eigene Pap.

Suchlistenstrukturen zur Darstellung gerichteter Graphen und deren Anwendung bei Synthese und Minimierung spezieller endlicher Automaten

A linked list structure for the description of directed graphs and its application to synthesis and reduction of special finite state automata

Elektron. Rechenanl. 12 (1970), H. 4, S. 208—216
Manuskripteingang: 25. 4. 1970

von R. HARTENSTEIN, Institut für Nachrichtenverarbeitung und Nachrichtenübertragung der Universität Karlsruhe

Es wird die Programmierung eines Verfahrens zur Synthese und Minimierung endlicher Akzeptoren und ähnlicher Automaten beschrieben. Das Verfahren beruht auf einer Modifikation von Signalflußgraphen-Techniken zur Lösung linearer algebraischer Gleichungssysteme. Zur Darstellung gerichteter Graphen für endliche Automaten und reguläre Ausdrücke wird hierbei eine spezielle Listenstruktur angewandt, die eine rechenzeit- und speicher-effektive Programmierung gestattet.

The principles of programming an algorithm for synthesis and reduction of finite state acceptors and similar structures are described. The algorithm is based on a modified signal flow graph technique, known from the solution of linear algebraic equation systems. For the description of directed graphs for finite state automata and regular expressions a special list structure is used, which allows effectivity in storage requirements and computing time.

1. Einleitung

Zur Spezifizierung endlicher Automaten bei Computer-Eingabe, etwa für Minimierungszwecke, wird die Überführungsfunktion meist in Matrixform verwendet. An Hand zweier Anwendungsbeispiele soll gezeigt werden, daß demgegenüber eine aus dem gerichteten Graphen abgeleitete Form der Listendarstellung bei speziellen Typen des endlichen Automaten überlegen ist. Es handelt sich hierbei um den endlichen Akzeptor und verwandte Automatentypen, wie sie insbesondere für die Lösung von Klassifikationsproblemen und Problemen der Informationsreduktion interessant sind. Die Überlegenheit der verwendeten Listenstruktur wird bei der Synthese aus regulären Ausdrücken sowie bei der Minimierung aufgezeigt.

2. Voraussetzungen

Der deterministische endliche Akzeptor sei ein endlicher Automat nach folgender Definition.

Definition 1. Der deterministische *endliche Akzeptor* $\mathfrak{A}$ ist ein vollständig definierter endlicher Automat $\mathfrak{A} = (X, Z, \{z_0\}, \delta, Z' \subseteq Z)$. Hierbei ist Z die Zustandsmenge mit $Z = \{z_1, z_2, z_3, \ldots z_r\}$, X das Eingabealphabet mit $X = \{x_1, x_2, \ldots x_m\}$, $z_0 \in Z$ der *Anfangszustand*, $Z' \subseteq Z$ die Menge der *Endzustände* und δ die Überführungsfunktion mit $\delta: Z \times X \rightarrow Z$.

Bild 1a zeigt das Beispiel eines Zustandsgraphen. Die übliche Schreibweise (vgl. Beispiel in Bild 1a) täuscht eine unvollständige Definition vor, sie ist in Wirklichkeit jedoch eine abgekürzte Schreibweise. Diese geht davon aus, daß

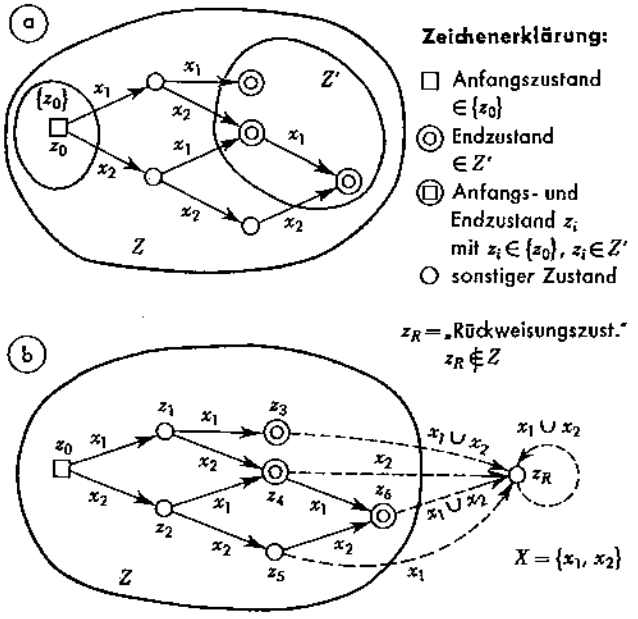


Bild 1. Zur Schreibweise bei der Spezifizierung endlicher Akzeptoren.
a) Beispiel des Zustandsgraphen eines endlichen Akzeptors: übliche, gekürzte Schreibweise.
b) Zustandsgraph des endlichen Akzeptors nach a) in ungekürzter Schreibweise (Interpretation).

Übergänge in einen nicht eingetragenen *Rückweisungszustand* z_R in der Automatentafel oder im Graphen weggelassen werden. Bild 1b zeigt zum Vergleich die ungekürzte Spezifikation des Akzeptors nach Bild 1a. Bei der konventionellen Automatentafel in Matrixform kommt die Abkürzung dadurch zustande, daß die Eintragung z_R durch eine Leerstelle ersetzt wird und die Zeile z_R fehlt. Wichtig ist die Klasseneinteilung der Zustände in *Endzustände* und in sonstige Zustände, die später auch *Zustände ohne Ausgabesymbol* genannt werden. Weiter sind folgende Definitionen wichtig.

Definition 2. Eine Eingabefolge s gilt dann als *akzeptiert* durch einen Akzeptor $\mathfrak{A}$, wenn sie $\mathfrak{A}$ vom Anfangszustand z_0 in einen Endzustand überführt. Andernfalls gilt s als *zurückgewiesen* durch $\mathfrak{A}$.

Definition 3. Eine Menge R von Folgen gilt als akzeptiert durch $\mathfrak{A}$, wenn $\mathfrak{A}$ jede Folge $s \in R$ akzeptiert.

Die Übergangsfunktion δ eines Akzeptors gibt zu jedem geordneten Paar (z_i, x_j) mit z_i als vorherigem Zustand und $x_j \in X$ den Folgezustand z_f an mit $z_f = \delta(z_i, x_j)$, sofern für (z_i, x_j) ein z_f existiert.

Zu jeder Übergangsfunktion δ eines endlichen Automaten läßt sich eine inverse Übergangsfunktion δ^{-1} ermitteln, die zu jedem geordneten Paar (z_i, x_j) den vorherigen Zustand z_v *(Vorzustand)* angibt mit $z_v = \delta^{-1}(z_i, x_j)$, sofern z_v für dieses Paar definiert ist. Zwischen der δ-Funktion und der δ^{-1}-Funktion eines endlichen Akzeptors bestehen Dualitätsbeziehungen, die sich zur vereinfachten Formulierung der Regeln für Synthesealgorithmen ausnützen lassen.

Für Zwecke der formalen Beschreibung von Strukturen in gerichteten Graphen sei folgendes definiert.

Definition 4. Wenn $z_f = \delta(z_i, x_j)$, sei das geordnete Paar (x_j, z_f) als *eine Übergangsregel des Knotens (oder Zustandes)* z_i bezeichnet (Bild 2f). Wenn $z_v = \delta^{-1}(z_i, x_j)$, sei das geordnete Paar (x_j, z_v) als *eine inverse Übergangsregel des Knotens (oder Zustandes)* z_i bezeichnet (Bild 2g).

Eine Zusammenfassung von Übergangsregeln eines Knotens z_i sei je nach Zusammensetzung als *ankommendes Bündel* (Bild 2c), *abgehendes Bündel* (Bild 2d), oder *gemischtes Bündel* (Bild 2e) des Knotens z_i bezeichnet. Die Einbettung eines Knotens in einen gerichteten Graphen läßt sich vollständig spezifizieren durch die vollständige Menge seiner Übergangsregeln. Die in folgenden Abschnitten behandelte Listenstruktur zur Beschreibung gerichteter Graphen ist im Prinzip eine Liste aller Übergangsregeln des Graphen. Die Gleichheit von Bündeln zweier Knoten sei im Graphen durch die Symbolik nach Bild 2h und Bild 2j gekennzeichnet.

Kleene hat eine Mengen-Algebra angegeben, mit deren Hilfe alle diejenigen Mengen R von Folgen beschrieben werden können, zu denen ein Akzeptor existiert, der R akzeptiert. Die Ausdrücke dieser Algebra heißen *reguläre Ausdrücke* und eignen sich zur Formulierung von Syntheseproblemen. Es sei kurz folgende Terminologie vereinbart. (Eine detailliertere Einführung in die *Kleene*sche Algebra ist u. a. in [1] zu finden.)

