

Physikalische Modellbildung

Um das Verhalten eines beliebigen Systems untersuchen oder gezielt beeinflussen zu können, ist es oftmals erforderlich, ein geeignetes Modell des Systems zur Verfügung zu stellen, anhand dessen das Verhalten des realen Systems analysiert und durch Simulation studiert werden kann. Ziel der Modellbildung ist es daher, eine geeignete mathematische Beschreibungsform zu finden, die benötigten Gleichungen zur Systembeschreibung aufzustellen und die Parameter des Systems (z. B. Massen, Kapazitäten, etc.) zu bestimmen. Vor dem Hintergrund der Vielfalt an zu beschreibenden Systemen und zugehörigen unterschiedlichen mathematischen Beschreibungen kann man Klassen von Modellen unterscheiden. Hierbei gibt es z. B.

- **Statische Systeme**
Systeme, die keine inneren Zustände besitzen und deren Ausgangsgrößen nur von den Eingangsgrößen abhängen.
- **Dynamische Systeme**
Systeme, deren Ausgangsgrößen unter anderem von den inneren Zuständen abhängen.
- **Wertediskrete Systeme**
Systeme, deren Zustände nur Werte aus einer diskreten Menge annehmen, die endlich oder unendlich sein kann.
- **Wertekontinuierliche Systeme**
Systeme, deren Zustände innerhalb der Grenzen des Wertebereichs jeden reellen Wert annehmen können.
- **Zeitdiskrete Systeme**
Systeme, deren Größen nur an diskreten aequidistanten Stellen der Zeitachse betrachtet werden. Zwischen diesen Stellen sind Zustände und Ausgänge des Systems nicht definiert.
- **Zeitkontinuierliche Systeme**
Systeme, für die Zustände und Ausgänge an jedem reellen Zeitpunkt definiert sind.

- Ereignisdiskrete Systeme
Systeme, deren Größen nur an diskreten Stellen der Zeitachse Änderungen erfahren. Die Zeitpunkte werden durch das Auftreten von Ereignissen festgelegt. Einige Definitionen des Begriffs fordern zusätzlich, dass das System wertediskret ist.
- Systeme mit konzentrierten Parametern
Dies sind Systeme, die sich durch gewöhnliche Differentialgleichungen beschreiben lassen, z. B. Systeme mit Punktmassen.
- Systeme mit verteilten Parametern
Hiervon spricht man, wenn partielle Differentialgleichungen zur Systembeschreibung verwendet werden, z. B. bei räumlichen Wärmeleitungsproblemen.
- Lineare Systeme
Dies sind Systeme, bei denen das Superpositions- und Homogenitätsprinzip gilt.
- Nichtlineare Systeme
Systeme, bei denen die o.g. Prinzipien nicht gelten – hierbei sind sehr viele unterschiedliche Erscheinungsformen eingeschlossen.
- Zeitinvariante Systeme
Systeme, deren Parameter und Eigenschaften nicht von der Zeit abhängen.
- Zeitvariante Systeme
System, deren Parameter und Eigenschaften mit der Zeit veränderlich sind, z. B. bei technischen Prozessen Alterung, Drift.
- Deterministische Systeme
System, dessen Verhalten ausgehend von den Anfangsbedingungen und Eingangssignalen eindeutig vorhersagbar ist.
- Nicht-deterministische Systeme
Bei nicht deterministischen Systemen ist das Verhalten auch bei bekannten Anfangsbedingungen und Eingangssignalen nicht eindeutig vorhersagbar.
- ...

Anhand dieser (unvollständigen) Aufzählung erkennt man, wie vielfältig die Unterscheidungsmöglichkeiten sind, wobei bei der detaillierten Beschreibung von realen Prozessen und Systemen immer auch Mischformen einzelner Klassen auftreten können. Das liegt daran, dass Modellbildung immer bedeutet, ein reales System für einen bestimmten Zweck abzubilden. Man kann ein und dasselbe System für einen bestimmten Zweck ereignisdiskret modellieren und für einen anderen auf eine kontinuierliche Beschreibung mittels einer Differentialgleichung zur Wiedergabe der Dynamik zurückgreifen, für eine dritte Anwendung mag die Betrachtung des stationären Verhaltens genügen. In allen drei Beispielfällen ist das Modell aber nur eine grobe Näherung, mit der die für das Modellierungsziel relevanten Verhaltensweisen abgebildet werden.

Im Folgenden wird zunächst eine Beschränkung auf lineare zeitinvariante Systeme mit konzentrierten Parametern vorgenommen, weil diese Systemklasse aufgrund ihrer relativen Schlichtheit in der Regelungstechnik bevorzugt ver-

wendet wird. Zumindest Systeme mit stetigen Nichtlinearitäten können durch Linearisierung in diese überführt werden.

Anschließend erfolgt eine Einführung in die Modellbildung von ereignisdiskreten Systemen. Ein Ausblick auf die Modellierung hybrider Systeme und der Abstraktion zu kontinuierlichen oder ereignisdiskreten Systemen schließt dieses Kapitel ab.

3.1 Kontinuierliche Modellbildung

Lineare zeitinvariante Systeme mit konzentrierten Parametern werden durch lineare, gewöhnliche Differentialgleichungen mit konstanten Koeffizienten beschrieben.

Wie bereits in Kapitel 2 dargestellt, stellt

$$y^{(n)} + \dots + a_2 \ddot{y} + a_1 \dot{y} + a_0 y = b_0 u + b_1 \dot{u} + \dots + b_m u^{(m)}, \quad (3.1)$$

die allgemeine Form für eine solche Differentialgleichung für ein Glied mit der Eingangsgröße u und der Ausgangsgröße y , wobei für kausale Systeme $m \leq n$ gilt.

