des Lexikons der Kybernetik nicht geändert, nur das Spektrum der – in Übereinstimmung mit den höheren gesellschaftlichen Anforderungen – zu lösenden Aufgabenstellungen ist für bestimmte Gebiete konkreter und ganz allgemein umfangreicher geworden. Die Forcierung der weiteren Entwicklung der Mikroelektronik, der zielgerichtete und zweckentsprechende Einsatz der Robotertechnik, die stärker als bisher geforderte Rationalisierung in der Sphäre der materiellen Produktion, d. h. die Vervollkommnung der Automatisierungsmittel, die Automatisierung ganzer Produktionsabschnitte bis hin zur Schaffung flexibler Fertigungssysteme, sowie die im Interesse der Produktivitätssteigerung zunehmende Notwendigkeit der rechnergestützten Arbeitsweise in allen gesellschaftlichen Bereichen sind anspruchsvolle Aufgaben, zu denen die Kybernetik-Forschung entscheidende praxiswirksame Beiträge zu leisten hat, ohne dabei die notwendige disziplingebundene Vertiefung ihrer Methoden zu vernachlässigen und ihre Grenzen zu eng zu ziehen.

Der Praxisbezug der Kybernetik-Forschung war deshalb auch das wesentliche Kriterium für die Auswahl und Darstellung der in diesen Ergänzungsband aufgenommenen Schlagworte. Das wissenschaftliche Redaktionskollektiv sah seine Verantwortung darin, diese Auswahl mit größter Sorgfalt vorzunehmen, um gegenwärtige Arbeitsrichtungen – allerdings unter dem Zwang der festgelegten alphabetischen Reihenfolge der Schlagworte – möglichst in ihrer Komplexität repräsentativ darzustellen. Mit voller Absicht sind deshalb auch Begriffe berücksichtigt worden, die nicht mehr neuestes Wissen darstellen, sondern inzwischen zur Allgemeinbildung des jeweiligen Fachmannes gehören. Wenn seit dem Redaktionsschluß im Januar 1984 bis zum Erscheinen dieses Bandes die dargestellten Ergebnisse und Kenntnisse schon wieder durch neueres, aktuelleres Wissen übertroffen werden, so sind sich die verantwortlichen Bearbeiter dieser Tatsache bewußt, ebenso

wie sie wissen, daß Vollständigkeit – insbesondere auf einem sich in so starker Entwicklung befindlichen Gebiet – nicht Aufgabe eines Lexikons sein kann, und sie sich nur trösten können mit dem im Vorwort zu H. KOBLISCHKES Abkürzungsbuch (Leipzig 1969) gefundenen Ausspruch des englischen Schriftstellers und Lexikographen SAMUEL JOHNSON (1709–1794): „Wörterbücher sind wie Uhren; die schlechteste ist besser als gar keine, und von der besten kann man nicht erwarten, daß sie immer ganz richtig geht".

Die freundliche Aufnahme des vierbändigen Lexikons der Kybernetik, die in vielen Rezensionen in den verschiedensten Fachzeitschriften zum Ausdruck kommt, aber auch die zahlreichen kritischen Hinweise und förderlichen Ratschläge waren für die Erarbeitung dieses Ergänzungsbandes eine große Hilfe. Dafür möchte sich der Herausgeber auf diesem Wege noch einmal herzlich bedanken. Recht herzlich dankt er auch allen Autoren und den verantwortlichen Bearbeitern für die von ihnen geleistete Arbeit. Ein besonderer Dank gilt Herrn Prof. Dr. sc. nat. VOLKER KEMPE, der als Direktor des Zentralinstituts für Kybernetik und Informationsprozesse der AdW der DDR dieses Vorhaben jederzeit unterstützte und insbesondere der Arbeit des Herausgebers viel Verständnis entgegenbrachte. Der Dank gilt nicht zuletzt der im Verlag zuständigen Lektorin, Frau Dipl.-Phys. GISELA LAGOWITZ, für die jederzeit vertrauensvolle Zusammenarbeit, die sich schon bei der Erarbeitung der bisherigen vier Bände so förderlich auf das Gelingen des Werkes ausgewirkt hat.

Trotz aller Sorgfalt werden Unzulänglichkeiten an dieser oder jener Stelle sicher nicht ganz vermieden worden sein. Herausgeber und Verlag sind deshalb für alle Bemerkungen und kritischen Hinweise dankbar und bitten darum, diese an das Zentralinstitut für Kybernetik und Informationsprozesse der AdW der DDR, DDR - 1086 Berlin, Kurstraße 33 zu richten.

G. LAUX

Hinweise für den Benutzer

Der vorliegende Ergänzungsband ist in strenger Übereinstimmung mit dem vierbändigen Lexikon der Kybernetik aufgebaut. Er umfaßt einen Textteil, in dem die Beiträge alphabetisch nach Schlagworten geordnet sind, ein Verzeichnis von in der Literatur gebräuchlichen Abkürzungen sowie eine Schlagwortliste aller in den fünf Bänden enthaltenen Schlagworte in deutscher, englischer, russischer und französischer Sprache.

Die Schlagworte stehen im Nominativ Singular, Ausnahmen im Plural gibt es nur dort, wo es die inhaltliche Darstellung des Begriffes verlangt. Im zugehörigen Text wird zugelassen, das Schlagwort kommentarlos nur noch mit seinem (bzw. seinen) Anfangsbuchstaben und Punkt zu verwenden, z. B. „Datenbeschreibungssprache" mit „D." oder „Textkommunikation, elektronische" mit „e. T.".

Im Text *kursiv* gedruckte Wörter weisen ausschließlich auf eigenständige Schlagworte an anderer Stelle des nunmehr fünfbändigen Lexikons hin.

Für einfach zusammengesetzte Schlagworte erfolgt die Ordnung nach dem Substantivum regens, z. B. „*Compiler, optimierender*" oder „*Datenmodell, hierarchisches*". Ausnahmen bilden Schlagworte, die als Terminus technicus zum Allgemeingut des jeweiligen Fachgebietes geworden sind, z. B. „*Abstrakte Datentypen*" oder „*Digitale Bildverarbeitung*". Komposita werden als Substantiv nach ihrem Anfangsbuchstaben eingeordnet, z. B. „*Compiler-Compiler*", „*Mehrprozessorsystem*" oder „*Pipeline-Prinzip*".

Bei Schlagworten, die aus mehreren Wörtern bestehen, ist der für die Kennzeichnung des Inhalts wesentlichste Begriff vorangestellt, z. B. „*Automatisierung wissenschaftlicher Forschungsarbeiten*", „*Informationsprozesse, Grundoperationen*" oder „*Operative Lenkung der Produktion*".

Bei Schlagworten, die Eigennamen enthalten, wird dieser vorangestellt, z. B. „BOOLE*scher Differentialkalkül*", „OSTROWSKI-*Bänder*" oder „SHEGALKIN-*Polynom*".

Schlagworte aus dem Englischen, die in ihrer Originalschreibweise Eingang in die deutsche Fachsprache gefunden haben, werden auch in dieser Schreibweise nach ihrem Anfangsbuchstaben alphabetisch eingeordnet, z. B. „*Computer Aided Design*", „*Debugging*", „*Fuzzy-Steuerung*" oder „*Scheduling*".

Zusätzlich sind zahlreiche Verweise aufgenommen worden, die jeweils auf das Schlagwort hinweisen, das Aufschluß über den betreffenden Begriff gibt.

Es wurde angestrebt, durchgängig für ein und denselben Begriff auch ein und dasselbe Formelzeichen zu verwenden. Unterschiede gibt es allerdings dort, wo die in den verschiedenen Fachgebieten benutzten Schreibweisen seit Jahren üblich sind und eine kompromißlose Vereinheitlichung nur zu Mißverständnissen führen würde.

Im Abkürzungsverzeichnis wird der sprachlichen Auflösung der Abkürzung eine kurze textliche Erläuterung hinzugefügt. Darauf verzichtet wurde bei solchen Abkürzungen, die schon durch die sprachliche Auflösung genügend erklärt sind oder die an anderer Stelle im Lexikon ausführlich erläutert werden.

Die Schlagwortliste ist nach den deutschen Schlagworten alphabetisch geordnet und kann mit der zusätzlich vorhandenen Angabe des betreffenden Bandes und der Seitenzahl zugleich als Schlagwortverzeichnis zum schnelleren Auffinden der gewünschten Textstelle benutzt werden. Alphabetische Verzeichnisse in englischer, russischer und französischer Sprache ermöglichen auch den Zugang zu den Textstellen aus der Fremdsprache.

Für interessierte Leser enthält das Lexikon – allerdings nicht mehr im Ergänzungsband, sondern nur in den bisherigen vier Bänden – in beschränktem Umfang einschlägige Firmen, Institutionen, Organisationen und Zeitschriften. Sie werden unter ihrem Originalnamen geführt und sind entsprechend alphabetisch eingeordnet. Sowjetische Institutionen, Organisationen und Zeitschriften werden in der deutschen Übersetzung angegeben. Die Originalbezeichnung in kyrillischer Schrift steht, nach dem russischen Alphabet geordnet, unter den in das Lexikon aufgenommenen Hinweisschlagworten „*Firmen*“, „*Institutionen*“, „*Organisationen*“ und „*Zeitschriften*“.

A

Ablaufsteuerung – Steuerung, bei der die Steuergröße oder die Führungsgröße (für eine untergeordnete Steuerung) von den Werten (von den Abläufen) bestimmter Zustandsgrößen des gesteuerten Systems und einem gegebenen Programm abhängig ist. Beispiele hierfür sind eine Aufzugssteuerung und die Stillsetzung bzw. Umkehr des Vorschubs einer Werkzeugmaschine durch einen Grenzwertschalter, der den Weg des Werkzeugs begrenzt. Auf diese Weise kann ein Arbeitstakt beendet und entsprechend einem eingegebenen Programm ein neuer ausgelöst werden. Zur Aufnahme des Programms dient ein Programmgeber. Meist werden die maßgebenden *Zustandsgrößen* nur auf die Erfüllung bestimmter Bedingungen geprüft und nehmen auch die Ausgangsgrößen (Steuer- oder Führungsgrößen) nur zwei Werte an. Der *Steuerungsalgorithmus* ist dann eine BOOLEsche *Funktion* und kann deshalb durch ein *Schaltsystem* realisiert werden. Die A. werden zusammen mit den Zeitplansteuerungen auch als Programmsteuerungen bezeichnet.

K. REINISCH

Literatur: REINISCH, K.: Kybernetische Grundlagen und Beschreibung kontinuierlicher Systeme. Berlin: VEB Verlag Technik 1974.

Abstrakte Datentypen – algebraisch spezifizierte Datenstrukturen. Jeder a. D. wird beschrieben durch eine Menge von Operationen (über einen definierten Gültigkeitsbereich) und eine Menge von Axiomen, die von diesen Operationen erfüllt werden müssen. Die Lösungen dieses Gleichungssystems sind Datenstrukturen.

Als Beispiel sei die Spezifikation eines Kellers S über Datenelementen E angegeben. Die Operationen sind:

$\emptyset$:	$\rightarrow S$	
push:	$S \times E \rightarrow S$	(Einkellern eines Datenelements)
pop:	$S \rightarrow S$	(Löschen der Kellerspitze)
top	$S \rightarrow E$	(Abfragen der Kellerspitze)

Die dazugehörigen Axiome lauten:

pop ($\emptyset$)	= error	(Fehlersituation)
pop(**push**(s, e))	$= s$	

top (Ø) = error (Fehlersituation)
top(**push**(s, e)) = e

In neueren *Programmiersprachen* sind Konzepte zur Unterstützung der Arbeit mit a. D., allerdings meist in Form der funktionalen Implementierung eines Modells enthalten.

W. ASSMANN

Literatur: GOGUEN, J. A., J. W. THATCHER a. E. G. WAGNER: An initial algebra approach to the specification, correctness and implementation of abstract data types. In: Current Trends in Programming Methodology, ed. by R. T. YEH, vol. IV. Englewood Cliffs: Prentice Hall, Inc. 1978.

Abtastregelung – s. *Regelung*.

Abtastsystem – System der *Informationsübertragung* oder Steuerung, in dem kontinuierliche Signale (z. B. Sprachsignale oder gemessene Prozeßgrößen) nur zu diskreten Zeitpunkten abgetastet und als diskontinuierliche Signale (Folge von Meßwerten) übertragen bzw. verarbeitet werden. Das ist erforderlich bei zeitgestaffelter Übertragung mehrerer Signale über einen Übertragungskanal, bei diskontinuierlich arbeitenden (Analyse-) Meßgeräten und bei Einsatz von digitalen (diskontinuierlich arbeitenden) *Prozeßrechnern* an kontinuierlichen Prozessen. Für die Rekonstruierbarkeit eines kontinuierlichen Signals aus den Abtastwerten bestehen Mindestforderungen an die Abtastfrequenz (s. *Abtasttheorem*). Die mathematische Beschreibung von Abtastsystemen erfolgt durch Differenzengleichungen und (zeit-) diskrete Zustandsgleichungen oder im Bildbereich mit Hilfe der *z-Transformation*, die eine *diskrete* LAPLACE-*Transformation* (für zeitdiskrete Signale) darstellt.

K. REINISCH

Literatur: GÜNTHER, M.: Analyse und Synthese zeitdiskreter/digitaler Steuerungssysteme. In: Taschenbuch Elektrotechnik in 6 Bänden, hrsg. von E. PHILIPPOW, 2. vollständig neu bearbeitete Neuauflage. Berlin: VEB Verlag Technik 1985; FÖLLINGER, O.: Lineare Abtastsysteme, 2. Auflage. München – Wien: R. Oldenbourg Verlag 1982; ZYPKIN, J. S.: Grundlagen der Theorie automatischer Systeme. Berlin: VEB Verlag Technik 1981.

Abtasttheorem – (Probensatz, Sampling-Theorem) ein Satz der *Informationstheorie*, der die Bedingungen formuliert, unter denen eine Zeitfunktion durch eine Impulsfolge ohne Informationsverlust dargestellt werden kann.

Es seien $x(t)$ eine Zeitfunktion mit frequenzbeschränktem Spektrum im Intervall und $x_m = x\left(\frac{m}{2f_0}\right)$, $m = -\infty, \ldots, +\infty$, eine Folge äquidistanter Abtastwerte von $x(t)$.

Dann gilt $x(t) = \sum_{m=-\infty}^{m=+\infty} x_m \cdot \frac{\sin \pi (2f_0 t - m)}{\pi (2f_0 t - m)}$.

Die Funktionen $s_m(t) = \frac{\sin \pi (2f_0 t - m)}{\pi (2f_0 t - m)}$ sind die sogenannten Spaltfunktionen. $s_m(t)$ nimmt am Abtastpunkt $t = \frac{m}{2f_0}$ den Wert Eins an und verschwindet an allen übrigen Abtastpunkten. Die Spaltfunktionen sind auf der unendlich langen Zeitachse definiert, bei Beschränkung auf endliche Intervalle entstehen Seitenbänder mit unendlich hohen Frequenzen. Durch Beschränkung auf das Frequenzband $0 \leqq f \leqq f_0$ läßt sich $x(t)$ in jedem Falle aus der Summandenstellung der Spaltfunktionen zurückgewinnen.

Das A. kann auf den Fall eines beliebigen endlichen Frequenzbandes $0 \leqq f_1 < f \leqq f_2$ verallgemeinert werden.

Das A. formuliert einen Idealfall, der in der Praxis nicht erreicht wird, da ideale Bandpässe physikalisch nicht realisierbar sind. Es gilt streng nur für deterministische Signale. Bei stationären, nichtperiodischen Zufallssignalen gilt die Darstellung nur in einem zeitlich beschränkten Intervall. Trotz dieser Einschränkungen besitzt das A. erhebliche theoretische und praktische Bedeutung.

K. Fritzsch

Literatur: Spătaru, A.: Theorie der Informationsübertragung. Berlin: Akademie-Verlag 1973.

Ada – Die universelle *Programmiersprache* Ada ist nicht nur wegen der Vielzahl der enthaltenen Sprachkonzepte interessant, sondern auch wegen ihrer ungewöhnlichen Entwicklungsgeschichte. Wegen der ständig steigenden Anzahl von existierenden und eingesetzten Programmiersprachen und der dadurch verursachten Problematik vor allem in großen Programmsystemen orientierte das US-amerikanische DoD (Department of Defense) gegen 1974 auf den zukünftigen Einsatz nur einer einzigen Programmiersprache. Hierzu wurden zunächst die Anforderungen an eine solche Sprache formuliert und auf breiter Ebene diskutiert und konkretisiert. Das Ergebnis lag 1978 mit dem sog. Steelman-Bericht vor. Die bereits existierenden Sprachen wurden sorgfältig mit diesem Forderungsprogramm verglichen, wobei festgestellt wurde, daß keine dieser Sprachen hinreichend geeignet ist. Innerhalb dieser Untersuchung wurden ferner *PASCAL*, *PL/1* und *ALGOL* 68 als Ausgangspunkte für die Entwicklung der endgültigen Sprache festgelegt sowie einige Eigenschaften von RTL/2 und LIS als besonders interessant klassifiziert. Auf dieser Basis wurden schließlich Sprachvorschläge von verschiedenen Compiler-Entwicklern ausgearbeitet, von denen 1979 das bei CII Honeywell

Bull unter Leitung von J. D. ICHBIAH ausgearbeitete Projekt angenommen wurde. Diese Sprache wurde nach AUGUSTA ADA LOVELACE, der Assistentin des genialen Konstrukteurs von Rechenmaschinen CHARLES BABBAGE (1792–1871), benannt, die als erste Programmiererin der Welt gilt. Die vorläufige Sprachdefinition vom Juni 1979 wurde nochmals überarbeitet und als endgültige Version im Juli 1980 herausgegeben. In der Zwischenzeit wurden im Rahmen der weltweiten *Implementierung* von *Compilern* weitere Änderungsvorschläge gemacht und bei der endgültigen Standardisierung berücksichtigt. Ada ist die leistungsfähigste Programmiersprache der Gegenwart und gleichzeitig aufgrund ihrer Eigenschaften eine Herausforderung an die Compiler-Hersteller, so daß erst 1983 erste Compiler bereitgestellt wurden, die die strengen Qualitätsforderungen erfüllen.

Die wesentlichsten Eigenschaften von Ada sind das sehr breit ausgebaute Typkonzept, das auch die (prozedurale) Definition *abstrakter Datentypen* zuläßt, die Möglichkeiten der *separaten Compilierung*, die Unterstützung der Parallelverarbeitung und die Behandlung von Ausnahmebedingungen während der Programmabarbeitung. Ferner ist sowohl der Top-down-Entwurf als auch der Bottom-up-Entwurf von großen Programmsystemen möglich, wobei die Ada-Programmier- und Testumgebung einen großen Beitrag zur effektiven Herstellung zuverlässiger *Software* leistet.

Die Lexik von Ada gestattet die Verwendung von Groß- und Kleinbuchstaben sowie von Ziffern und des Unterstreichungszeichens innerhalb von Identifikatoren, die außerdem beliebig lang sein dürfen. Zusammen mit der Kommentarschreibweise (Kommentare beginnen mit „– –“ und enden am Zeilenende) ermöglicht sie gut lesbare, selbstdokumentierende Programme. Durch die Unterteilung großer Programmsysteme in kleinere Einheiten, insbesondere in Programmpakete (packages), die neben der Definition spezifischer *Datentypen* auch alle Operationen mit Objekten dieser Datentypen enthalten, wird ein Grundprinzip der strukturierten Programmierung unterstützt.

Ein solches Programmpaket wird unterteilt in einen von außen (also vom Nutzer) sichtbaren Teil, der alle Informationen enthält, die für einen Nutzer des Programmpakets von Interesse sind, also letztlich das *Interface* des Programmpakets, und einen nicht sichtbaren Teil (Programmpaket-Körper), der interne Deklarationen und die Algorithmen enthält. Durch das folgende Programmpaket wird der Datentyp „Keller“ bereitgestellt, und zwar für reelle Zahlen und eine maximale Kellergröße von 100:

```
package STACK is                    – – Spezifikationsteil
  MAX: constant:=100;               – – Kellergröße
  S: array (1. .MAX) of REAL;       – – Keller
  procedure PUSH(X: REAL);          – – Interface für PUSH
```

```
    function POP return REAL;          --Interface für POP
end;

package body STACK is                  --Programmpaketkörper
    PTR: INTEGER range 0..MAX;         --Kellerzeiger
    procedure PUSH(X: REAL) is         --Einkellerungsoperation
    begin
        PTR:=PTR+1;
        S(PTR):=X;
    end PUSH;
    function POP return REAL is        --Auskellerungsoperation
    begin
        PTR :=PTR-1;
        return S(PTR+1);
    end POP;
begin                                  --Programmpaket-Initialisierung
    PTR :=0;
end STACK;
```

Der Aufruf dieser Prozeduren kann mittels STACK.PUSH(Y) bzw. Y:=STACK.POP(); durchgeführt werden.

Spezifikationsteil und Körper bilden selbständige Compilierungseinheiten und können unabhängig voneinander übersetzt werden, wobei allerdings vom Compiler die Übereinstimmung der Interface-Beschreibung mit der tatsächlichen Implementierung im Programmpaketkörper überprüft wird.

