Siegfried Wendt
Universität Kaiserslautern

Operationszustand versus Steuerzustand – eine äußerst zweckmäßige Unterscheidung

Das Problem

Die systemtheoretische Begründung für die Einführung des Zustandsbegriffs findet man im Mosaikstein "Der Zustandsbegriff in der Systemtheorie". Während sich die dortige Betrachtung sowohl mit kontinuierlichen als auch mit diskreten Systemen befaßt, wird hier die Betrachtung auf diskrete Systeme beschränkt. Außerdem wird hier von vorneherein eine Vektorisierung des Zustands in Betracht gezogen. Den hier angestellten Überlegungen liegt das Modell des deterministischen, diskreten und sequentiellen Systems zugrunde, welches durch die beiden folgenden Formeln charakterisiert wird:

$$Y(n) = \omega [Z(n), X(n)]$$

$$Z(n+1) = \delta [Z(n), X(n)]$$

Wenn man über das im oben erwähnten Mosaikstein über Zustände Gesagte hinauskommen will, muß man nach dem Nutzen des jeweils betrachteten Systems fragen.

Die Zustandsvariablen eines Systems ergeben sich aus der Aufgabenstellung, die mit diesem System erfüllt werden soll. Jede Zustandsvariable dient ja dazu, eine Information zu speichern, die später noch gebraucht wird. Aus der Anzahl der verschiedenen Zustandsvariablen und der Mächtigkeit ihrer Wertebereiche ergibt sich für komplexe Systeme meistens eine riesige Zahl potentieller Zustände, die leicht die Trillionengrenze überschreiten kann. Obwohl eine solche Zustandsmenge endlich ist, kommt eine explizite, von Menschen zur Kenntnis nehmende Aufzählung praktisch nicht in Frage. Deshalb steht man vor dem Problem, wie man solche Automaten spezifizieren und realisieren soll.

Es gibt Aufgabenstellungen, die ohne explizite Aufzählung der Zustände nicht zu lösen sind. Allerdings genügt es in diesen Fällen, daß die Aufzählung algorithmisch innerhalb eines Computersystems erfolgt und von keinem Menschen beobachtet wird. Derartige Aufgabenstellungen sind dadurch gekennzeichnet, daß für alle Zustandsvariablen die gleiche intuitive Semantik gilt. Diese Fälle werden hier aber nicht weiter betrachtet [Burch et al–90], [Burch et al–94].

Die im folgenden betrachteten Systeme sind dadurch gekennzeichnet, daß jede Zustandsvariable ihre individuelle intuitive Semantik haben kann. In diesen Fällen ist es grundsätzlich sinnvoll, zwei Typen von Zustandsvariablen zu unterscheiden, die man in den Beschreibungs– und Konstruktionsverfahren völlig unterschiedlich behandeln muß. Es handelt sich um die grundsätzliche Unterscheidung zwischen *Operationsvariablen* und *Steuervariablen*. Daß diese Unterscheidung eine gewaltige Vereinfachung des Problems großer Zustandsmengen mit sich bringt, wurde im Bereich der Automaten– und Schaltwerkstheorie bereits in der zweiten Hälfte der 60–iger Jahre von verschiedenen Autoren festgestellt und beschrieben [Glushkov–65], [Wendt–70].

Das Unterscheidungskriterium

> Ob eine Zustandsvariable als *Operationsvariable* oder als *Steuervariable* zu klassifizieren ist, entscheidet sich an der Frage, wie man den Wertebereich dieser Variablen definieren kann.

Es werden nun Beispiele betrachtet, die diesen Unterschied veranschaulichen und die es leicht machen, den Unterschied zwischen Operationsvariablen und Steuervariablen in einer strengen allgemeinen Definition zu fassen. In einem ersten Beispiel werden die Ruhezustände eines vereinfachten fahrkartenverkaufenden Automaten betrachtet. Die Vereinfachung besteht darin, daß der Automat nur eine einzige Art von Fahrkarten verkaufen kann, so daß der Käufer nicht angeben muß, welche Art Fahrkarte er haben will. Die Anstöße, die der Käufer gibt, bestehen lediglich in dem Einwurf von Münzen.

Man wird bei diesem Automaten den Zustand zweckmäßigerweise in drei Komponenten zerlegen, weil man jeweils in drei unterschiedliche Behälter im Innern des Automaten schauen muß, wenn man seine Reaktion Y auf eine geplante Eingabe X vorhersagen will.

Durch die erste Zustandskomponente wird die aktuelle Füllung des Fahrkartenbehälters berücksichtigt. Dieser Behälter hat ja – wie jeder andere Behälter innerhalb des Automaten auch – nur eine endliche Kapazität. Deshalb ist es möglich, daß dieser Behälter durch den früheren Verkauf vieler Fahrkarten inzwischen leer geworden ist. In diesem Fall wird der Automat auf eine Eingabe zwangsläufig anders reagieren, als wenn der Behälter noch nicht leer ist.

Durch die zweite Zustandskomponente wird die Füllung des Eigengeldbehälters berücksichtigt. In diesen Eigengeldbehälter wird man bei Inbetriebsetzung des Automaten eine bestimmte Menge Münzen in einer bestimmten Münzenverteilung hineinfüllen, damit bereits beim Verkauf der ersten Fahrkarte Wechselgeld herausgegeben werden kann. Nach jedem erfolgreichen Verkauf einer Fahrkarte wird man auch die von dem Käufer eingeworfenen Münzen in den Eigengeldbehälter aufnehmen. Es ist leicht einzusehen, daß der aktuelle Inhalt des Eigengeldbehälters Einfluß auf das Verhalten des Automaten hat. So könnte es beispielsweise sein, daß der Automat kein weiteres Geld mehr annehmen kann, weil der Eigengeldbehälter für eine bestimmte Münzenart bereits bis zur Kapazitätsgrenze gefüllt ist. Oder aber es könnte der Fall eintreten, daß der Automat das erforderliche Wechselgeld nicht herausgeben kann, weil er die dafür notwendigen Münzen nicht in seinem Behälter findet.

Durch die dritte Zustandskomponente wird ebenfalls ein Münzenbehälter berücksichtigt. Es muß nämlich neben dem Eigengeldbehälter noch einen separaten Behälter geben, über den der aktuelle Verkauf einer Fahrkarte abgewickelt wird. Solange der Verkauf einer Fahrkarte noch nicht vollständig abgewickelt ist, dürfen die vom Käufer eingeworfenen Münzen noch nicht in den Eigengeldbehälter übernommen werden, weil sie noch nicht Eigentum der fahrkartenverkaufenden Institution geworden sind. Es könnte ja sein, daß der Käufer durch den Einwurf seiner letzten Münze eine Situation herbeiführt, wo ein bestimmtes Wechselgeld herausgegeben werden müßte, das der Automat aber mangels eines entsprechenden Münzenvorrats nicht ausgeben kann. In diesem Fall muß der Automat dem Käufer alle Münzen, die dieser bisher eingeworfen hat, wieder zurückgeben. Der separate Münzenbehälter für die Abwicklung eines Fahrkartenverkaufs ist selbstverständlich auch allein deshalb schon erforderlich, weil man sonst ja nicht mehr feststellen könnte, wieviel der Käufer inzwischen schon angezahlt hat, ohne einen Gegenwert vom Automaten erhalten zu haben.

In diesem Beispiel haben alle drei Zustandsvariablen jeweils das typische Merkmal von Operationsvariablen:

> Man kann den Wertebereich einer Operationsvariablen definieren,
> ohne explizit alle möglichen Wertübergänge aufzählen zu müssen.

Häufig, aber nicht immer trifft auf Operationsvariable auch noch der Sachverhalt zu, daß kein Bedarf an einer expliziten Aufzählung der Elemente des Wertebereichs besteht, d.h. daß die Interpretation der Elemente des Wertebereichs beschrieben werden kann, ohne daß diese Elemente explizit aufgezählt werden müssen. Im betrachteten Beispiel besteht kein Bedarf an einer expliziten Aufzählung der Elemente der Wertebereiche der Operationsvariablen.