Es seien die Variablen $\alpha, \beta, \gamma, \ldots$ gegeben und die neutralen Elemente Φ und Λ mit folgenden Interpretationen:

Φ ist die Leermenge mit $\Phi = \{\ \}$, während Λ die Leerfolge ist mit $\Lambda = \langle\ \rangle$. Die wichtigsten Typen von regulären Ausdrücken haben folgende Form.

Disjunktion:	$\alpha \cup \beta$	
(Ketten-)Produkt:	$\alpha \cdot \beta$	(abgekürzt: $\alpha\beta$)
(Ketten-)Potenz:	α^n	$(\alpha^n \cdot \alpha = \alpha^{n+1},\ \alpha = \alpha^1,\ \alpha^0 = \Lambda)$
Iteration:	$\alpha*$	$(\alpha* = \Lambda \cup \alpha^1 \cup \alpha^2 \cup \alpha^3 \cup \ldots.)$

Die durch α und β bezeichneten Variablen können hierbei bedeuten: Symbole des Eingabealphabetes X, Folgen über X oder reguläre Ausdrücke. (Diese Algebra ist rekursiv.) Den drei obigen Ausdrücken entsprechen die Strukturen nach Bild 3b.

Ein regulärer Ausdruck ϱ heißt *atomar*, wenn $\varrho \in X$. Ein Zweig eines Graphen heißt atomar, wenn sein Index atomar ist. Ein Graph heißt atomar, wenn er nur atomare Zweige besitzt. Ein Zweig heißt *Leerzweig*, wenn er den Index Λ hat.

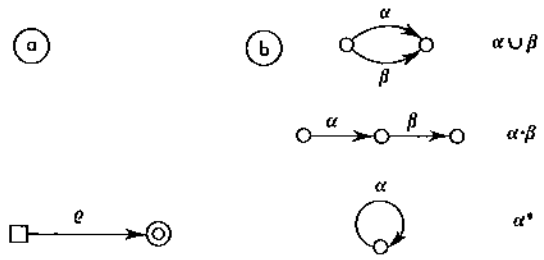


Bild 3. Graphen regulärer Ausdrücke.
a) Residuum des endlichen Akzeptors, der ϱ akzeptiert;
b) Graphen der wichtigsten regulären Ausdrücke.

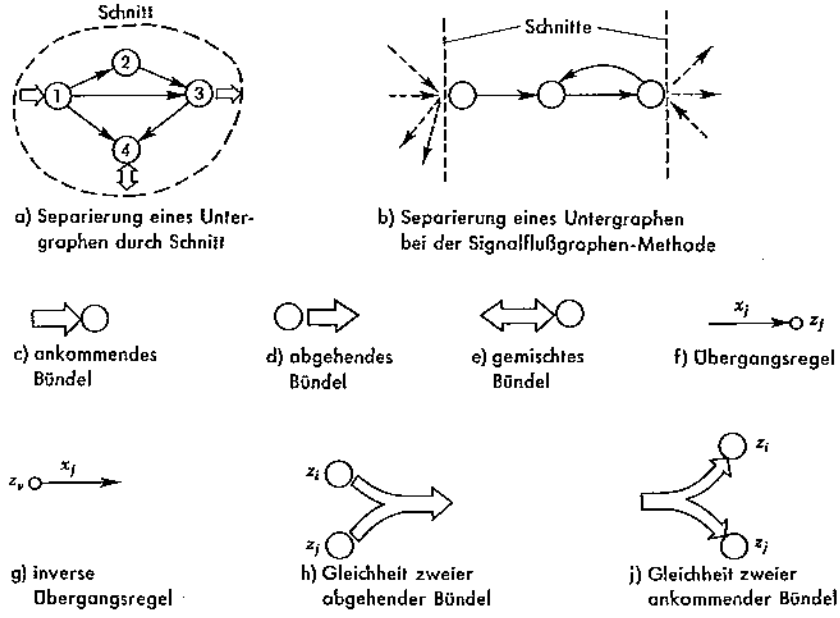


Bild 2. Charakterisierung von Schnittstellen in gerichteten Graphen.

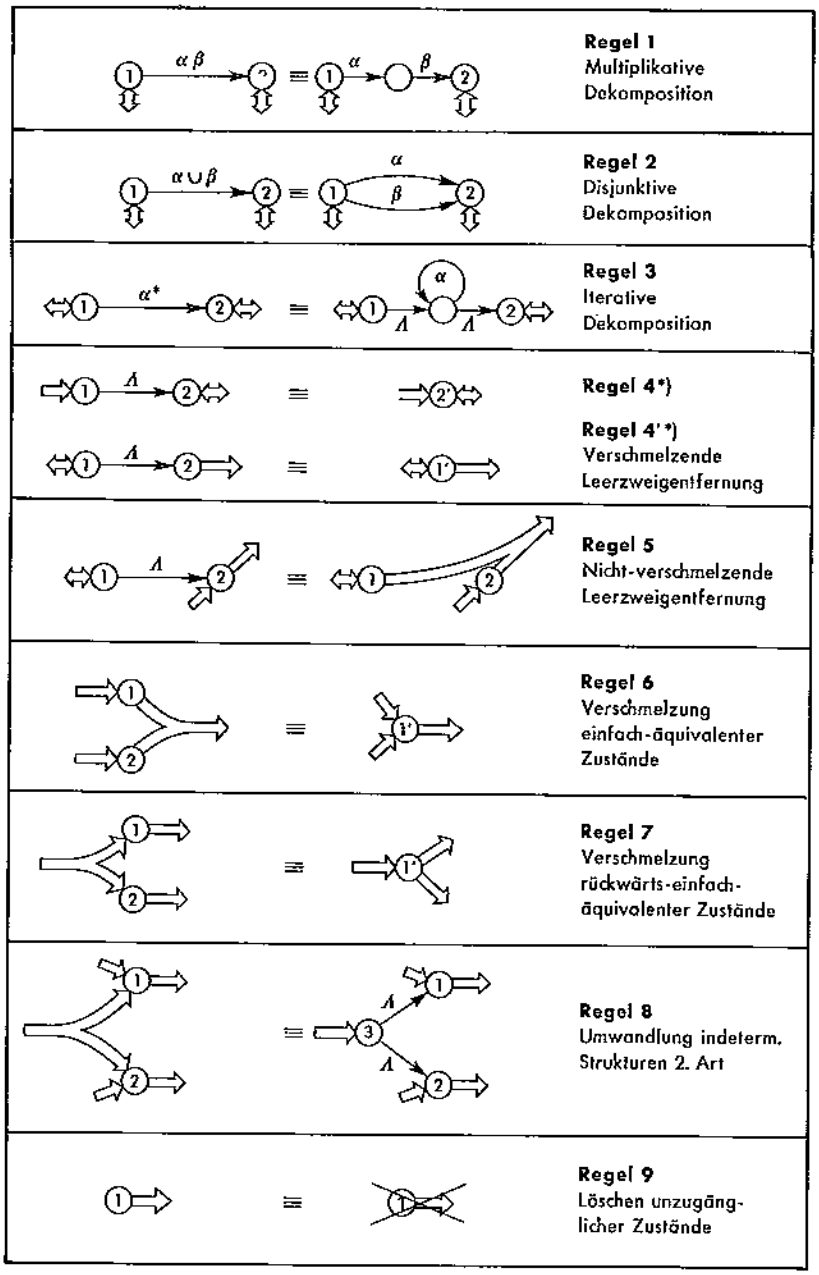


Bild 4. Elementare Transformationsregeln zur Akzeptor-Synthese.
*) Diese beiden Regeln können durch Regel 5 und 9 ersetzt werden, falls Vereinfachung des Repertoirs erwünscht.

3. Synthese endlicher Akzeptoren und ähnlicher Strukturen

Im Folgenden wird die Anwendung eines in [1] beschriebenen Syntheseverfahrens auf die genannte Listenstruktur behandelt. Das Verfahren beruht auf einer Anwendung modifizierter Signalflußgraphen-Techniken auf echte und unechte Zustandsgraphen. Die Problemformulierung besteht dabei in einem *Residuum* genannten unechten Zustandsgraphen gemäß Bild 3a, wobei ϱ für den als Problem gegebenen regulären Ausdruck ϱ steht. Dieser Graph heißt deswegen *unecht*, da noch nicht für jeden Zustand ein Knoten vorhanden ist. Die Synthese erfolgt durch sukzessive Substitution von Unterstrukturen des Graphen nach Regeln in Bild 4 unter stufenweiser Vermehrung der Knoten solange, bis ein echter Zustandsgraph vorliegt. Bild 5 zeigt ein Synthesebeispiel für dieses Verfahren.

Bei Graphen für Unterstrukturen sei hierbei für die Charakterisierung der Verbindungen zwischen lokalen Schnittstellen-Knoten und globalen Knoten das Bezeichnungsschema nach dem Beispiel in Bild 2a benützt anstelle der bei Signalflußgraphen-Technik üblichen Schreibweise (Bild 2b).