Falls der Datentyp Keller nicht nur für reelle Zahlen benötigt wird, würde das für jeden weiteren Elementtyp ein weiteres Programmpaket erfordern. Durch Anwendung von generierbaren Programmpaketen kann nun ein einziges Programmpaket nach Bedarf weitere Programmpakete für andere Kellergrößen und Elementtypen erzeugen. Dazu ist lediglich im obigen Beispiel der Spezifikationsteil wie folgt abzuändern:

```
generic
    MAX: INTEGER;                      --1. Parameter: Kellergröße
    type ELEM is private;              --2. Parameter: Elementtyp
package STACK is
    S: array (1..MAX) of ELEM;
    procedure PUSH(X: REAL);
    function POP return ELEM;
end;
```

Um wiederum mit einem Keller der maximalen Länge 100 für reelle Zahlen arbeiten zu können, ist nun folgende Anweisungsfolge notwendig:

```
declare
  package MY_STACK is new STACK (100, REAL);
                                    - - Anlegen des speziellen Kellers
  use MY_STACK;                     - - Vereinfachte Schreibweise
begin
  ...
  PUSH(Y);                          - - anstelle MY_STACK.PUSH
  ...
  Y:=POP();                         - - anstelle MY_STACK.POP
  ...
end;
```

Es ist auf die gleiche Weise aber auch möglich, ein Programmpaket für die Verarbeitung beliebig komplizierter Datentypen (Felder, Records) herzustellen.

Als weitere Ergänzung zu diesem Beispiel sei die Behandlung von Ausnahmebedingungen dargestellt, wie sie hier beim Überschreiten der Kellergrenzen auftreten können. Ergänzt man im Spezifikationsteil die Deklaration einer Ausnahmebedingung

```
  ERROR: exception;
```

sowie in den Prozeduren PUSH bzw. POP die Anweisungen

```
  if PTR=MAX then                   - - Überlauf in PUSH
    raise ERROR
  end if;
```

bzw.

```
  if PTR=0 then                     - - Unterlauf in POP
    raise ERROR
  end if;
```

so führt eine Überschreitung der Kellergrenzen zur Ausnahmebedingung ERROR. Im aufrufenden Programm kann darauf geeignet reagiert werden:

```
declare                             - - Anlegen des Kellers
  package MY_STACK is new STACK (100, REAL);
  use MY_STACK;
begin
  ...
  PUSH(Y);
  ...
  Y:=POP();
```

```
exception                          - - Ausnahmebehandlungsteil
  when ERROR = >
  ...                              - - Anweisungsfolge zur Behandlung
                                       von Indexüberschreitungen
end;
```

Für die Echtzeitverarbeitung bietet Ada eine Reihe spezieller Anweisungen. Solche Programmteile, die parallel zueinander ablaufen sollen, werden ähnlich wie Programmpakete als **task**'s deklariert. Der Start einer **task** erfolgt automatisch, wenn der Anweisungsteil der Programmeinheit aktiviert wird, in dem die Deklaration der **task** erfolgte. Fundamentales Kommunikations- und Synchronisationsmittel zwischen **task**'s ist das sog. Rendezvous. Hierzu werden bei der Deklaration einer **task** auch Eingänge (**entry**) deklariert und die dazugehörigen Aktionen in **accept**-Anweisungen zusammengefaßt. Soll ein bestimmter Eingang einer **task** von einer anderen **task** aus aktiviert werden, so erfolgt dies durch eine einem Prozeduraufruf ähnliche Anweisung, bei der neben dem **entry**-Namen auch Parameter übergeben werden. Ein solches Rendezvous wird in mehreren Etappen durchgeführt:

- Synchronisationsphase: Die aufrufende **task** arbeitet bis zum Erreichen des **entry**-Aufrufs, die gerufene **task** bis zum Erreichen dieses Eingangs (**accept**-Anweisung); beide warten jeweils aufeinander.
- Datenaustausch und -verarbeitung: Der gerufenen **task** werden die Parameterwerte übergeben, die **accept**-Anweisung wird ausgeführt, und gegebenenfalls erfolgt die Bereitstellung von Resultaten als Parameterwert für die rufende **task**.
- Rendezvousbeendigung: Beide **task**'s arbeiten unabhängig voneinander weiter bis zum Erreichen eines neuen **entry**-Aufrufs oder Eingangs.

Durch verschiedene Möglichkeiten zur Beeinflussung des Rendezvousverhaltens (**select**-Anweisung) und zur Zeitsteuerung (**delay**-Anweisung) wird Ada allen Anforderungen an eine Echtzeitverarbeitungssprache gerecht.

Das Typkonzept von Ada enthält numerische Typen, Zeichenketten, Aufzählungen, Felder, Records und Pointer, ferner lassen sich zu jedem Typ Teilmengen (**subtypes**) definieren. Durch die Verwendung von sog. Attributen können während der Laufzeit im Anweisungsteil Informationen über Typeigenschaften gewonnen werden (z. B. über die aktuellen Feldgrenzen eines Feldparameters in einem Unterprogramm). Durch die Möglichkeit, Operatoren zu „überladen", kann die Lesbarkeit von Programmen wesentlich erhöht werden (Verwendung von Operatoren wie „+" und „−" anstelle von Funktionsaufrufen und typgerechtes Verarbeiten durch den Compiler).

Ferner kann auf die rechnerinterne Datendarstellung Einfluß genommen werden. Das betrifft einerseits die Genauigkeit von Zahlen, andererseits aber

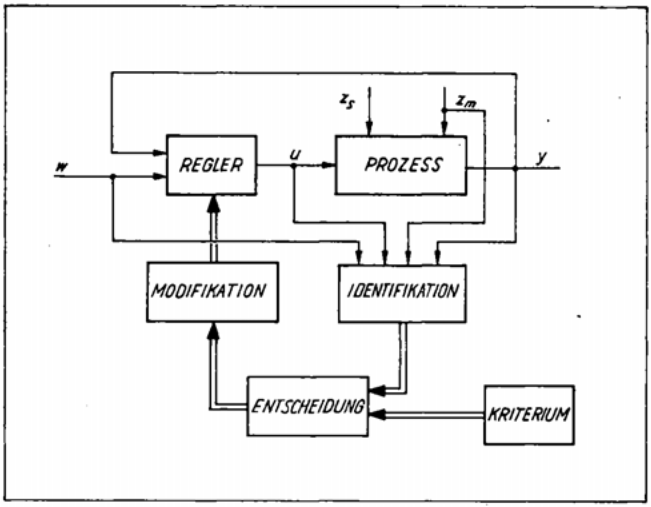
auch die Codierung und Adressierung von Objekten. Es ist auf diese Weise möglich, das Programmverhalten maschinenunabhängig zu machen, aber auch spezielle Maschineneigenschaften zu berücksichtigen. Zusammen mit den strengen Qualitätsforderungen an die Ada-Compiler ist deshalb die *Portabilität* von Ada-Programmen weitgehend gewährleistet.

W. ASSMANN

Literatur: LEDGARD, H.: Ada – an introduction. Ada reference manual (July 1980). Berlin, Heidelberg, New York: Springer-Verlag 1981.

Adaptive Regelung – selbstanpassende, selbsteinstellende, selbstoptimierende Regelung; ein *Steuerungssystem* mit der Fähigkeit, bei einer gewissen Anfangsunbestimmtheit und sich ändernden Arbeitsbedingungen einen bestimmten – gewöhnlich optimalen – Systemzustand zu erreichen und aufrechtzuerhalten. Die Anpassung (*Adaption*) an die jeweiligen Arbeitsbedingungen erfolgt durch Veränderung der Parameter, der Systemstruktur und, wenn möglich, der steuernden Einwirkungen auf Grund von Informationen, die während der Steuerung erworben wurden. Zur Beurteilung des Systemzustandes wird i. allg. ein Kriterium herangezogen. Aus dem laufenden Vergleich von erreichter und gewünschter Güte werden Maßnahmen abgeleitet, die in Richtung Güteverbesserung zielen. Das Adaptionsproblem ist somit interpretierbar als Optimierungsproblem, das unter der Bedingung unvollständiger bzw. fehlender A-priori-Information zu lösen ist. Die Steuerung dient nicht nur der eigentlichen Prozeßführung, sondern auch der Beschaffung von Information zum Abbau der Anfangsunbestimmtheit (Systemerkennung, *Identifikation*). Diesem zweifachen (dualen) Charakter der Stellgröße trägt das Konzept der dualen Steuerung Rechnung, das die Probleme „Identifikation" und „Optimierung" unter Berücksichtigung ihrer Abhängigkeit vereinigt. Die Lösung dualer Steuerungsaufgaben ist jedoch sehr aufwendig und daher nur in einfachen Fällen durchführbar.

Die Entwicklung adaptiver Regler wurde motiviert durch die begrenzten Möglichkeiten konventioneller *Regelungssysteme*, Prozesse mit unbekannten oder veränderlichen Übertragungseigenschaften zufriedenstellend zu steuern. Die Adaptionsfähigkeit ist somit eine natürliche Erweiterung des klassischen Regelungskonzepts. Diese Erweiterung führt zu einer Überlagerung des konventionellen Regelungssystems durch ein Anpassungssystem (s. Abb.) mit den Funktionen Identifikation (Erkennung), Entscheidungsprozeß (Entwurf), Modifikation (Einstellung). Die Identifikationsstufe verarbeitet die verfügbare Meßinformation zur Systemerkennung (z. B. Parameter- und Zustandsschätzung), wobei A-priori-Information einfließen kann. Der Entscheidungsprozeß umfaßt die Umsetzung der Identifikationsergebnisse in



Adaptives Regelungssystem

eine Reglerveränderung gemäß dem vorgegebenen Gütemaß (z. B. neue Reglerparameter). Die Modifikation schließlich führt die Reglerveränderung aus (z. B. Parameterverstellung), die der Regelung einen nichtlinearen Charakter verleiht.

Bei vielen a. R. ist eine klare Abgrenzung der drei Funktionseinheiten nicht möglich. Je nach regelungstechnischer Aufgabenstellung wird unterschieden in parameteradaptive Regelungen (Die Prozeßparameter sind unbekannt bzw. veränderlich. Der adaptive Regler reagiert mit entsprechenden Reglerparameteränderungen.), signaladaptive Regelungen (Der adaptive Regler paßt sich dem jeweiligen Charakter der Signale, z. B. Führungssignale, an.) sowie strukturadaptive Regelungen (Die Anpassung an unterschiedliche Arbeitsbedingungen erfolgt durch strukturelle Veränderungen — Strukturumschaltungen. Dazu steht i. allg. eine begrenzte Anzahl von (Regler-)Strukturen zur Auswahl.). Gegenwärtig besitzen parameteradaptive Regelungen die größte Bedeutung. Zu ihnen gehören die selbstoptimierenden adaptiven Regelungen und die *modelladaptiven Regelungen.*

Rekursive *Parameterschätzverfahren* bzw. Verfahren zur simultanen Zustands- und Parameterschätzung (adaptive *Beobachter*) sind eine spezielle Form *adaptiver Systeme* zur Lösung von Identifikationsaufgaben. Ein einstellbares Modell gegebener Struktur wird durch selbsttätige Parameterverstellung dem statischen und dynamischen Verhalten eines realen Objekts angepaßt. Als Bestandteil selbstoptimierender adaptiver Regelungssysteme müssen diese Verfahren im On-line/Echtzeitbetrieb arbeiten.

Die Realisierung a. R. erfordert auf Grund des Umfangs und des Charak-

ters (nichtlinear, zeitvariabel) entsprechender Algorithmen den Einsatz von *Digitalrechnern* (Prozeßrechner, *Mikrorechner*). Adaptive Prozeßregelungen sind somit eine spezielle Form von *DDC*-Regelungen.

G. Billerbeck

Literatur: Asher, R. B., D. Andrisani a. P. Dorato: Bibliography on adaptive control systems. Proc. IEEE, New York **64** (1976) 8; Zypkin, J. S.: Adaption und Lernen in kybernetischen Systemen. Berlin: VEB Verlag Technik 1970; Фелдьбаум, А. А.: Теория дуального управления, ч. I–IV. Автоматика и Телемеханика, Москва *9* и *11* (1960), *1* и *2* (1961).

Adaptives System – ein Begriff, der sich auf verschiedene Arten kybernetischer Systeme bezieht, so z. B. gesteuerte Systeme, erkennende Systeme, identifizierende Systeme, Filter usw., die unter Bedingungen der Unbestimmtheit arbeiten, d. h., die in einer Umgebung oder unter Bedingungen arbeiten, über die nicht genügend A-priori-Informationen vorliegen. Als zusammenfassender Begriff ersetzt er solche bisher gebräuchlichen Begriffe wie selbstanpassendes System, selbstorganisierendes System u. a.

Während ein nichtadaptives System die gewünschte Funktion der Steuerung, Erkennung usw. für eine einzige spezielle Umgebung gewährleistet, erfüllt das a. S. diese Funktion für alle Umgebungen einer bestimmten Klasse. Das erste adaptive Steuerungssystem war der adaptive *Autopilot*, der ein Flugzeug unter beliebigen Wetterbedingungen auf einem vorgegebenen Kurs hält.

Zunächst erwartete man, daß man das Verhalten eines zu steuernden Objekts unter a priori unbekannten Bedingungen nach den biologischen und physiologischen Gesetzmäßigkeiten steuern könne, nach denen sich lebende Organismen an eine unbekannte Umgebung anpassen oder adaptieren. Diese Erwartungen haben sich nicht erfüllt, führten aber zu dem Namen „Adaptive Systeme".

Die formale Behandlung a. S., die gegenwärtig die naive Behandlung zunehmend ersetzt, führt zu einer strengen mathematischen Theorie. In der formalen Theorie wird ein zu steuerndes Objekt und ein Ziel der Steuerung eingeführt. Die gewöhnliche, klassische Steuerungstheorie untersucht Methoden, *Algorithmen*, *Strategien*, um das gewünschte Ziel für das betrachtete Objekt zu erreichen. Im Unterschied dazu werden in der Theorie der adaptiven Steuerung Algorithmen oder Strategien zur Steuerung für eine Klasse von Objekten abgeleitet, wobei vorher und während der Steuerung nicht bekannt ist, welches Objekt der Klasse zu steuern ist. Die fehlende Information über das zu steuernde Objekt führt dazu, daß im Vergleich zur nichtadaptiven Steuerung die „Geschwindigkeit" der Annäherung an das Ziel langsamer ist und die Algorithmen komplizierter sind. In der Regel sind

adaptive Strategien nicht stationär, benötigen ein großes Gedächtnis und verwenden die Vorgeschichte des zu steuernden Prozesses. Das Ziel wird allgemein für den Grenzfall einer unendlich langen Zeit formuliert.

Eine relativ allgemeine Methode, eine adaptive Strategie zu erhalten, ist die sog. identifizierende Methode. Sie besteht darin, gleichzeitig unbekannte Parameter des zu steuernden Systems zu schätzen und nach den jeweils aktuell geschätzten Parametern die optimale, zum Ziel führende Strategie zu berechnen. Diese Vorgehensweise ist verbreitet, jedoch nicht universell. Andere Methoden sind spezifischer zugeschnitten auf Objekte mit diskreter oder kontinuierlicher Zeit, zur Optimierung eines Funktionals oder zur Stabilisierung eines Prozesses. Das Problem der Stabilisierung bei stetiger Zeit wird seit den 50er Jahren in einer sehr großen Anzahl von Arbeiten für Klassen von linearen *Differentialgleichungen* untersucht. Dabei ist eine Strategie zu finden, die die Lösung einer unbekannten Differentialgleichung der betrachteten Klasse asymptotisch stabil oder dissipativ macht. Oftmals wird die Strategie in Form einer quadratischen Differentialgleichung angegeben. Für diskrete Zeit werden besonders lineare Differenzengleichungen betrachtet. Besondere Aufmerksamkeit ruft dabei die sog. Etalon-Aufgabe hervor. Die Bewegung eines beliebigen Objekts der betrachteten Klasse ist bei dieser Aufgabe an eine vorgegebene Bewegung des *Etalons* anzunähern.

Am weitesten entwickelt scheinen adaptive Optimierungsaufgaben mit diskreter Zeit zu sein, obwohl sie erst seit Mitte der 60er Jahre bearbeitet werden. Zur Zeit kann die optimale adaptive Steuerung von Klassen homogener MARKOW-Ketten als abgeschlossen angesehen werden. Es wurde gezeigt, daß adaptive Strategien existieren, die ein *Funktional* – den Grenzwert des mittleren Gewinns pro Schritt – für eine beliebige unbekannte MARKOW-*Kette* optimieren. Diese Strategien sind konstruktiv bestimmt. Auch ist die Aufgabe gelöst, das *Extremum* für ein Funktional bei Beschränkungen für andere Funktionale zu bestimmen. Besonders zahlreich sind die Arbeiten zur Steuerung von für die Anwendungen wichtigen homogenen Prozessen mit unabhängigen Zufallsgrößen. Für diese Prozesse ist die Methode der *stochastischen Approximation* ausgearbeitet worden, und *endliche Automaten* wurden erstmals zu ihrer Steuerung verwandt. Die Theorie der Steuerung von MARKOW-*Prozessen* mit stetiger Zeit wird gegenwärtig erarbeitet. Im Falle kompakter Zustandsmengen wurden fast abschließende Resultate erzielt. Außerdem wird die Steuerung von gewissen Klassen von *stochastischen Differentialgleichungen* untersucht, wobei es bisher aber noch wenig fundierte Resultate gibt. Ferner existiert eine Theorie der adaptiven Steuerung *stationärer Zufallsprozesse*. Begonnen wurden auch Untersuchungen von teilweise beobachtbaren MARKOW-Ketten.

Auch in der *mathematischen Statistik* entwickelten sich adaptive Verfahren,

unter ihnen sog. robuste Verfahren, d. h. Verfahren, die unempfindlich bezüglich der Art der *Wahrscheinlichkeitsverteilung* der betrachteten *Zufallsgröße* sind.

W. G. SRAGOWITSCH

Literatur: Срагович, В. Г.: Адаптивное управление. Москва: Изд. Наука 1981.

Aggregat-Modellierung großer Systeme – Konzeption der algorithmischen Modellierung. Die Methodologie der algorithmischen Modellierung wird gegenwärtig auf der Ebene der *mathematischen Modellierung* und auf der Ebene der programmtechnischen Umsetzung weiterentwickelt. Die Methodik der von BUSLENKO begründeten Aggregat-Darstellung besteht aus den folgenden zwei Etappen: 1. Das zu untersuchende System wird als eine Gesamtheit relativ selbständiger Aggregate dargestellt, die in Abhängigkeit vom System selbst abgesondert werden. Diese Aggregate können durch vereinheitlichte *mathematische Modelle* beschrieben werden. 2. Das Verhalten als System insgesamt wird durch ein sog. A-System realisiert, das sich durch Herstellung der Verbindung zwischen den Aggregaten ergibt.

Die Modellierung beruht auf der Zustandsdarstellung, wobei der Zustandsraum in disjunkte Gebiete aufgeteilt wird. Der *Algorithmus* teilt sich auf in a) Zustandserkennung, d. h. Zuordnung des konkreten Systemzustands zu einer Klasse sich bezüglich der Zustandsänderung entsprechender Systemzustände, und b) Realisierung der dieser Klasse entsprechenden Entscheidung über den weiteren Verlauf der *Trajektorie*.

Modellierung des Aggregats: Grundlegendes Prinzip der Modellierung des Systems, das als Aggregat bezeichnet wird, ist die Zustandsdarstellung. Dabei gilt $\boldsymbol{x} \in X$, wobei $\boldsymbol{x}$ den Zustandsvektor und X den n-dimensionalen Zustandsraum darstellen. Im allgemeinen ändern sich die *Zustandsgrößen* mit der Zeit, d. h., der Vektorfunktion $\boldsymbol{x}(t)$ entspricht eine Trajektorie im Zustandsraum. Dabei kann $x_i(t)$ eine Realisierung einer *Zufallsgröße* $X_i(t)$ mit bekannter *Wahrscheinlichkeitsverteilung* darstellen. Ist das Aggregat autonom, so wird sein Verhalten vollständig durch seinen Anfangszustand $\boldsymbol{x}(t_0)$ gekennzeichnet. Unter der Voraussetzung, daß die Systemparameter und die Systemstruktur konstant bleiben, ergibt sich

$$\boldsymbol{x}(t) = H[\boldsymbol{x}(t_0), t] ,$$

wobei der Übergangsoperator H im allgemeinen einen Zufallsoperator darstellt. In bestimmten Zeitpunkten t_j wirken auf das System durch die Umgebung bedingte diskrete äußere Einwirkungen ein, so Eingangsgrößen $\boldsymbol{f}^j \in F$, die in den Zeitpunkten t_j $(j=1, 2, \ldots, t_{j+1} \geqq t_j)$ eintreten. Im allgemeinen kann $(t_j, \boldsymbol{f}^j)$ eine Realisierung eines Zufallsvektors mit bekannter Wahrscheinlichkeitsverteilung darstellen, wobei $\boldsymbol{f}^j$ als Eingangsvektor be-

zeichnet wird. Die Trajektorie des Aggregats kann durch zielgerichtete Einwirkungen (Steuergrößen) $\boldsymbol{u}^i \in U$ gesteuert werden, die in den Zeitpunkten $\tau_i (i=1, 2, \ldots, \tau_{i+1} \geqq \tau_i)$ eintreten. Auch der Steuervektor $\boldsymbol{u}^i$ stellt im allgemeinen eine Realisierung eines Zufallsvektors dar.