Im Beispiel des fahrkartenverkaufenden Automaten fragt man, wie eine aktuelle Füllung des jeweils betrachteten Behälters aussehen kann und was sie für den Betrieb des Automaten bedeutet:
– Die Füllung des Fahrkartenbehälters legt fest, wie viele Fahrkarten in Zukunft noch verkauft werden können, ohne daß eine Nachfüllung erfolgen muß.
– Die Füllung des Eigengeldbehälters gibt an, wieviel Geldeigentum der fahrkartenverkaufenden Institution sich innerhalb des Automaten befindet und in welcher Münzenverteilung dieses Eigentum vorliegt. Dieses Eigentum ergibt sich als Summe aus dem ursprünglich hineingesteckten Wechselgeld und den bisherigen Verkaufserlösen.
– Die Füllung des Verkaufsabwicklungsbehälters gibt an, wieviel der Käufer bisher einbezahlt hat und welche Münzen er dafür verwendet hat.

Immer wenn auf eine explizite Aufzählung der Elemente des Wertebereichs verzichtet werden kann, kann auch eine extreme Erweiterung des Wertebereichs keinerlei Probleme mit sich bringen. Man betrachte hierzu zwei unterschiedliche Varianten des fahrkartenverkaufenden Automaten:
In der Variante 1 betrage der Fahrkartenpreis 1 DM, und in der Variante 2 betrage der Preis eine Million DM. Weiterhin wird angenommen, daß man in beiden Fällen nur mit 10–Pfennig–Münzen bezahlen kann. Der Wertebereich der Zustandsvariablen, die dem Verkaufsabwicklungsbehälter zugeordnet ist, umfaßt im Fall der Variante 1 zehn Elemente, im Fall der Variante 2 zehn Millionen Elemente. Trotz dieser riesigen Zustandszahl ist aber die Variante 2 nicht schwieriger zu verstehen und zu behandeln als die Variante 1 mit ihrer kleinen Zustandszahl. Denn die Verhaltensbeschreibung

$$\begin{pmatrix} Y(n) \\ Z(n+1) \end{pmatrix} = \begin{cases} \begin{pmatrix} \text{nichts} \\ Z(n) + 10\ \text{Pfg.} \end{pmatrix} & \text{falls } (Z(n) + 10\text{Pfg.}) < \text{Fahrkartenpreis} \\ \begin{pmatrix} \text{Fahrkarte} \\ 0\ \text{Pfg.} \end{pmatrix} & \text{sonst} \end{cases}$$

ist unabhängig von der Mächtigkeit des Wertebereichs der Zustandsvariablen Z. (Der Einfachheit halber blieben in dieser Verhaltensbeschreibung die Möglichkeiten des Überlaufens des Eigengeldbehälters und des Leerlaufens des Fahrkartenbehälters unberücksichtigt.)

Zustandsvariable, auf die das Kriterium für Operationsvariable nicht zutrifft, sind Steuervariable.

> Man kann den Wertebereich einer Steuervariablen nur definieren,
> indem man explizit alle Werte und möglichen Wertübergänge aufzählt.

Da die einzelnen Elemente des Wertebereichs einer Steuervariablen explizit aufgezählt werden müssen und da außerdem explizit die möglichen Wertübergänge festgelegt werden müssen, ist der Zustandsgraph das geeignete Mittel zur Darstellung der Sachverhalte, die eine Steuervariable betreffen. Demgegenüber brächte es keinerlei Vorteile, den Wertebereich und die Wertübergänge einer Operationsvariablen in einem Zustandsgraphen zu erfassen, selbst wenn die Mächtigkeit des Wertebereiches dieses zuließe. Damit entpuppt sich das Problem der riesigen Zustandszahlen als ein Scheinproblem, denn die eigentliche Ursache für die großen Zustandszahlen liegt in den Mächtigkeiten der Wertebereiche der Operationsvariablen, die man aber gar nicht elementweise zu betrachten braucht. Der Sachverhalt, daß man den Wertebereich einer Steuervariablen gar nicht definieren kann, ohne seine Elemente und die Wertübergänge explizit aufzuzählen, sorgt zwangsläufig dafür, daß die Mächtigkeit solcher Wertebereiche innerhalb beherrschbarer Grenzen bleibt.

Es kann durchaus vorkommen, daß man in bestimmten Fällen nicht auf den ersten Blick erkennt, ob eine Zustandsvariable als Operations– oder als Steuervariable einzuordnen ist. In diesen Fällen ist es zweckmäßig, sich zu überlegen, wie man jemand anderem auf didaktisch optimale Weise Einsicht in den Wertebereich dieser Variablen vermitteln würde. Die folgenden Beispiele sollen in diesem Sinne als Anregung dienen.

Beispiele

(1) In vielen Autotypen gibt es heutzutage die Möglichkeit, sich in einem kleinen Anzeigefeld nacheinander die Uhrzeit, die Außentemperatur, den aktuellen Kraftstoffverbrauch in Liter pro 100 km, die Länge in km der seit dem letzten Anlassen des Motors gefahrenen Strecke, die mittlere Geschwindigkeit für diese Strecke und ähnliches mehr anzeigen zu lassen. Die Auswahl der aktuellen Wertequelle wird durch den Zustand eines Automaten bestimmt, den der Fahrer jeweils durch einen Tastendruck in den nächsten Zustand weiterschalten kann. Die Zustandsfolge ist zyklisch, d.h. es gilt

$$\forall n:\ Z(n+q) = Z(n)\ ,\quad \text{wobei q die Anzahl der unterschiedlichen Anzeigequellen ist.}$$

Als Zustandsrepertoire repZ wird man hier den Zahlenwertebereich { 1, 2, 3, ... (q–1), q } wählen, weil man dann den Zustand mit der Nummer der ausgewählten Wertequelle gleichsetzen kann:

$$\text{Nummer der aktuell ausgewählten Wertequelle} = Z(n)$$

Für die Zustandsübergangsfunktion gilt dann die einfache Formel

$$Z(n+1) = 1 + (\ Z(n)\ )_{\text{mod q}}$$

Offensichtlich konnte man hier den Wertebereich repZ der Zustandsvariablen einführen, ohne die Werte und die Wertübergänge einzeln explizit aufzuzählen. Man konnte sogar die Mächtigkeit dieses Wertebereichs offen lassen, denn es genügt zu wissen, daß sie gleich der Anzahl q der unterschiedlichen Anzeigequellen ist.

Allerdings wurde dabei angenommen, daß die Festlegung, wie die unterschiedlichen Wertequellen numeriert sind, nicht zur Automatenspezifikation gehört. Es wurde also ein Systemaufbau angenommen, wie er links in Bild 1 gezeigt ist.