Unter Verwendung der Laplace-Transformation (vgl. Kapitel 2.3) ist es möglich, die Übertragungsfunktion

$$G(s) = \frac{b_m s^m + \dots + b_1 s + b_0}{s^n + \dots + a_1 s + a_0} = \frac{Y(s)}{U(s)} \quad (3.2)$$

anzugeben, die das Übertragungsverhalten für die laplacetransformierten Signale $Y(s)$ und $U(s)$ beschreibt.

Analog erhält man den Frequenzgang zu

$$G(j\omega) = \frac{b_m (j\omega)^m + \dots + b_1 j\omega + b_0}{(j\omega)^n + \dots + a_1 j\omega + a_0} = \frac{\underline{y}}{\underline{u}} \quad (3.3)$$

als Ausschnitt der Übertragungsfunktion

$$G(j\omega) = G(s)|_{s=j\omega} \quad , \quad (3.4)$$

der das Übertragungsverhalten für harmonische Signale in Betrag und Phase wiedergibt. Die Koeffizienten der jeweiligen Polynome entsprechen denen der Differentialgleichung (3.1).

Wie in Kapitel 2.10 beschrieben kann man das in Gl.(3.1) dargestellte System auch äquivalent im Zustandsraum darstellen als

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix} \cdot \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \cdot u \quad (3.5)$$

$$y = [b_0 - b_n a_0 \quad b_1 - b_n a_1 \quad \dots \quad b_{n-1} - b_n a_{n-1}] \cdot \mathbf{x} + b_n u$$

wobei die Koeffizienten der Matrizen ebenfalls den Koeffizienten der Differentialgleichung entsprechen. Für den Fall $m < n$ sind die Koeffizienten b_{m+1} bis b_n zu null zu setzen. Das Ein- Ausgangsverhalten des Systems wird äquivalent dargestellt.

Aus der Systembeschreibung als Differentialgleichung lassen sich also die in der Regelungstechnik häufig verwendeten Beschreibungsformen leicht gewinnen. Im Folgenden wird erläutert, wie Systembeschreibungen in Form von Differentialgleichungen aus den aus der Physik und anderen Disziplinen bekannten Grundgleichungen aufgestellt werden können.

Alternativ zu dieser analytischen Modellbildung ist es auch möglich, Modelle durch eine Identifikation zu gewinnen; hierauf wird in Kapitel 4 eingegangen.

3.1.1 Aufstellen von Differentialgleichungen

Bei den allermeisten signalübertragenden Anordnungen müssen die Auswirkungen von Speichern für Materie oder Energie berücksichtigt werden. Beim Aufstellen von Differentialgleichungen für komplexe Zusammenhänge empfiehlt sich oftmals ein modulares Vorgehen, etwa

1. Speicher identifizieren und durch geeignete Grundgleichungen beschreiben (Teilsysteme bilden),
2. Verbindungen identifizieren und beschreiben (Zusammenwirken der Teilsysteme beschreiben),
3. Teilsysteme mit Hilfe der Verbindungen zusammenfassen und überflüssige Variable eliminieren.

Bei dem Aufstellen von Zustandsraummodellen werden im Allgemeinen die die Speicher beschreibenden Größen als Zustandsgrößen gewählt und gemäß dem oben angegebenen Vorgehen mit Grundgleichungen von niedriger Ordnung beschrieben. Diese Grundgleichungen können wiederum als ein System von Gleichungen erster Ordnung dargestellt werden und bilden damit Blöcke im Zustandsraummodell. Mit Hilfe der Verbindungsgleichungen werden dann die Kopplungen der Zustände ausgedrückt.

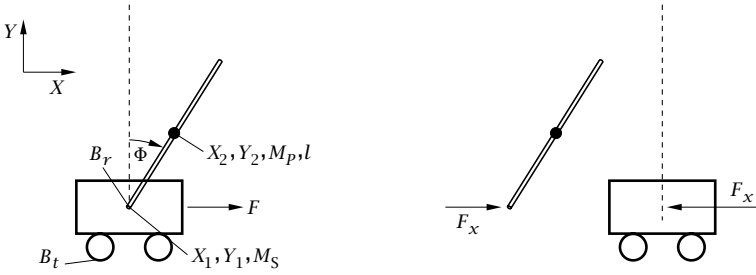
Dieses Vorgehen soll am Beispiel eines in Abb. 3.1 abgebildeten ebenen inversen Pendels, das mit einem Schlitten angetrieben werden kann, dargestellt werden.

Gemäß der oben genannten Vorgehensweise werden zunächst die beiden Elemente (Schlitten und Pendelstab) freigeschnitten; sie stellen die Speicher für kinetische bzw. potentielle Energie dar.

Für den Schlitten gilt nach Newton mit dem Reibungskoeffizient B_t

$$M_s \ddot{X}_1 + B_t \dot{X}_1 = F - F_x \quad . \quad (3.6)$$

Weitere Bewegungsgleichungen sind zur Beschreibung des Schlittens nicht erforderlich, da sein Freiheitsgrad eins beträgt.