Das Aggregat besitzt nun einige grundlegende Zustände $r=1, 2, \ldots, N$. In jedem dieser Zustände r ist das Aggregat gekennzeichnet durch die Zustandsgrößen $\boldsymbol{x}^r(t)$, wobei $\boldsymbol{x}^r(t) \in \Delta X_r \subset X$. ΔX_r ist ein bestimmtes Gebiet im n-dimensionalen Zustandsraum. Diese Gebiete $r \in R$ besitzen dabei die folgende Eigenschaft: Wenn gilt

$$\boldsymbol{x}(t_j) \in \Delta X_r, \quad \boldsymbol{x}(t_j - \varepsilon) \overline{\in} \Delta X_r, \quad \varepsilon > 0, \quad \text{oder}$$

$$\boldsymbol{x}(t_j + 0) \in \Delta X_r, \quad \boldsymbol{x}(t_j) \in \Delta X_r \quad \text{und} \quad r \in R,$$

wobei für beliebiges $t > t_j$ gilt $t_j + 0 \in (t_j, t)$, so entsteht ein Ausgangssignal $\boldsymbol{y}_r^j$ mit

$$\boldsymbol{y}_r^j = G_r[\boldsymbol{x}(t_j), \boldsymbol{u}^s],$$

wobei der Ausgangsoperator G im allgemeinen einen Zufallsoperator darstellt. Auf der Trajektorie $\boldsymbol{x}(t)$ des Aggregats werden darüber hinaus einige besondere Zustände unterschieden. Der Übergangsoperator H hängt davon ab, ob im betrachteten Zeitintervall besondere Zustände enthalten sind. Befindet sich das Aggregat in einem solchen Zustand, so geht es sprungförmig in einen neuen Zustand über. Als besondere Zustände werden unterschieden: 1. das Eintrittsmoment eines Eingangssignals $\boldsymbol{f}^j \in F$, 2. das Eintrittsmoment eines Steuersignals $\boldsymbol{u}^j \in U$, 3. das Eintrittsmoment eines Steuersignals $\boldsymbol{u}^j \in U$ und eines Eingangssignals $\boldsymbol{f}^j \in F$, 4. das Austrittsmoment eines Ausgangssignals $\boldsymbol{y}^j$.

Stückweise lineare Aggregate: Als Spezialfall der Aggregate wurden von KOWALENKO die stückweise linearen Aggregate eingeführt. Für diese Aggregate ist kennzeichnend, daß sich der Zustandsvektor $\boldsymbol{x}_i(t)$ innerhalb des Gebietes ΔX_i, $i \in R$, des n-dimensionalen Zustandsraumes nach der linearen Differentialgleichung

$$\frac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{x}_i(t) = v_i, \quad i \in R,$$

ändert, wobei v_{ik} bestimmte Systemkonstanten darstellen. Die Gebiete ΔX_i stellen *konvexe Polyeder* dar, die im allgemeinen durch Beziehungen der Art

$$\bigvee_r \bigwedge_j \left\{ \sum_k a_{rjk}^i x_{ik} + a_{rj0}^i \geqq 0 \right\}$$

bestimmt sind. Der Zustandsvektor $\boldsymbol{x}_i(t)$ bewegt sich auf der Trajektorie. Trifft die Trajektorie in t_1 auf eine der das Polyeder ΔX_i begrenzenden Hyperflächen G_k^i, d. h., genügt der Zustandsvektor für t_1 den Gleichungen

der Hyperfläche, so erfolgt für t_1 eine Zustandsänderung des Aggregats, und außerdem wird ein Ausgangssignal ausgesondert. Analog ergeben sich die Zustandsänderungen des stückweise linearen Aggregats, wenn Eingangsgrößen oder Steuergrößen eintreten.

A-Systeme: Existiert eine solche meist nicht eindeutige Aufgliederung des Systems in Elemente, wobei jedes erhaltene Element ein Aggregat darstellt, so wird ein solches System ein System von Aggregaten oder ein A-System genannt. Die Aufgabe der Strukturanalyse und Synthese von A-Systemen sowie die Entwicklung von Algorithmen der maschinellen Analyse und Synthese erfordern formale Regeln der Beschreibung verschiedener Konstruktionen des Aggregats sowie der Wechselwirkung untereinander, die bereits weitgehend ausgearbeitet sind.

J. A. MÜLLER

Literatur: MÜLLER, J. A.: Einige Möglichkeiten zur Beschreibung großer Systeme. messen. steuern. regeln, Berlin **14** (1971) 11 u. **14** (1971) 12; БУСЛЕНКО, Н. Б.: Моделирование сложных систем. Москва: Изд. Наука 1978.

Algorithmen, Äquivalenz von – (s. a. Teil A–E, S. 61) binäre Relation über Algorithmen eines bestimmten Typs. Bei auf eine bestimmte Art äquivalenten Ausgangsdaten sind zwei Algorithmen (schwach) äquivalent, falls die Resultate der Berechnung übereinstimmen (bzw. stark äquivalent, falls die Resultate der Berechnung und die Berechnungsgeschichte übereinstimmen).

Einige Beispiele:

Es werden rekursive Schemata betrachtet, d. h. Gleichungssysteme, die n-stellige partiell rekursive Funktionen definieren. Zwei Schemata, die die Funktionen φ und ψ definieren, heißen funktional äquivalent, falls für beliebige natürliche Zahlen $x_1, x_2, \ldots, x_n$, für die φ und ψ definiert sind, die Gleichung

$$\varphi(x_1, x_2, \ldots, x_n) = \psi(x_1, x_2, \ldots, x_n)$$

gilt.

S. C. KLEENE zeigte, daß für einen überall definierten Operator $\boldsymbol{R}$ über rekursiven Schemata ein Schema S existiert, so daß S und $\boldsymbol{R}(S)$ funktional äquivalent sind (KLEENEscher Fixpunktsatz). Hieraus folgt der Satz von USPENSKIJ-RICE über die Nichtentscheidbarkeit einer beliebigen, bez. der funktionalen Äquivalenz invarianten Eigenschaft rekursiver Schemata unter der Bedingung, daß Schemata S_1 und S_2 existieren, die sich bez. dieser Eigenschaft unterscheiden. Daraus folgt die Nichtentscheidbarkeit der funktionalen Äquivalenz.

Es werden MARKOW-Algorithmen über einem bestimmten Alphabet A betrachtet. Zwei MARKOW-Algorithmen $\mathfrak{A}_1$, $\mathfrak{A}_2$ heißen äquivalent bez. A, falls für jedes Wort p, $p \in A^*$, aus der Eigenschaft $\mathfrak{A}_1(p) \in A^*$ die Gültigkeit der Gleichung $\mathfrak{A}_1(p) = \mathfrak{A}_2(p)$ folgt. Zwei Algorithmen heißen vollständig äquivalent bez. A, wenn für alle Worte p, $p \in A^*$, für die $\mathfrak{A}_1$ und $\mathfrak{A}_2$ definiert sind, die Gleichung $\mathfrak{A}_1(p) = \mathfrak{A}_2(p)$ gilt. Beide Probleme sind nicht entscheidbar.

Es werden verschiedene Programmschemata untersucht. Ein Standardprogrammschema wird induktiv graphisch oder funktional über einer Basis, bestehend aus Variablen, Funktionssymbolen, Prädikatensymbolen und den Sondersymbolen start, stop, definiert. Zwei Standardprogrammschemata heißen funktional äquivalent, falls bei jeder Interpretation der Basis beide zyklisch werden oder beide mit dem gleichen Resultat die Berechnung beenden. N. A. KRINITZKI zeigte, daß für Schemata ohne Zyklen das Äquivalenzproblem entscheidbar ist. A. A. LETITSCHEWSKI, D. LUCKHAM, D. M. PARK und M. S. PATERSON zeigten die Nichtentscheidbarkeit des Äquivalenzproblems für Standardprogrammschemata. N. A. MARTINJUK wies die Nichtentscheidbarkeit der syntaktischen Äquivalenz von Standardprogrammschemata nach. Zwei Standardprogrammschemata heißen syntaktisch äquivalent, falls die Mengen der Berechnungswege übereinstimmen. Zwei Standardprogrammschemata heißen schwach äquivalent, falls, wenn beide ihre Berechnung beenden, die Resultate gleich sind. Die schwache Äquivalenz und jede stärkere Äquivalenz sind nicht entscheidbar. Entscheidbar ist das Äquivalenzproblem für JANOW-Schemata, d. h. Standardprogrammschemata mit nur einstelligen Funktions- und Prädikatensymbolen. Diese Lösung gilt auch noch, falls Konstanten und Hilfsregistervariablen zugelassen werden. Es existiert ein vollständiges Verifikationssystem.

Es existieren vielfältige weitere Untersuchungen einzelner Äquivalenzbeziehungen von Klassen von Algorithmen, die insbesondere von praktischen Problemen wie der Übersetzung von Algorithmen (in eine andere algorithmische Sprache) und der Optimierung von Algorithmen stimuliert werden. Besonders werden Klassen von Algorithmen untersucht, für die das Äquivalenzproblem entscheidbar ist und vollständige Verifikationssysteme existieren.

B. THALHEIM

Literatur: GREIBACH, S. A.: Theory of program structures – schemes, semantics, verification. Berlin, Heidelberg, New York: Springer-Verlag 1975 (Lecture Notes in Computer Science, Bd. 36); KLEENE, S. C.: Introduction to metamathematics. Amsterdam: North Holland Publ. Co. 1952; КОТОВ, В. Е.: Введение в теорию схем программ. Москва: Изд. Наука 1978; МАРКОВ, А. А.: Теория алгоритмов. Труды мат. института АН СССР, т. 42, Москва 1954.

Algorithmen, Klassifizierung – Gliederung von *Algorithmen* nach inhaltlichen und modellseitigen Kriterien. Nach dem Charakter der zu verarbeitenden Information werden Algorithmen mit vollständiger und mit unvollständiger Information unterschieden.

Algorithmen mit vollständiger Information, bei denen alle wesentlichen Eigenschaften des gesteuerten Prozesses, die Einflußgrößen auf diesen Prozeß, die Zustände und Transformationsoperationen des Prozesses und die Ergebnisse der Transformationen bekannt sind, existieren in Form von Algorithmen mit streng determinierten Übergängen, bei denen das Ergebnis des Überganges von einem Prozeßzustand in einen anderen eindeutig bestimmt werden kann, sowie in Form von Algorithmen mit nicht streng determinierten Übergängen, bei denen die Einwirkung eines Operators O_n auf einen Prozeßzustand a nicht eindeutig zu einem anderen Prozeßzustand b, sondern mit Übergangswahrscheinlichkeiten p_i zu mehreren Prozeßzuständen $b, c, \ldots, m$ führt. Nach der folgenden Tabelle kann der Übergang z. B. mit

Übergangswahrscheinlichkeiten

→	b	c	d
O_n (a)	0,7	0,2	0,1

den Werten $p_b = 0{,}7$; $p_c = 0{,}2$ und $p_d = 0{,}1$ erfolgen. Bei Algorithmen mit unvollständiger Information sind Eigenschaften, Einflußgrößen, Zustände, Transformationsoperatoren und Ergebnisse der Transformationen nicht oder nur teilweise bekannt. Ihre Modellierung kann unter Einsatz von sog. Fuzzy-Methoden *(s. Unscharfe Modelle)* erfolgen.

Die Abstufung nach dem Kompliziertheitsgrad der Struktur von Algorithmen umfaßt fünf Stufen: 1. Lineare Algorithmen mit sequentiellem Ablauf der funktionalen Operatoren $O_1, O_2, \ldots, O_i$ ohne logische Operatoren E_j (Verwendung für elementare Prozesse, z. B. betriebliche Kostensummierung in Abrechnungsprozessen); 2. Algorithmen als Folge funktionaler Operatoren $O_1, O_2, \ldots, O_i$ mit einzelnen logischen Operatoren E_j, die zu vorwärtsgerichteten Sprüngen oder zu zyklischer Wiederholung von Operationen führen (z. B. Durchführung des konstruktiven und technologischen Änderungsdienstes); 3. Algorithmen als Folgen funktionaler und logischer Operatoren O_i, E_j unter Verwendung einzelner *Unterprogramme* U_k, die selbständige Teilprozesse auslösen (z. B. Änderungen des operativen Produktionsplanes eines Betriebes mit selbständiger Disposition von Maschinenzeit und Arbeitszeit-Ressourcen); 4. Algorithmen als Folge von Unterprogrammabläufen $U_1, U_2, \ldots, U_r$, die nach einem Superprogramm bausteinartig zusammengefügt werden (z. B. Erarbeitung von Planvarianten für die Reihenfolge der

Auftragsbearbeitung und der Maschinenbelegung); 5. Selbständige Ableitung von Algorithmen aus einer definierten Ziel- und Aufgabenstellung als Auswahl oder Kombination einer Menge existierender oder möglicher Abläufe bzw. im Ergebnis eines Lernprozesses.

Die Klassifizierung nach typischen inhaltlichen Funktionen des zu steuernden Prozesses unterscheidet in der *ökonomischen Kybernetik* für mikroökonomische Prozesse folgende Gruppen von Algorithmen: Algorithmen der Prognose, der Strategienbildung und der langfristigen Planung; Algorithmen der konstruktiven und technologischen Vorbereitung der Produktion; Algorithmen der technisch-ökonomischen Jahresplanung; Algorithmen der organisatorischen Produktionsvorbereitung; Algorithmen der *operativen Lenkung der Produktion* und der Kontrolle; Algorithmen zur Steuerung der Absatztätigkeit sowie Algorithmen zur Abrechnung und Analyse des betrieblichen Reproduktionsprozesses.

Entsprechend der Stellung der Algorithmen in der Steuerhierarchie existieren zwei Kategorien, und zwar Einebenen-Algorithmen zur Steuerung horizontal strukturierter Prozesse (z. B. Steuerung der Vorfertigung, der Baugruppen-Montage, der Oberflächenbehandlung und der Endmontage in Produktionssystemen des Fahrzeugbaus) und Mehrebenen-Algorithmen (s. *Hierarchischer Steueralgorithmus*) zur Steuerung komplexer Prozesse mit hierarchischer Struktur, z. B. zur Steuerung eines betrieblichen Reproduktionsprozesses (als (0)-Ebene) mit seinen nachgeordneten Teilprozessen Produktionsvorbereitung, Produktionsdurchführung, Material- und Lagerwirtschaft, Absatzorganisation usw. (als (-1)-Ebene) innerhalb des übergeordneten Reproduktionsprozesses des Kombinates (als $(+1)$-Ebene). Mehrebenen-Algorithmen bilden die typische Form der Steuermodelle *großer Systeme*.

H.-G. Lauenroth

Algorithmensystem – Gesamtheit horizontal und/oder vertikal verknüpfter Algorithmen zur Funktionsbeschreibung und zur Steuerung *großer Systeme*. Ein A. besteht nach

$$S_{\Pi} = [M_{\Pi}, R_{\Pi}] \tag{1}$$

– aus der Menge der Algorithmen

$$M_{\Pi} = \{\Pi_{\mathrm{P}}, \Pi_{\mathrm{St}}\} \tag{2}$$

mit

$$\Pi_{\mathrm{P}} = \{\Pi_{\mathrm{P}_1}, \Pi_{\mathrm{P}_2}, \ldots, \Pi_{\mathrm{P}_r}\} \tag{3}$$

als der Menge der Algorithmen des Ablaufs des gesteuerten Prozesses (Produktions-, Transport-, Lagerprozesse in industriellen Systemen) mit den

produktionstechnologischen Teilalgorithmen $\Pi_{P_i} \in \Pi_P$ und mit

$$\Pi_{St} = \{\Pi_{St_1}, \Pi_{St_2}, \ldots, \Pi_{St_j}\} \tag{4}$$

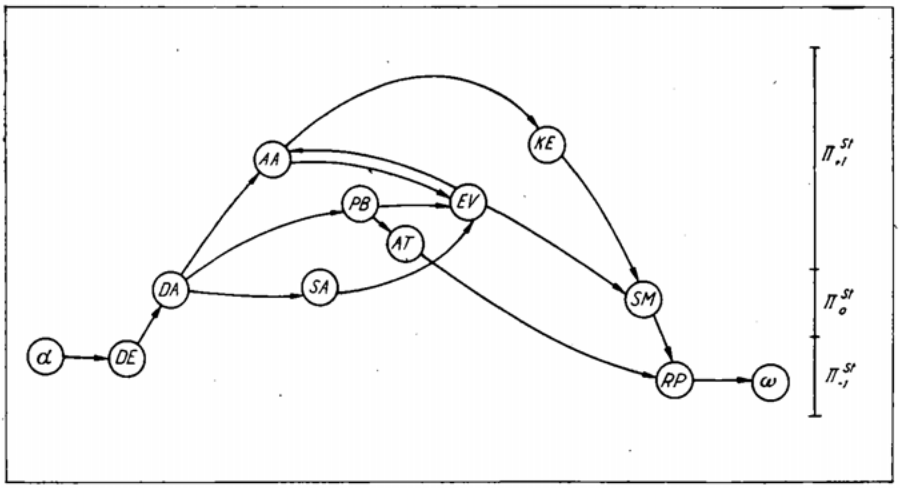
als der Menge der Algorithmen des Ablaufs der prospektiven und korrektiven Steuerung (Steuerprozesse) mit den informationstechnologischen Teilalgorithmen $\Pi_{St_m} \in \Pi_{St}$;

– aus der Relationsmenge

$$R_\Pi = \{R_{\Pi_1}, R_{\Pi_2}, R_{\Pi_3}\} \tag{5}$$

mit den Relationen R_{Π_1} als Kopplungen zwischen den produktionstechnologischen Teilalgorithmen $\Pi_{P_i} \in \Pi_P$, den Relationen R_{Π_2} als Kopplungen zwischen den informationstechnologischen Teilalgorithmen $\Pi_{St_m} \in \Pi_{St}$ und den Relationen R_{Π_3} als Kopplungen zwischen den Algorithmen Π_P und Π_{St} zur Sicherung einer effektiven Funktion des Steuersystems als Ganzes.

In großen Systemen sind die Steueralgorithmen Π_{St} hierarchisch aufgebaut (s. *Hierarchischer Steueralgorithmus*); für die Steuerung des Reproduktionsprozesses eines Fahrzeugbetriebes zum Beispiel erhält ein A. die in der Abb. dargestellte Form. Die Tab. gibt den Inhalt der Algorithmen an. In der unteren Ebene der Steuerung Π^{St}_{-1} werden die Algorithmen der Datenerfassung DE (nach Start-Symbol α) und der Durchsetzung der in den übergeordneten Ebenen Π^{St}_0 und Π^{St}_{+1} getroffenen Entscheidungen zur Prozeß-



Komplexes hierarchisches Algorithmensystem zur Steuerung eines Fahrzeugbetriebes

Inhalt der Algorithmen des komplexen hierarchischen Algorithmensystems

Lfd. Nr.	Bezeichnung	Inhalt
1	DE	Zeitgerechte Datenerfassung aus Produktion, Material- und Lagerwirtschaft, Transport und Absatz
2	DA	Aufbereitung der Daten und Bereitstellung für analytische und vorausschauende Berechnungen
3	PB	Berechnung technisch-ökonomischer Parameter in definierten Kontrollfrequenzen
4	SA	Situationsanalyse über den Stand der Abweichungen und die Notwendigkeit von Steuermaßnahmen
5	AA	Analyse der Ausrüstungen, Maschinengruppen und Technologien unter Berücksichtigung von Instandhaltungsmaßnahmen
6	EV	Vorbereitung lang-, mittel- und kurzfristiger Entscheidungen unter Verwendung von Optimierungs- und Simulationsprogrammen
7	AT	Ermittlung von Änderungstendenzen technisch-ökonomischer Parameter
8	KE	Berechnung der Kapazitätsentwicklung unter Berücksichtigung von Investitionsvorhaben und Kooperationsbeziehungen
9	SM	Erarbeitung von Steuermaßnahmen über die Umverteilung von Ressourcen sowie die Änderung von Führungsgrößen
10	RP	Realisierung getroffener Entscheidungen zur Stabilisierung und Weiterentwicklung des betrieblichen Reproduktionsprozesses

steuerung RP realisiert; mit dem Ende-Symbol ω ist ein Steuerzyklus abgeschlossen. Die Entscheidungen tragen lang-, mittel- und kurzfristigen Charakter, sie werden unter Berücksichtigung gegenwärtiger Situationen und voraussehbarer künftiger Entwicklungen getroffen.