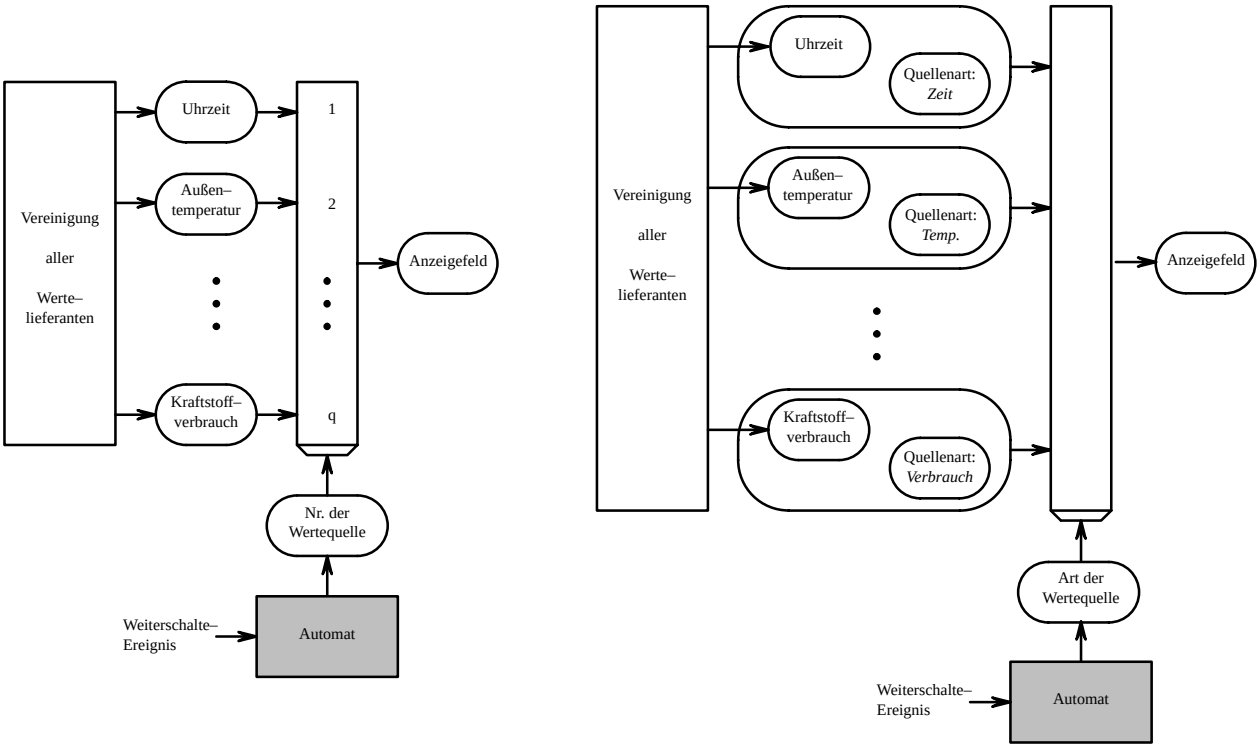

Bild 1 Zwei Varianten eines Systems zur Verbindung alternativer Wertequellen mit einem Anzeigefeld

Wenn man dagegen annimmt, daß die Numerierung der Wertequellen zur Automatenspezifikation gehört, d.h. wenn man einen Systemaufbau annimmt, wie er rechts in Bild 1 gezeigt ist, dann kann man repZ nicht mehr mitteilen, ohne alle Elemente einzeln explizit aufzuzählen. Denn im Unterschied zur Angabe

$$\text{repZ} = \{\ 1,\ 2,\ \ldots\ ,\ q\ \},$$

wo man genau weiß, was anstelle von "..." zu denken ist, gibt es im Falle von

$$\text{repZ} = \{\ \textit{Zeit},\ \textit{Temperatur},\ \ldots\ ,\ \textit{Verbrauch}\ \}$$

ein entsprechendes implizites Wissen nicht. Aber immerhin kann man die Elemente von repZ vollständig aufzählen und begründen, ohne die Zustandsübergänge aufzählen zu müssen. Deshalb handelt es sich bei der Zustandsvariablen in beiden Fällen um eine Variable für einen Operationszustand, der entweder die Nummer oder die Art einer Wertequelle angibt.

(2) Als zweites Beispiel wird das Automatenmodell einer Maschine betrachtet, der man eine endliche Folge übergeben kann, die von der Maschine sortiert werden soll. Damit die Möglichkeit des Sortierens überhaupt gegeben ist, muß für jedes Paar (e_i, e_k) von Elementen aus der Folge entschieden werden können, ob $e_i \leq e_k$ gilt oder nicht. Die Maschine soll nach dem sog. "Quicksort"–Algorithmus verfahren.

Die Grundidee zu diesem Algorithmus ist die folgende: Man sucht eine Trennstelle, an der die Folge in zwei kürzere Folgen zerschnitten werden kann, welche unabhängig voneinander sortiert werden können. Die beiden Teilfolgen sollen also die Bedingung erfüllen, daß das größte Element in der vorderen Teilfolge nicht größer ist als das kleinste Element in der hinteren Teilfolge. Die Suche einer solchen Trennstelle be-

ginnt damit, daß man willkürlich ein Element aus der Folge auswählt, zu dem es nach der erfolgreichen Auftrennung in der vorderen Teilfolge kein größeres und in der hinteren Teilfolge kein kleineres Element geben soll. Dieses ”Referenzelement” wird das größte der vorderen oder das kleinste der hinteren Teilfolge werden.

Bild 2 zeigt ein Beispiel, anhand dessen der Algorithmus veranschaulicht werden soll, der zu der Zerlegung der ursprünglichen Folge in die zwei gewünschten Teilfolgen führt. Ganz oben im Bild ist die zu sortierende Folge dargestellt. Man kann sich vorstellen, daß die Folge a[1:N] aus N=18 nebeneinanderliegenden quadratischen Kärtchen besteht, wobei jedes Kärtchen mit einer natürlichen Zahl beschriftet ist. Die beiden Variablen v und h dienen dazu, die Grenzpositionen ”vorne” und ”hinten” der jeweils aktuell zu bearbeitenden Teilfolge festzuhalten. Da zu Beginn die ganze Folge zu bearbeiten ist, muß als Initialisierung die Zuweisung (v, h):=(1, 18) erfolgen. Als Referenzelement wird das Element auf der Position

$$m = \frac{(v + h) - (v + h)_{\bmod 2}}{2}$$

gewählt, also das Element, welches genau in der Mitte oder – falls es kein Mittenelement gibt – unmittelbar links von der Mitte liegt. Im Beispiel ist m=9; das Element an dieser Position ist schattiert dargestellt.

Die beiden INTEGER–Variablen i und j dienen dazu, die Trennstelle, an der die Folge zerschnitten werden soll, einzugrenzen. In der Folge unten in Bild 2 ist die Trennstelle eingezeichnet. Links von der Trennstelle gibt es keine Elemente, die größer als das schattierte Element, also größer als 14 sind, und rechts von der Trennstelle gibt es keine Elemente, die kleiner als 14 sind. Das schattierte Element wurde in diesem Beispiel zum kleinsten Element der hinteren Teilfolge.