Der Satz von Substitutionsregeln nach Bild 4 gestattet in jedem Fall die Erzielung der Minimallösung des Syntheseproblemes, da Regel 6 für sich allein eine vollständige Minimierung nach der Methode des sukzessiven Verschmelzens einfach-äquivalenter Zustände ermöglicht [2]. Der linke Graph von Regel 6 zeigt also zwei solche *einfach-äquivalenten* Zustände. In Analogie hierzu seien die beiden Knoten des linken Graphen in Regel 7 als *rückwärts einfach-äquivalent* bezeichnet.

Bei Anwendung des Substitutionsverfahrens können zwischenzeitlich indeterministische Strukturen auftreten, die im Lauf des Syntheseprozesses wieder *determiniert* werden müssen. Indeterministische Strukturen können in 2 Kategorien eingeteilt werden, weshalb die Bezeichnungen *indeterministische Struktur 1. Art* (Beispiel in Bild 6a) und *indeterministische Struktur 2. Art* (Beispiel in Bild 6b) benützt seien. Zwei rückwärts einfach-äquivalente Zustände bilden dem-

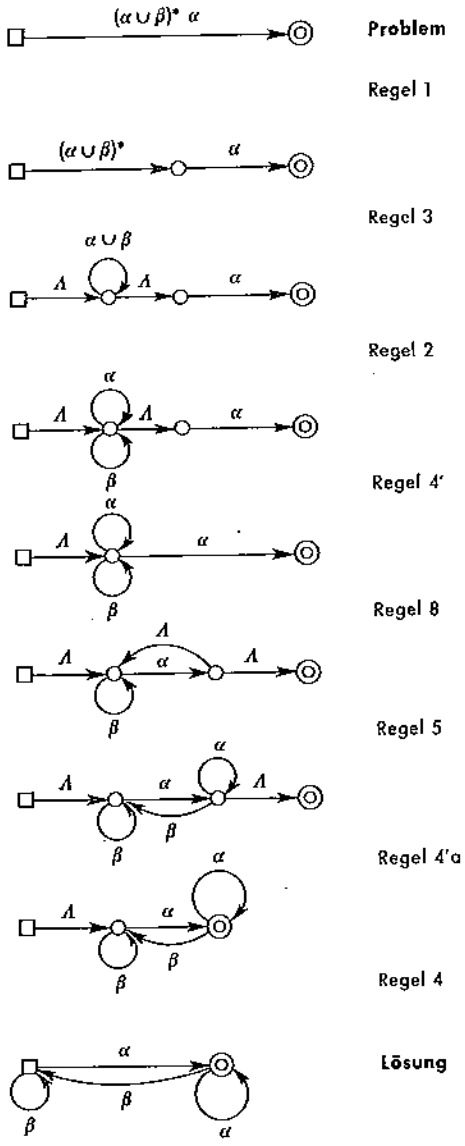


Bild 5. Synthesebeispiel zur Anwendung der Substitutionsmethode.

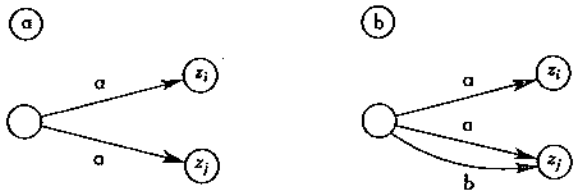


Bild 6. Indeterministische Struktur 1. und 2. Art (Beispiele).
a) Indeterministische Struktur erster Art;
b) Indeterministische Struktur zweiter Art.

nach eine indeterministische Struktur 1. Art, deren Determinierung nach Regel 7 erfolgt. Die Determinierung indeterministischer Strukturen 2. Art erfolgt nach Regel 8 [1]. Regeln 1—3 dienen der Dekomposition mit dem Ziel der Atomisierung, Regeln 4 und 5 der Entfernung von Leerzweigen, die bei Anwendung der Regeln 3 oder 8 anfallen. Regel 4 und 4' können durch Regel 5 ersetzt werden, wobei evtl. verbleibende unzugängliche Knoten durch Regel 9 entfernt werden.

Die wesentlichen Prozesse, auf denen das genannte Syntheseverfahren beruht, sind also die folgenden.

1. Atomisierung: Auflösung regulärer Ausdrücke über eine Knotenvermehrung;
2. Determinierung: Beseitigung von Indeterminiertheiten;
3. Minimierung: Reduktion nicht minimaler Strukturen.

Der algorithmische Ablauf eines Syntheseprozesses nach der Substitutionsmethode hat — ebenso, wie die Minimierung nach der Methode des sukzessiven Verschmelzens — etwa folgenden Verlauf: Der vorliegende Graph (bzw. die vorliegende Liste) wird zunächst nach Unterstrukturen abgesucht, auf welche Substitutionsregeln anwendbar sind. Jedesmal dann, wenn eine entsprechende Struktur gefunden wurde, wird sofort die zugehörige Substitution durchgeführt. Daran anschließend wird der Suchprozeß fortgesetzt. Der Prozeß ist dann beendet, wenn keine substituierbare Struktur mehr gefunden werden kann.

Die Suche nach substituierbaren Unterstrukturen erfolgt in der Weise, daß nacheinander alle Unterstrukturen zunächst einem einzigen Test unterzogen werden. In einer darauffolgenden weiteren Schleife wird ein anderer Test durchgeführt, usw. In [1] wird gezeigt, daß für die Minimierung nach der Methode des sukzessiven Verschmelzens und für die Akzeptor-Synthese die beiden folgenden Typen von Prüfungen ausreichen, die sich auf Unterstrukturen aus höchstens 2 Knoten mit ihren Schnittbündeln erstrecken.

A. Einstellige Prüfungen (betr. nur einen Knoten z_i)

1. Knotenprüfung
   a) auf Endzustandseigenschaft ($\lambda \neq 0$?)
   b) auf Unzugänglichkeit (existiert inv. Überg.-Regel?)
2. Prüfung der einzelnen Zweige des Knotens
   a) einzeln auf Atomarität ($x \in X$?)
   b) einzeln auf Leerzweig-Eigenschaft ($x = \Lambda$?)
   c) paarweise auf *Indeterminiertheit* (Gleichheit)

B. Zweistellige Prüfungen (betr. 2 Knoten z_i und z_j)
Prüfung von Bündelpaaren auf Äquivalenz oder Rückwärts-Äquivalenz (Gleichheit der Bündel)

Die Substitution einer der jeweils durch eine dieser Prüfungen ermittelten Unterstrukturen kann durch Transformation mittels Anwendung der nachstehenden Elementaroperationen erfolgen.

C. Einstellige Operationen (ein einziger Bezugsknoten z_i)

1. Veränderung der Knoten-Anzahl (Einführen oder Löschen eines Knotens)
2. Bündel-Veränderungen (Einführen oder Löschen eines Zweiges)

D. Zweistellige Operationen (zwei Bezugsknoten z_i und z_j)

1. Operationen am Ausgabesymbol (Doppelung einer Endzustandseigenschaft)
2. Operationen an Bündeln
   a) Doppelung eines Zweig-Bündels von z_i nach z_j
   b) Verschiebung eines Zweig-Bündels von z_i nach z_j

E. Zusammengesetzte Operationen

1. algebraische Operationen (Dekompositionen)
2. nicht-algebraische Operationen (Verschmelzung oder Spaltung von Knoten)

In den folgenden Abschnitten werden Listenstrukturen behandelt, die mittels der genannten Tests und Operationen eine rechenzeiteffektive Bearbeitung von Minimisierungs- und Syntheseproblemen an Automaten des behandelten Typs ermöglichen.

4. Suchlistendarstellung von Automatentafeln bei Synthese- und Minimierungsprozessen

Ein Nachteil der konventionellen Automatentafel für die Darstellung im Computer ist ihre Unbrauchbarkeit zur Darstellung indeterministischer Strukturen, da infolge Matrixform der Tabelle nur jeweils ein δ-Wert je Paar (z_v, x_i) angegeben werden kann. Es muß also bei einer Anwendung des oben beschriebenen Substitutionsverfahrens auf Listenstrukturen bereits für den rein atomaren Fall eine andere Listendarstellung der δ-Funktion gefunden werden. Hierzu eignet sich die *Suchlisten-Darstellung*, bei welcher im Gegensatz zur konventionellen Automatentafel die einzelnen Zeilen unterschiedliche Längen haben können. Die Länge der Zeile für einen Zustand entspricht dabei genau der Anzahl der vom entsprechenden Knoten abgehenden Zweige bei Darstellung durch einen Zustandsgraphen. Im folgenden Abschnitt wird die Suchlistendarstellung für Zustandsgraphen beschrieben.