A. bilden das Kernstück *automatisierter Leitungssysteme*.

H.-G. Lauenroth

Algorithmische Logik – A. L. umfassen eine Vielzahl formaler Kalküle, deren Entwicklung durch die maschinelle Informationsverarbeitung angeregt wurde und die eine Erweiterung der klassischen Logik, insbesondere des *Prädikatenkalküls erster Stufe*, darstellen. Charakteristisch für diese Erweiterungen ist das Vorhandensein von syntaktischen Objekten und Konnektoren

für die Bildung und Repräsentation von *Programmen* als Beschreibung von *Algorithmen.*

Die ersten Systeme dieser Art wurden 1958 von J. YANOV für den Nachweis der Äquivalenz von Programmen und 1962 von J. MCCARTHY für den Korrektheitsnachweis geschaffen. Das erste logische System für eine a. L. mit vollständig ausgearbeiteter Syntax und Semantik entwickelte 1966 H. THIELE. E. ENGELER interpretierte Programme durch unendlich lange Formeln (1967), und C. A. R. HOARE schuf 1969 einen Kalkül für den Beweis der Korrektheit von Programmen unter Verwendung einer prädikativen Semantik. Die von A. SALWICKI entwickelten a. L. ähneln den höheren *Programmiersprachen* vom Typ *ALGOL.* Die Idee der dynamischen Logiken von V. PRATT (1976) bestand darin, Programme als modale Operatoren aufzufassen und eine relationale Semantik, die der KRIPKE-Semantik ähnelt, zugrunde zu legen. Weitere inzwischen entwickelten Kalküle sind die effektiven Definitionen von J. TIURYN, die auf einer Formalisierung des Algorithmenbegriffes von H. FRIEDMAN aufbauen, sowie die Programmlogik von R. L. CONSTABLE (1978).

A. L. sind eine geeignete Grundlage für die Formalisierung und die Untersuchung von Konzepten der Software-Technologie, z. B. der Syntax und Semantik von Programmiersprachen, der Datenstrukturen, der Korrektheit von Programmen sowie der Ausdrucksfähigkeit und der Klassifikation von Programmschemata. Für das Studium dieser Fragen werden Methoden der *mathematischen Logik* eingesetzt. Im Mittelpunkt stehen Untersuchungen zur Axiomatisierbarkeit und Interpretierbarkeit sowie zur *Modelltheorie* dieser Systeme. Von besonderem Interesse ist der Aufbau von Beweissystemen, die eine wichtige Grundlage für die *Programmverifikation* darstellen.

Diese Resultate liefern wichtige Beiträge für den Entwurf von Programmiersprachen, die Verifikation und Synthese von Programmen sowie die Theorie der Datenstrukturen. Darüber hinaus bilden die a. L. eine geeignete Grundlage für die Ausbildung von Informatikern.

H. HERRE

Literatur: YANOV, J.: On equivalence of operator schemes. Problems of Cybernetics **1** (1959), 1–100; MCCARTHY, J.: Towards a mathematical science of computation. In: Information Processing 1962. Proc. of IFIP Congress 62, ed. by C. M. POPPLEWELL. Amsterdam: North-Holland Publ. Co. 1963; THIELE, H.: Wissenschaftstheoretische Untersuchungen in algorithmischen Sprachen. Berlin: Dt. Verl. d. Wiss. 1966; ENGELER, E.: Algorithmic properties of structures. Mathem. Systems Theory, Berlin, Heidelberg, New York **1** (1967), 183–195; HOARE, C. A.: An axiomatic basis for computer programming. Comm. of ACM, New York **12** (1969), 576–583; SALWICKI, A.: Formalized algorithmic language. Bull. Acad. Pol. Science,

Warszawa 18 (1970), 227–232; FRIEDMAN, H.: Algorithmic procedures, generalized Turing algorithms and elementary recursion theorie. In: Logic Colloquium '69, ed. by GANDY/YATES. Amsterdam: North-Holland Publ. Co. 1971; PRATT, V.: Semantical considerations of Floyd-Hoare logic. Proc. 17th Symposium on Foundations of Computer Science, Houston 1976; CONSTABLE, R. L.: On the theory of programming logics. Proc. 9th Annual ACM Symposium on Theory of Computing, New York 1978; TIURYN J.: A survey of the logic of effective definitions. MIT/LCS TR-246, 1980.

Algorithmus, deterministischer – *Algorithmus* im Sinne der klassischen mathematischen *Algorithmentheorie*.

Algorithmus, effizienter – Begriff wird einerseits unpräzisiert verwendet für einen günstigen *Algorithmus* (z. B. bezüglich Speicher- und/oder Zeitaufwand, Unterbrechbarkeit oder spezieller Kriterien), andererseits im Sinne der Komplexitätstheorie für *polynomialen Algorithmus* gebraucht (s. a. *Komplexität*).

Algorithmus, nichtdeterministischer – *Algorithmus*, bei dem es Elementarschritte gibt, in denen aus einer Menge (von möglichen Fortsetzungen, Instruktionen, Elementen, . . .) beliebig gewählt werden kann. Die Anwendung eines n. A. führt zu einem Ergebnis aus einer Menge von möglichen Ergebnissen. Mathematisch präzisiert ist der Begriff durch die Arbeitsweise einer „nichtdeterministischen TURING-Maschine", d. h. einer TURING-*Maschine*, deren *Zentraleinheit* ein *nichtdeterminierter Automat* ist.

HEINZ D. MODROW

Algorithmus, polynomialer – *Algorithmus* mit einer Zeitkomplexität von der Ordnung eines Polynoms; s. *Komplexität* (von Algorithmen).

Amplitudengang – s. *Frequenzgang*.

Anfahrautomatik – Regelungs- oder Steuereinrichtungen zur Erfüllung spezieller Forderungen bezüglich dynamischer Übergangsvorgänge bei der Inbetriebnahme (Anfahren) einer automatisierten technischen Anlage. Die A. enthält neben Überwachungs- und Steuereinrichtungen für Sicherheitsfunktionen auch Regler, deren Struktur und Parameter zur Erzielung optimaler Einschwingvorgänge veränderbar sind. Häufig werden Baueinheiten der A. auch für das Abfahren technischer Anlagen (Abfahrautomatik) verwendet. Neben den zeitabhängigen oder ereignisabhängigen Parameterveränderungen des Hauptprozesses (z. B. zulässige Temperaturgradienten) muß die A. eine Vielzahl von Hilfsvorgängen (z. B. Entwässerungsventile bei

Dampfleitungen, Spülvorgänge) steuern. Dadurch muß die A. gewöhnlich wesentlich mehr Signale verarbeiten und Stellbefehle (überwiegend binär) abgeben als die Automatik für den Normalbetrieb.

R. MÜLLER

Antihavarietraining – Durchführen von anlagenspezifischen Schulungs- und Übungsprogrammen zur Vorbereitung des Anlagen- und Wartenpersonals automatischer technischer Anlagen (Kraftwerke, verfahrenstechnische Anlagen) auf ihre Arbeitstätigkeit, insbesondere zur Bekämpfung von Betriebsstörungen. Das A. wird meist in Simulationsanlagen durchgeführt, wobei entweder nur einige wichtige Anlagenzustände und dynamische Prozeßverläufe simuliert oder alle Reaktionen und Eigenschaften der Anlage komplex und zeitlich nachgebildet werden (Kraftwerkstrainer). Das Ziel des A. ist das Erlernen und Üben von Arbeitsabläufen in bestimmten Situationen des Normal-, An- und Abfahrbetriebes, von Algorithmen bei der Störerkennung und von technischen und organisatorischen Gegenmaßnahmen bei eingetretenen Störungen. Das Personal soll in die Lage versetzt werden, außergewöhnliche Anlagenzustände sicher zu beherrschen und Havarien zu verhindern.

R. MÜLLER

Attributgrammatik – Attributgrammatiken wurden 1968 von KNUTH zur formalen Beschreibung der Semantik in *Programmiersprachen* eingeführt. Wegen ihrer günstigen Eigenschaften finden sie häufig Verwendung in Compiler-Generatoren (s. *Compiler-Compiler*).

Eine Attributgrammatik ist ein 5-Tupel $AG=(G, A, Dom, R, C)$. Dabei ist $G=(V_T, V_N, P, S)$ eine reduzierte (enthält nur notwendige Produktionen) *kontextfreie Grammatik* mit der Terminalmenge V_T, der hierzu disjunkten Metasymbolmenge V_N, der Regelmenge P und dem Startsymbol $S \in V_N$. Alle Regeln $p \in P$ haben die Form $p\colon X_0^p \to X_1^p X_2^p \ldots X_{n_p}^p$ mit $X_0^p \in V_N$ und $X_i^p \in V = V_T \cup V_N$ $(i=1, \ldots, n_p)$. Alle Sätze der durch G definierten Sprache entstehen durch Ableitung aus S, indem in beliebiger Reihenfolge alle Metasymbole durch Anwendung von Regeln p eliminiert werden. Dabei wird X_0^p stets durch die rechte Seite der Regel ersetzt.

Jedem Symbol $X \in V$ wird nun eine Menge $A(X)$ von Attributen zugeordnet, wobei $A(X) \in A$, der Menge der vorhandenen Attribute. Der Wertebereich jedes Attributes a wird durch $Dom\,(a) \in Dom$ definiert.

Die Menge R enthält die Berechnungsregeln für die Attribute. Jeder Produktion p werden die Attributregeln $R(p)$ zugeordnet, die für jedes Attribut a, das in $A(X_j^p)$ mit $j=0, \ldots, n_p$ enthalten ist, die Berechnung aus den anderen Attributen dieser Menge definieren: $X_j^p \cdot a = f_{ja}^p(X_0^p \cdot a_1, \ldots, X_{n_p}^p \cdot a_l)$.

Schließlich werden jeder Produktion p Kontextbedingungen $g^p(X_0^p \cdot a_1, \ldots, X_{n_p}^p \cdot a_l)$ zugeordnet. Diese zweiwertigen Relationen müssen bei der Ableitung von Sätzen der durch AG definierten Sprache $L(AG)$ erfüllt sein, so daß sie stets eine Untermenge der durch die zugehörige kontextfreie Grammatik definierten Sprache $L(G)$ ist.

Für die praktische Verwendung von Attributgrammatiken zur Definition von Programmiersprachen und deren Nutzung in Compiler-Generatoren werden im allgemeinen weitere Bedingungen an die Semantikfunktionen f_{ja}^p und g^p gestellt, die eine effektive Auswertung ermöglichen. Auch aus diesem Grunde existieren verschiedene Definitionen von Attributgrammatiken.

W. ASSMANN

Literatur: KNUTH, D.: Semantics of context-free languages. Mathem. Systems Theory, New York **2** (1968) 2; BOCHMANN, G. V.: Semantic evaluation from left to right. Communications of the ACM, New York **19** (1976) 2.

Aufgabenverfügbarkeit — s. *Verfügbarkeit.*

Ausfall – Ereignis, das die Arbeitsfähigkeit einer Betrachtungseinheit für immer oder vorübergehend beendet. Dabei versteht man unter Arbeitsfähigkeit jeden Zustand der Betrachtungseinheit, in dem sie ihre Funktion erfüllen kann und ihre Hauptparameter in zulässigen Grenzen bleiben. In diesem Sinne ist nicht jede Beschädigung der Betrachtungseinheit ein Ausfall. Unwesentliche Beschädigungen verletzen lediglich die „Unversehrtheit" der Betrachtungseinheit. Jedoch können manche unwesentlichen Beschädigungen nach einer gewissen Zeit einen Ausfall hervorrufen. So kann aus einem Lackschaden durch Korrosion eine wesentliche Beschädigung werden, die schließlich zum Ausfall führt. Andererseits gibt es auch Ausfälle, die nicht mit Beschädigungen der Betrachtungseinheit verkoppelt sind. Beispielsweise kann die fehlerhafte Bedienung von Steuergliedern die ordnungsgemäße Inbetriebsetzung einer Automatisierungsanlage verhindern. Es kommt zum Betriebsausfall der Automatisierungsanlage, obwohl sie keinerlei Beschädigungen aufweist.

Ausfälle lassen sich nach verschiedenen Merkmalen klassifizieren. Eine mögliche Einteilung zeigt die Tabelle. Die wichtigsten Begriffe dieser Tabelle sind wie folgt erklärt.

Abhängiger Ausfall eines Elements – Ausfall eines Elements der Betrachtungseinheit, der durch Beschädigungen oder Ausfälle anderer Elemente der Betrachtungseinheit bedingt ist.

Unabhängiger Ausfall eines Elements – Ausfall eines Elements, der nicht durch Beschädigungen oder Ausfälle anderer Elemente der Betrachtungseinheit bedingt ist.

Tabelle 1. Klassifizierung von Ausfällen

Klassifizierungsmerkmal	Ausfallart
Charakter der Änderung eines Hauptparameters bis zum Ausfallzeitpunkt	Sprungausfall Driftausfall
Verwendungsmöglichkeit der Betrachtungseinheit nach Ausfalleintritt	Teilausfall Totalausfall
Zusammenhang zwischen verschiedenen Ausfällen	abhängiger Ausfall unabhängiger Ausfall
Dauer der Arbeitsunfähigkeit	andauernder Ausfall, Störung intermittierender Ausfall
Äußere Kennzeichnung des Ausfalls	offensichtlicher Ausfall versteckter Ausfall
Ausfallursache	
Unzulänglichkeit des Konstruktionsverfahrens oder des Konstrukteurs	konstruktionsbedingter Ausfall
Unzulänglichkeit des technologischen Verfahrens, Nichteinhaltung der technologischen Vorschriften	fertigungsbedingter Ausfall
Verletzung der Bedienungsvorschriften, unvorhergesehene äußere Einwirkungen	nutzungsbedingter Ausfall
Art der Ausfallentstehung	natürlicher Ausfall, z. B. durch Verschleiß, künstlicher Ausfall, z. B. durch Sabotage
Zeitpunkt des Ausfalleintritts	Ausfall bei Prüfung Ausfall in der Einlaufphase (Frühausfall) Ausfall in der Phase der normalen Nutzung Ausfall in der letzten Phase der Nutzung (Spätausfall)
Ausfallbeseitigungsmöglichkeit	behebbarer Ausfall nicht behebbarer Ausfall

Driftausfall (allmählicher Ausfall) – Ausfall, der durch allmähliche Änderung der Werte eines oder mehrerer Hauptparameter der Betrachtungseinheit charakterisiert wird.

Sprungausfall (Spontanausfall) – Ausfall, der durch sprunghafte Änderung der Werte eines oder mehrerer Hauptparameter der Betrachtungseinheit charakterisiert wird.

Fertigungsbedingter Ausfall – Ausfall, der sich als Folge einer Unzulänglichkeit des technologischen Prozesses bei der Fertigung oder Reparatur der Betrachtungseinheit ergibt.

Konstruktionsbedingter Ausfall – Ausfall, der sich als Folge einer Unzulänglichkeit des Konstruktionsverfahrens oder eines Fehlverhaltens des Konstrukteurs ergibt.

Nutzungsbedingter Ausfall – Ausfall, der sich als Folge einer Nichteinhaltung der Bedienungsanweisungen oder als Folge des Einflusses unvorhergesehener äußerer Einwirkungen ergibt.

Intermittierender Ausfall – wiederholt auftretende Störung ein und desselben Typs.

Störung (Fehlfunktion) – sich selbst behebender (selbstheilender) Ausfall, der zu einem kurzzeitigen Verlust der Arbeitsfähigkeit führt.

Teilausfall – Ausfall, nach dessen Eintreten eine Verwendung der Betrachtungseinheit gemäß ihrer Zweckbestimmung noch möglich ist, obwohl sich die Werte eines oder mehrerer Hauptparameter außerhalb der zulässigen Grenzen befinden.

Totalausfall – Ausfall, nach dessen Eintreten eine Verwendung der Betrachtungseinheit gemäß ihrer Zweckbestimmung unmöglich ist.

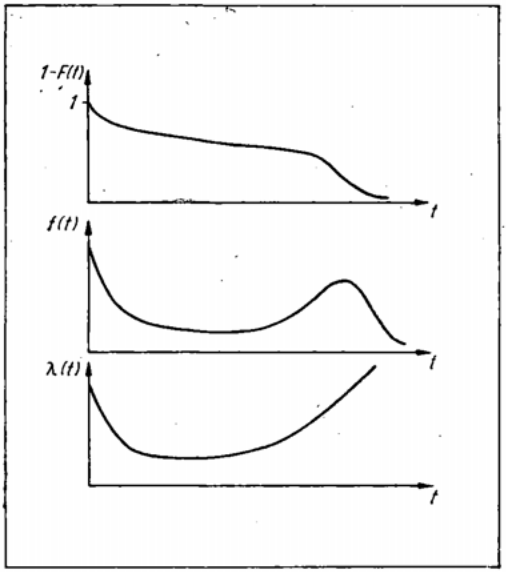
K. Reinschke

Literatur: DDR-Standard TGL 26096: Zuverlässigkeit in der Technik; Koslow, B. A., u. I. A. Uschakow: Handbuch zur Berechnung der Zuverlässigkeit in Elektronik und Automatentechnik (In dt. Sprache hrsg. und ergänzt von K. Reinschke). Berlin: Akademie-Verlag 1978.

Ausfallrate – Zuverlässigkeitskenngröße. Bezeichnet man mit $F(t)$ die Verteilungsfunktion der zufälligen Dauer von der Inbetriebnahme bis zum Ausfall einer Betrachtungseinheit, so wird die Ausfallrate der Betrachtungseinheit durch die zeitabhängige Funktion

$$\lambda(t) = \frac{1}{1-F(t)} \frac{\mathrm{d}}{\mathrm{d}t} F(t) = \frac{f(t)}{1-F(t)}$$

definiert. Die Funktion $f(t)$ im Zähler heißt Ausfalldichte, die Funktion $1-F(t)$ im Nenner heißt Überlebenswahrscheinlichkeit. Die Zusammenhänge



Zusammenhang zwischen Überlebenswahrscheinlichkeit, Ausfalldichte und Ausfallrate

zwischen einander entsprechenden Überlebenswahrscheinlichkeiten, Ausfalldichten und Ausfallraten wurden in der Abb. skizziert.

Nur bei exponentialverteilten Dauern der Arbeitsfähigkeit der Betrachtungseinheit, d. h. bei $F(t) = 1 - e^{-\lambda t}$, ist die Ausfallrate eine zeitunabhängige Konstante.

Die angegebene Formel für $\lambda(t)$ kann man auch wie folgt lesen: $\lambda(t)$ ist der Wert der bedingten Ausfalldichte – unter der Bedingung, daß bis zum Zeitpunkt t der Ausfall der Betrachtungseinheit nicht eingetreten ist – im Zeitpunkt t.

K. Reinschke

Literatur: Reinschke, K.: Zuverlässigkeit von Systemen. Band I. Berlin: VEB Verlag Technik 1973; DDR-Standard TGL 26096: Zuverlässigkeit in der Technik; Koslow, B. A., u. I. A. Uschakow: Handbuch zur Berechnung der Zuverlässigkeit in Elektronik und Automatentechnik (In dt. Sprache hrsg. u. ergänzt von K. Reinschke), Berlin: Akademie-Verlag 1978.

Automat, zellularer – Eine umfangreiche Darstellung zu anderen Aspekten zellularer Automaten (ZA) befindet sich im Teil A–E, S. 227–230. Als Synonyme bzw. als Begriffe mit sehr ähnlicher Bedeutung werden verwendet: Wiederholstrukturen, Mosaikautomat, Automatennetz. In der englisch- bzw. russischsprachigen Fachliteratur werden folgende Begriffe verwendet: cellu-

lar automaton, iterative structure, tesselation automaton однородная структура, клеточный автомат.

Der Begriff ZA wurde durch JOHN VON NEUMANN eingeführt. ZA werden durch die Zellautomaten A (oder kurz die Zelle), das (lokale) Verbindungsschema 'S' zwischen den Zellautomaten (oder die Nachbarschaftskopplung) und das räumliche (Gesamt-) Anordnungs-/Positionierungsschema G für die Zellautomaten (oder das Gitter) charakterisiert. Zur Verdeutlichung dienen die in Abb. 1 gegebenen Beispiele, bei denen das Anordnungsschema eine Gerade ist, auf der alle Zellen (oder Zellautomaten) des ZA angeordnet sind und bei denen jede Zelle mit der darauf folgenden (Fall a) und jede Zelle mit der übernächsten und der vorhergehenden (Fall b) verbunden oder verkoppelt ist. ZA dieses Anordnungstyps werden als eindimensionale ZA bezeichnet.

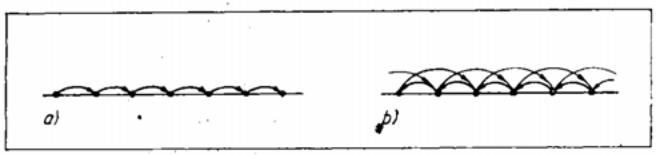


Abb. 1
Beispiele für eindimensionale zellulare Automaten

In Abb. 2 ist als weiteres Beispiel ein zweidimensionaler ZA gegeben, dessen Anordnungsschema ein gleichförmiges (rechteckiges) rektanguläres Punktgitter ist. In diesem Beispiel ist eine Nachbarschaftskopplung – die sogenannte VON-NEUMANN-Kopplung – angegeben, bei der jede Zelle mit sich und mit ihrer oberen, unteren, linken und rechten Nachbarzelle gekoppelt ist. Neben diesen Beispielfällen lassen sich eine Vielzahl weiterer Gittertypen (z. B. 1-dimensionale, 2-dimensionale, ..., n-dimensionale, trianguläre, hexagonale Gitter) und eine Vielzahl von Nachbarschaftskopplungen angeben. Die Zellautomaten sind stochastische oder deterministische Automaten.