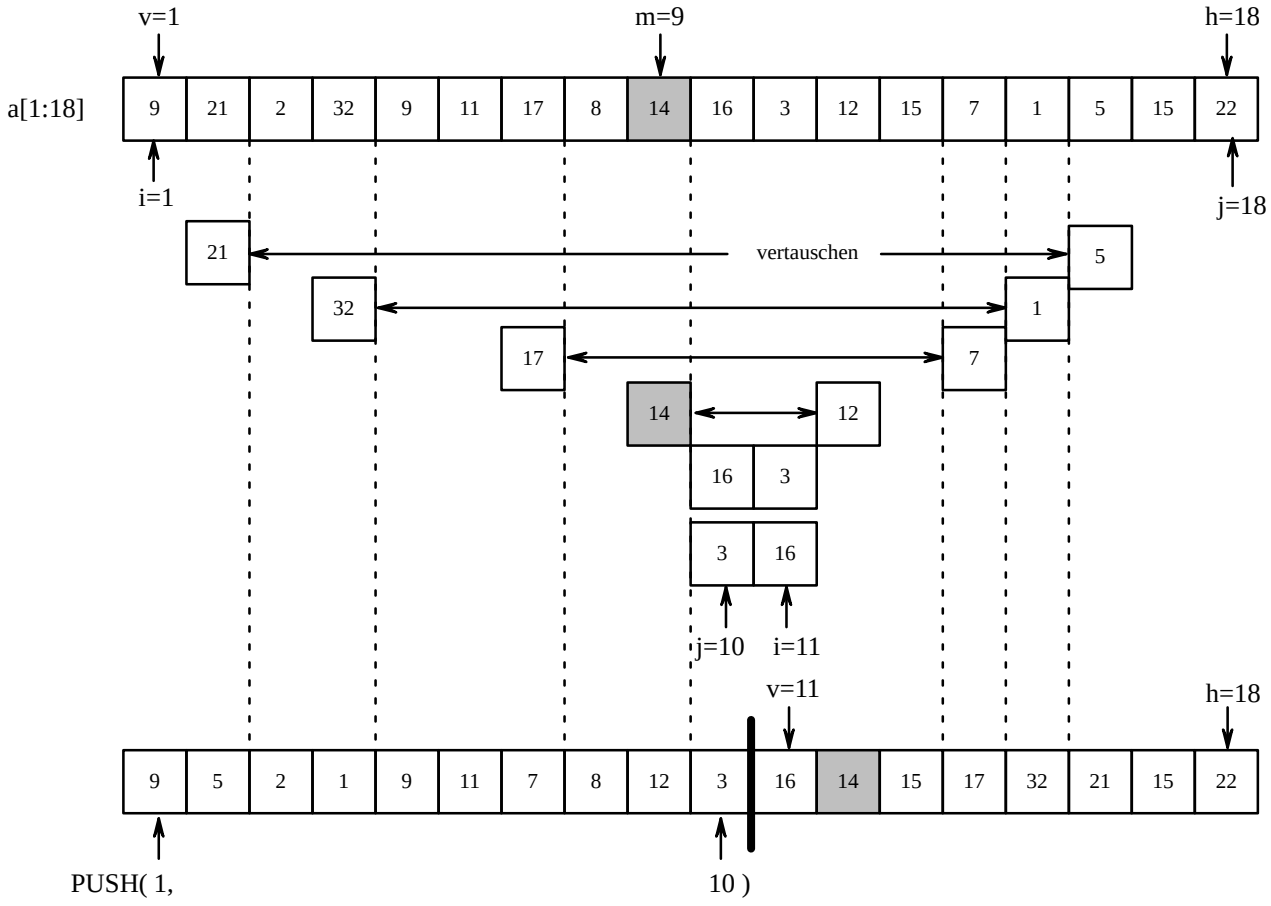

Bild 2 Veranschaulichung des ”Quicksort”–Algorithmus

Ausgehend von der oben im Bild vorgegeben Folge erhält man die unten im Bild dargestellte Folge, indem man wie folgt verfährt: Die Variablen i und j werden zu Beginn auf die Werte (v, h)=(1, 18) gesetzt. Nun werden von außen nach innen Elementpaare gesucht, bei denen das linke, auf der Position i liegende Partnerelement größer oder gleich dem schattierten Element ist, während das rechte, auf der Position j liegende Partnerelement kleiner oder gleich dem schattierten Element ist. Als erstes wird das Paar (a[2]=21, a[16]=5) gefunden. Die beiden Partnerelemente des Paares werden nun bezüglich ihrer Positionen vertauscht. Diese Paarsuche und Vertauschung wird so lange wiederholt, bis sich die Positionen i und j gekreuzt haben, d.h. bis i>j gilt. Dies ist in Bild 2 bei der Situation (a[i]=a[11]=16, a[j]=a[10]=3) erreicht.

Man muß sich nun entscheiden, welche der beiden durch die Teilung gewonnenen Teilfolgen man als nächste bearbeiten will. Die Entscheidung wird so gefällt, daß grundsätzlich immer zuerst die hintere Teilfolge bearbeitet wird. Im Beispiel äußert sich dies darin, daß die Zuweisung (v, h):=(11, 18) erfolgt und daß das Paar der Grenzpositionen der vorderen Teilfolge, also (1, 10) zur späteren Bearbeitung auf den Stack gelegt wird.

Wenn man die Sortiermaschine als Automat modellieren will, muß man zuerst einmal danach fragen, über welche Ein– und Ausgabekanäle die Maschine mit ihrer Umgebung kommunizieren kann und welche Elemente über diese Kanäle fließen können. Diese Elemente bilden die Repertoires repX und repY. Bild 3 zeigt die Maschine als Black–Box mit ihren Kanälen. Man erkennt zwei Eingabekanäle und drei Ausgabekanäle. Alle drei Ausgabekanäle sollen vom Display–Typ sein, d.h. über diese Kanäle wird etwas angezeigt, aber es können auf diesen Kanälen keine Ausgabeereignisse auftreten – es kann also kein Material fließen, und es kann keine Sprachausgabe erfolgen. Zwei der Ausgabekanäle dienen der Möglichkeit, Eingabekanäle zu sperren, d.h. den Einwurfschlitz zu verschließen bzw. die Drucktaste zu verriegeln.

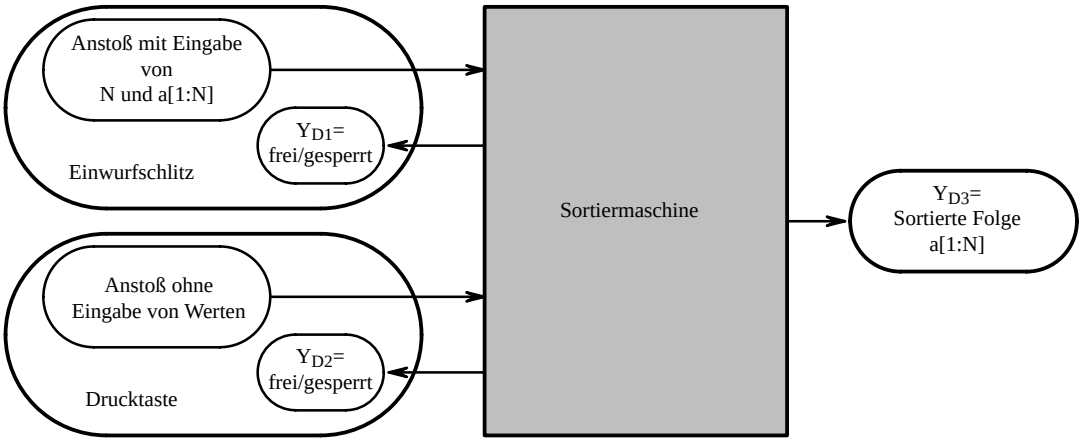

Bild 3 Sortiermaschine als Black–Box mit Kanälen zur Kommunikation mit der Umgebung

Den Eingabekanal oben in Bild 3 kann man sich als Einwurfschlitz vorstellen, in den, wenn er geöffnet ist, eine beschriftete Karte eingeworfen werden soll, auf der die Folgenlänge N und die zu sortierende Folge a[1:N] steht. Es wird hier vorausgesetzt, daß keine unsinnig beschrifteten Karten eingeworfen werden. Den Eingabekanal unten in Bild 3 kann man sich als Drucktaste vorstellen, die man, wenn sie nicht verriegelt ist, drücken soll, um den nächsten Arbeitsschritt der Maschine auszulösen. Nachdem die Aufgabenkarte eingeworfen wurde, wird diese Taste freigegeben. Man muß so oft auf diese Taste drücken, bis im vorher leeren Display–Fenster Y_{D3} die Anzeige der sortierten Folge erscheint. Ein einmaliges weiteres Drücken führt wieder in die Anfangssituation, in der der Einwurfschlitz geöffnet, die Taste verriegelt und das Ergebnis–Display leer sind.

In den Bildern 4 und 5 sind zwei Zustandsgraphen dargestellt, die zwei Varianten des Automatenverhaltens beschreiben. Der Gesamtzustand des Automaten setzt sich aus sieben Komponenten zusammen:

$$Z = (z_1, z_2, z_3, z_4, z_5, z_6, z_7) = (a[1:N], STACK, v, h, i, j, Steuerzustand) .$$

Es besteht eine 1:1–Zuordnung zwischen der Menge der Zustandsknoten des jeweiligen Graphen und dem Wertebereich der Zustandsvariablen *Steuerzustand*. Die Wertebereiche der anderen sechs Zustandsvariablen haben dagegen keine solche Entsprechung im Zustandsgraphen. Diese Zustandsvariablen kommen aber explizit in den Zustandsübergangsformeln vor, die in den Transitionen stehen, während die Variable *Steuerzustand* im Zustandsgraphen nicht vorkommt.

Was es zu den Zuständen 1, 2 und 3 zu sagen gibt, ist für beide Varianten gleich: Die Symbolisierung des Zustands 1 durch einen doppelt berandeten Kreisknoten soll darauf hinweisen, daß dieser Zustand der Anfangszustand ist, in dem sich die Maschine vor der allerersten Eingabe befindet. In diesem Zustand ist der Einwurfschlitz geöffnet, die Drucktaste ist verriegelt, und das Ausgabedisplay ist leer. Im Zustand 2 ist der Einwurfschlitz geschlossen, die Drucktaste ist freigegeben, und das Ausgabedisplay ist immer noch leer. Im Zustand 3 ist der Einwurfschlitz immer noch geschlossen, die Drucktaste ist freigegeben, und im Ausgabedisplay wird die sortierte Folge angezeigt.