Abb. 3.1. Einzelpendel, freigeschnitten

Für die Schwerpunktbewegung des Pendelstabs in X -Richtung gilt

$$M_p \ddot{X}_2 = F_x \quad , \quad (3.7)$$

die Kopplung der beiden Systeme ist durch die Kinematik gegeben

$$X_2 = X_1 + \frac{l}{2} \sin(\Phi) \quad . \quad (3.8)$$

Zweimaliges Ableiten von Gl.(3.8) nach der Zeit und Einsetzen in Gl.(3.7) führt zu

$$F_x = M_p \left(\ddot{X}_1 - \frac{l}{2} \sin(\Phi) \dot{\Phi}^2 + \frac{l}{2} \cos(\Phi) \ddot{\Phi} \right) \quad (3.9)$$

und mit Gl.(3.6) folgt

$$(M_s + M_p) \ddot{X}_1 + B_t \dot{X}_1 + \frac{M_p l}{2} \left(\ddot{\Phi} \cos(\Phi) - \dot{\Phi}^2 \sin(\Phi) \right) = F \quad (3.10)$$

als erste Differentialgleichung, die das System beschreibt. Der Drallsatz für den Pendelstab liefert die Momentenbilanz, mit dem Reibbeiwert B_r gilt

$$J \ddot{\Phi} + B_r \dot{\Phi} = \frac{l}{2} M_p g \sin(\Phi) - \frac{l}{2} M_p \ddot{X}_1 \cos(\Phi) \quad . \quad (3.11)$$

Der letzte Summand in Gl.(3.11) stellt bereits die Kopplung von Schlitten und Pendelstab dar, da die Beschleunigung des Drehpunktes berücksichtigt werden muss. Mit dem Trägheitsmoment

$$J = \frac{1}{3} M_p l^2 \quad (3.12)$$

folgt

$$\frac{2}{3} l \ddot{\Phi} + \frac{2B_r}{M_p l} \dot{\Phi} = g \sin(\Phi) - \ddot{X}_1 \cos(\Phi) \quad (3.13)$$

als zweite Differentialgleichung zur Systembeschreibung. Bei näherer Betrachtung von Gl.(3.10) und Gl.(3.13) wird ersichtlich, dass das Pendel eine nicht-lineare Charakteristik aufweist.

Für die weitere Betrachtung mit dem Ziel der Systemdarstellung als System linearer Differentialgleichungen ist es zweckmäßig, das Pendel in seiner aufrechten Position zu linearisieren. Für diesen Arbeitspunkt gilt dann (zur Darstellung des Kleinsignalverhaltens werden Formelzeichen in Kleinbuchstaben für die Abweichungen von den Arbeitspunktwerten verwendet)

$$(M_s + M_p)\ddot{x}_1 + B_t\dot{x}_1 + \frac{M_p l}{2}\ddot{\varphi} = f \quad (3.14)$$

$$\frac{2}{3}l\ddot{\varphi} + \frac{2B_r}{M_p l}\dot{\varphi} + \ddot{x}_1 - g\varphi = 0 \quad . \quad (3.15)$$

Die beiden gekoppelten Differentialgleichungen zweiter Ordnung können als ein System von vier Differentialgleichungen erster Ordnung dargestellt werden, indem z. B. Gl.(3.15) nach \dot{x}_1 aufgelöst und in Gl.(3.14) eingesetzt wird. Mit Wahl des Zustandsvektors

$$\mathbf{x} = [x_1 \ \varphi \ \dot{x}_1 \ \dot{\varphi}]^T \quad (3.16)$$

ergibt sich die Darstellung im Zustandsraum zu

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{\varphi} \\ \ddot{x}_1 \\ \ddot{\varphi} \end{bmatrix} &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{3gM_p}{4M_s+M_p} & -\frac{4B_t}{M_p+4M_s} & -\frac{6B_r}{l(M_p+4M_s)} \\ 0 & \frac{6g(M_s+M_p)}{l(4M_s+M_p)} & \frac{6B_t}{l(M_p+4M_s)} & \frac{12B_r(M_s+M_p)}{M_p l^2(M_p+4M_s)} \end{bmatrix} \begin{bmatrix} x_1 \\ \varphi \\ \dot{x}_1 \\ \dot{\varphi} \end{bmatrix} \\ &+ \begin{bmatrix} 0 \\ 0 \\ \frac{4}{4M_s+M_p} \\ \frac{-6}{l(4M_s+M_p)} \end{bmatrix} f \end{aligned} \quad (3.17)$$

mit der Messgleichung

$$\mathbf{y} = \begin{bmatrix} x_1 \\ \varphi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x} \quad , \quad (3.18)$$

wenn man beispielsweise Schlittenposition und Pendelwinkel als Ausgangsgrößen betrachten möchte.

Die numerischen Eigenwerte der Systemmatrix \mathbf{A} ergeben sich exemplarisch für die Parameter $M_s = 30\text{kg}$, $M_p = 0,91\text{kg}$, und $l = 0,38\text{m}$ sowie Reibwerten von $B_t = 35 \text{ Nsm}^{-1}$ und $b_2 = 0,002 \text{ Nmsrad}^{-1}$ zu

$$\begin{aligned} \lambda_1 &= 0 \\ \lambda_2 &= 6,31 \\ \lambda_3 &= -6,28 \\ \lambda_4 &= -1,13 \quad , \end{aligned} \quad (3.19)$$

woraus aufgrund eines positiven Eigenwertes die Instabilität der oberen Gleichgewichtslage ersichtlich ist (vgl. Abschnitt 2.2). Bekannterweise kehrt das Pendel bei einer geringen Auslenkung aus der oberen Ruhelage auch nicht in diese zurück.

An dieser Stelle soll auch noch das inverse Doppelpendel vorgestellt werden, das für die Betrachtungen in Kapitel 5 als zu regelndes System Verwendung findet.