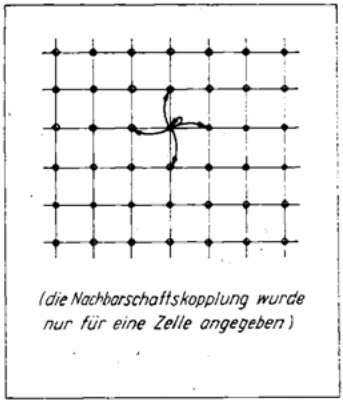


Abb. 2
Beispiel eines zweidimensionalen zellularen Automaten

Die Zellautomaten arbeiten taktweise, wobei der Takt für die Zellen gleichzeitig (synchronisiert) oder nicht gleichzeitig (unsynchronisiert) gegeben werden kann. Konfigurationen auf ZA werden durch die Zustände der einzelnen Zellen beschrieben.

ZA dienen als Modell paralleler (und räumlich verteilter) Informationsverarbeitungsvorgänge technischer und biologischer Informationsverarbeitungssysteme. Die räumliche Struktur des ZA ermöglicht die realitätsnahe Modellierung des zeitlich verkoppelten Ablaufs dieser Informationsverarbeitungsvorgänge für die verschiedenen Zellen des ZA. Die Verwendung von ZA als Modelle diskreter räumlich verteilter physikalischer Prozesse wurde ebenfalls untersucht. In den letzten Jahren wurden wegen der sich beim Entwurf und der Produktion von mikroelektronischen Schaltkreisen ergebenden Vorteile zunehmend Rechner- und Prozessorstrukturen als regelmäßige zellulare Strukturen realisiert. Damit wurde für ZA ein direkter Anwendungsbezug erschlossen.

Rechner mit einer einheitlichen Gesamtstruktur, ohne die Einbeziehung anderer (inhomogener oder konventioneller) Strukturtypen, wurden noch nicht realisiert, obwohl bereits Lösungen mit einer weitgehend homogenen Grundstruktur vorliegen (z. B. Distributed-Array-Processor (DAP) der Fa. ICL). Das gegenwärtig vorherrschende Einsatzgebiet für homogen realisierte Prozessoren sind *Spezialprozessoren* für die *digitale Bildverarbeitung* und Bilderkennung. Darüber hinaus sind homogene Realisierungen zumindest teilweise für universellere Einsatzgebiete erfolgt (z. B. der teilweise homogen realisierte Chip 32-b-MP INTEL Typ iAPX 432). In einer Reihe von Forschungseinrichtungen wurden als Experimentierrechner *Mehrprozessorsysteme* mit homogener Grundstruktur aufgebaut. Die theoretische Untersuchung von ZA zur Klärung ihrer potentiellen Leistungsfähigkeit bzw. ihrer Leistungsgrenzen erfolgte bisher vorwiegend für homogene ZA, d. h. die Klasse von ZA, bei der Zelle, Nachbarschaftskopplung und Gitterstrukturen an jedem Gitterpunkt identisch sind. Die Untersuchung inhomogener ZA – manchmal auch als Automatennetze bezeichnet – steckt noch in den Anfängen.

Wesentliche Ergebnisse wurden bei homogenen ZA zu Problemen der Charakterisierung von umfassenden Teilklassen von ZA erzielt. Die Lösung dieser Fragen steht im Zusammenhang mit der Untersuchung von Problemen außerhalb der engeren Theorie ZA, z. B. die Untersuchung von parallelen Algorithmen auf TURING-*Maschinen* mit mehreren Köpfen (und ggf. mit einem mehrdimensionalen Speichermedium). Im Zusammenhang damit werden Probleme zur wechselseitigen Darstellung und Simulierbarkeit von unterschiedlichen Teilklassen von ZA – damit zusammenhängend die Aufklärung möglichst einfacher Bezugsklassen von ZA – untersucht.

Gegenstand eines weiteren Problemkreises sind Komplexitäts- und Generierungsprobleme auf ZA, d. h. die Bearbeitung komplexer Rechenvorgänge oder die Strukturierung komplexer Zustände mit Hilfe einfacher Grundelemente und Grundoperationen. Diese Untersuchungen haben gezeigt, daß es Klassen von Konfigurationen gibt, die nicht generierbar sind und deshalb als Garten-Eden-Konfiguration (englisch: garden-of-eden-configuration) bezeichnet werden. In diesen Rahmen mehr grundlegender Untersuchungen zu ZA ordnen sich die Ergebnisse ein, die von J. VON NEUMANN zur Selbstgenerierung und Selbstreproduktion von ZA erarbeitet wurden und die über den engeren Bereich der Theorie ZA hinaus Grundlagenuntersuchungen in künstlichen bzw. technischen Systemen überhaupt sind.

Mit ZA wurde – im Hinblick auf praktische Anwendungen – eine Vielzahl von Problemen paralleler Algorithmen zur Synchronisation der Arbeit von verteilten Komponenten (firing-squad, early-bird-problems), zur *Mustererkennung* und -manipulation und zur *Spracherkennung* behandelt und geklärt. Von der bereits oben erwähnten, in den letzten Jahren zunehmenden Verwendung von Rechner- und Prozessorstrukturen nach dem direkten Vorbild homogen strukturierter ZA wurde auch die Entwicklung der Theorie ZA beeinflußt. Dabei zeigt sich, daß die schnelle technische und technologische Entwicklung auf dem Gebiet der *Mikroelektronik* zu neuen eigenständigen Begriffsbildungen und ingenieurtechnischen Lösungen (z. B. *Pipelineprozessor*, Array-Prozessor) geführt hat, die einerseits eindeutig zum Begriffsbild des ZA gehören, andererseits im Hinblick auf den notwendigen theoretischen Hintergrund noch unzureichend in die Theorie eingeordnet sind. Damit rücken neben den oben erwähnten Problemkreisen weitere Fragen in den Blickpunkt. Für derartige Probleme werden im folgenden einige Beispiele gegeben, die aber nur Ausschnitte aus dem notwendigen Gesamtumfang darstellen: Untersuchung endlicher ZA einschließlich ihrer Umgrenzung (sschaltungen), Untersuchung von nichtautonomen ZA (*systolische Arrays*), Untersuchung inhomogener ZA, Fehlererkennung und -toleranz in ZA.

G. WOLF

Literatur: v. NEUMANN, J.: Theory of Self-reproducing Automata. Urbana/Ill.: Urbana Press Univ. of Ill. 1966; BURKS, A. W.: Essays on Cellular Automata. Urbana/Ill.: Urbana Press Univ. of Ill. 1970; ZUSE, K.: Rechnender Raum. Braunschweig: Vieweg u. Sohn 1969; ALADJEV, V.: On the Theory of the homogeneous Structures. Tallinn: Acad. of Sc. Esnt. SSR 1972; VOLLMAR, R.: Algorithmen in Zellularautomaten. Stuttgart: B. G. Teubner 1979.

Automatisierung wissenschaftlicher Forschungsarbeiten – Teilprozeß der mit der wissenschaftlich-technischen Revolution einhergehenden Automa-

tisierung, in dessen Verlauf manuelle und geistige Tätigkeiten des Menschen in der wissenschaftlichen Forschung durch die Funktion künstlicher Systeme (Automatisierungsmittel) ersetzt oder ergänzt werden. In dem Maße, wie die Wissenschaft zur Produktivkraft wird, erweitert sich ihre Einflußsphäre und wächst die Komplexität wissenschaftlicher Apparaturen und wissenschaftlich-technologischer Anlagen, während die materiellen Ressourcen und das Arbeitskräftepotential begrenzt sind. Zwangsläufig muß daher die Forschung intensiviert werden. Einen effektiven Weg der Intensivierung der Forschung, der Steigerung der wissenschaftlichen Arbeitsproduktivität und damit auch der Beschleunigung des Forschungstempos stellt die A.w.F. dar.

Wichtigste Automatisierungsmittel sind Geräte- und Programmsysteme der elektronischen Rechentechnik. Jede Anwendung der Groß-, Klein- und Mikrorechentechnik in der experimentellen und theoretischen Forschung, in der Organisation und Leitung der Forschung und in der wissenschaftlichen Information ist streng genommen ein Beitrag zur A.w.F. Im engeren Sinne wird unter A.w.F. die Experimentautomatisierung, d. h. die On-line-Kopplung experimenteller Apparaturen mit Klein-, Prozeß- und Mikrorechnern verstanden. Für die Automatisierung traditioneller experimenteller Verfahren gibt es bereits eine Reihe *automatisierter Systeme zur Verarbeitung experimenteller Daten* (z. B. Fourierspektrometer mit Spezialprozessoren, Bildverarbeitungssysteme). Da die wissenschaftliche Forschung ein zutiefst evolutionärer Prozeß ist und im naturwissenschaftlichen Experiment auf jede Antwort eine neue Frage des Experimentators folgt, müssen sich Geräte- und Programmsysteme für die Experimentautomatisierung schnell und einfach ändern lassen, und die Gestaltung des Mensch-Maschine-Systems „Experimentator — Automatisierungsmittel — Untersuchungsobjekt" verdient eine besondere Beachtung. Darin unterscheidet sich die Experimentautomatisierung grundlegend von der Automatisierung technologischer Prozesse in der Industrie, bei der vorwiegend sich ständig wiederholende Abläufe automatisiert werden.

Um der Forderung hoher Flexibilität zu genügen, werden die Geräte- und Programmsysteme zur Experimentautomatisierung aus universell verwendbaren Bausteinen, d. h. modular aufgebaut. Auf diese Weise können gleichartige Automatisierungsmittel in unterschiedlichen Disziplinen der wissenschaftlichen Forschung eingesetzt werden, und die A.w.F. erhält einen interdisziplinären Charakter. Voraussetzung dafür sind aufwärts kompatible Familien von Mikro- und Kleinrechnern, die dadurch gekennzeichnet sind, daß sie eine gleichartige *Datenstruktur* besitzen und daß der Befehlssatz eines leistungsschwächeren Rechners eine Untergruppe des Befehlssatzes eines leistungsstärkeren Rechners bildet. Eine derartige Familie ist z. B. das System der Kleinrechner SKR der RGW-Länder mit standardisierten *Be-*

triebssystemen. Unter der Vielzahl standardisierter modularer Instrumentierungssysteme zur Rechner-Experiment-Kopplung hat sich besonders der ursprünglich für die Kernphysik entwickelte *CAMAC*-Standard durchgesetzt, der in einigen sozialistischen Ländern auch zum Standard der industriellen Automatisierung erhoben wurde.

Die Experimentautomatisierung führt neben der Intensivierung der Forschung auch zu einer qualitativen Verbesserung und Erweiterung der experimentellen Methodik hinsichtlich Auflösung, Meßgenauigkeit, Stabilität, Zuverlässigkeit und statistischer Signifikanz, zur Entwicklung neuer komplexer Methoden, die in einem Experiment sehr viele Parameter erfassen und verarbeiten oder mehrere, bisher unabhängig voneinander ausgeführte Meßverfahren über den Rechner miteinander verkoppeln, und eröffnet neue experimentelle Möglichkeiten für in-situ Messungen, für zerstörungsarme Messungen, für Experimente in aggressiver Umwelt, die bisher ohne rechentechnische Hilfsmittel nicht ausführbar waren.

R. A. Pose

Literatur: Выставкин, А. Н.: Процесс исследования как объект автоматизации. Автоматика и вычислительная техника, Москва (1981) 2.

Automatisierungsgrad – Kenngröße zur Charakterisierung des Niveaus der Automatisierung von Geräten, Maschinen, Anlagen und Prozessen. Die Berechnung des A. kann auf Basis der Arbeitskräfte, der Ausrüstungen und des Zeitaufwandes erfolgen:

– arbeitskraftbezogen: $A^0_{Ak} = \frac{Ak_{aut}}{Ak_{ges}}$

Ak_{aut} – Anzahl der Produktionsarbeiter und des ingenieurtechnischen Personals mit Kontroll- und Überwachungsfunktionen an automatisierten Anlagen einer Produktionseinheit,

Ak_{ges} – Gesamtzahl der Arbeitskräfte der Produktionseinheit;

– ausrüstungsbezogen: $A^0_{Ar} = \frac{BW_{aut}}{BW_{ges}}$

BW_{aut} – Bruttowert der automatisierten Ausrüstungen [Mark],

BW_{ges} – Bruttowert aller Ausrüstungen [Mark];

– zeitbezogen: $A^0_T = \frac{T_{aut}}{T_{ges}}$

T_{aut} – Zeitaufwand zur Durchführung der automatisierten Operationen eines Prozesses,

T_{ges} – Zeitaufwand für den Gesamtprozeß.

Entsprechend der Größenordnung des A. kann eine qualitative Charakteristik von komplexen Prozessen nach der folgenden Aufstellung erfolgen:

Qualitative Charakteristik von Prozessen nach dem Automatisierungsgrad

A^0	Qualitative Prozeßcharakteristik
0,0 . . . 0,1	nicht automatisiert
>0,1 . . . 0,4	teilautomatisiert
>0,4 . . . 0,6	halbautomatisiert
>0,6 . . . 0,9	automatisiert
>0,9 . . . 1,0	vollautomatisiert

H.-G. LAUENROTH

Automatisierungskonzeption — langfristig orientierte technisch-ökonomische Ziel- und Aufgabenstellung von Produktions- und Informationsprozessen. Die A. als Bestandteil der Intensivierungskonzeption eines Kombinates enthält als Ergebnis systemanalytischer Untersuchungen: Aussagen über die zu erreichenden Ziele hinsichtlich der erforderlichen Entwicklung von Produktionsumfang, Sortiment, Innovationsraten, Produktivität, Qualität, Kosten, Gestaltung der Arbeitsbedingungen und weiterer wesentlicher Kenngrößen sowie über die Grundrichtungen der Intensivierung und Rationalisierung des Reproduktionsprozesses eines Kombinates; Analysen über die Bedarfsfelder der zu automatisierenden Produktionsprozesse (mechanische Fertigung, Montage, Meß- und Prüfprozesse, Transport, Lagerhaltung u. a.) und der zu automatisierenden Informationsprozesse (Planung, Entwicklung, Konstruktion, Projektierung, Produktionsvorbereitung und -steuerung, Abrechnung und Analyse) im Hinblick auf die Bestimmung des automatisierwürdigen Potentials; Untersuchungen über die Lösungsfelder, d. h. die einsetzbare automatisierte Produktionstechnik (*NC-Maschinen, Industrieroboter*, Transferstraßen, *Fertigungszellen, flexible Fertigungssysteme* u. a.) und die einsetzbare automatisierte Informationstechnik (*Mikroprozessoren, Bürorechner, Prozeßrechner, EDVA, Kommunikationstechnik* u. a.) im Hinblick auf die Bestimmung eines effektiven *Automatisierungsgrades*; Angaben über das Realisierungsfeld in Form notwendiger und verfügbarer materieller Ressourcen (Ausrüstungen, Geräte, Bauelemente), personeller Ressourcen (Umfang, Qualifikation) und finanzieller Ressourcen (Investitionsmittel u. a.); Varianten von Automatisierungsstrategien als Ergebnis der Abstimmung von Bedarfs-, Lösungs- und Realisierungsfeldern, wobei sich die Varianten sowohl auf qualitativ und quantitativ als auch zeitlich unterschiedlich gestaltete Strategien beziehen; Bewertungsmaßstäbe und Bewertungen für die Automatisierungsstrategien mit dem Ziel der Auswahl der effektivsten Varianten; Maßnahmen und Termine zur Vorbereitung und Realisierung der A.

Wesentlich für die Effektivität von Automatisierungsprojekten ist dabei die Berücksichtigung der Wechselwirkung von Produkt- und Prozeßinnovationen (s. *Innovation*) und der stimulierenden Wirkung der *flexiblen Automatisierung* auf die Produkt-Innovationsrate (s. *Systemanalyse von Innovationsprozessen*).

H.-G. Lauenroth

Automatisierungsziel – Gesamtheit der zu erreichenden Wirkungen und Effekte durch die Automatisierung von Produktions- und Informationsprozessen. Im Maßstab eines Kombinates wird das A. als komplexe Kategorie technischer, ökonomischer und sozialer Natur behandelt; es umfaßt die qualitative Bestimmung des anteiligen Beitrages der Automatisierung zur Sicherung der geplanten Produktionssteigerung, Produktivitätserhöhung, Qualitätsverbesserung, Kostensenkung, Verbesserung der Arbeitsbedingungen und der Arbeitsinhalte sowie zur Erhöhung der Produkt- und Prozeß-Innovationsraten (s. *Innovation*); die quantitative Bestimmung der Entwicklung des *Automatisierungsgrades* in den Bereichen der mechanischen Fertigung, der Montage, des Meß- und Prüfwesens, der Transport- und Lagerhaltungsprozesse im Bereich der Produktion sowie der Planung, Entwicklung, Konstruktion, Projektierung, Produktionsvorbereitung und -steuerung, Abrechnung und Analyse im technischen und ökonomischen Bereich im Hinblick auf die Entwicklung von Ergebnis-, Aufwands- und Effektivitätskenngrößen; die zeitbezogene Bestimmung der Realisierungsetappen des A. nach Varianten in Abhängigkeit von den Automatisierungsstrategien (s. *Automatisierungskonzeption*).

H.-G. Lauenroth

B

Backtracking – Programmierverfahren zum Suchen eines Weges durch einen *Graphen* von einem (oder mehreren) vorgegebenen Anfangspunkt zu einem (oder mehreren) Endpunkt. Durch B. können immer auch alle Wege gefunden werden. Dazu ist der letzte erreichte Endpunkt als Sackgasse (engl. blind alley , dead end) zu interpretieren. B. ist häufig als implizite Fähigkeit von Spezialsprachen implementiert. Programme in solchen Sprachen können dann als einfache Geradeaus-Programme geschrieben und gesehen werden, das System übernimmt die Sicherung von Zuständen (Schnappschuß), deren Folgezustände nicht eindeutig sind, und reaktiviert sie, wenn ein erreichter

Wesentlich für die Effektivität von Automatisierungsprojekten ist dabei die Berücksichtigung der Wechselwirkung von Produkt- und Prozeßinnovationen (s. *Innovation*) und der stimulierenden Wirkung der *flexiblen Automatisierung* auf die Produkt-Innovationsrate (s. *Systemanalyse von Innovationsprozessen*).

H.-G. Lauenroth

Automatisierungsziel – Gesamtheit der zu erreichenden Wirkungen und Effekte durch die Automatisierung von Produktions- und Informationsprozessen. Im Maßstab eines Kombinates wird das A. als komplexe Kategorie technischer, ökonomischer und sozialer Natur behandelt; es umfaßt die qualitative Bestimmung des anteiligen Beitrages der Automatisierung zur Sicherung der geplanten Produktionssteigerung, Produktivitätserhöhung, Qualitätsverbesserung, Kostensenkung, Verbesserung der Arbeitsbedingungen und der Arbeitsinhalte sowie zur Erhöhung der Produkt- und Prozeß-Innovationsraten (s. *Innovation*); die quantitative Bestimmung der Entwicklung des *Automatisierungsgrades* in den Bereichen der mechanischen Fertigung, der Montage, des Meß- und Prüfwesens, der Transport- und Lagerhaltungsprozesse im Bereich der Produktion sowie der Planung, Entwicklung, Konstruktion, Projektierung, Produktionsvorbereitung und -steuerung, Abrechnung und Analyse im technischen und ökonomischen Bereich im Hinblick auf die Entwicklung von Ergebnis-, Aufwands- und Effektivitätskenngrößen; die zeitbezogene Bestimmung der Realisierungsetappen des A. nach Varianten in Abhängigkeit von den Automatisierungsstrategien (s. *Automatisierungskonzeption*).

H.-G. Lauenroth

B

Backtracking – Programmierverfahren zum Suchen eines Weges durch einen *Graphen* von einem (oder mehreren) vorgegebenen Anfangspunkt zu einem (oder mehreren) Endpunkt. Durch B. können immer auch alle Wege gefunden werden. Dazu ist der letzte erreichte Endpunkt als Sackgasse (engl. blind alley , dead end) zu interpretieren. B. ist häufig als implizite Fähigkeit von Spezialsprachen implementiert. Programme in solchen Sprachen können dann als einfache Geradeaus-Programme geschrieben und gesehen werden, das System übernimmt die Sicherung von Zuständen (Schnappschuß), deren Folgezustände nicht eindeutig sind, und reaktiviert sie, wenn ein erreichter

Zustand keinen Folgezustand hat. B. kann deshalb auch als ein Verfahren zur Simulation *nichtdeterminierter Automaten* verstanden werden.

Es gibt vielfältige Anwendungen von B.: Parsingalgorithmen zur Analyse formaler und natürlicher Sprachen, *automatisches Theorembeweisen* und Inferieren, Computerspiele, *Formelmanipulation*, *Handlungsplanung* für *Roboter* u. v. a.