Beim Übergang aus dem Zustand 1 in den Zustand 2 wird der Initialzustand für den Quicksort–Algorithmus hergestellt. Der Übergang aus dem Zustand 2 in den Zustand 3 kennzeichnet das Ende der Abwicklung des Quicksort–Algorithmus. Die beiden Varianten unterscheiden sich in der Anzahl der unterschiedlichen Steuerzustände, die der Automat im Laufe der Abwicklung des Quicksort–Algorithmus einnimmt. Diese Zustände – bei der Variante I bilden sie die Menge {2, 4, 5, 6, 7} und bei der Variante II die Menge {2, 4} – sind anhand der Erscheinungen auf den Ausgangskanälen nicht zu unterscheiden, denn in all diesen Zuständen ist der Einwurfschlitz geschlossen, die Drucktaste freigegeben und das Ausgabedisplay leer.

Die unterschiedliche Anzahl von Steuerzuständen für den Quicksort–Algorithmus ergibt sich aus den unterschiedlichen Umfängen der Zustandszuweisungen, die in den Transitionen stehen. In der Variante I werden manche Zuweisungen in aufeinanderfolgenden Schritten ausgeführt, die man in einem Schritt machen könnte und die in der Variante II auch tatsächlich in einem Schritt erledigt werden. So sind beispielsweise die beiden Zuweisungen, die in der Variante I beim Übergang vom Zustand 6 nach 7 einerseits und von 7 nach 4 andererseits ausgeführt werden, in der Variante II zu einer einschrittigen Zuweisung zusammengefaßt, die beim Übergang vom Zustand 4 nach 4 erfolgt.

Man sollte beachten, daß es sich bei der Reduktion der Anzahl der Steuerzustände beim Übergang von der Variante I zur Variante II nicht um eine Zustandsreduktion handelt, wie sie aus der Theorie der endlichen Automaten bekannt ist. Denn dort ist an eine Zustandsreduktion die Bedingung geknüpft, daß die Reduktion keine Änderung des außen beobachtbaren Automatenverhaltens bewirken darf. Im vorliegenden Falle dagegen zeigen die beiden Varianten durchaus ein experimentell feststellbares unterschiedliches Verhalten, wobei aber angenommen wurde, daß dieser Unterschied für den Benutzer der Maschine irrelevant sei. Denn der Unterschied besteht lediglich darin, daß die Variante I zur Erledigung einer Sortieraufgabe mehr Anstöße braucht als die Variante II.