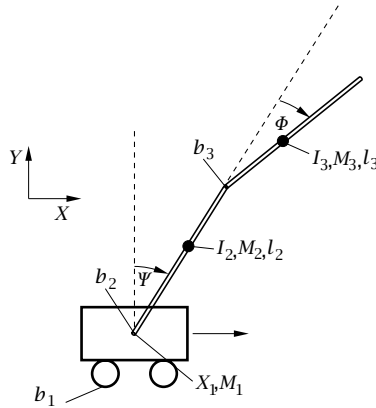


Abb. 3.2. Doppelpendel

Ohne weitere Herleitung werden seine Gleichungen, die man durch Freischneiden oder durch Anwendung der Lagrangeschen Formalismen erhalten kann, angegeben zu

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{Q} \tag{3.20}$$

in den verallgemeinerten Koordinaten

$$\mathbf{q} = [X_1 \ \Psi \ \Phi]^T \quad , \tag{3.21}$$

die die Freiheitsgrade des Systems darstellen. Der Term $\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}}$ repräsentiert die Trägheitskräfte des Systems; daher wird die Matrix \mathbf{H} auch als Trägheitstensor bezeichnet. Sie ist symmetrisch und beinhaltet die Elemente

$$H_{1,1} = M_1 + M_2 + M_3 \tag{3.22}$$

$$H_{1,2} = M_2 l_2 c_{\Phi - \Psi} + M_3 (l_3 c_{\Phi} + a_2 c_{\Phi - \Psi}) \tag{3.23}$$

$$H_{1,3} = M_3 l_3 c_{\Phi} \tag{3.24}$$

$$H_{2,2} = M_2 l_2^2 + I_2 + M_3 (l_3^2 + a_2^2 + 2a_2 l_3 c_{\Psi}) + I_3 \tag{3.25}$$

$$H_{2,3} = M_3 (l_3^2 + a_2 l_3 c_{\Psi}) + I_3 \tag{3.26}$$

$$H_{3,3} = M_3 l_3^2 + I_3 \tag{3.27}$$

mit den Abkürzungen $s_{\Phi+\Psi} = \sin(\Phi+\Psi)$ und $c_{\Phi+\Psi} = \cos(\Phi+\Psi)$, I_i bezeichnen die Trägheitsmomente der Stäbe, l_i die Schwerpunktabstände und a_i die Stabblängen. Im Term $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}})$ sind die Zentrifugal-, Coriolis- und Gewichtskräfte zusammengefasst, es gilt

$$h_1 = -(M_2 l_2 s_{\Phi-\Psi} + M_3 (l_3 s_{\Phi} + a_2 s_{\Phi-\Psi})) (\dot{\Phi} - \dot{\Psi})^2 - 2M_3 l_3 s_{\Phi} (\dot{\Phi} - \dot{\Psi}) \dot{\Psi} - M_3 l_3 s_{\Phi} \dot{\Psi}^2 \quad (3.28)$$

$$h_2 = -2M_3 a_2 l_3 s_{\Psi} (\dot{\Phi} - \dot{\Psi}) \dot{\Psi} - M_3 a_2 l_3 s_{\Psi} \dot{\Psi}^2 - M_2 g l_2 s_{\Phi-\Psi} - M_3 g (l_3 s_{\Phi} + a_2 s_{\Psi-\Psi}) \quad (3.29)$$

$$h_3 = M_3 a_2 l_3 s_{\Psi} (\dot{\Phi} - \dot{\Psi})^2 - M_3 g l_3 s_{\Phi} \quad , \quad (3.30)$$

und der Vektor \mathbf{Q} beinhaltet die äußere Kraft u_A sowie die geschwindigkeitsproportionale Reibung b_i in den Gelenken mit

$$Q_1 = u_A - b_1 \dot{X}_1 \quad (3.31)$$

$$Q_2 = -b_2 (\dot{\Phi} - \dot{\Psi}) \quad (3.32)$$

$$Q_3 = -b_3 \dot{\Psi} \quad . \quad (3.33)$$

Strukturell ist erkennbar, dass Gl.(3.20) ein nichtlineares dynamisches System der Ordnung 6 darstellt. Zur Linearisierung wird erneut die inverse Gleichgewichtslage betrachtet, indem

$$\mathbf{q} = \dot{\mathbf{q}} = 0 \quad (3.34)$$

gewählt wird. Durch Definition des Zustandsvektors zu

$$\mathbf{x} = [x_1 \ \varphi \ \psi \ \dot{x}_1 \ \dot{\varphi} \ \dot{\psi}]^T \quad (3.35)$$

lässt sich das System beschreiben als

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u \quad (3.36)$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} \quad . \quad (3.37)$$

Bestimmt man mit einem positionsgeregelten Schlitten für die gewählten Parameter von $M_2 = 1,2$ kg, $M_3 = 0,63$ kg, $l_2 = 0,326$ m, $l_3 = 0,196$ m, $a_2 = 0,53$ m, $I_2 = 0,0604$ kgm², $I_3 = 0,0163$ kgm², $b_2 = 0,0233$ Nms und $b_3 = 0,0161$ Nms die jeweiligen Koeffizienten, erhält man für die Matrizen

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 27,4482 & -7,5528 & 0 & -0,2524 & 0,1625 \\ 0 & -44,3515 & 42,1121 & 0 & 0,8054 & -0,6601 \end{bmatrix} \quad (3.38)$$

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ -2,0281 \\ 0,2283 \end{bmatrix} \quad (3.39)$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} . \quad (3.40)$$

Die Systemmatrix \mathbf{A} hat folgende Eigenwerte:

$$\begin{aligned} \lambda_1 &= 0 \\ \lambda_2 &= 0 \\ \lambda_3 &= -3,9028 \\ \lambda_4 &= 3,8594 \\ \lambda_5 &= 6,9605 \\ \lambda_6 &= -7,8296 \quad , \end{aligned} \quad (3.41)$$

die auch in diesem Fall auf die Instabilität der oberen Gleichgewichtslage schließen lassen.

Diese Darstellung wird in Kapitel 5 zur Auslegung eines Reglers, der das System in der Gleichgewichtslage (3.34) stabilisieren soll, verwendet.

Anhand dieses Beispiels wird ersichtlich, dass die Beschreibung speziell nichtlinearer Systeme mit Differentialgleichungen relativ aufwändig ist. Eine zweckmäßige Vorgehensweise, die oftmals zu einfachen und übersichtlicheren Beschreibungen führt, ist die Darstellung als Wirkungsplan, in dem häufig eine linearisierte Beschreibung in der Nähe eines Arbeitspunktes durchgeführt wird.