Die *Implementierung* von B. hängt von der Definition der Spezialsprache ab. Werden in ihr alle strukturbildenden Aktionen kopierend vorgenommen, genügt es, die relevanten Zustandsvariablen mit ihren Werten (Adressen von Strukturen) zu sichern.

Sind jedoch strukturzerstörende Aktionen zugelassen, muß die zugehörige Umkehraktion gespeichert werden. Eine besondere Schwierigkeit erwächst bei rekursiven Spezialsprachen. Es muß gesichert werden, daß bereits abgeschlossene *Unterprogramme* (Funktionen, Prozeduren) wieder betreten werden können, dort einen anderen Verlauf nehmen, obwohl sich ihre aktuellen Parameter scheinbar nicht geändert haben. Ein übliches Verfahren zur Realisierung der Wiederbetretbarkeit besteht in der Ausstattung dieser Unterprogramme mit einem Vorspann, der auf die globalen Backtrackinformationen zurückgreift und damit den Programmfluß ändert, indem z. B. die aktuellen äußeren Parameter von innen geändert werden. Ein anderes Verfahren besteht darin, daß man den gesamten Prozeduraufruf mit den neuen aktuellen Werten im Schnappschuß sichert und diesen Aufruf im Backtracking realisiert. Ein solches Verfahren ist jedoch nur anwendbar, wenn die Wirtssprache, in der die Spezialsprache eingebettet ist, über einen *Interpreter* verfügt bzw. keinen prinzipiellen Unterschied zwischen Daten und Programmen vornimmt (s. *LISP* oder *ASSEMBLER*).

Man spricht von blindem Backtracking, wenn der durchlaufene Weg schrittweise zurückverfolgt (Ariadnefaden, Prinzip der kleinsten Änderungen) und die letzte mögliche Entscheidung geändert wird. Es hängt vom Problemkomplex ab, ob ein effizienteres Verfahren gewählt werden kann.

Eine Effizienzsteigerung kann man einerseits dadurch erreichen, daß man nur inhaltlich relevante Entscheidungspunkte sichert, sie also von nur scheinbaren Alternativen abgrenzt, und andererseits, daß man die Schnappschüsse mit Zusatzinformationen versieht, die später zur Auswahl geeigneter Reaktivierungszustände benutzt werden können, wenn man entsprechende Informationen über die Ursachen von Sackgassen hat (gezieltes Backtracking).

D. Koch

Back-up – Sonderform der *Redundanz* zum Verringern der Ausfallwahrscheinlichkeit (Zuverlässigkeit), vor allem beim Steuern technologischer

Prozesse mittels Rechner (DDC). Bei (kurzzeitigem) Ausfall des *Prozeß rechners* wird durch Back-up-Regler ein (Not-)Betrieb aufrechterhalten, die Notabschaltung (Prozeßsicherung) vermieden.

R. MÜLLER

BASIC – (Beginners All Purpose Symbolic Instruction Code) eine *Programmiersprache*, 1964 im Dartmouth College (USA) entwickelt. Seit 1974 werden Bemühungen für eine Standardisierung von BASIC unternommen. Entsprechende Standardisierungsvorschläge von ECMA und ANSI für ein minimales BASIC liegen vor. Von den Herstellern von Rechnern werden Interpreter bzw. *Compiler* bereitgestellt, die i. allg. einen erweiterten Sprachumfang realisieren.

BASIC zeichnet sich durch geringe Komplexität, einfache Handhabbarkeit und Erlernbarkeit sowie problemadäquate Notierung aus, ist besonders geeignet für die Dialogprogrammierung und stellt die gegenwärtig bevorzugte höhere Programmiersprache für *Mikrorechner* dar. Besondere Bedeutung hat BASIC für den verstärkten Einsatz von Rechnern des „Systems der *Kleinrechentechnik*" (SKR).

Am Beispiel der Berechnung von Mittelwert und Varianz einer Reihe von Meßwerten soll die Programmierung im Rahmen eines minimalen BASIC dargestellt werden.

```
10  REM MITTELWERT UND VARIANZ
20  DIM X(50)
30  PRINT „ANZAHL DER MESSWERTE"
40  INPUT N
50  PRINT N
60  PRINT „MESSWERTE"
70  FOR I=1 TO N
80  INPUT X(I)
90  PRINT X(I);
100 NEXT I
110 LET M=0
120 FOR I=1 TO N
130 LET M=M+X(I)
140 NEXT I
150 LET M=M/N
160 LET V=0
170 FOR I=1 TO N
180 LET V=V+(X(I)-M)*(X(I)-M)
190 NEXT I
200 LET V=V/(N-1)
```

```
21Ø PRINT „MITTELWERT“, „VARIANZ“
22Ø PRINT M,V
23Ø GOTO 3Ø
24Ø END
```

Erweiterungen von BASIC bezüglich des Sprachumfanges betreffen insbesondere die Verarbeitung von Zeichenketten, die kompakte Behandlung von Matrizenoperationen sowie die Ein- und Ausgabe von graphischen Informationen.

H.-M. Voigt

Literatur: Strelocke, K., und P. Hoffmann: Die Dialogprogrammiersprache BASIC. Berlin: Verlag Die Wirtschaft 1981; Schneider, W.: Einführung in BASIC. Braunschweig, Wiesbaden: Friedr. Vieweg und Sohn 1980; American National Standards Institute (ANSI): Proposed American National Standard for Minimal BASIC X 3 J 2/76–35, Dez. 1976; European Computer Manufacturers Association (ECMA) Standard ECMA – Minimal BASIC – Final Draft ECMA / TC21 / 76 / 44, Dez. 1976.

Basisautomatisierung – derjenige Anteil an Automatisierung(stechnik), der zur Gewährleistung der Funktionserfüllung von Maschinen, Apparaten und technologischen Anlagen unabdingbar ist. Von den grundsätzlichen Automatisierungszielen, Beiträge zur Funktionstüchtigkeit, Wirtschaftlichkeit, Sicherheit sowie Menschen- und Umweltfreundlichkeit zu erbringen, dient die B. somit der Funktionstüchtigkeit und Sicherheit; sie kann ergänzt werden durch weitere Automatisierungsmaßnahmen bzw. -einrichtungen zugunsten weiterer Ziele. Da Maschinen, Apparate und Anlagen ohne B. nicht betreibbar sind, kann man sie auch der Technologie zurechnen.

G. Brack

Bauelemente der Mikroelektronik – sind vor allem *integrierte Schaltungen* (Schaltkreise), in denen eine Vielzahl von Transistorfunktionen bereits zur Funktionseinheit zusammengefaßt sind. Es werden analoge und digitale Schaltkreise unterschieden. Je nach der Anzahl der im Schaltkreis enthaltenen Transistoren (analog) bzw. der Anzahl der Gatter (digital) gilt folgende Einteilung:

- SSI: small scale integration (100, 10),
- MSI: median scale integration (1000, 100),
- LSI: large scale integration (10^4, 10^3),
- VLSI: very large scale integration (10^5, 10^4).

Die Anzahl der verschiedenen Bauelemente ist heute unwahrscheinlich groß. Vollständige Tabellen gibt es nicht. Allgemein wird das Spektrum auf mehrere hunderttausend Typen geschätzt. Bei den analogen Schaltkreisen sind folgende Funktionen in hoher Typenvielfalt vertreten:

- Operationsverstärker, Begrenzer, SCHMITT-Trigger,
- Leistungsendstufen, Stromversorgungsschaltungen,
- Modulatoren, Demodulatoren und
- viele Sonderschaltungen der Rundfunk-, Fernseh-, Foto- und Tonbandindustrie sowie Schaltkreise für kommerzielle Elektronik.

Bei den digitalen Schaltkreisen existieren drei Gruppen:

- Standardschaltkreise,
- Schaltkreise der Mikrorechentechnik,
- Schaltkreise für Sonderanwendungen.

Die Standard-Schaltkreise werden im wesentlichen Logikfamilien zugeordnet. An vorderster Stelle steht hier die TTL-Reihe mit mehreren Varianten. Die Typenvielfalt dieser Reihe liegt bei mehreren Hundert. Ihre Schnittstellen, Pegel, Belastungsfaktoren haben sich zu einem Stand entwickelt, dem sich alle anderen Reihen und digitalen Bauelemente weitgehend anzupassen haben.

Neben der TTL-Reihe haben die CMOS-Reihe (für geringe Leistungen) und die ECL-Reihe (für höchste Geschwindigkeit) größere Bedeutung erreicht. Die anderen Schaltkreisreihen sind weitgehend verschwunden.

Neben den Schaltkreisreihen haben bestimmte Logikkonzepte für den inneren Aufbau von Schaltkreisen Bedeutung. Hier seien nur die Varianten der p-MOS und n-MOS sowie I^2L und EFL erwähnt. Sie bestimmen die Logikbedingungen im Inneren der Schaltung, während an den Ein- und Ausgängen immer auf TTL-Pegel angepaßt wird.

Die Schaltkreise der Mikrorechentechnik sind um die zentralen Steuerbausteine, den *Mikroprozessor*, herum gegliedert. Hier hat sich die Entwicklung von den 4-bit- über die 8-bit- zu den 16- und 32-bit-Schaltungen vollzogen. Auf lange Sicht dürfte im Produktionsaufkommen die 8-bit-Verarbeitungsbreite deutlich überwiegen.

Wichtige Ergänzungsbausteine zum Mikroprozessor sind Speicherbausteine, In-out-Bausteine (seriell und parallel), Zeit- und Zählerbausteine, Controller-Bausteine, AD- und DA-Wandler sowie Busbausteine.

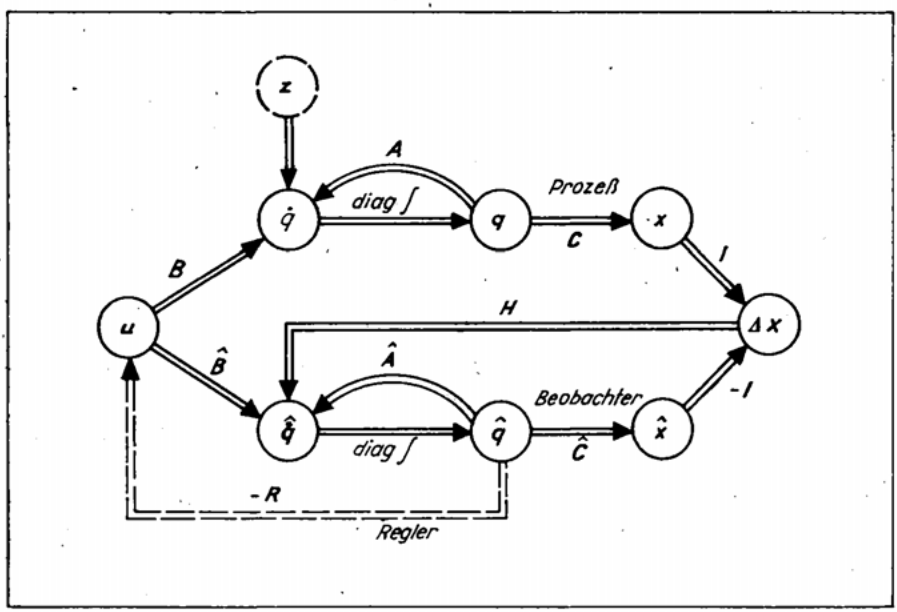
Schaltkreise für Sonderanwendungen werden benötigt, weil oft der Leistungsbedarf minimiert werden muß, sowie für Sonderaufgaben (z. B. Konsumgüter) mit sehr großen Stückzahlen und Spezialaufgaben der industriellen Elektronik. Sie können auf vielfältige Weise kostengünstig hergestellt werden. Eine wichtige Lösung bietet die Hybridtechnik, bei der verschiedene Bauelemente in Schaltungen integriert werden, die auf der Dick- oder Dünnschichttechnik basieren.

H. VÖLZ

Literatur: VÖLZ, H.: Elektronik, 4. Aufl. Berlin: Akademie-Verlag 1984.

Belastungsredundanz — s. *Redundanz.*

Beobachter — in der Steuerungs- und Regelungstheorie allgemein eine (dynamische) Einrichtung, die es gestattet, meßtechnisch nicht erfaßbare Größen, die wesentlich das Verhalten eines Systems charakterisieren, aus gemessenen Ein- und Ausgangsgrößen zu rekonstruieren und damit die Weiterverarbeitung z. B. im Sinne einer *automatischen Steuerung* zu ermöglichen. Wesentliche Bedeutung haben sog. Zustandsbeobachter in der Theorie linearer Systeme erlangt, die auf dem Begriff des Systemzustandes aufbaut. Der zuerst von LUENBERGER 1964 angegebene Beobachter stellt ein zum (als bekannt vorausgesetzten) System parallel geschaltetes Modell dar, dessen Eingang die Steuergröße ist (s. Abb.). Differenzen in den Zustandsgrößen von



Modell des Beobachters nach LUENBERGER

Original und Modell (Beobachter) können, da beide mit dem gleichen Eingangssignal beaufschlagt werden, nur durch verschiedene Anfangswerte oder durch Störungen im Original verursacht werden. Letztere führen, wenn sie dauernd einwirken und nicht meßbar sind, prinzipiell zu Abweichungen. Das Funktionsprinzip eines Beobachters ist das eines *Folgeregelungssystems.* Es können nur beobachtbare Zustandsvariable geregelt werden, d. h., zur Rekonstruktion des vollen Zustandes des Systems muß vollständige *Beobacht-*

barkeit gefordert werden. Für den rekonstruierten Zustand erhält man

$$\dot{\hat{q}} = \hat{A}\hat{q} + \hat{B}u + Hx\,; \qquad \hat{x} = \hat{C}\hat{q}\,.$$

Mit

$$\hat{B} = B, \quad \hat{A} = A - HC, \quad \hat{C} = C$$

und Wahl der Reglermatrix H so, daß die Eigenwerte von $\hat{A}$ genügend weit links von denen von A liegen, ergibt sich für den Beobachterzustand:

$$\dot{\hat{q}} = A\hat{q} + Bu + HC \triangle p\,.$$

Die Dimensionierung eines Beobachters ist einfach für Eingrößensysteme, die in der sog. Beobachtungsnormalform vorliegen. Eine problemlose Erweiterung auf *Mehrgrößenregelungssysteme* in P-kanonischer Struktur ist möglich. Aus Aufwandsgründen haben Beobachter reduzierter Ordnung große Bedeutung. Für ein System n-ter Ordnung mit m linear unabhängigen Ausgangsgrößen genügt dann zur Rekonstruktion des vollen Zustandes ein Beobachter der Ordnung $n-m$. Der Entwurf erfolgt ähnlich wie der des vollständigen Beobachters.

Auch für bestimmte nichtlineare Systeme ist der Entwurf entsprechender (Zustands-)Beobachter möglich.

H. Ehrlich

Bilanzregelung – eine der Aufgaben der *Basisautomatisierung* zur Sicherung der Funktionsfähigkeit von Apparaten und technologischen Anlagen. Aufgabe der B. ist es, Zuflüsse und Abflüsse (Stoffströme, Energieströme) eines Bilanzraumes, technisch als Stoff- oder Energiespeicher in einem Apparat oder einer Anlage realisiert, dadurch im Gleichgewicht zu halten, daß der Speicherinhalt durch Regelungen in bestimmten Grenzen gehalten wird. Die gebräuchlichsten Regelgrößen bei Stoffspeichern sind Füllstände bzw. Gasdrücke, bei Speichern thermischer Energie Temperaturen. Als Stellgröße der Bilanzregelung wird in einen der Zu- oder Abflüsse eingegriffen und dadurch das Gleichgewicht herbeigeführt. Beim Entwurf des Automatisierungssystems einer technologischen Anlage sollte mit der B. begonnen werden.

G. Brack

Bildschirmsystem – (s. a. Teil A–E, S. 409) Anzahl unterschiedlicher Gerätetypen, die für verschiedene Einsatzfälle letztlich die Ausgabe von – i. allg. alphanumerischen Informationen – auf Bildschirmen realisieren und der Eingabe von Informationen über die dazugehörigen Tastaturen dienen. Unterschieden wird nach Art des Rechneranschlusses und Anzahl der Bildschirme je Rechneranschluß.

Bei der Anschlußart wird zwischen Nah- und Fernaufstellung unter-

schieden, die wieder als Einzel- oder Bildschirmgruppen möglich sind. Beim Nahanschluß ist das Steuergerät direkt mit dem lokalen Interface des Rechners verbunden, und die über Spezialkabel angeschlossenen Bildschirmgeräte können davon bis zu 1200 m entfernt aufgestellt werden. Beim Fernanschluß werden die Signale des lokalen Parallelinterface durch einen Leitungsmultiplexer in ein Serieninterface umgewandelt und dann über postalische Zwei- oder Vierdrahtverbindungen zu einem beliebig weit entfernten Punkt übertragen. Dort übernimmt wieder ein Steuergerät die Anpassung der Signale an das Interface der Bildschirmgeräte, die dann wieder im Umkreis von 1200 m zu diesem Steuergerät installiert sein müssen.

Der Nahanschluß zeichnet sich durch hohe Datenraten und relativ einfache Bedienungsprozeduren aus und erzeugt damit im steuernden Rechner wenig System-Overhead. Kennzeichnend für den Fernanschluß ist die Überbrückung beliebiger Entfernungen.

An die Stelle der Bildschirmgeräte mit Tastatur – Aus- und Eingabemöglichkeit – können auch Seriendrucker – nur Ausgabemöglichkeit –, die im allg. als Protokolldrucker Verwendung finden, treten. Typische Vertreter der B. sind das EC 7920 des *ESER* und IBM 3270 von *IBM*.

H.-J. Schiller

Bildvorverarbeitung – Teilgebiet der *digitalen Bildverarbeitung* mit dem Ziel der Verbesserung der Bildqualität sowie der Aufbereitung des Bildinhaltes zur nachfolgenden automatischen Klassifizierung dieses Inhaltes bzw. zur besseren visuellen Verarbeitung durch den Menschen. Das Ergebnis der B. ist stets wieder ein Bild.

Wesentliche Teilgebiete der B. sind Methoden zur Rausch- und Störungsunterdrückung, zur Kantendetektion, Richtungsdetektion, Binarisierung, Verdünnung und Verdickung, Restaurierung u. a. Dabei existieren sowohl analoge optisch-holographische Verfahren als auch Verfahren der digitalen Bildverarbeitung mit Hilfe von Rechenanlagen. Im zweiten Fall wird mit einer Abtastkamera ein Rasterbild mit i. allg. 128×128 bis 4096×4096 Rasterpunkten erzeugt. Die Helligkeitswerte der einzelnen Rasterpunkte sind zwischen 2 und 256 Stufen quantisiert. Auch das Ergebnis der digitalen B. ist wieder ein digitalisiertes Rasterbild. Bei der Störungsunterdrückung sind sowohl Filterverfahren (Hochpaßfilterung, Tiefpaßfilterung) als auch Maskenoperatoren im Einsatz, die für eine Realisierung als *Spezialprozessor* besonders geeignet sind. Der bekannteste Maskenoperator zur Störungsunterdrückung ist der Rangordnungsoperator mit dem Spezialfall Medianoperator. Bei diesem Operator werden die innerhalb einer Maske um den betrachteten Bildpunkt liegenden Grauwerte (gebräuchlich sind 3×3-, 5×5- und 7×7-Masken) nach ihrem Wert in einer Reihe geordnet und der

Wert des mittleren Elements der Reihe als neuer Grauwert in den Bildpunkt geschrieben. Der Vorteil des Medianoperators besteht darin, daß er Störungen unterdrückt, ohne Grauwertkanten zu verflachen. Verfahren zur Kantendetektion beruhen auf der Auswertung von Grauwertgradienten in einer Bildpunktumgebung (lokale Kantenoperatoren) bzw. unter Beachtung von Optimalkriterien, die sich auf das gesamte Bild beziehen (globale Kantenoperatoren). Der bekannteste lokale Kantenoperator ist der SOBEL-Operator. Durch Auswertung von Grauwertgradienten lassen sich ebenso Richtungen von Grauwertkanten detektieren. Zur Vereinfachung der nachfolgenden Bildverarbeitung wird das Bild in vielen Fällen binarisiert (Grauwerte 0 und 1). Das Problem besteht dabei in der Selektion einer geeigneten Binarisierungsschwelle. Hier ist es nur in Spezialfällen möglich, eine Schwelle fest vorzugeben. In der Regel wird die Schwelle entweder von Bild zu Bild neu bestimmt, oder sie wird innerhalb eines Bildes in Abhängigkeit von den Grauwertverhältnissen verschiedene Werte annehmen (adaptive Schwelle). Verfahren zur Schwellenselektion beruhen auf der Auswertung von Grauwerthistogrammen (Häufigkeiten der Grauwerte), evtl. nach deren vorheriger Modifizierung. Zur bessern Auswertung von Objektkonturen im (meist schon binarisierten) Bild dienen Methoden zu deren Verdünnung (ggf. bis zur 1-bildpunktbreiten Skelettlinie) und Verdickung bzw. Restaurierung auf Grund von Modellvorstellungen über den möglichen Verlauf von Objektkonturen.