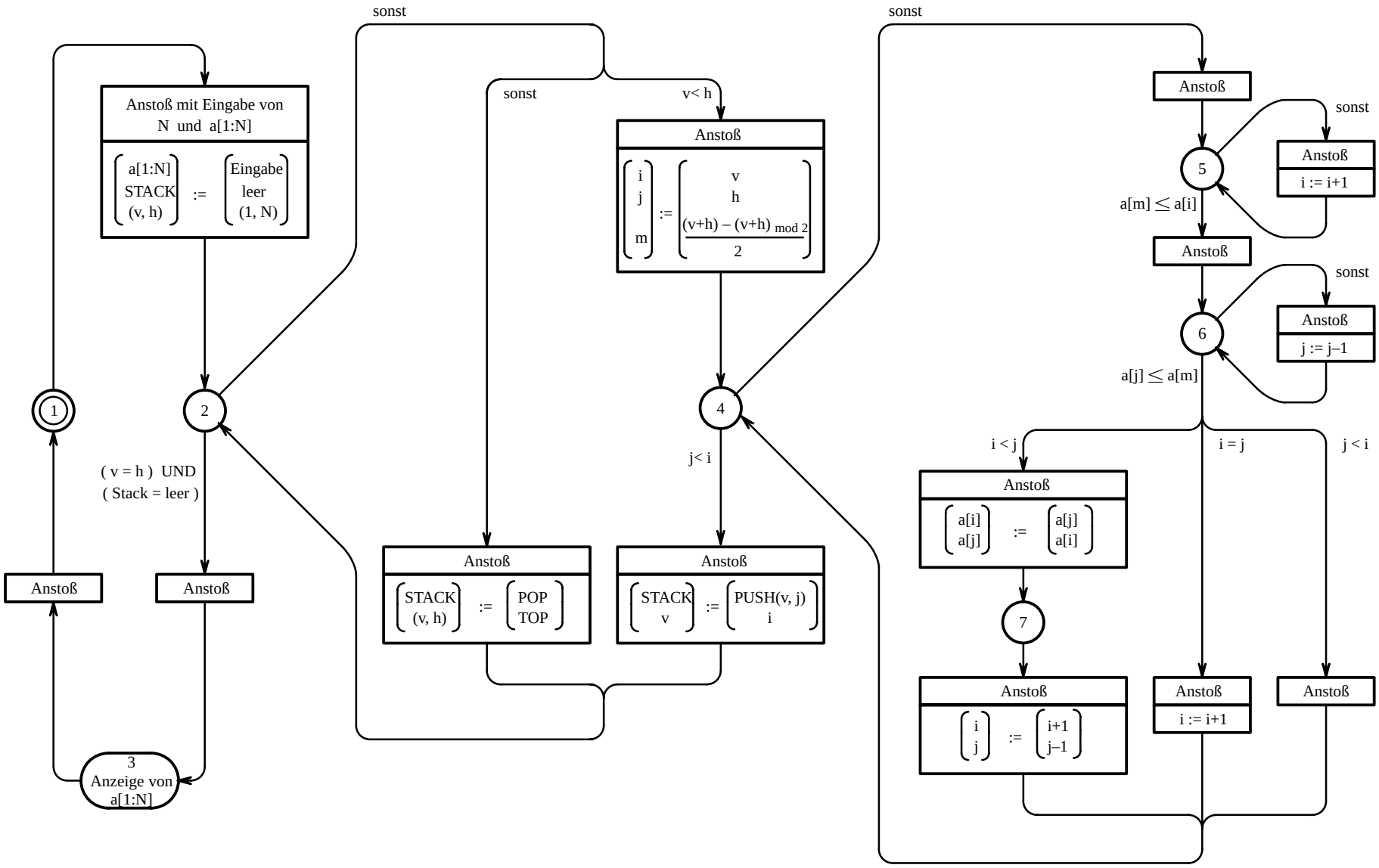

Bild 4 Variante I des Quicksort–Algorithmus

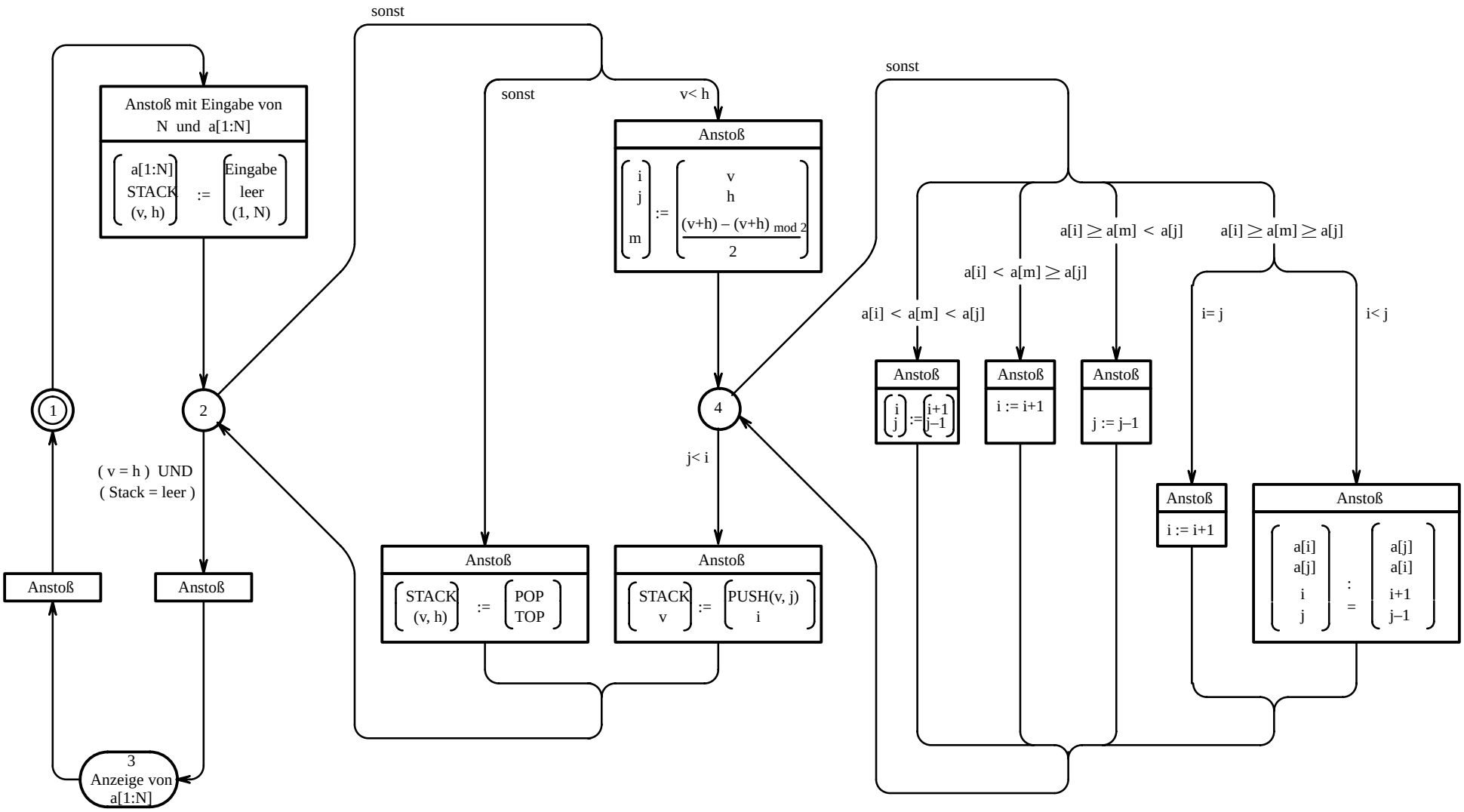

Bild 5 Variante II des Quicksort–Algorithmus

(3) Vergleich der beiden Beispiele:

Im Beispiel (1), d.h. beim Automaten zur Auswahl einer Wertequelle ist repZ={ 1, 2, ... , q }. Im Beispiel (2) ist das Repertoire der Steuerzustände im Fall der Variante I gleich der Menge {1, 2, 3, 4, 5, 6, 7}. Abgesehen von der offen gelassenen Mächtigkeit q des ersten Repertoires kann man keinen Unterschied zwischen den beiden Mengen erkennen. Daß es sich dennoch bei der ersten Menge um ein Repertoire von Operationszuständen und bei der zweiten Menge um ein Repertoire von Steuerzuständen handelt, kann man nur erkennen, wenn man fragt, welche Bedeutungen man denn zu den natürlichen Zahlen in diesen Mengen assoziiert. Zu den Zahlen der ersten Menge assoziiert man die Elemente einer Menge von Wertequellen, und zu den Zahlen der zweiten Menge assoziiert man die Knoten im Graphen in Bild 4. Man kann also jeweils eine Menge von Assoziationspaaren angeben:

zum 1. Beispiel: { (1, Uhrzeit), (2, Außentemperatur), ... (q, Kraftstoffverbrauch) }

zum 2. Beispiel: { (1, Knoten 1), (2, Knoten 2), ... (7, Knoten 7) }

Man sieht auf den ersten Blick, daß die erste Menge in eine ganz andere Kategorie fällt als die zweite: Während man der ersten Menge nach Festlegung des Wertes von q und nach entsprechender Vervollständigung der Aufzählung der Elemente alle Information entnehmen kann, die man für das Verständnis braucht, muß man im Falle der zweiten Menge den Graphen kennen, um mit einem Assoziationspaar (j, Knoten j) überhaupt ein Verständnis verbinden zu können. Die Notwendigkeit, den Graphen zu kennen, bliebe auch erhalten, wenn man statt ”Knoten j” jeweils einen Text verwenden würde, dem man Information über die ”Bedeutung des Knoten im Algorithmus” entnehmen könnte, denn die Vereinigung all dieser Texte könnte letztlich nichts anderes sein als eine mehr oder weniger vollständige und eindeutige Umschreibung des Graphen. Um dies einzusehen, braucht man nur einmal zu versuchen, einen angemessenen Text zu formulieren, aus dem man ein Verständnis für die Bedeutung des Knotens 5 in Bild 4 gewinnen kann.

Das Steuerkreis–Modell

Die Möglichkeit, einen Zustand Z als Tupel ($z_1, z_2, z_3, \ldots z_k$) zu betrachten, bedeutet keineswegs, daß es sinnvoll sein muß, den Automaten mit dem Zustand Z nun als Komposition aus k kleineren Automaten mit dem jeweiligen Zustand z_j zu betrachten. Wenn man aber das Zustandstupel unterteilen kann in den Steuerzustand Z_{ST} und das Tupel Z_{OP} der Operationszustände, dann ist es sowohl bezüglich der Analyse als auch bezüglich der Synthese äußerst zweckmäßig, den Automaten als Steuerkreis zu modellieren.

Die Bezeichnung ”Steuerkreis” wurde vom Autor eingeführt in Anlehnung an den Begriff des Regelkreises. Unter einem Regelkreis versteht man ein aus den beiden Komponenten *Regelstrecke* und *Regler* aufgebautes System, dessen Struktur links in Bild 6 gezeigt ist. Rechts im Bild ist eine Steuerkreisstruktur dargestellt, die sich für die Erklärung des Prinzips besonders gut eignet.

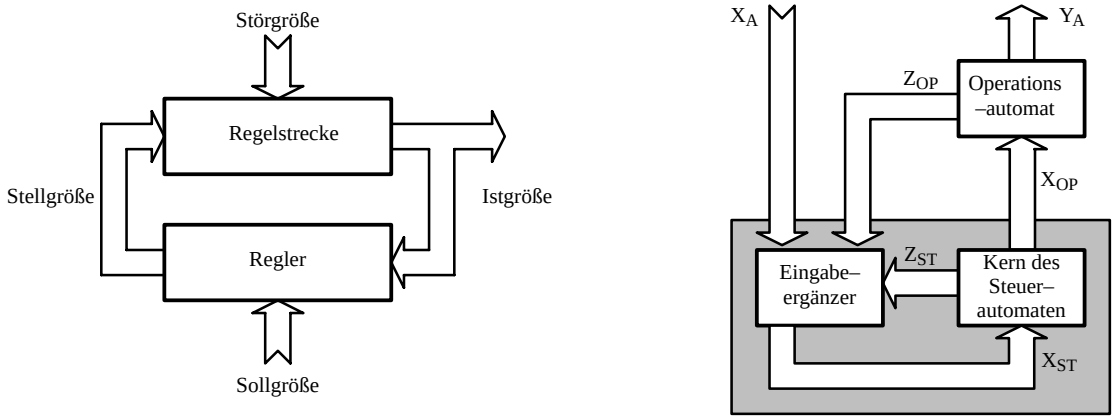

Bild 6 Gegenüberstellung der beiden Systemstrukturen ”Regelkreis” und ”Steuerkreis”

> Der Steuerkreis ist eine Systemstruktur,
> bei der eine steuernde Komponente und eine gesteuerte Komponente
> so miteinander verbunden sind, daß gerichtete Kanäle einen Kreis bilden.

Grundsätzlich schließt der Begriff des Steuerkreises das Vorkommen nebenläufiger Vorgänge nicht aus. Hier aber soll nur der sequentiell arbeitende Steuerkreis betrachtet werden, der aus einem Operationsautomaten und einem Steuerautomaten besteht. Obwohl er aus zwei Automaten zusammengesetzt ist, kann man den Steuerkreis noch formal als einen einzigen Automaten betrachten: Er hat je einen Kanal für die Eingabe X_A und die Ausgabe Y_A , und sein Zustand Z ist gleich dem Tupel (Z_{OP}, Z_{ST}).