3.1.2 Wirkungsplan

Der Wirkungsplan ist eine schematische Darstellung der Wirkungszusammenhänge innerhalb eines Systems und eignet sich, um die Abhängigkeiten der einzelnen dynamischen Elemente darzustellen. Hierfür werden klare Definitionen und feste Regeln und Bezeichnungen, die in der DIN 19226 festgelegt sind, verwendet. Er beschreibt wirkungsmäßige Zusammenhänge zwischen Größen.

Die Werte der jeweils interessierenden physikalischen Größen werden häufig auch als Signale bezeichnet. Der Wirkungsplan beschreibt daher die in den zugehörigen Geräten, Anlagen usw. stattfindende Signalübertragung. Durch diese Beschränkung auf Fragen der Signalübertragung ist die Regelungstechnik unabhängig von speziellen Eigenschaften des jeweiligen technischen Problems.

Die Elemente des Wirkungsplans stellen gerichtete Operationen zur Veränderung und Verknüpfung von Signalen dar. Diese Operationen und damit auch die Elemente, die sie veranschaulichen, werden als rückwirkungsfrei angesehen. Rückwirkungsfreiheit bedeutet hier, dass Änderungen der Ausgangsgröße eines Elementes keinen Einfluss auf die zugehörige Eingangsgröße haben.

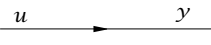
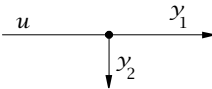
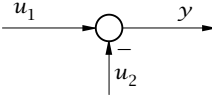
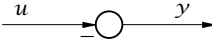
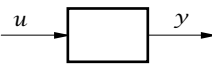
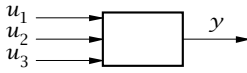
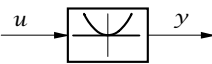
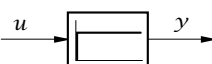

Im Wirkungsplan wird die Übertragung von Signalen durch einfache Linien mit Angaben der Wirkungsrichtung, die Verzweigung von Signalen mit der Verzweigungsstelle, auch Verzweigungspunkt genannt, und die Addition von Signalen unter Beachtung von Vorzeichen durch die Additionsstelle, auch Summenpunkt genannt, dargestellt. Die Übertragung und Verknüpfung von Signalen wird durch Rechtecke, Blöcke genannt, wiedergegeben. Durch zusätzliche Angaben in oder an den Blöcken kann die Übertragung oder Verknüpfung näher bezeichnet werden. Eine Zusammenstellung der Elemente des Wirkungsplans gibt Tabelle 3.1.

Das positive Vorzeichen an Summenpunkten darf i. Allg. fortgelassen werden. In Blöcke wird häufig eine Zeichnung mit der qualitativen Darstellung des Zusammenhangs zwischen Eingangs- und Ausgangsgröße eingetragen. Dies kann eine Kennlinie sein (z. B. $y = K \cdot u^2$ in Tabelle 3.1) oder der zeitliche Verlauf der Ausgangsgröße nach einem Sprung der Eingangsgröße (z. B. für $y = K \cdot u$ und $T \cdot \dot{y} + y = K \cdot u$ in Tabelle 3.1).

Der Wirkungsplan ist eine der wichtigsten Darstellungsformen regelungstechnischer Aufgaben und Lösungen. Nur korrekte und zuverlässige Wirkungspläne führen zu technisch brauchbaren Lösungen. Beim Aufstellen komplizierterer Wirkungspläne ist dringend zu empfehlen, entgegen der Wirkungsrichtung der Größen vorzugehen, d. h. ausgehend von einer Größe nach deren Ursachen zu fragen und diese Antworten festzuhalten. Dadurch kann man leichter sicherstellen, dass alle auf eine Größe wirkenden Einflüsse erfasst werden.

Die Darstellungsform Wirkungsplan verkörpert die grundsätzliche Betrachtungsweise und auch das wesentliche Ziel des Faches Regelungstechnik, nämlich Hilfsmittel bereitzustellen, um dynamische technische Systeme mit komplexer Struktur analysieren, zielgerichtet beeinflussen und auch an deren Gestaltung mitzuwirken zu können. Solche komplexen Strukturen entstehen z. B. in Systemen mit mehreren, auf einander einwirkenden Einflussgrößen oder auch durch interne Rückwirkungen, die oft – aber nicht ausschließlich – auf Regelkreise zurückgehen. Zu Gunsten einer allgemein, d. h. in allen Fachdisziplinen anwendbaren Methodik setzen die entsprechenden Analyse- und Entwurfsverfahren der Regelungstechnik auf einer mathematischen Beschreibung der betrachteten realen Prozesse auf, die von deren spezieller technischer Ausprägung abstrahiert und eine (theoretische und/oder experimentelle) Modellbildung voraussetzt.