Bei Abspeicherung konventioneller Automatentafeln in Matrixform ist jeder definierte Übergang durch Angabe des Folgezustandes z_f als nicht-negative ganze Zahl angegeben, wobei jeder nicht explizit definierte Übergang durch ein besonderes Leerzeichen in der Tabelle gekennzeichnet ist. Bei der Suchlistendarstellung wird nur jeder explizit definierte Übergang durch ein geordnetes Paar in der Form (x_i, z_f) spezifiziert, d. h. durch eine *Übergangsregel*. Die Gesamtheit aller Übergangsregeln des gleichen Vorzustandes z_h wird als *Zeilenblock* Nr. h bezeichnet (Bild 7). Jedem Zeilenblock bei Suchlistendarstellung entspricht eine Zeile bei Vorliegen einer konventionellen Automatentafel. Ein Zeilenblock h einer Automatentafel in Suchlistendarstellung ist die listenmäßige Beschreibung des abgehenden Gesamtbündels vom Knoten h eines entsprechenden Zustandsgraphen.

Eine Automatentafel in Suchlistendarstellung besteht aus drei verschiedenen Listen, die *Suchliste, Hilfsliste* und *Ausgabeliste* genannt werden. Die Gesamtheit aller Zeilenblöcke bildet die *Suchliste*. Wenn ein Vorzustand z_h gegeben ist und somit der diesem zugeordnete Zeilenblock bekannt ist, wird der Folgezustand auf ein gegebenes Eingabesymbol x_i nicht durch Indizierung, sondern durch einen Suchprozeß ermittelt. Hierbei wird der aktuelle Zeilenblock abgesucht nach einer Übergangsregel, die das Eingabesymbol x_i enthält (Bild 8). Wird eine solche Übergangsregel (x_i, z_f) gefunden, so ist damit ein Folgezustand z_f ermittelt. Wird bei Fortsetzung der Suche mindestens eine weitere solche Übergangsregel gefunden, so liegt eine indeterministische Struktur vor (vgl. Bild 8).

Die *Hilfsliste* hat die Aufgabe, zu jedem Vorzustand z_h den zugehörigen Zeilenblock Nr. h zugänglich zu machen. Dies erfolgt dadurch, daß jedem Zeilenblock ein Wertepaar

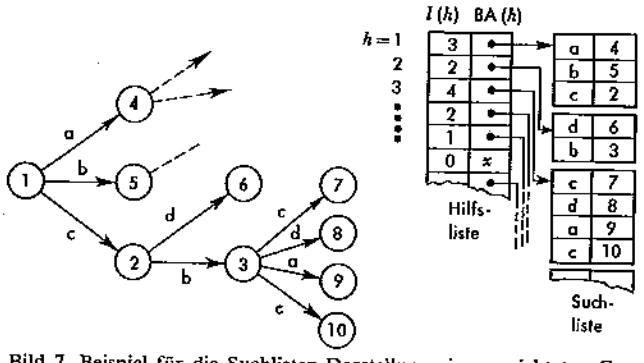


Bild 7. Beispiel für die Suchlisten-Darstellung eines gerichteten Graphen.

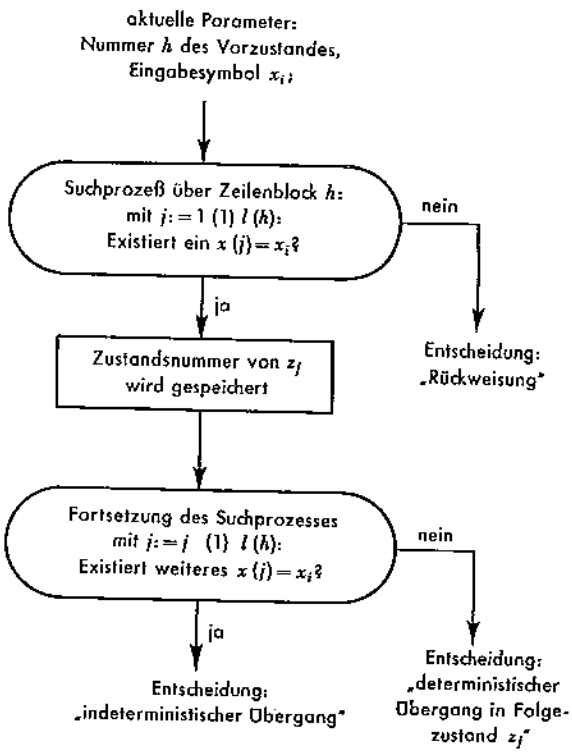


Bild 8. Simulation des Klassifikationsbetriebs eines Automaten in einfacher Suchlistendarstellung: Beispiel für Ermittlung eines Übergangs.

$(l(h), BA(h))$ der Hilfsliste zugeordnet ist, wobei die *Basis-Adresse* $BA(h)$ angibt, wo Zeilenblock Nr. h beginnt und die *Zeilenlänge* $l(h)$ angibt, aus wievielen Übergangsregeln der Zeilenblock h besteht. Eine beliebige Position h der Hilfsliste kann unmittelbar durch Indizierung aufgefunden werden. Bei der Suchlistendarstellung von Zustandsgraphen entspricht jedem Knoten ein Doppelwort der Hilfsliste, jedem Zweig eine Übergangsregel und jedem Zeilenblock ein Gesamtbündel.

Die Ausgabefunktion wird bei der Suchlistendarstellung in der sogenannten *Ausgabeliste* spezifiziert. Für die Darstellung solcher Automaten, die sich nach Art eines Moore-Automaten beschreiben lassen, wird die Hilfsliste um eine Spalte (Ausgabespalte) erweitert, so daß jedem Zustand z_h ein Tripel $(l(h), BA(h), \lambda(z_h))$ zugeordnet ist. Bei Mealy-Automaten wird statt dessen die Suchliste um eine Spalte erweitert, wobei jede Übergangsregel nicht als Paar, sondern als Tripel in der Form $(x_i, \delta(z_h, x_i), \lambda(z_h, x_i))$ erscheint.

Bei deterministischen vollständig definierten Automaten ist die Zeilenlänge m in der Automatentafel gleich der Länge m des Eingabe-Alphabetes. Ist jedoch z. B. einer der Übergänge zweideutig, so wird die Zeilenlänge $m + 1$ benötigt, was bei Suchlistendarstellung realisiert werden kann, bei Anwendung der konventionellen Darstellung jedoch nicht. Sind in einer Zeile beispielsweise 2 Übergänge nicht spezifiziert, so hat die Suchlistendarstellung die Zeilenlänge $m - 2$, was dem Fortfall von Zweigen bei graphischer Darstellung entspricht. Diese Analogie gegenüber dem Zustandsgraphen gilt auch bei mehrdeutigen Übergängen, für die mehrere Zweige je Eingabesymbol benützt werden.

Die Suchlistendarstellung hat vor allem bei Akzeptoren und ähnlich strukturierten Automaten, wie sequentielle Klassifikatoren und Transduktoren mit Anfangszustand, gegenüber der konventionellen Automatentafel wichtige Vorteile:

a) Meist erhebliche Speicherplatz-Ersparnis durch Weg-

lassen der Rückweisungs-Spezifikation (gemäß Akzeptor-Definition);
b) Darstellbarkeit mehrdeutiger Übergänge;
c) Sehr hohe Ersparnis an Testschritten bei globalen Prüfungen durch Analyse und Vergleich von Zuständen und ihrer Umgebung;
d) Abkürzung der Einzelschritte zu c;
e) Sehr starke Rechenzeit-Ersparnis bei Reduktionsprozessen infolge c und d.

Die Zeitersparnis ist darauf zurückzuführen, daß die Menge der Zustände aufgrund der unterschiedlichen Zeilenlängen in Klassen unterteilt ist. Durch die Erfassung der Zeilenlängen in der Hilfsliste liegt also diese Klassenunterteilung der Knoten aufgrund der Größe ihrer abgehenden Bündel unmittelbar ablesbar vor. Da nur Knoten mit gleichgroßen abgehenden Bündeln einfach-äquivalent sein können (Regel 6 in Bild 4), kann der paarweise Vergleich der Zeilen entsprechend der aktuellen Zeilenlänge auf die jeweilige Untermenge von Zuständen beschränkt werden. Zur Veranschaulichung diene folgendes Beispiel eines Rechenzeitvergleiches. Ein Akzeptor habe r Zustände und ein Eingabe-Alphabet der Länge m, wobei Gleichverteilung der Zeilenlängen vorausgesetzt sei. Die Kombinatorik lehrt, daß bei paarweisem Vergleich von r Elementen $r(r-1)/2$ einzelne Vergleichsschritte notwendig sind. Für große r ist die Anzahl von Schritten näherungsweise $r^2/2$. Bei nach Längenklassen getrenntem Vergleich ist die Anzahl der Schritte $m \cdot r/m(r/m-1)/2$ oder für große r näherungsweise $r^2/2\,m$. Dies bedeutet eine Einsparung an Vergleichsschritten um den Faktor $1/m$, wobei noch nicht berücksichtigt wurde, daß der einzelne Vergleichsschritt dadurch eine zusätzliche Einsparung bringt, daß die durchschnittliche Länge der zu vergleichenden Zeilen geringer ist und beispielsweise bei Gleichverteilung nur die Hälfte beträgt. (Durch Einhaltung eines Ordnungsschemas für Zeilenblöcke ist ein Zeilenvergleich ohne Permutation der Übergangsregeln möglich.) Zur vollständigen Reduktion von Moore-Klassifikatoren mit $r \approx 8000$ und $m = 9$ wurden auf einem Digitalrechner CD 3300 Rechenzeiten zwischen 5 und 10 Minuten benötigt [3].