Der Steuerautomat, der in Bild 6 durch das schattierte Rechteck symbolisiert wird, besteht aus zwei Komponenten. Der Eingabe–Ergänzer hat Zuordnerverhalten, d.h. er hat keinen veränderlichen Zustand, sondern realisiert die Funktion

$$X_{ST} = \lambda(X_A, Z_{OP}, Z_{ST}) .$$

Damit der Eingabe–Ergänzer die aktuellen Zustände Z_{OP} und Z_{ST} als Argumentinformationen heranziehen kann, müssen der Operationsautomat und der Kern des Steuerautomaten ihren Zustand jeweils über einen Ausgabekanal vom Display–Typ anzeigen.

Wenn der Eingabe–Ergänzer eine Eingabe X_A erhält, reichert er diese möglicherweise um bestimmte Informationen an und gibt sie in der angereicherten Form als Eingabe X_{ST} an den Kern des Steuerautomaten weiter. Man kann sich dies veranschaulichen, indem man annimmt, die Eingabe X_A bestehe im Einwurf einer beschrifteten Karte, deren Beschriftung der Eingabe–Ergänzer möglicherweise ergänzt, bevor er die Karte als Eingabe X_{ST} in den Einwurfschlitz des Steuerautomatenkerns wirft. Dies sei am Beispiel der Sortiermaschine konkretisiert, wobei die Variante 1 betrachtet wird:

Der Steuerautomat sei im Zustand 2. In diesem Zustand kann als Eingabe X_A nur ein Anstoß über die Drucktaste erfolgen. Als Eingabe X_{ST} für den Steuerautomatenkern reicht aber ein bloßer Anstoß nicht aus, denn der Steuerautomatenkern benötigt weitere Informationen, damit er entscheiden kann, welche der drei Transitionen, die vom Zustand 2 ausgehen – von 2 nach 2, von 2 nach 3 oder von 2 nach 4 – er schalten soll. Diese Information kann der Eingabe–Ergänzer in X_{ST} hineinstecken, denn aus Z_{OP}=(a[1:N], STACK, v, h, i, j) kann er sie wie folgt gewinnen:

$$X_{ST} = \lambda(\text{Anstoß}, Z_{OP}, 2) = \begin{cases} \text{"v < h"} & \text{falls dies zutrifft} \\ \text{"(v} \geq \text{h) UND (STACK} \neq \text{leer)"} & \text{falls dies zutrifft} \\ \text{"(v} \geq \text{h) UND (STACK = leer)"} & \text{falls dies zutrifft} \end{cases}$$

Aus $X_{ST}(n)$ und $Z_{ST}(n)$ bestimmt der Steuerautomatenkern seinen Folgezustand $Z_{ST}(n+1)$ und den Teil seiner Ausgabe, der als $X_{OP}(n)$ an den Operationsautomaten weitergegeben wird. Die Eingabe X_{OP} kann man sich wieder als Einwurf einer beschrifteten Karte vorstellen. Die Beschriftung muß dem Operationsautomaten sagen, wie er seinen Zustand Z_{OP} verändern und welche Ausgabe Y_A er liefern soll. Dazu genügt es, als Beschriftung der Karte X_{OP} die Beschriftung der jeweiligen Transition aus dem Zustandsgraphen zu übernehmen. Wenn man im betrachteten Beispiel annimmt, daß im Zustand 2 ein Anstoß X_A erfolgt und der Eingabe–Ergänzer daraufhin die Eingabe X_{ST}="v < h" für den Steuerautomatenkern liefert, dann muß der Steuerautomatenkern dem Operationsautomaten eine Eingabekarte X_{OP} übergeben, welche die Beschriftung "(i, j, m) := (v, h, $((v+h) - (v+h)_{mod\ 2})/2$)" trägt.

Der Operationsautomat ist ein Schrittrepertoire–Automat,
und der Steuerautomat ist ein Schrittfolgengenerierungsautomat.

Mit den Begriffen Schrittrepertoire–Automat und Schrittfolgengenerierungsautomat sind die folgenden Vorstellungen zu verbinden:

Bei einem *Schrittrepertoire–Automaten* sind die Elemente des Eingaberepertoires repX als Anweisungen zu interpretieren, die dem Automaten sagen, was er in diesem Automatenschritt tun soll. Bei der Konstruktion eines solchen Automaten braucht der Konstrukteur also überhaupt nicht über irgendwelche Schrittfolgen nachzudenken, sondern er braucht nur dafür zu sorgen, daß der Automat in der Lage ist, jede Anweisung aus dem Repertoire auszuführen. Ein Schrittrepertoire–Automat kann somit keine Ziele verfolgen, sondern ist ein sturer Befehlsempfänger, der einen Befehl nach dem anderen ausführt. Nur die befehlende Eingabequelle kann mit der Folge der erteilten Befehle ein Ziel verfolgen.

Daß der Operationsautomat im Beispiel des Sortierungssteuerkreises ein Schrittrepertoire–Automat ist, erkennt man daran, daß man sich die Elemente seines Eingaberepertoires als beschriftete Karten vorstellen kann, wobei auf einer Karte jeweils die Beschriftung einer Transition aus dem Zustandsgraphen des Steuerkreises zu finden ist. Und diese Transitionsbeschriftungen sind tatsächlich als Anwei-

sungen an einen Sklaven zu interpretieren, der Zugang zu den Speicherzellen des Zustandstupels Z_{OP}=(a[1:N], STACK, v, h, i, j) hat.

Bei einem *Schrittfolgengenerierungsautomaten* sind die Elemente des Eingaberepertoires repX als möglicherweise parametrisierte Aufforderungen zu interpretieren, in der Schrittfolgengenerierung den nächsten Schritt zu tun. Bei der Konstruktion eines solchen Automaten muß also der Konstrukteur die zu generierenden Schrittfolgen kennen, d.h. sie müssen ihm als Teil der Spezifikation des zu konstruierenden Automaten vorgegeben werden. Die anschaulichste Form einer solchen Spezifikation ist der Zustandsgraph. Wenn die Elemente des Eingaberepertoires repX nicht parametrisiert werden könnten, d.h. wenn es sich um bloße Anstöße ohne mitgegebene zusätzliche Information handeln würde, dann könnte der Automat nur eine einzige Folge generieren, d.h. dann könnte der Zustandsgraph keine Verzweigungen enthalten. Durch die Parametrisierung hat die Eingabequelle die Möglichkeit vorzuschreiben, in welcher Richtung die Folgengenerierung an einer Verzweigungsstelle fortgesetzt werden soll. Daneben können in die Eingabeparameter aber auch Informationen gesteckt werden, die nicht der Verzweigungsentscheidung im Automaten dienen, sondern die einfach als Ausgabeparameter weitergegeben werden.

Daß der Steuerautomat im Beispiel des Sortierungssteuerkreises ein Schrittfolgengenerierungsautomat ist, erkennt man daran, daß der Zustandsgraph als Spezifikation dieses Automaten vorgegeben werden muß. Jedes Eingabeelement des Automaten ist ein Tupel (X_A, Z_{OP}), wobei der jeweilige Anstoß in X_A enthalten ist, während Z_{OP} immer nur eine Parametrisierung ist. Wenn X_A über die Drucktaste eingegeben wird, ist es nur ein bloßer Anstoß; X_A kann aber auch – wenn der Einwurfschlitz nicht geschlossen ist – als Karte mit der Beschriftung (N, a[1:N]) eingegeben werden, was als parametrisierter Anstoß zu deuten ist. Während die Eingabeparametrisierung durch Z_{OP} jeweils nur zur Verzweigungsentscheidung gebraucht wird, wird die Information (N, a[1:N]) vom Steuerautomaten nicht ausgewertet, sondern wird als Auftragsparameter an den Operationsautomaten weitergereicht.

Extended Finite State Machines

Die Erkenntnisse, die in den voranstehenden Abschnitten dargestellt sind, wurden in zwei getrennten Disziplinen ungefähr zur gleichen Zeit unabhängig voneinander gefunden, nämlich einerseits im Bereich des Hardware–Entwurfs, wo Methoden für den Entwurf komplexer Schaltwerke gesucht wurden [Wendt–70], und andererseits im Bereich des Sprach– und Protokollentwurfs, wo Mittel zur formalen Erfassung von Semantik gesucht wurden [Burstall–69]. Zwar gibt es grundsätzlich schon im Konzept der Turing–Maschine [Turing–36] eine Aufteilung des Zustands in zwei Anteile – den Zustand des endlichen Automaten im Zentrum und den Bandzustand, der durch die Bandbeschriftung und die Bandposition bestimmt ist – , aber die explizite Verallgemeinerung dieser Zweiteilung kam erst viel später.