Der Wirkungsplan unterstützt die Modellbildung komplexer Systeme, indem eine Zerlegung in Teilsysteme vorgenommen und das daraus resultierende Wirkungsgefüge transparent gemacht wird. Zum Aufstellen von

Bezeichnung	Symbol	Funktion
Wirkungslinie Signalübertragung		$y = u$
Verzweigungsstelle Verzweigungspunkt		$y_1 = u$ $y_2 = u$
Additionsstelle Summenpunkt		$y = u_1 - u_2$
Umkehrpunkt		$y = -u$
Übertragungsblock		$y = f(u)$
		$y = f(u_1, u_2, u_3)$
		$y = K \cdot u^2$
		$y = K \cdot u$
		$T \cdot \dot{y} + y = K \cdot u$

Tab. 3.1. Elemente des Wirkungsplans

Wirkungsplänen, und zwar im Sinne einer damit empfohlenen top-down-Vorgehensweise, kann der folgende Leitfaden aufgestellt werden:

1. Klärung der Eingangs- und Ausgangsgrößen
Diese ergeben sich aus der Aufgabenstellung des zu modellierenden technischen Systems. So gilt, dass eine (ungeregelte) Regelstrecke als Eingangsgrößen die Stell- und Störgrößen und als Ausgangsgrößen die Regelgrößen besitzt. Ein als Wirkungsplan abzubildender Regler wird als Eingangsgrößen jedoch die Regel- und Führungsgrößen und als Ausgangsgrößen die Stellgrößen aufweisen. Schließlich kennt ein geregeltes System (der geschlossene Regelkreis) als Eingangsgrößen die Stör- und Führungsgrößen und als Ausgangsgrößen die Regelgrößen.
2. Zerlegung in Teilsysteme
Bei der Zerlegung eines Gesamtsystems in Teilsysteme wird nach unmittelbaren Ursache-/Wirkungszusammenhängen gesucht, wobei die zuvor empfohlene Methode Anwendung findet, ausgehend von den Ausgangsgrößen des Gesamtsystems (und damit denen des Wirkungsplans) sukzessive rückwärts vorzugehen, bis schließlich nur noch die in Schritt 1 festgelegten Eingangsgrößen als solche auftreten. Das Ziel besteht darin, ein erstes Wirkungsgefüge von Teilsystemen aufzustellen, aus denen ein Überblick über Struktur, Dynamik und Vorzeichen der Wirkzusammenhänge hervorgeht. Dabei sei angemerkt, dass die im Wirkungsplan vorzusehenden Vorzeichen zur Vermeidung von Doppeldeutigkeiten stets am Summenpunkt, und zwar in Pfeilrichtung rechts vom Pfeil anzutragen sind (nach DIN 19226).
3. Übertragungsverhalten der Teilsysteme
Für viele der in Schritt 2 definierten Teilsysteme wird das im Wirkungsplan abzubildende dynamische Übertragungsverhalten unmittelbar aus der technischen Ausführung des zu modellierenden technischen Systems zu entnehmen sein. Bei den übrigen, weniger trivialen Teilsystemen hilft die Aufstellung der das Übertragungsverhalten beschreibenden Differentialgleichungen weiter, die aus den formalen Beschreibungen der betreffenden Fachdisziplinen hervorgehen (z. B. Energieerhaltungssätze, Gleichgewichtsbeziehungen von Kräften und Drehmomenten, Bewegungsgleichungen, Wärmeübergang und -speicherung, elektrische Netzwerke, Strömungsvorgänge, chemische Reaktionen, etc.).

Als kurzes Beispiel wird erneut das Einzelpendel in seiner oberen Gleichgewichtslage betrachtet. Zunächst werden nach obiger Vorgehensweise die Eingangs- und Ausgangsgrößen definiert; als Ausgangsgrößen mögen die Position x_1 des Schlittens und der Pendelwinkel φ als Abweichungen von der oberen Gleichgewichtslage dienen, als Eingangsgröße wird die vom Antrieb auf den Schlitten ausgeübte Kraft f gewählt.

Die Zerlegung des Systems in Teilsysteme führt – wie bereits bei der Aufstellung der nichtlinearen Differentialgleichungen – zu der Beschreibung von

Schlitten und Pendel. Diese wird jedoch aufgrund der Beschreibung des Systems in der Nähe des Arbeitspunkts durch implizit linearisierte Gleichungen durchgeführt, wobei der Schwerpunkt auf der Bestimmung von Ursachen und Wirkungen liegt.

Das Übertragungsverhalten wird erneut durch mechanische Grundgleichungen beschrieben, wobei nach den Ursachen für die Bewegungen von x_1 und φ gefragt werden muss. Die Position des Schlittens ergibt sich durch zweifache Integration aus seiner Beschleunigung; diese wird von den auf den Schlitten einwirkenden Kräften verursacht. Als näherungsweise Beschreibung gilt mit Abb. 3.1

$$M_1 \ddot{x}_1 = \sum_i f_i = f - f_x - B_t \dot{x}_1 \quad . \quad (3.42)$$

Für den Arbeitspunkt ergibt sich hierbei durch Betrachtung von Gl.(3.9)

$$f_x = M_p \ddot{x}_1 + \frac{1}{2} M_p l \ddot{\varphi} \quad . \quad (3.43)$$

Für die Drehbewegung des Pendelwinkels findet man analog angreifende Momente als Ursachen, es gilt

$$J \ddot{\varphi} = \sum_i m_i = \frac{1}{2} l M_p g \varphi - \frac{1}{2} l M_p \ddot{x}_1 - B_r \dot{\varphi} \quad . \quad (3.44)$$

Die Ausgangsgrößen werden also durch zweifache Integration aus den Beschleunigungen gewonnen. Die zugehörigen Gleichungen sind an den Summenpunkten ablesbar; als Eingangsgröße für das System bleibt die Kraft f , die auf den Schlitten wirkt, übrig.

3.1.3 Modularisierte Umsetzung in Simulink

SIMULINK erlaubt als grafische Programmierumgebung die wirkungsplanorientierte Programmierung von Simulationen dynamischer Systeme mit MATLAB. Hierzu ist es möglich, aus unterschiedlichen Bibliotheken eine Reihe von vordefinierten und parametrierbaren signalübertragenden Blöcken auszuwählen und über die jeweiligen Ein- und Ausgangssignale zu verbinden. Hierbei beginnt ein Signal an einer Signalquelle, kann verzweigt und durch mehrere Blöcke hindurch übertragen werden und endet in einer Signalsenke. Es können auch eigene Blöcke erstellt und in eigenen Bibliotheken gesammelt werden.