5. Doppelte Suchliste zur Darstellung gerichteter Graphen bei Synthese- und Minimierungsprozessen

Es ist für gerichtete Graphen charakteristisch, daß sowohl die Vorwärts- als auch die Rückwärts-Umgebung eines jeden Knotens durch visuelle Inspektion unmittelbar erfaßt werden kann. Eine solche unmittelbare Erfassung ist bei matrixförmigen Automatentafeln nur für die Vorwärts-Umgebung von Zuständen möglich, während die Rückwärtsumgebung nur durch allumfassende Suchprozesse zugänglich gemacht werden kann. Dieser Nachteil der konventionellen Automatentafel gegenüber dem Graphen ist darauf zurückzuführen, daß im ersteren Fall nur die Übergangsfunktion δ, bei letzterem hingegen gleichzeitig mit δ auch die inverse Übergangsfunktion δ^{-1} explizit beschrieben ist. Zur Nutzung sämtlicher Vorzüge des gerichteten Graphen auch für Computer-Anwendung, was bei gewissen Prozessen Rechenzeitersparnisse um mehrere Größenordnungen ermöglichen kann, eignet sich die *doppelte Suchlistendarstellung*, die gleichzeitig die Funktionen δ und δ^{-1} in Suchlistendarstellung erfaßt, wobei für diese beiden Funktionen getrennte, aufeinander abgestimmte Listensysteme mit eigenen Hilfslisten vorhanden sind. Durch eine doppelte Automatentafel in konventioneller Form wäre selbst im deterministischen Fall die Spezifikation von Automaten unmöglich, da die Inversion δ^{-1} einer eindeutigen δ-Funktion nicht notwendigerweise eindeutig ist.

Ein Vorteil der Doppeldarstellung besteht darin, daß die paarweisen Vergleiche von Knoten bezüglich ihrer Rückwärtsumgebung, die zur Feststellung indeterministischer Strukturen erster Art erforderlich sind (Regel 7), mit dem gleichen Unterprogramm durchgeführt werden können, welches durch Vergleiche der Vorwärtsumgebung von Knoten einfach-äquivalente Zustände ermittelt (Regel 6). Es ist also eine Vorwärts-Minimierung und eine Rückwärts-Minimierung mit Hilfe des gleichen Unterprogrammes möglich. Ein weiterer Vorteil der Doppeldarstellung, der sich bei einer computergerechten Simulationsstrategie für Substitutionsverfahren auswirkt, wird später beschrieben.

Ein weiterer Vorteil der doppelten Suchliste ist eine erhebliche Rechenzeit-Ersparnis bei Verschmelzungsprozessen. Bei einer Einfach-Darstellung von δ werden zwei Zustände z_i und z_j zu z_i verschmolzen, indem die Zeile j für z_j eliminiert wird und unter Absuchen der gesamten δ-Liste alle Folgezustände z_j durch z_i ersetzt werden. Bei doppelter Darstellung kann auf diesen Suchprozeß verzichtet werden, da Vorzustände von z_j, deren Zeilen zu ändern sind, durch die inverse Funktion δ^{-1} direkt definiert sind. Zur Veranschaulichung sei ein Rechenzeitvergleich zwischen einfacher konventioneller und doppelter Suchlistendarstellung für diesen Suchprozeß angestellt. Bei einer Alphabetlänge m und r Zuständen erfordert die einfache konventionelle Automatentafel $r \cdot m$ Suchschritte. Bei einer mittleren Zeilenlänge von $m/2$ für beide Suchlisten erfordert die doppelte Suchlistendarstellung $m/2 \cdot m/2 = m^2/4$ Suchschritte ($m/2$ Vorzustände mit $m/2$ Zeilenlänge als Durchschnittswerte). Für die Zahlenwerte $r = 1000$ und $m = 10$ ergibt sich beispielsweise eine Schritt-Anzahl von 10000 gegenüber 25.

6. Suchlisten-Darstellung nicht-atomarer Graphen und ihre Anwendung bei Syntheseprozessen

Mit den beschriebenen Methoden der Suchlistendarstellung können nur echte Zustandsgraphen dargestellt werden, d. h. solche Graphen, die ausschließlich atomare Zweige enthalten. Zur Anwendung der Substitutionsmethode zur Synthese muß die Listenstruktur jedoch zur Aufnahme regulärer Ausdrücke befähigt werden. Zu diesem Zweck wird das erste Wort einer Übergangsregel durch eine Verweis-Adresse AD und einen „Indirekt-Merker" ersetzt, wenn es sich anstelle eines Symbols aus dem Alphabet X (atomarer Fall) um einen Ausdruck handelt.

Der „Indirekt"-Merker soll Verwechslungen zwischen Symbolen und Adressen verhindern. Die Adresse AD ist ein Index, der das Auffinden des regulären Ausdruckes in einer *algebraischen Liste* ermöglicht, wie in Bild 9 gezeigt wird. Die algebraische Liste hat die gleiche Struktur, die auch dem Aufbau der Suchliste zugrunde liegt. Ein Doppelwort einer *Sekundär-Hilfsliste* gibt Basisadresse und Zeilenlänge eines Blockes der algebraischen Liste an. Die Zeilenlänge ist durch die Länge des in diesem Block gespeicherten regulären Ausdrucks bestimmt.

Für die Struktur eines solchen Blocks wird eine *polnische Darstellung* des regulären Ausdruckes gewählt. (Das Prinzip der *polnischen Darstellung* algebraischer Ausdrücke ist die Wiedergabe ohne Verwendung von Klammern.) Die Priorität der einzelnen Operatoren innerhalb des Ausdruckes ergibt sich dabei eindeutig aus der Folge. Das bekannte Schema nach *Lukasiewicz* (beschrieben beispielsweise in [4]) muß für die *Kleene*sche Algebra zwecks Einbeziehung der

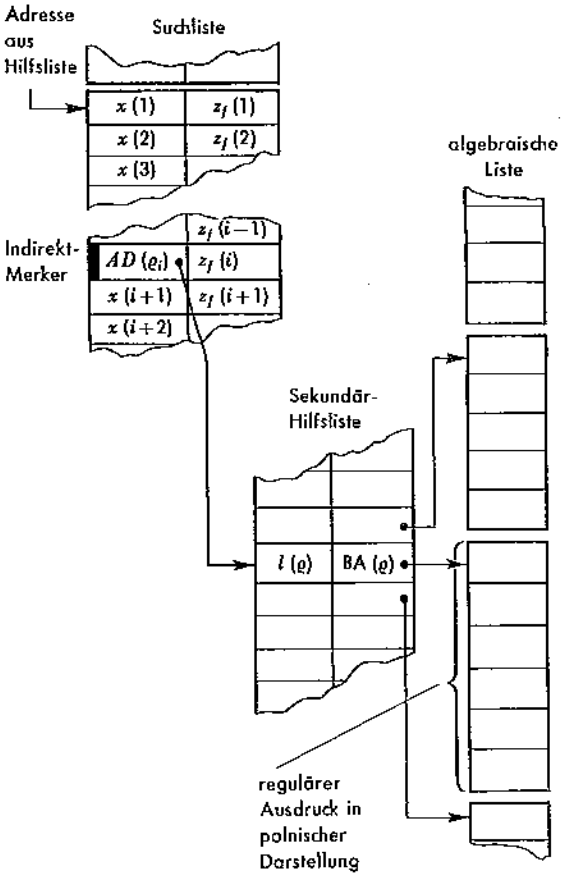


Bild 9. Darstellung nicht-atomarer Zweige durch rekursive Suchlistendarstellung.

einstelligen Iterations-Operation modifiziert werden. Folgendes Beispiel zeigt die polnische Darstellung eines regulären Ausdruckes.

Beispiel: Gegeben ist $\varrho = \{a \cup b(cd)*\}*$.

Die polnische Darstellung des regulären Ausdruckes ϱ lautet:

$$* \cup a \cdot b * \cdot cd.$$

Die Reihenfolge der Operator-Symbole in einem regulären Ausdruck in polnischer Darstellung ist ein geeigneter Ansatz zur Formulierung eines Dekompositions-Algorithmus für uneigentliche Zustandsgraphen. Ein einfaches Beispiel in Bild 10 veranschaulicht die Anwendung der polnischen Darstellung in Suchlistenform zur Dekomposition eines regulären Ausdruckes.