Der derzeitige Forschungsschwerpunkt auf dem Gebiet der B. besteht sowohl in der Realisierung von Algorithmen als Spezialprozessoren mit integrierten Schaltkreisen bzw. in VLSI-Technik oder als Arrayprozessoren und deren Implementierung auf speziellen Bildverarbeitungssystemen als auch in der Entwicklung von Algorithmen unter Beachtung von Prinzipien, die eine besonders effektive Hardware-Realisierung erlauben (z. B. systolische Algorithmen). Spezialprozessoren realisieren einen bzw. über eine Mikroprogrammierung mehrere ähnliche Operatoren. Beispiele sind der Prozessor GIPP (grayscale image preprocessor) vom ZKI der AdW der DDR und der Prozessor CYTO III der amerikanischen Firma Synthetic Vision System für Maskenoperatoren. Demgegenüber zeichnen sich Arrayprozessoren durch eine Zusammenschaltung vieler gleichartiger Prozessoren bzw. Zentralprozessoreinheiten aus, die durch entsprechende Programmierung einen speziellen Algorithmus realisieren können. Damit ist mit einem Arrayprozessor eine Vielfalt unterschiedlicher Operatoren realisierbar. Der technische Aufwand ist gegenüber Spezialprozessoren wesentlich höher. Als Beispiel sei das Multiprozessorsystem ZMOB genannt, das aus 256 Z80A-Mikroprozessoren besteht und eine Fülle von Bildverarbeitungsoperationen einschließlich diskreter Transformationen, statistischer Operationen und geometri-

scher Operationen auszuführen gestattet. Ein weiterer Arrayprozessor ist der Prozessor CLIP, aus 96×96 Einzelprozessoren bestehend, bei denen je vier Nachbarn zusammengeschaltet sind.

W. Uebel

Literatur: Simon, H., K. D. Kunze, K. Voss u. W. R. Herrmann: Automatische Bildverarbeitung in Medizin und Biologie. Dresden: Verlag T. Steinkopff 1975; Kowalewski, W. A.: Image Pattern Recognition. Berlin, Heidelberg, New York: Springer-Verlag 1980; Duff, M. J. B., a. S. Levialdi: Languages and Architecture for Image Processing. New York: Academic Press 1981; Rosenfeld, A.: Digital Picture Analysis. Berlin, Heidelberg, New York: Springer-Verlag 1976; Ballard, D. H., a. C. M. Brown: Computer Vision. Princeton: Prentice-Hall 1982; Rosenfeld, A., a. A. C. Kak: Digital Picture Processing. New York: Academic Press 1976; Höhne, K. H.: Digital Image Processing in Medicine. Berlin, Heidelberg, New York: Springer-Verlag 1981; Chung, T., u. G. Schwarze: Ein programmierbarer Prozessor für die Vorverarbeitung von Grautonbildern. Patent, Berlin 1983; Kazmierczak, H.: Erfassung und maschinelle Verarbeitung von Bilddaten. Berlin: Akademie-Verlag 1980; Klette, R., U. Rösler u. G. Sommer: Digitale Bildverarbeitung in der Automatisierungstechnik. messen.steuern.regeln, Berlin **26** (1983) 11, S. 607–615.

Binärsteuerung – In Binärsteuerungen werden zweiwertige Eingangssignale durch *logische Verknüpfung* zu zweiwertigen Ausgangssignalen verarbeitet. Die beiden Werte des Informationsparameters müssen deutlich unterscheidbar sein; ihnen werden i. allg. für die formale Beschreibung die Werte Null und Eins zugewiesen. Man unterscheidet prinzipiell die getaktete (synchrone) und die ungetaktete (asynchrone) Betriebsart von Binärsteuerungen sowie statische (kombinatorische) und dynamische (sequentielle oder Folge-) Steuerungen. Dynamische Steuerungen enthalten Speicher für binäre Größen. Die algebraische Beschreibung einer Binärsteuerung erfolgt durch die (i. allg. nichtlineare) Zustandsgleichung

$$^{k}\boldsymbol{z} = f(^{k-1}\boldsymbol{z}, {}^{k}\boldsymbol{x})$$

und die Ausgabegleichung

$$^{k}\boldsymbol{y} = g(^{k}\boldsymbol{z}, {}^{k}\boldsymbol{x}) .$$

$^{k-1}\boldsymbol{z}$ ist der Zustand der Steuerung vor Auftreten einer Änderung des Eingangssignalvektors $\boldsymbol{x}$. $^{k}\boldsymbol{z}$ ist der daraus resultierende Folgezustand; f wird als Überführungsfunktion bezeichnet. $\boldsymbol{y}$ ist der Ausgangssignalvektor, der durch die Ausgabe- oder Ergebnisfunktion g mit dem Zustand und dem Eingang verknüpft ist.

Bei der Analyse und dem Entwurf von Binärsteuerungen werden außer

den BOOLEschen Gleichungssystemen der Problemstellung angepaßte Beschreibungsmittel, z. B. *formale Sprachen*, Tabellen (*Automatentabelle, Entscheidungstabelle*), *Graphen* (Automatengraph, Programmablaufsprache, Steuergraph, *Petrinetze*) sowie Zeitdiagramme verwendet. Der systematische Entwurf einer Binärsteuerung setzt die Formalisierung der Aufgabenstellung unter Verwendung der genannten Beschreibungsmittel voraus und hat aufwandsarme Lösungen, gegebenenfalls unter Berücksichtigung von Zusatzforderungen (z. B. Zuverlässigkeit, automatische Fehlererkennung), zum Ziel.

Die Realisierung erfolgt z. B. mit elektrischen, pneumatischen und hydraulischen Relais. Weit verbreitet sind elektronische Steuerungssysteme mit diskreten oder integrierten Dioden- und Transistorschaltkreisen. Hochintegrierte Binärsteuerungen sind entweder durch PROMs bzw. PLA-Schaltkreise speicherprogrammierbar oder durch Einsatz von gewöhnlichen oder speziellen *Mikrorechnern* frei programmierbar.

H. EHRLICH

Binärvektor – s. BOOLE*scher Raum.*

Binder – s. *Verbinder.*

Black-box-Methode, rechnergestützte – Methode zur Analyse und Simulation der Funktions- und Verhaltensweisen komplexer Systeme bei bekannten Eingangs- und Ausgangsgrößen und unbekannter Struktur und unbekannten Prozeßabläufen innerhalb dieser Systeme (s. *Schwarzer Kasten*) unter Einsatz der elektronischen *Rechentechnik*. Ergebnis der Anwendung der r. B. bilden:

- die Ermittlung des Transformationsoperators T des analysierten Systems, z. B. in Form einer Matrix $\boldsymbol{A}$ (s. *Matrix der partiellen Wirkung*);
- die Bestimmung voraussichtlicher Prozeßergebnisse

  $$Y = T \cdot X$$

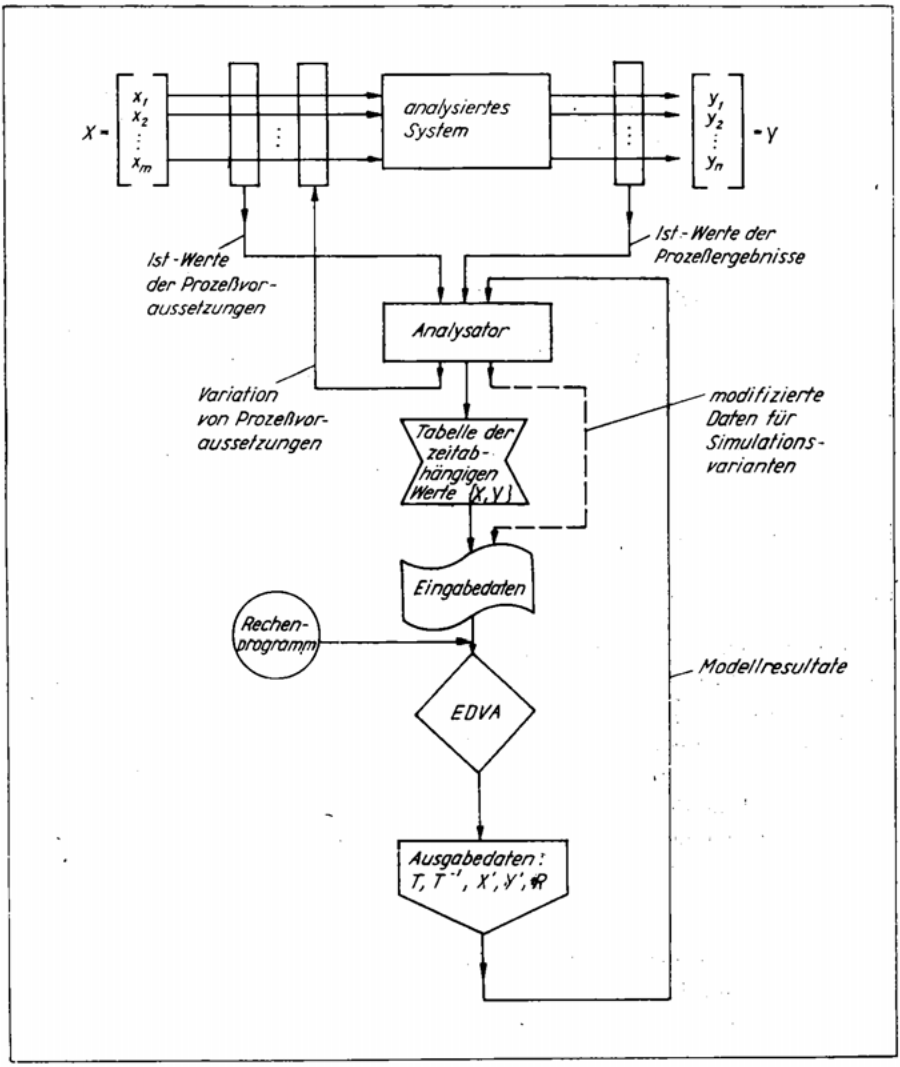
  bei gegebenen Prozeßvoraussetzungen X;
- die Berechnung benötigter Prozeßvoraussetzungen

  $$X' = T^{-1} \cdot Y'$$

  bei Vorgabe zu erreichender Prozeßergebnisse Y';
- die Ableitung hypothetischer Aussagen über die Struktur R des analysierten Systems.

Entsprechend der Abbildung (s. S. 52) ergibt sich der dargestellte Ablauf der Analyse und Simulation unter Nutzung einer EDVA; die Simulation kann dabei zyklisch in Varianten durchgeführt werden.

H.-G. LAUENROTH



Prinzip der rechnergestützten black-Box-Analyse

Literatur: LAUENROTH, H.-G., u. K. BÖHM: Kybernetik in der industriellen Organisation. Berlin: VEB Verlag Technik 1979.

BOOLEsche Ableitung — s. BOOLEscher *Differentialkalkül*.

Boolesche Differentialgleichung — s. Boolescher *Differentialkalkül.*

Boolesche Differenz —s. Boolescher *Differentialkalkül.*

Boolesche Gleichung — Durch $\boldsymbol{x}=(x_1,\ldots, x_k)$, $x_i \in \{0,1\}$, seien Binärvektoren (s. Boolescher *Raum*) beschrieben. Mit zwei Booleschen *Funktionen* $f(\boldsymbol{x})$ und $g(\boldsymbol{x})$ wird die Gleichung

$$f(\boldsymbol{x})=g(\boldsymbol{x}) \tag{1}$$

gebildet. Ihre Lösungsmenge bilden alle $\boldsymbol{x}$, die (1) identisch erfüllen. Die gleiche Lösungsmenge besitzt die homogenisierte Gleichung

$$f(\boldsymbol{x}) \nsim g(\boldsymbol{x})=0\,, \tag{2}$$

so daß es genügt, sich mit der Lösung und Interpretation homogener Gleichungen vom Typ

$$h(\boldsymbol{x})=0 \tag{3}$$

zu befassen. Sie treten in allen Anwendungsbereichen der Booleschen *Algebren* auf (Logik, Theorie der *Schaltsysteme, Automatentheorie, Algorithmentheorie, Graphentheorie, diskrete Optimierung* u. a.). Die Bestimmung der Lösungsmenge ist, im Gegensatz zur einfachen Prüfbarkeit eines Vektors $\boldsymbol{x}$ auf Lösungseigenschaften, eine i. allg. äußerst komplexe Aufgabe, da die Zahl der Lösungsvektoren exponentiell mit der Variablenzahl steigt. Praktisch ist sie bis zu einigen 10 Variablen allgemein möglich (s. *Ternärvektorliste*), darüber hinaus führen in einigen Fällen spezielle Funktionenklassen, Dekompositions- und Näherungsmethoden weiter.

Ist ein System von Gleichungen $h_1(\boldsymbol{x})=0, \ldots, h_n(\boldsymbol{x})=0$ gegeben, so läßt es sich stets äquivalent durch

$$h_1(\boldsymbol{x}) \vee \ldots \vee h_n(\boldsymbol{x})=0 \tag{4}$$

ersetzen und damit auf den Typ (3) zurückführen.

Praktisch wichtig ist das Auflösungsproblem Boolescher Gleichungen. Es sei die Gleichung

$$h(x_1,\ldots, x_k, y_1,\ldots, y_n)=0$$

gegeben, die lösungsäquivalent durch ein Gleichungssystem

$$\begin{aligned} y_1&=f_1(\boldsymbol{x})\\ &\vdots\\ y_n&=f_n(\boldsymbol{x}) \end{aligned} \tag{5}$$

zu ersetzen ist. Die Auflösbarkeit ist gegeben, wenn die Bedingung

$$\min_{\boldsymbol{y}}{}^{n} h(\boldsymbol{x}, \boldsymbol{y})=0 \tag{6}$$

erfüllt ist (s. BOOLE*scher Differentialkalkül*), sie führt jedoch auch bei Gültigkeit von (6) i. allg. nicht zu eindeutigen Lösungen. Für jede Funktion f_i läßt sich ein Funktionenverband angeben, dessen Elemente zulässige Werte von f_i bilden. Mit $\boldsymbol{y}_0=(y_1,\ldots,y_{i-1},y_{i+1},\ldots,y_n)$ gilt für f_i

$$\min_{\boldsymbol{y}_0}{}^{n-1}h(\boldsymbol{x},\boldsymbol{y})|_{y_i=0}\leqq f_i\leqq\min_{\boldsymbol{y}_0}{}^{n-1}h(\boldsymbol{x},\boldsymbol{y})|_{x_i=1}. \tag{7}$$

Die Auflösung ist eindeutig, wenn untere und obere Grenze des Verbandes zusammenfallen. Bemerkenswert ist der Satz: Sei $h(\boldsymbol{x},\boldsymbol{y})=0$ nach $\boldsymbol{y}$ auflösbar, dann ist die Gleichung auch nach jedem anderen Vektor $\boldsymbol{y}^*$ auflösbar, der wenigstens die Variablen von $\boldsymbol{y}$ enthält.

D. BOCHMANN

Literatur: BOCHMANN, D., u. CH. POSTHOFF: Binäre dynamische Systeme. Berlin: Akademie-Verlag 1981, München, Wien: Oldenbourg-Verlag 1981; BOCHMANN, D., CH. POSTHOFF u. A. D. ZAKREWSKIJ: BOOLEsche Gleichungen – Theorie, Anwendungen, Algorithmen. Berlin: VEB Verlag Technik 1984; RUDEANU, S.: Boolean Functions and Equations. Amsterdam: North-Holland Publ. Co. 1974.

BOOLEscher Differentialkalkül – Innerhalb der BOOLE*schen Algebren* gibt es zwei Operationen mit Gruppeneigenschaft, die Antivalenz ($\curlyvee$, $\oplus$) und die Äquivalenz ($\equiv$), die deshalb als Summen- und auch Differenzoperationen interpretiert werden können. Auf ihnen aufbauend konnte ein Differenzen- und Differentialkalkül geschaffen werden (AKERS 1959, TALANZEW 1959, BOCHMANN 1963, THAYSE 1970), in dem jedoch wegen der praktischen Anschaulichkeit (Identität der Operanden $\Leftrightarrow$ Ergebnis der Differenzbildung $=0$) die Antivalenz bevorzugt wurde. Der Kalkül erlaubt die Behandlung zeitlicher, räumlicher und anderer Änderungen in binären dynamischen Systemen.

Es sei $\boldsymbol{x}=(x_1,\ldots,x_k)$, $x_i\in\{0,1\}$, ein Binärvektor. Ihm wird das Differential $\mathrm{d}\boldsymbol{x}=(\mathrm{d}x_1,\ldots,\mathrm{d}x_k)$, $\mathrm{d}x_i\in\{0,1\}$, als Vektor von Änderungsgrößen (i. allg. unabhängig von $\boldsymbol{x}$) zugeordnet. Nach einer Änderung von $\boldsymbol{x}$ durch $\mathrm{d}\boldsymbol{x}$ entsteht ein neuer Vektor $\boldsymbol{x}^*$. Es gilt

$$\boldsymbol{x}^*=\boldsymbol{x}\curlyvee\mathrm{d}\boldsymbol{x}\,, \tag{1}$$

$$\mathrm{d}\boldsymbol{x}=\boldsymbol{x}\curlyvee\boldsymbol{x}^*\,. \tag{2}$$

Durch (2) wird der Begriff BOOLEsche Differenz nahegelegt, der aber in der Literatur auch in anderer Bedeutung auftritt.

Als (totales) Differential einer Funktion $f(\boldsymbol{x})$ wird definiert

$$\mathrm{d}f(\boldsymbol{x})=f(\boldsymbol{x})\curlyvee f(\boldsymbol{x}\curlyvee\mathrm{d}\boldsymbol{x}). \tag{3}$$

Wird die Änderung nur auf eine Variable x_i bezogen (nur $\mathrm{d}x_i \neq 0$), so spricht man vom partiellen Differential

$$\underset{x_i}{\mathrm{d}} f(x_i) = f(x_i) \nprec f(x_i \nprec \mathrm{d}x_i). \tag{4}$$

(Jeweils konstante Parameter sollen nicht extra geschrieben werden.)

In (4) ist der Wert von $\mathrm{d}x_i$ noch beliebig. Setzt man $\mathrm{d}x_i = 1$, so entsteht die von x_i unabhängige partielle Ableitung von f nach x_i:

$$\frac{\partial f}{\partial x_i} = f(x_i) \nprec f(\bar{x}_i) . \tag{5}$$

Eine ähnliche Festlegung für den gesamten Vektor $\boldsymbol{x}$ führt zur vektoriellen Ableitung von f nach $\boldsymbol{x}$:

$$\frac{\partial f}{\partial \boldsymbol{x}} = f(\boldsymbol{x}) \nprec f(\bar{\boldsymbol{x}}) . \tag{6}$$

Ableitungen können mehrfach ausgeführt werden. Es gelten für die Ausführung der Ableitungsoperatoren Kommutativitäts- und Assoziativitätsgesetze und die wichtige Eigenschaft

$$\frac{\partial}{\partial x_i} \frac{\partial f}{\partial x_i} = 0 . \tag{7}$$

Weitere Begriffe des B. D. entstanden (unter Verzicht auf die Gruppenaussagen) dadurch, daß statt der Antivalenzoperation die Konjunktion und Disjunktion verwendet werden:

$$\mathrm{Min}\, f(\boldsymbol{x}) = f(\boldsymbol{x}) \wedge f(\boldsymbol{x} \nprec \mathrm{d}\boldsymbol{x}) , \tag{8}$$

$$\mathrm{Max}\, f(\boldsymbol{x}) = f(\boldsymbol{x}) \vee f(\boldsymbol{x} \nprec \mathrm{d}\boldsymbol{x}) , \tag{9}$$

$$\min_{x_i} f(x_i) = f(x_i) \wedge f(\bar{x}_i) , \tag{10}$$

$$\max_{x_i} f(x_i) = f(x_i) \vee f(\bar{x}_i) . \tag{11}$$

Die Operatoren Min und Max sind hier als totales Minimum und Maximum geschrieben und hängen eng mit dem totalen Differential zusammen,

$$\mathrm{d}f(\boldsymbol{x}) = \mathrm{Min}\, f(\boldsymbol{x}) \nprec \mathrm{Max}\, f(\boldsymbol{x}) . \tag{12}$$

Die mit den Ableitungen verwandten (hier partiell geschriebenen) Operatoren min und max genügen

$$\frac{\partial f}{\partial x_i} = \min_{x_i} f(x_i) \nprec \max_{x_i} f(x_i) \tag{13}$$

und können ebenfalls mehrfach ausgeführt werden. Außerdem ist es für

manche Anwendungen zweckmäßig, gerichtete Differentiale und Ableitungen einzuführen, bei denen die Änderungsrichtungen $0 \to 1$ und $1 \to 0$ unterschieden werden.

Anschaulich sind folgende Interpretationen wichtig:

$f\partial/\partial x_i$ ist genau dann gleich 1, wenn eine Änderung von x_i zu einer Änderung von f führt.

$\partial \boldsymbol{x}/\partial f$ ist genau dann gleich 1, wenn eine Änderung des gesamten Vektors $\boldsymbol{x}$ (z. B. $(0, ..., 0) \to (1, ..., 1)$ zu einer Änderung von f führt.

$\mathrm{d}f(\boldsymbol{x})$ ist in Abhängigkeit von den Werten der $\mathrm{d}\boldsymbol{x}$ (der gerade auftretenden Änderungen) gleich 1, wenn sich f ändert, d. h., es enthält alle Ableitungen als durch die Parameter $\mathrm{d}x_i$ festzulegende Spezialfälle.