Häufig verwendete Standard-Bibliotheken enthalten u.a. folgende Blöcke:

- Continuous
Übertragungsfunktion, Differenzierer, Integrierer, Zustandsraummodell, ...
- Discontinuous
Tote Zone, Begrenzer, Quantisierer, Hysterese, ...

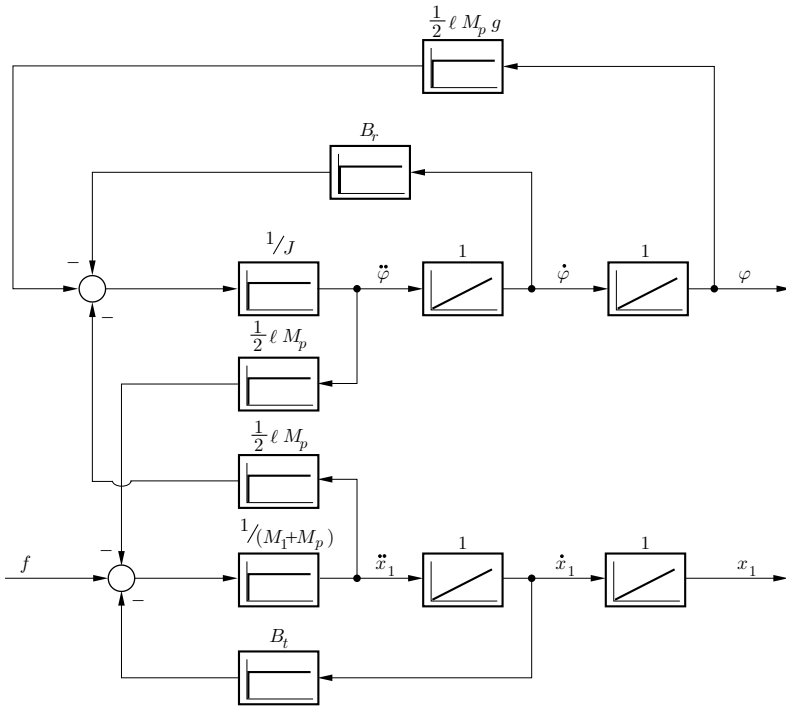


Abb. 3.3. Wirkungsplan des Einzelpendels

- Discrete
Zeitdiskrete Übertragungsfunktion, zeitdiskreter Integrierer, Halteglied, ...
- Math Operations
Verstärkung, Produkt, Summe, trigonometrische Funktionen, ...
- Sinks
Anzeige, Oszilloskop, Speicher, ...
- Sources
Sprung, Signalgenerator, Rampe, Zufallszahl, ...

Neben den originär zu SIMULINK gehörenden Bibliotheken gibt es noch eine Reihe von Erweiterungen, sog. *blocksets*, die zusätzliche anwendungsspezifische Blöcke zur Verfügung stellen, die z. B. die Funktionalität von Toolboxen in SIMULINK integrieren.

Jeder Block besteht aus einem oder mehreren Eingängen u , inneren Zuständen x und einem oder mehreren Ausgängen y .

Der Ausgang eines Blocks wird als Funktion von Eingang, Zuständen und der Zeit dargestellt. Die Zustände beinhalten hierbei möglicherweise die „Vergangenheit“ des Blocks, also z. B. vergangene Eingangssignale. Zustände werden daher benötigt, um innerhalb eines Blocks Dynamik abzubilden; Blöcke ohne Zustände und damit ohne Dynamik sind z. B. Summen- oder Produkt-

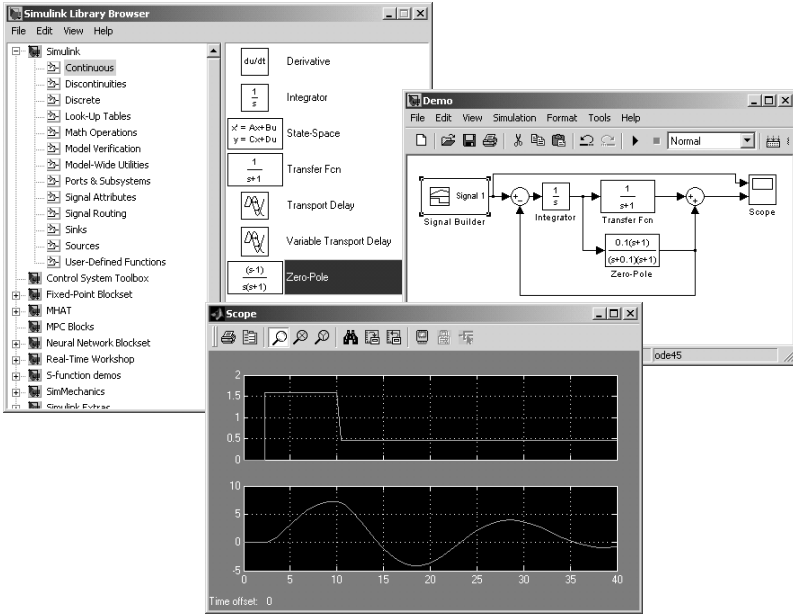


Abb. 3.4. Wirkungsplanähnliche Darstellung in SIMULINK

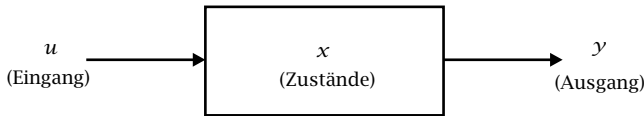


Abb. 3.5. Allgemeiner Block in SIMULINK

blöcke, da ihr Ausgangssignal unmittelbar aus den anliegenden Eingangssignalen verzugsfrei errechnet werden kann.