Bild 10a zeigt als Beispiel einer Dekompositions-Aufgabe einen Ausschnitt aus einem uneigentlichen Zustandsdiagramm in Suchlistendarstellung. Der zu verarbeitende reguläre Ausdruck in der Form $\varrho = 1 \cdot 2 \cup 3$ hat die polnische Darstellung $\cup 3 \cdot 2\,1$, welche in inverser Folge ihrer Symbole in der algebraischen Liste abgespeichert ist. Die den vorliegenden Übergang $\delta(z_7, \varrho) = z_8$ beschreibende Übergangsregel in der Suchliste enthält anstelle eines Eingabesymbols (wie im atomaren Fall) eine Adresse, was durch einen Indirekt-Merker angezeigt ist. Die Simulation der Dekomposition von ϱ durch Transformation der in Teil a) des Bildes gezeigten Suchlistendarstellung erfolgt in den folgenden Schritten:

1. Schritt: Transformation von Darstellung a) in Darstellung b). Das letzte Symbol im vorliegenden Block der algebraischen Liste bestimmt, welche Transformation zuerst vorgenommen wird: aus $\cup$ folgt die disjunktive Dekomposition. Der Block der algebraischen Liste wird gespalten, indem ein neues Wertepaar in die Sekundär-Hilfsliste eingeführt und das alte Wertepaar in seiner Längenangabe $l(\varrho)$ korrigiert wird von 5 auf 3 (Bild 10b). Das Symbol für den Disjunktions-Operator wird gelöscht, was im Bild durch Schraffur angedeutet ist. Für den durch Abspaltung eines Unterausdruckes gebildeten zusätzlichen Block der algebraischen Liste wird eine neue Übergangsregel definiert durch Verlängerung des Zeilenblockes in der Suchliste (Bild b). Der Graph hat nunmehr zwei Pfade von Knoten 7 nach Knoten 8.

2. Schritt: Transformation von Darstellung b) in Darstellung c). Der durch Blockspaltung in der algebraischen Liste durch Schritt 1 entstandene neue Block hat die Länge 1, was durch das zugeordnete neue Wertepaar der Sekundär-Hilfsliste angezeigt wird. Dies bedeutet, daß der reguläre Ausdruck atomar ist, also ein Einzelsymbol des Eingabealphabetes ist (Symbol: 3). Der als *Atomisierung* bezeichnete Schritt verlagert den Ort der Darstellung dieses Symbols von der algebraischen Liste in die Suchliste, indem in der entsprechenden (5.) Position des zugeordneten Zeilenblocks Adresse und Indirekt-Merker gemeinsam durch das Eingabesymbol ersetzt werden. Der dadurch überflüssig gewordene Block in der algebraischen Liste und das diesem zugeordnete Parameter-Paar in der Sekundär-Hilfsliste werden gelöscht, was durch Schraffur angedeutet wurde (Bild 10c).

3. Schritt: Transformation von Darstellung c) in Darstellung d). Die Operation in diesem Schritt wird durch das letzte

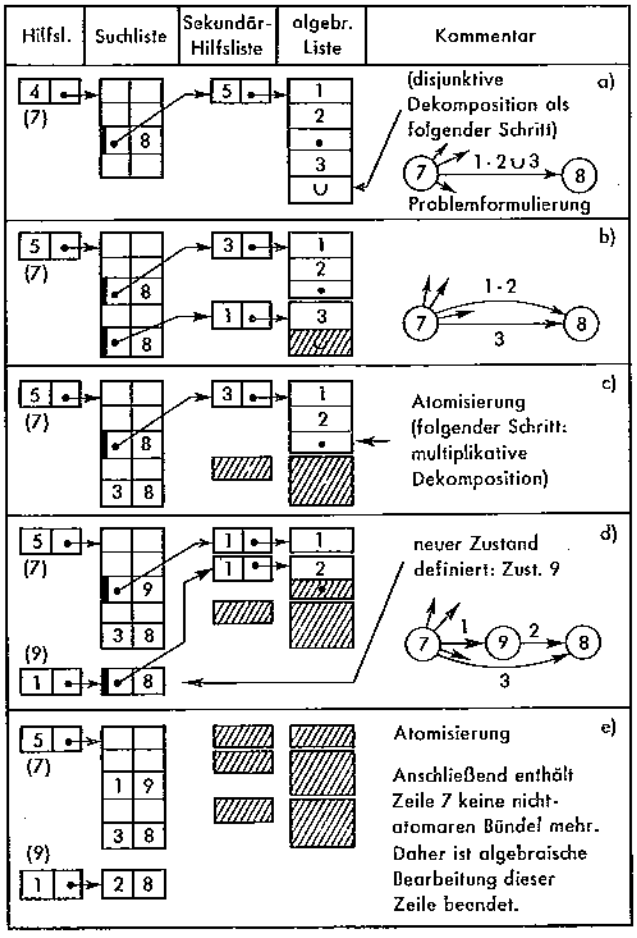


Bild 10. Beispiel für die Anwendung der Substitutionsmethode auf eine rekursive Suchliste.

Symbol im vorliegenden Block der algebraischen Liste bestimmt. Aus diesem Symbol „." folgt die Multiplikative Dekomposition. Hierzu muß ein neuer Knoten eingeführt werden (Knoten Nr. 9 nach Bild 10d). Dies geschieht durch Anlegen einer neuen Zeile über die Einführung eines neuen Wertepaares in eine freie Position der Hilfsliste (Pos. 9 in diesem Beispiel). Der reguläre Ausdruck in der algebraischen Liste wird gespalten, wie in Schritt 1, jedoch mit folgenden Unterschieden: 1). Die neu zu definierende Übergangsregel für den zusätzlichen Pfad im Graphen wird in den Zeilenblock des *neu definierten* Zustands z_9 eingeführt. 2). Die Folgezustands-Eintragung (Pos. 3 in Zeilenblock von Zustand 7) für den ersten vorliegenden Block der algebraischen Liste muß von 8 auf 9 geändert werden. Bild 10d zeigt den Graphen der durch Schritt 3 transformierten Listenstruktur.

4. Schritt: Transformation von Darstellung d) in Darstellung e). Durch den 3. Schritt sind in der algebraischen Liste 2 atomare Ausdrücke entstanden. Diese werden durch 2 Atomisierungs-Schritte wie Schritt 2 in die Suchliste verlagert. Bild 10c zeigt die als Endergebnis auftretende Listenstruktur für den Graphen nach Bild 10d.

Die oben eingeführte *einfach-rekursive Suchlisten-Darstellung* eines uneigentlichen Zustandsgraphen beruht auf zwei Ebenen der Darstellung. In der ersten Ebene, die durch direkte Adressierung in der Suchliste zugänglich ist, werden atomare Zweige des Graphen erfaßt. In der zweiten Ebene, die durch indirekte Adressierung (algebraische Liste) zugänglich ist, werden solche Zweige des Graphen wiedergegebenen, denen Ausdrücke mit atomaren Komponenten zugeordnet sind. Dies bedeutet, daß in der zweiten Ebene nur Symbole des Eingabe-Alphabetes oder Konnektive gespeichert werden können. Durch Zulassen des aus der ersten Ebene (Suchliste) bekannten „Indirekt"-Merkers in der algebraischen Liste kann die Listenhierarchie rekursiv um beliebig viele Ebenen erweitert werden. Hierdurch können auch solche reguläre Ausdrücke erfaßt werden, die nichtatomare Komponenten enthalten, wie z. B. Kompositionen von Residuen-Graphen zur Formulierung komplexer Syntheseprobleme. Die Synthese endlicher Automaten für Zwecke der Informationsreduktion und der Klassifikation über diesen Ansatz mittels des oben beschriebenen Syntheseverfahrens wird in [6] behandelt.

7. Strategien zur Synthese und Minimierung bei Anwendung von Suchlistendarstellungen

Die Möglichkeiten von Computern lassen sich über zyklische Unterprogramme häufig auf bequeme Weise effektiv nützen. Aus diesem Grunde sind auch bei Minimierung und Synthese einfache schematische Techniken strukturellen Methoden vorzuziehen. Aus Gründen des Programmier-Aufwandes und der Speicher-Effektivität sind möglichst elementar operierende Unterprogramme in möglichst geringer Anzahl wünschenswert. Die in Bild 4 gezeigten Regeln lassen sich, wie bereits angedeutet, in der gewünschten Weise auf wenige elementare Operationen zurückführen. Bei Zugrundelegung der dualen Darstellung der δ-Funktion (doppelte Suchliste) ergibt sich für einen Teil der Operationen eine Version a und eine Version b, wobei der Unterschied sich lediglich darauf bezieht, ob Anwendung auf die δ-Funktion oder die δ^{-1}-Funktion erfolgt.