Auch für alle anderen Operatoren lassen sich solche anschaulichen Interpretationen angeben, so daß die Mittel des B. D. eine praktisch bequeme Formulierung vielfältiger dynamischer Problemstellungen unterstützen. Für komplexere Änderungsbegriffe (z. B. Änderung von f durch die Änderung wenigstens einer Variabler aus einer gegebenen Teilmenge) wurden ebenfalls Operatoren bereitgestellt, ihre Eigenschaften untersucht und beschrieben.

Wir geben einige für partielle, aber auch für vektorielle und mehrfache Ableitungen geltende Zusammenhänge an:

$$\frac{\partial}{\partial x_i}(f(\boldsymbol{x}) \curlywedge g(\boldsymbol{x})) = \frac{\partial f}{\partial x_i} \curlywedge \frac{\partial g}{\partial x_i}, \tag{14}$$

$$\frac{\partial}{\partial x_i}(f(\boldsymbol{x}) \wedge g(\boldsymbol{x})) = g\frac{\partial f}{\partial x_i} \curlywedge f\frac{\partial g}{\partial x_i} \curlywedge \frac{\partial f}{\partial x_i}\frac{\partial g}{\partial x_i}, \tag{15}$$

$$\frac{\partial}{\partial x_i}(f(\boldsymbol{x}) \vee g(\boldsymbol{x})) = \bar{g}\frac{\partial f}{\partial x_i} \curlywedge \bar{f}\frac{\partial g}{\partial x_i} \curlywedge \frac{\partial f}{\partial x_i}\frac{\partial g}{\partial x_i}. \tag{16}$$

Ähnliche Beziehungen gelten auch für die Differentiale. Die Umkehrung der Ableitungs- und Differentialoperatoren (z. B. das Bestimmen von $f(\boldsymbol{x})$ aus

$$g(\boldsymbol{x}) = \frac{\partial f}{\partial \boldsymbol{x}} \tag{17}$$

bei gegebenen $g(\boldsymbol{x})$) kann als Spezialfall der Untersuchung und Lösung Boolescher Differentialgleichungen angesehen werden, für die eine Reihe analytischer und rechnerorientierter Lösungsverfahren erarbeitet wurden. Zwei Zusammenhänge betonen dabei die besondere praktische Bedeutung:

1. Lösungen von Booleschen Differentialgleichungen, die Ableitungsoperationen enthalten, sind Mengen binärer Funktionen. Es wurde gezeigt,

daß zu jeder Menge binärer Funktionen eine generierende Differentialgleichung gehört. Damit entstand für die Untersuchung von Klassen binärer Funktionen ein neues Instrument, das zur Analyse, Optimierung, Funktionsauswahl u. a. eingesetzt werden kann.

2. Die besonders einfachen Differentialgleichungen vom Typ

$$f(\boldsymbol{x}, \mathrm{d}\boldsymbol{x}) = 0 \tag{18}$$

definieren *Graphen* und werden erfolgreich zu deren Untersuchung eingesetzt. Dabei werden die $\boldsymbol{x}$ als Punkte in einem Boole*schen Raum* und $\mathrm{d}\boldsymbol{x}$ als Richtung einer von $\boldsymbol{x}$ wegführenden Kante interpretiert. Alle Lösungen von (18) gelten als Kanten des Graphen. Die Handhabung, Analyse, Synthese usw. eines endlichen Graphen wird damit auf entsprechende Operationen mit binären Funktionen zurückgeführt.

Anwendungen findet der B. D. bisher vorrangig bei Aufgaben der Diagnose, Testung, Testsatzermittlung und Empfindlichkeitsanalyse binärer Automaten. Weitere Anwendungsgebiete sind Hasardanalyse, Berücksichtigung und Bestimmung dynamischer Effekte und Nebenbedingungen in Automaten, Lösung und Auflösung Boolescher Gleichungssysteme, Verhaltensanalyse von Automaten.

Es gibt eine Reihe von Ergebnissen, die bei den Spezialfällen zeitlicher und räumlicher Änderungen den B. D. bereichern und weiterführen.

D. Bochmann

Literatur: Bochmann, D., u. V. N. Roginskij: Dynamische Prozesse in Automaten. Berlin: VEB Verlag Technik 1977; Bochmann, D., u. Ch. Posthoff: Binäre dynamische Systeme. Berlin: Akademie-Verlag 1981, München, Wien: Oldenbourg-Verlag 1981; Thayse, A.: Boolean Calculus of Differences. Berlin, Heidelberg, New York: Springer-Verlag 1981; Bochmann, D.: Automatengraphen. Berlin: Akademie-Verlag 1982; Bochmann, D., Ch. Posthoff u. A. D. Zakrewskij: Boolesche Gleichungen – Theorie, Anwendungen, Algorithmen. Berlin: VEB Verlag Technik 1984.

Boolescher Differentialoperator – s. Boole*scher Differentialkalkül.*

Boolescher Raum – Es sei $\boldsymbol{x} = (x_1, \ldots, x_k)$ ein Binärvektor (k-Tupel), dessen Komponenten x_i nur die Werte 0 und 1 annehmen, die als Definitions- und Wertebereich Boolescher Funktionen aufgefaßt werden. Die Gesamtheit aller $\boldsymbol{x}$ für festes k bildet einen Booleschen Raum B^k, der vielfältige theoretische und praktische Bezüge besitzt. Praktisch bedeutungsvoll ist insbesondere die bei vielen Problemen auftretende Forderung, in hochdimensionalen B. R. Rechnungen auszuführen.

Für die x wird eine Halbordnungsrelation eingeführt: Für zwei Vektoren $x_1 \leqq x_2$ ist komponentenweise nur eine der Kombinationen $0 \leqq 0$, $0 \leqq 1$, $1 \leqq 1$ erlaubt. Ordnungsrelationen lassen sich in vielfältiger Weise festlegen. Am wichtigsten sind die durch das Dezimaläquivalent (Interpretation von x als Dualzahl) oder einen GRAY-Code vorgegebenen Reihenfolgen. Für B^k kann gezeigt werden, daß es sich um einen topologischen, regulären, kompakten Raum handelt, der die Eigenschaften eines linearen Vektorraums erfüllt. Die wichtigste Metrik in B^k ist die HAMMING-Distanz $H\mathrm{d}(x_1, x_2)$: die Zahl der unterschiedlichen Komponenten der Vektoren x_1 und x_2. Es sei $x_0 \in B^k$, dann bilden alle Vektoren mit $H\mathrm{d}(x_0, x) \leqq h$ eine Hyperkugel um x_0 mit dem Radius h. Der gesamte Raum B^k bildet um jeden seiner 2^k Punkte eine Hyperkugel mit dem Radius k.

Es seien x_1 und x_2 zwei Punkte mit $x_1 \leqq x_2$ und $H\mathrm{d}(x_1, x_2) = h$. Dann stimmen x_1 und x_2 in $k-h$ Stellen überein, während in h Stellen bei x_1 eine 0 und bei x_2 eine 1 steht. Die Menge aller x mit $x_1 \leqq x \leqq x_2$ bildet dann einen BOOLEschen Unterraum B^h, der allen schon genannten Bedingungen ebenfalls genügt. Kleinste Elemente (Infima) sind in B^k der Nullvektor 0 und in B^h x_1, größte Elemente (Suprema) entsprechend 1 und x_2. Der von uns gewählte Unterraum B^h läßt sich zweckmäßig durch einen Ternärvektor der Länge k kennzeichnen, der in den festgelegten Stellen 0 oder 1 und in den variablen Stellen das Symbol „–" enthält. Jede Punktmenge in B^k kann durch geeignete Zerlegung in BOOLEsche Unterräume als *Ternärvektorliste* (TVL) dargestellt werden. Sind die Unterräume disjunkt, so kommt man zu orthogonalisierten TVL, die das effektivste rechnerinterne Mittel zur Ausführung von Rechnungen in B. R. sind. BOOLEsche Unterräume werden auch als Hyperkuben bezeichnet.

Eine Punktmenge in B^k kann durch eine charakteristische Funktion gekennzeichnet werden, die genau für die Punkte der Menge den Wert 1 annimmt. Jede Funktion $f(x)$ kennzeichnet in diesem Sinne einen Teilraum. Maximal große Hyperkuben, die nur 1-Punkte der Funktion enthalten, bilden ihre Primimplikanten. Die zugehörigen Ternärvektoren haben maximale Strichzahlen. Eine Überdeckung aller 1-Punkte durch eine minimale Zahl von Primimplikanten führt zu einer (unter bestimmten Voraussetzungen) aufwandsminimalen Schaltung, die die gegebene Funktion realisiert.

Von großem Interesse ist das Einbetten endlicher *Graphen* in B. R. hinreichend großer Dimension. Die Knoten des Graphen werden mit Punkten des Raumes identifiziert, die Kanten werden zu Übergängen zwischen diesen. Eine Kante wird dann durch einen Punkt-Richtungs-Vektor $(x, \mathrm{d}x)$ charakterisiert (BOOLE*scher Differentialkalkül*), der gesamte (schlichte, ungewichtete) Graph durch eine charakteristische Funktion $g(x, \mathrm{d}x)$ oder eine BOOLEsche Gleichung $f(x, \mathrm{d}x) = 0$ mit $f = \bar{g}$. Die für die Behandlung BOOLE-

scher Räume und Funktionen bereitgestellten Mittel werden damit auch für Graphen verfügbar.

Die Abbildung eines B. R. auf sich selbst oder in (auf) einen anderen B. R. wird im allgemeinen als Umcodierung der Punkte (z. B. der Knoten des Graphen) zu verstehen sein. Sie ist technisch durch Schaltnetzwerke mit mehreren Ausgängen zu realisieren und bewirkt eine Reihe mathematischer Zusammenhänge, wie z. B. Vergröberung eines Graphen (oder anderer in B^k auftretender Objekte), Stetigkeit der Abbildung (verbunden mit Hasard-Problemen) u. a.

Eine Boolesche *Funktion* von k Variablen kann ebenfalls als 2^k-dimensionaler Binärvektor aufgefaßt werden. Jedem Punkt in B^{2^k} ist damit eine Funktion über B^k zugeordnet. Boolesche Unterräume in B^{2^k} beschreiben dann den wichtigen Fall der Verbände Boolescher Funktionen.

D. Bochmann

Literatur: Rudeanu, S.: Boolean Functions and Equations. Amsterdam: North-Holland Publ. Co. 1974; Bochmann, D.: Einführung in die strukturelle Automatentheorie. Berlin: VEB Verlag Technik 1975, München, Wien: Hanser-Verlag 1975; Bochmann, D., u. Ch. Posthoff: Binäre dynamische Systeme. Berlin: Akademie-Verlag 1981, München, Wien: Oldenbourg-Verlag 1981; Thayse, A.: Boolean Calculus of Differences. Berlin, Heidelberg, New York: Springer-Verlag 1981; Bochmann, D., Ch. Posthoff u. A. D. Zakrewskij: Boolesche Gleichungen – Theorie, Anwendungen, Algorithmen. Berlin: VEB Verlag Technik 1984.

Boolesches Differential – s. Boolesc*her Differentialkalkül.*

Boolesches Gleichungssystem – s. Boolesc*he Gleichung.*

Bootstrapping – Methode zur Erzeugung neuer Programmversionen unter Benutzung bereits vorhandener Versionen geringeren Leistungsvermögens; hat bei der Erzeugung von *Compilern* für höhere *Programmiersprachen* zu, sammen mit der Verwendung von Cross-Compilierung und *Compiler-Com-*

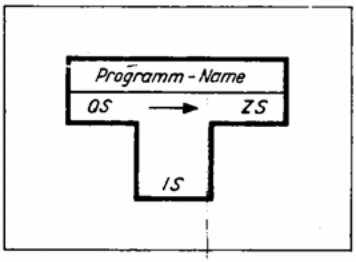


Abb. 1

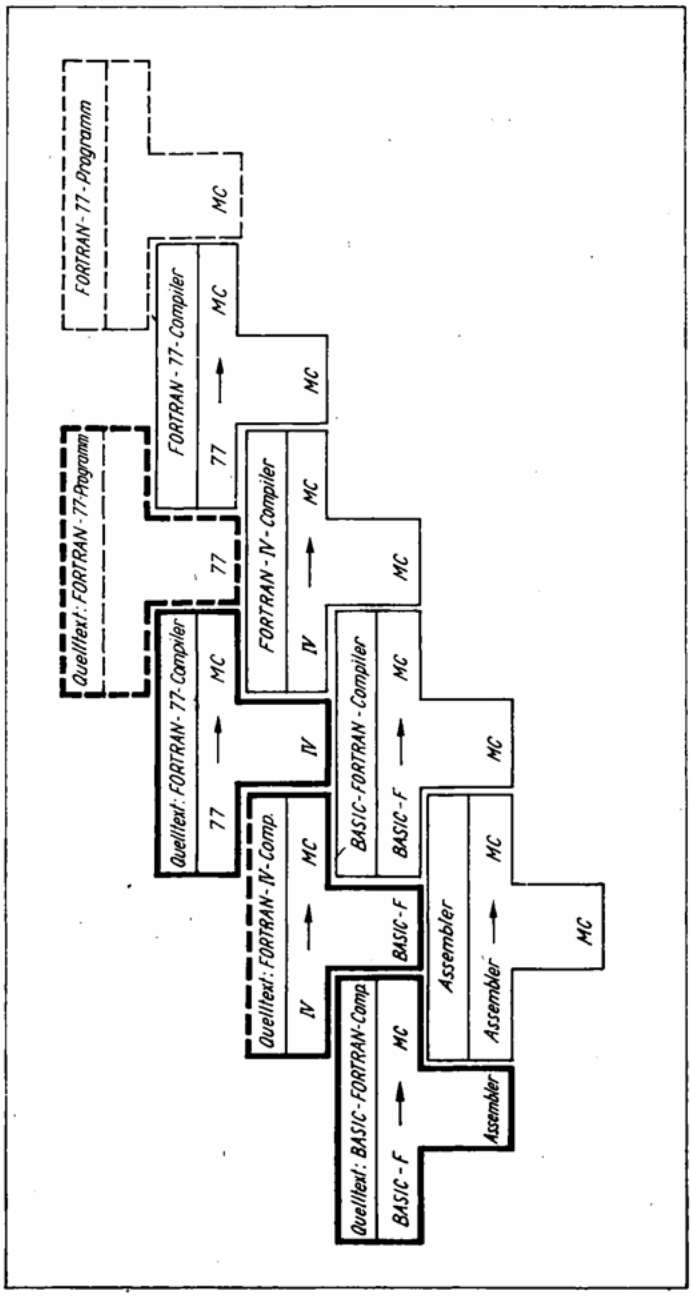

Quelltext: FORTRAN-77-Programm
FORTRAN-77-Programm
Quelltext: FORTRAN-77-Compiler
77 → MC
77
FORTRAN-77-Compiler
77 → MC
MC
Quelltext: FORTRAN-IV-Comp.
IV → MC
IV
FORTRAN-IV-Compiler
IV → MC
MC
Quelltext: BASIC-FORTRAN-Comp.
BASIC-F → MC
BASIC-F
BASIC-FORTRAN-Compiler
BASIC-F → MC
MC
Assembler
Assembler
Assembler → MC
MC
MC

Abb. 2

pilern besondere Bedeutung erlangt. Auch der Transport von Compilern zwischen verschiedenen Rechnertypen wird häufig durch B. durchgeführt.

Beim B. von Compilern wird zunächst ein Compiler für eine Untermenge der zu implementierenden Sprache erzeugt. Der eigentliche Compiler wird anschließend unter Verwendung dieser Sprachuntermenge programmiert und mit dem bereits implementierten Compiler übersetzt. Als Ergebnis dieser Übersetzung entsteht ein Compiler für den gesamten Sprachumfang, der auch in der Lage ist, sein eigenes Programm zu übersetzen (Selbstübersetzung). Soll später eine Obermenge der Sprache erzeugt werden (Sprachерweiterung), dann kann das Compilerprogramm unter Ausnutzung aller bisher implementierten Spracheigenschaften modifiziert werden.

Zur Beschreibung von Bootstrapping-Prozessen dienen die sog. T-Diagramme. Sie haben die in Abb. 1 gezeigte Form und enthalten die vom Compiler akzeptierte Quellsprache QS, die erzeugte Zielsprache ZS und die Implementierungssprache IS für den Compiler selbst. Ein *FORTRAN*-77-Compiler könnte z. B. mit dem in Abb. 2 angegebenen Bootstrapping-Prozeß erzeugt werden: Ein *BASIC-FORTRAN*-Compiler wird in *Assembler-Sprache* programmiert und mittels des vorhandenen Assemblers in ein abarbeitungsfähiges Programm (IS = Maschinencode MC) überführt. Der FORTRAN-IV-Compiler kann nun bereits in BASIC-FORTRAN formuliert werden, also auf einem wesentlich höheren Sprachniveau. Durch Übersetzung mit dem BASIC-FORTRAN-Compiler wird ein lauffähiger FORTRAN-IV-Compiler erhalten. Schließlich wird der FORTRAN-77-Compiler in FORTRAN-VI programmiert. Dabei wäre auch eine Implementierung als FORTRAN-77-FORTRAN-IV-Konverter möglich, die lediglich noch einen zusätzlichen Übersetzungslauf erfordern würde. Der durch mehrere Übersetzungsschritte erhaltene FORTRAN-77-Compiler schließlich übersetzt beliebige FORTRAN-77-Programme.

W. Assmann

Brainware — Überlegungen und potentielle Lösungen als Basis zur Erarbeitung von Software und auch Hardware für naturwissenschaftliche, technische, ökonomische und andere Aufgaben, die in dieser Form oder modifiziert für eine automatisierte Informationsverarbeitung geeignet sind, jedoch noch nicht genutzt werden.

Der Begriff Brainware entstand mit der Entwicklung und Vorbereitung der Begriffe *Software* und *Hardware* und soll verdeutlichen, daß der Hard- und Software eines Rechners noch die geistigen Fähigkeiten des Menschen übergeordnet sind. Demzufolge wird die Entwicklung der Brainware einerseits von den Forderungen und Tendenzen der Hard-, Soft- und *Orgware*-Entwicklung, zum anderen aber auch von der allgemeinen und der disziplinären Wissenschaftsentwicklung direkt und indirekt bestimmt.

Brainware wurde im kapitalistischen Sinne zuweilen auch als jene Menge „geistigen Kapitals", welche eine Firma besitzt, verstanden.

H.-G. LAUENROTH, H. VÖLZ

Büroautomatisierung – Prozeß, bei dem der Mensch durch Einbeziehung mikroelektronischer und nachrichtentechnischer Mittel von monotonen Informationsprozessen in der Verwaltung entlastet wird bzw. durch den bestimmte Informationsprozesse in der Verwaltung beschleunigt oder erst möglich werden.

Die B. ist gekennzeichnet durch den Einsatz von Datenverarbeitungs-, Textverarbeitungs- und Informationsübertragungssystemen in Arbeitsplatzrechnern, zunächst getrennt voneinander, später als integriertes System, sowie durch den Aufbau von *Informationssystemen* für komplexe Sachgebiete. Durch Integration der Verarbeitung, Speicherung und Übertragung von Sprache, Text, Daten und Graphik wird mit einem Gerät der Zugriff zu einer großen Zahl von Informationsquellen sowie das Zusammenspiel mit anderen Menschen und Rechnern möglich.

Datenverarbeitungssysteme zur B. sind vorrangig als Informationssysteme für solche Gebiete wie Planung, Statistik, Organisation, Kader, Bibliothek, Buch- und Lagerhaltung ausgeprägt.

Textverarbeitungssysteme ermöglichen die Eingabe, Speicherung, Korrektur, Formatierung und Ausgabe von Textinformationen. Sie sind in unterschiedlichen Ausbaustufen auch mit Mitteln der Textgestaltung versehen.

Der Informationsübertragungsanschluß bietet je nach vorhandener Kommunikationstechnologie den Anschluß an ein lokales Netz zur Abwicklung der hausinternen Kommunikation oder auch an ein öffentliches Datennetz mit den dort verfügbaren Diensten, wie Teletex, Telefax, Videotext.

C. SATTLER

Bus – lineare Verbindungsstruktur für die zeitmultiplexe Datenübertragung zwischen Funktionseinheiten eines *Mikroprozessors*, *Digitalrechners* bzw. *Mehrrechnersystems*. Der Bus stellt eine globale Kommunikationsressource des Systems dar, die durch die angeschlossenen Funktionseinheiten angefordert werden kann und diesen konfliktfrei für die Realisierung ihrer Übertragungsanforderungen zugeordnet wird.

Der Bus umfaßt i. allg. mehrere parallele Signalleitungen (*Bus-Leitungen*), an die alle kommunizierenden Funktionseinheiten angeschlossen sind. Anzahl und Funktion der Bus-Leitungen werden durch das Arbeitsprinzip des Bus festgelegt und bilden die *Bus-Struktur*. Die Funktionseinheiten sind über Bus-Koppler, die einen Master- und/oder Slave-Teil enthalten, an den Bus angeschlossen. Der Master kann die Bus-Zuordnung anfordern, diese bei