Das Steuerkreismodell mit dem Begriffspaar (Steuerzustand, Operationszustand) findet man im Bereich der Entwicklung von Hardware–Systemen, während man auf das Begriffspaar (control state, data state) stößt, wenn es um die Modellierung von Software–Systemen geht [Keller–76]. Die Tatsache aber, daß in beiden Fällen vom Steuerzustand oder control state gesprochen wird, zeigt, daß man beidesmal zumindest intuitiv weiß, nach welchem Kriterium das Tupel der Zustandsvariablen des Gesamtsystems aufgeteilt werden soll.

Das Konzept der Zweiteilung des Tupels der Zustandsvariablen in eine ausgezeichnete Variable und die restlichen Variablen findet man in der Literatur seit ungefähr 1985 auch unter der Bezeichnung ”Extended Finite State Machine”, abgekürzt EFSM. Man findet dort das Begriffspaar (explicit states,

variables for implicit states). Allerdings wird dabei nicht mehr auf die Existenz eines klaren Aufteilungskriteriums hingewiesen, sondern man findet sogar die Aussage, daß ein solches Kriterium nicht existiere – beispielsweise in [Ellsb. et al–97], wo auf Seite 9 gesagt wird: "There are no clear rules on the use of explicit states and variables. ... A pragmatic guideline is to use explicit states to show important information as states and to reserve variables for less important information." Aus der Tatsache, daß ihnen ein klares Aufteilungskriterium nicht bekannt war, haben die Autoren der zitierten Aussage den irrigen Schluß gezogen, daß es nicht existiere.

Es sind zwei Arten von Fehlern, die man machen kann, wenn man das Aufteilungskriterium nicht beachtet: Einerseits kann man den Fehler machen, Steuerzustände als implizite Zustände einzuführen, und andererseits kann man den Fehler machen, Operationszustände als explizite Zustände, d.h. als Knoten im Zustandsgraphen darzustellen. In beiden Fällen erschwert man unnötigerweise das Verständnis der Systembeschreibung: Im ersten Fall führt man eine künstliche, d.h. eine nicht intuitiv erfaßbare algebraische Struktur ein anstelle eines Graphen, der unmittelbar das Wesentliche der Struktur zeigen würde, und im zweiten Fall verschleiert man wesentliche algebraische Zusammenhänge, die für das Verständnis des Systems von zentraler Bedeutung sind.

Ich muß gestehen, daß ich selbst schon Fehler der zweiten Art gemacht habe. So habe ich beispielsweise in meinem Buch [Wendt–91] in Bild 98 einen Graphen als Verhaltensmodell eines fahrkartenverkaufenden Automaten dargestellt, ohne darauf hinzuweisen, daß dieser Graph nicht die angemessene Darstellungsform für das betrachtete System ist. Er verschleiert nämlich die Tatsache, daß es hier um das Aufsummieren von Zahlungsbeiträgen geht, die geleistet werden, bis eine Schwelle – der Preis der Fahrkarte – erreicht oder überschritten wird. Die algebraische Formulierung, die auf Seite 3 des vorliegenden Aufsatzes steht, bringt diesen Sachverhalt viel unmittelbarer und damit viel verständlicher zum Ausdruck als der Graph.

Ergänzende Hinweise

In der Automaten– und Schaltwerkstheorie spielen Verfahren zur Reduktion von Zustandsmengen eine große Rolle. Die Frage nach einer möglichen Zustandsreduktion ist aber nur bezüglich der Steuervariablen sinnvoll; eine Operationsvariable kann von vorneherein aus semantischen Gründen nie Gegenstand einer Reduktionsüberlegung sein.

Bezüglich der Operationsvariablen kann es dagegen sinnvoll sein zu fragen, ob sich nicht durch Betrachtung von Tupeln der Codierungsaufwand minimieren läßt. Dies ist immer dann zu fragen, wenn der Wertebereich repT des Tupels nicht gleich dem kartesischen Produkt der Wertebereiche der Tupelkomponenten ist. Es sei hierzu ein zweistelliges Tupel (v_1, v_2) von Operationsvariablen betrachtet, wo zu den Tupelkomponenten die Wertebereiche $repv_1=\{\alpha, \beta\}$ und $repv_2=\{R, S, T\}$ gehören sollen. Es könnte nun sein, daß der Algorithmus, der durch den Steuerautomaten abgewickelt wird, das Vorkommen der Wertekombinationen (α, T) und (β, S) ausschließt. Dann kommt zwar immer noch jeder Wert aus dem Wertebereich einer Tupelkomponente als Belegung vor, aber es gilt

$$\begin{aligned} repT &= \{ (\alpha, R), (\alpha, S), (\beta, R), (\beta, T) \} \subset repv_1 \times repv_2 \\ |repT| &= 4 < |repv_1| \cdot |repv_2| = 2 \cdot 3 = 6 \end{aligned}$$

Würde man die beiden Variablen v_1 und v_2 getrennt binär codieren, bräuchte man für v_1 ein Bit und für v_2 zwei Bit, für beide zusammen also drei Bit. Dagegen genügen für das Tupel (v_1, v_2) zwei Bit. Man sollte sich aber darüber im klaren sein, daß mit einer solchen Reduktion des Codierungsaufwands keine Reduktion der Zustandszahl des Automaten verbunden ist.

Während die Anzahl unterschiedlicher Operationsvariablen in einem System verhältnismäßig hoch sein kann, gibt es immer nur wenige unterschiedliche Steuervariable in einem System, häufig sogar nur eine einzige. Die Operationsvariablen entsprechen den frei deklarierten Variablen eines Programms, während die Steuervariable im Programm dem Befehlszähler entspricht, neben dem es normalerweise nicht noch weitere Befehlszähler gibt. Wenn in einem System mehrere Steuervariable vorkommen, dann ist es immer sinnvoll, dieses System in kommunizierende Teilsysteme zu zerlegen, von denen jedes jeweils nur eine Steuervariable enthält. Daß man auch sehr komplexe Systeme für das menschliche Verständnis gut faßbar darstellen kann, indem man das System als Netz aus kommunizierenden Teilsystemen darstellt, ist inzwischen in sehr vielen Fällen der Praxis demonstriert worden.

Literatur

[Burch et al–90] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, L. J. Hwang:
Symbolic Model Checking: 10^{20} States and Beyond.
Proceedings of Fifth Annual IEEE Symposium on Logic in Computer Science, 1990.

[Burch et al–94] J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, D. L. Dill:
Symbolic Model Checking for Sequential Circuit Verification.
IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems, vol. 13, pp. 401–424; 1994.

[Burstall–69] R. M. Burstall: Formal Description of Program Structure and Semantics in First Order Logic.
Machine Intelligence 5, pp. 79–98 (Eds. Meltzer, B. and Michie, D.), Edinburgh; 1969.

[Ellsb. et al–97] J. Ellsberger, D. Hogrefe, A. Sarma: SDL – Formal Object–oriented Language for Communicating Systems.
Prentice Hall Europe, 1997.

[Glushkov–65] V. M. Glushkov: Automata Theory and Structural Design Problems of Digital Machines.
Cybernetics 1, H. 1, S. 3–9, 1965.

[Keller–76] Robert M. Keller: Formal Verification of Parallel Programs.
Communications of the ACM, Vol. 19, Nr. 7, pp. 371–384, 1976.

[Turing–36] A. M. Turing: On Computable Numbers, with an Application to the Entscheidungsproblem.
Proceedings of the London Mathematical Society, vol. 42; 1936.

[Wendt–70] Siegfried, Wendt: Eine Methode zum Entwurf komplexer Schaltwerke unter Verwendung spezieller Ablaufdiagramme.
Elektronische Rechenanlagen 12, H. 6, S. 314–323, 1970.

[Wendt–91] Siegfried, Wendt: Nichtphysikalische Grundlagen der Informationstechnik.
2. Auflage. Springer–Verlag, Berlin; 1991.