A. Einstellige Prüfungen (das Bündel oder die Eigenschaft eines einzigen Knoten z_t wird geprüft).

Test 1 Alle abgehenden Zweige werden einzeln auf Atomarität geprüft;

Test 2a/(b) Alle abgehenden (ankommenden) Zweige werden einzeln auf das Leersymbol Λ geprüft;

Test 3 Alle abgehenden Zweige werden paarweise auf Gleichheit geprüft (Zweideutigkeit);

Test 4a/(b) Knoten wird auf End-(Anfangs-)knoten-Eigenschaft geprüft;

Test 5a/(b) Knoten wird auf Unzugänglichkeit (Isoliertheit) geprüft, d. h. auf Existenz eines ankommenden (abgehenden) Zweiges; (vgl. Regel 9).

B. Zweistellige Prüfungen (die Bündel zweier Knoten z_i und z_j werden verglichen).

Test 6a/(b) Prüfe Knotenpaar auf Vorwärts-(Rückwärts-)Äquivalenz.

Aufgrund der Ergebnisse solcher Tests können die anzuwendenden Operationen gewählt werden. Die hierzu verfügbaren Regeln nach Bild 4 lassen sich in folgende Elementar-Operationen zerlegen.

C. Einstellige Operationen (primär ist ein Knoten z_v bzw. z_f betroffen).

Operation 1 Definition eines neuen Knotens;

Operation 2a/(b) Entfernung eines unzugänglichen (isolierten) Knotens und seines Bündels; (vgl. Regel 9)

Operation 3 Entfernung eines einzelnen abgehenden Zweiges;

Operation 4 Definition eines einzelnen abgehenden Zweiges.

D. Zweistellige Operationen (primär sind zwei Zustände z_i und z_j betroffen).

Operation 5 Doppelung eines abgehenden Gesamt-Bündels (das abgehende Gesamtbündel des Knotens z_i wird zusätzlich als Bündel des Knotens z_j definiert);

Operation 6a/(b) Verschiebung eines abgehenden (ankommenden) Gesamtbündels (das betr. Gesamtbündel des Knotens z_i wird für z_j definiert und bei z_i entfernt);

Operation 7 Doppelung einer Zustands-Eigenschaft.

E. Algebraische Operation (primär ist ein Knoten z_v betroffen).

Operation 8 Disjunktive, multiplikative oder iterative Dekomposition eines abgehenden nicht-atomaren Zweiges.

Es ist sinnvoll, Elementarkombinationen aus obigen Prüfungen und Operationen zu einfachen Schleifen zu program-

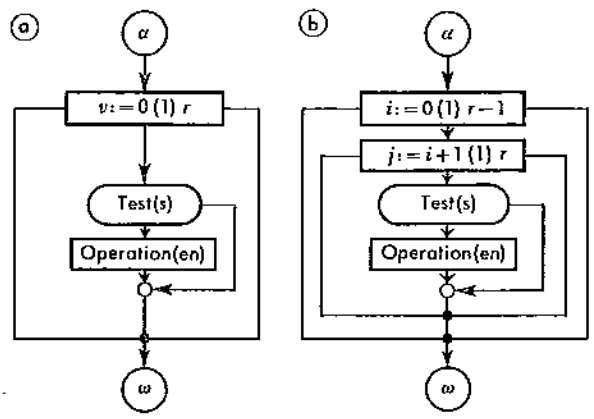


Bild 11. Aufbauschema für Transformationsschleifen.
a) einstellig, b) zweistellig.

mieren nach einem Schema, das in Bild 11 als Strukturdiagramm dargestellt ist. Diese Programmschleifen erstrecken sich über die Gesamtmenge aller Zustände oder Zustandspaare. Dies bedeutet, daß zwischen zwei Zyklen die gesamte Automatentafel Zwischenergebnis-Charakter annehmen kann, wie beispielsweise bei Bearbeitung indeterministischer Strukturen zweiter Art (1. Schleife: Prozedur 4, 2. Schleife: Prozedur 2, 3. Schleife: Prozedur 3a). Die folgende Liste zeigt den Satz von Schleifen für die Synthese aus regulären Ausdrücken. Dieser Satz enthält außerdem alle zur Reduktion explizit oder implizit vollständig definierter endlicher Automaten erforderlichen Operationen, wobei die Reduktion durch sukzessives Verschmelzen erfolgt (Bild 12).

F. Einstellige Transformations-Schleifen (betroffen ist die Zustandsmenge $\{z_v | v = 0, 1, 2, \ldots r\}$) (s. Bild 11a).

Prozedur 1 Dekomposition (Regeln 1—3)
(Zyklus: Test 1, Operation 8)

Prozedur 2 Leerzweig-Entfernung (Regeln 5 u. 9)
(Zyklus: Test 2a, Operationen 5, 3)

Prozedur 3a/(b) Entfernung von unzugänglichen (isolierten) Knoten mit $\lambda = 0$ (zusammen mit vorausgehender Prozedur 1: Regel 9 u. inverse)
(Zyklus: Tests 5a(b), 4b(a), Operation 2a(b),)

Prozedur 4 Determinierung vom Typ 2: nur Schritt 1 von Regel 8 nach Bild 13 (Schritt 2 durch nachfolgende Prozeduren 2 und 3));
(Zyklus: Test 3, Operationen 1, 1, 4, 4, 6b, 6b)

G. Zweistellige Transformationsschleifen (betroffen ist die Menge $\{(z_i, z_j) | i = 0, 1, \ldots (r-1); j = (i+1), \ldots r\}$ der Zustandspaare (s. Bild 11b).

Prozedur 5a/(b) Vorwärts-(Rückwärts-)Minimierung zusammen mit nachfolgender Prozedur 3a(b): (Regel 6 (7))
(Zyklus: Test 6a(b), Operation 6b(a),)

Prozedur 6 Doppelung von Knoten-Eigenschaften (wichtig bei Endzuständen, vgl. [6])
(Zyklus: Tests 2a(b), 4b(a), Operation 7).

Legt man die doppelte Suchlistendarstellung zugrunde, so kann man unter Verwendung der oben genannten Programmschleifen einen Synthesealgorithmus aufstellen, der nach der folgenden Strategie vorgeht:

Die Prozedur 1 wandelt den unechten Zustandsgraphen in einen echten mit Leerzweigen um. Die Leerzweige werden mit Prozedur 2 entfernt.

Da der entstandene Automat in der Regel nicht minimal ist, wird durch Prozedur 5a reduziert. Bei komplexeren Syntheseproblemen zwischenzeitlich auftretende Mehrdeutigkeiten werden durch Prozedur 4 beherrscht. Für die Minimierung vollständig oder quasi-vollständig definierter endlicher Automaten ist die Strategie nach Bild 12 ein sehr rechenzeit-effektiver Algorithmus, der auf einer Anwendung von Regel 6 beruht.

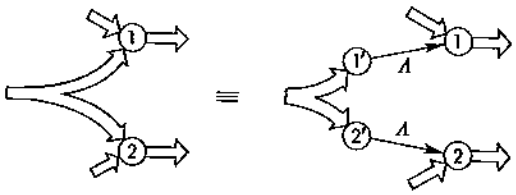


Bild 13. Determinierung einer indeterministischen Struktur zweiter Art.

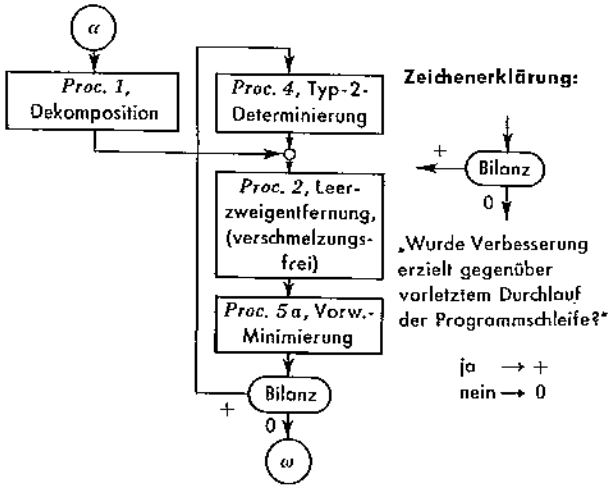


Bild 14. Beispiel für eine Strategie zur Simulation der Synthese von Minimalautomaten.

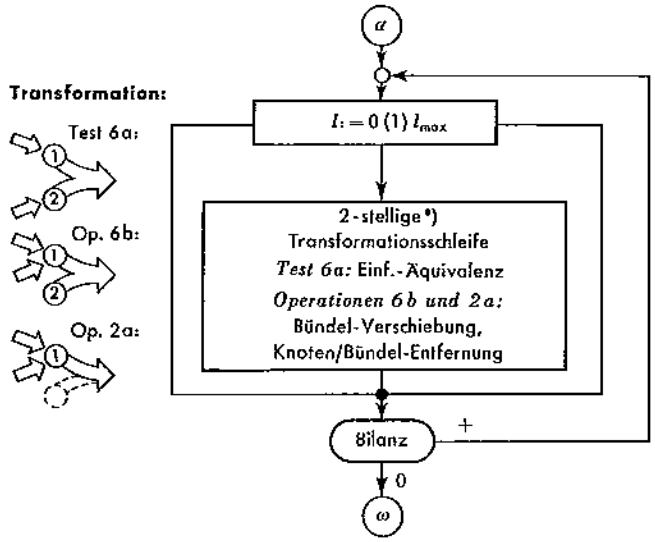


Bild 12. Minimierung endlicher Akzeptoren und ähnlicher Strukturen.

Literatur

[1] *Hartenstein, R.*, Über die Anwendung des endlichen Akzeptors auf das Zeichenerkennungsproblem. Dissertation, Karlsruhe, 1969.

[2] *Gill, A.*, Introduction to the Theory of Finite-State Machines. McGraw-Hill Publ., New York/Toronto/London 1962.

[3] *Binder, G.*, Simulation eines Erkennungsverfahrens für handgeschriebene Zeichen mit automatisiertem Klassifikator-Entwurf. Diplomarbeit am Institut für Nachrichtenverarbeitung und Nachrichtenübertragung der Universität Karlsruhe, 1968.

[4] *Fischer, K.*, Methoden der Formelübersetzung, in: Nicht-numerische Informationsverarbeitung (Hrsg. *R. Gunzenhäuser*). SpringerVerlag, Wien/New York 1968.

[5] *Ott, G. H., Feinstein, N. H.*, Design of Sequential Machines from their Regular Expressions. Journal ACM 8 (1961).

[6] *Hartenstein, R.*, Synthese endlicher Automaten bei Problemen der Erkennung, Klassifikation und Informationsreduktion, EIK 1970 (in Vorbereitung).

DIE BEITRÄGE DIESES HEFTES WURDEN GESCHRIEBEN VON

Dr.-Ing. *F. Wittgruber* (33). Studium der Nachrichtentechnik an der TH Darmstadt. 1964—1965 wiss. Mitarbeiter und Doktorand bei Prof. Dr.-Ing. E.h. K. Küpfmüller und Prof. Dr.-Ing. R. Piloty, TH Darmstadt. Seit 1966 wiss. Assistent bei Prof. Dr.-Ing. R. Piloty am Institut für Nachrichtenverarbeitung, TH Darmstadt. Promotion auf dem Gebiet der Speichertechnik.

Dipl.-Math. *Wolf Göhring* (30). Von 1950 bis 1959 Besuch des Realgymnasiums in Völklingen (Saar). Von 1959 bis 1965 Mathematik- und Physikstudium in Saarbrücken. 1965 Diplom in Mathematik mit der Arbeit „Vergleich von Lernprozessen".
Von 1965 bis 1969 wissenschaftlicher Mitarbeiter bei Prof. Händler — zunächst in Hannover, dann in Erlangen — auf dem Gebiet der Mustererkennung und des computer-aided-design.
Seit November 1969 Systemprogrammierer für den TR 440 bei AEG-Telefunken in Konstanz.

Dr.-Ing. *Reiner Hartenstein* (35) studierte von 1954 bis 1959 Nachrichtentechnik an der Universität Karlsruhe. Von 1960 bis 1965 war er Mitarbeiter des Kernforschungszentrums Karlsruhe, wo er zunächst die Tätigkeit eines Entwicklungsingenieurs für Elektronik ausübte und später eine Entwicklungsgruppe für digitale und hybride Elektronik leitete.
Seit 1965 ist er wissenschaftlicher Mitarbeiter am Institut für Nachrichtenverarbeitung und Nachrichtenübertragung der Universität Karlsruhe, wo er 1969 promovierte.

Dipl.-Ing. *Freiwalt Schön* (48). Beginn des Studiums Fachrichtung Maschinenbau 1947 an der Technischen Hochschule München. Volontär-Assistententätigkeit und halbjähriges Auslandspraktikum bis zum Eintritt 1953 in die Zentralkonstruktion der Fa. Siemens & Halske. Seit 1955 Führung eines Konstruktionsbüros für Automatisierung und 1961 Leitung einer Konstruktionsgruppe. Seit 1965 als Leiter einer Gruppe in der Abteilung Technologie im Bereich der Zentralen Forschung und Entwicklung, u.a. auf dem Gebiete des DVA-Einsatzes in der Konstruktion tätig.

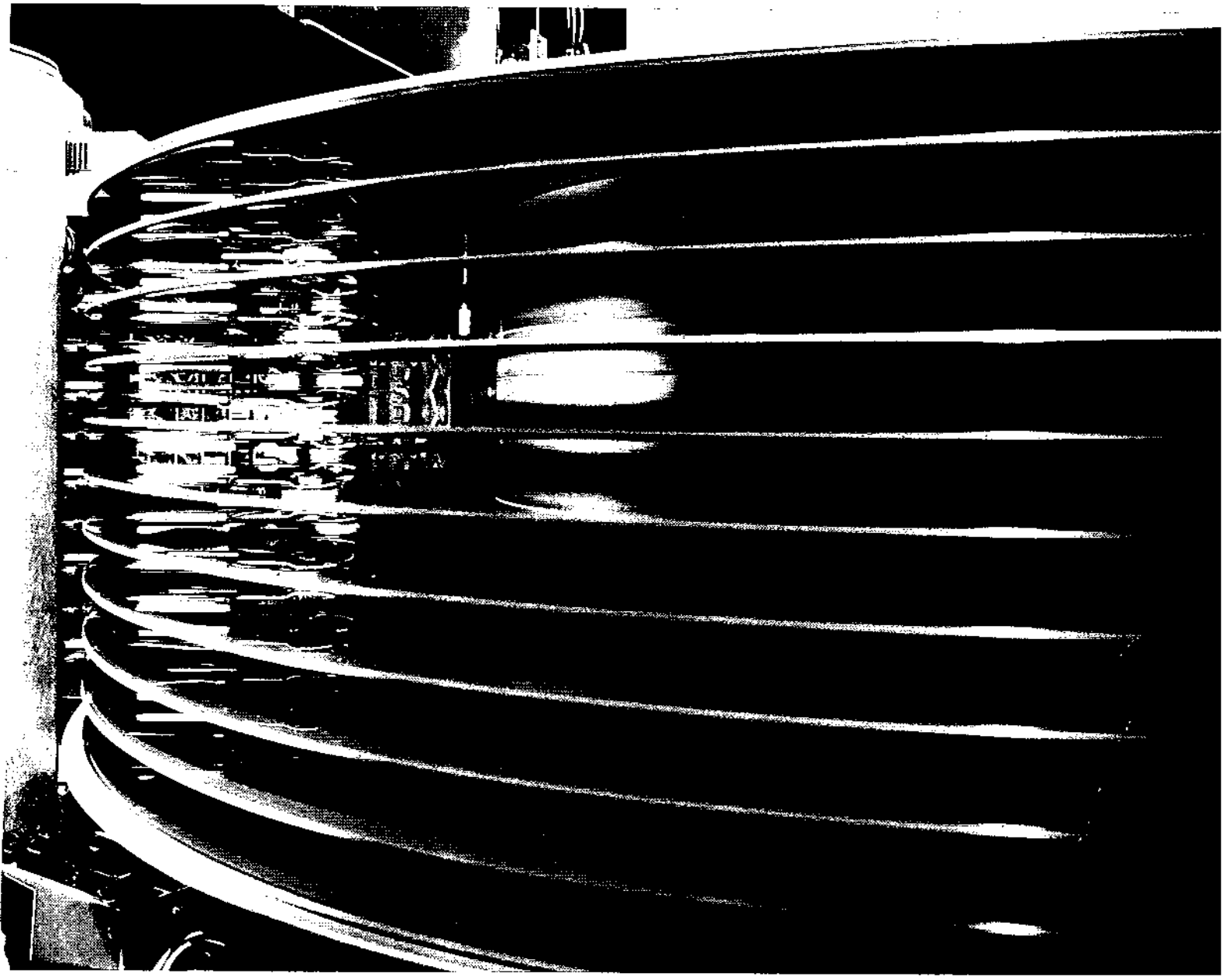
Postverlagsort München B 7922 F

ELEKTRONISCHE RECHENANLAGEN

eR

4 AUGUST 1970
12. Jahrgang
Seite 175–230
R. OLDENBOURG
MÜNCHEN & WIEN

ZEITSCHRIFT FÜR TECHNIK
UND ANWENDUNG
DER NACHRICHTENVERARBEITUNG
IN WISSENSCHAFT
WIRTSCHAFT UND VERWALTUNG



IBM-Pressefoto