In Anlehnung an die Zustandsraumdarstellung eines allgemeinen kontinuierlichen dynamischen Systems

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{y} &= \mathbf{g}(\mathbf{x}, \mathbf{u}) \end{aligned} \tag{3.45}$$

bzw. in der zeitdiskreten Darstellung

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{y}_k &= \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) \end{aligned} \tag{3.46}$$

wird der Block über die Implementierung von Systemfunktionen zur Berechnung von

- Ausgänge y
- Zustandsableitungen \dot{x} bzw.
- neuen diskreten Zustände x_{k+1}

in seinem Verhalten bestimmt.

Die Aufrufe der jeweiligen Gleichungen werden von dem SIMULINK-Solver koordiniert; hierdurch wird das dynamische System über der Zeit integriert. Hier hat der Benutzer eine Eingriffsmöglichkeit in der Auswahl des verwendeten Lösers (Einschritt- oder Mehrschrittverfahren mit fester oder variabler Schrittweite) und Vorgabe von ggf. Schrittweitengrenzen bzw. zulässigen Fehlern (relative und absolute Toleranzen). Hierauf wird in Kapitel 6 näher eingegangen. Neben der Simulation von kontinuierlichen Systemen lassen sich auch zeitdiskrete und gemischt kontinuierliche und zeitdiskrete Systeme modellieren und simulieren.

Um eine übersichtliche Modellstruktur durch eine Hierarchiebildung erzielen zu können, können einzelne Blöcke und sie verbindende Signale zu sog. *subsystems* zusammengefasst werden. Neben dem Vorteil, dass sich die Anzahl der im Simulationsfenster dargestellten Funktionsblöcke reduziert, können funktionell verwandte Blöcke sinnvoll zu Teilsystemen zusammengefasst werden. Diese Teilsysteme sind maskierbar, d. h. die in dem Teilsystem vorhandenen Parameter können über eine Maske eingegeben werden. Simulationsmodelle können somit in hierarchischen Strukturen mit beliebig tief verschachtelten Ebenen entworfen werden.

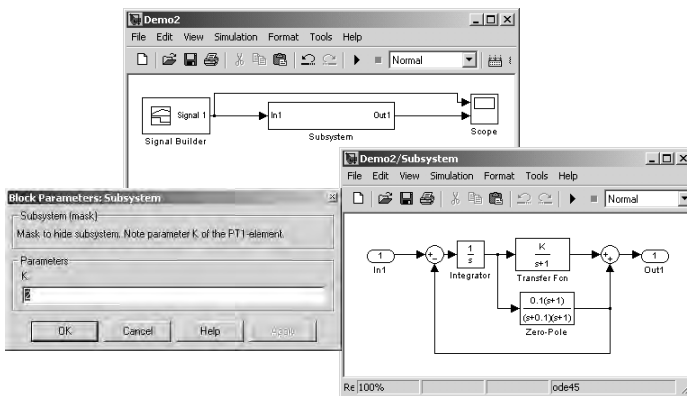


Abb. 3.6. Maskiertes Subsystem in SIMULINK

Für eine tiefere Beschreibung und technische Details zum Arbeiten mit SIMULINK sei an dieser Stelle auf die SIMULINK-Dokumentation und [53] verwiesen.

Auf ein paar für die praktische Arbeit hilfreiche Methodiken soll aber noch kurz hingewiesen werden.

- Interaktion mit MATLAB

Die Interaktionsmöglichkeiten von MATLAB und SIMULINK sind vielfältig. Sie beziehen sich zum einen auf den direkten Datenaustausch für die Simulation und zum anderen auf verwendete Funktionen. Datenaustausch bezieht sich hierbei einerseits auf die Parametrierung von SIMULINK-Blöcken durch Variablen, die im Matlab-Workspace definiert sind und andererseits auf die Möglichkeit, Eingangsdaten für eine SIMULINK-Simulation aus dem MATLAB-Workspace einzulesen bzw. Simulationsergebnisse in den MATLAB-Workspace zu schreiben, so dass sie anschliessend dort weiterverarbeitet werden können (Signalanalyse, grafische Darstellung etc.). Neben den in den SIMULINK-Bibliotheken definierten und parametrierbaren Blöcken gibt es noch zwei Möglichkeiten, den Funktionsumfang von SIMULINK zu erweitern. Ein einfacher Weg besteht in dem direkten Aufruf von MATLAB-Funktionen mit Argumenten, die aus SIMULINK-Signalen bestehen; hierüber kann der MATLAB-Funktionsumfang eingebunden werden. Der noch flexiblere Weg besteht in der Möglichkeit, eigene beliebige, ggf. dynamische Systeme definieren zu können, die dann von dem SIMULINK-Solver – wie das restliche dargestellte System auch – integriert wird. Diese Funktionen werden als S-Funktionen bezeichnet.

- Schreiben von S-Funktionen

Ausgehend von der allgemeinen Beschreibung eines dynamischen Systems nach Gl.(3.45) oder Gl.(3.46), bzw. in Kombination beider Darstellungen auch einer hybrid kontinuierlich-zeitdiskreten Darstellung wird für S-Funktionen ein Gerüst mit geeigneten Funktionsrümpfen zur Verfügung gestellt, das die Interaktion mit dem SIMULINK-Solver abwickelt. Hierdurch ist es möglich, Funktionen in das SIMULINK-Modell zu integrieren, die z. B. aufgrund ihrer algorithmischen Struktur für eine grafische Implementierung schlecht geeignet sind. Die Funktionen können über einen eigenen Speicherbereich verfügen, z. B. in Form des Zustandsvektors \mathbf{x} . Durch die Möglichkeit, diese Funktionen in C, C++, Fortran, als M-file oder in ADA zu implementieren, können einerseits Geschwindigkeitsvorteile entstehen, andererseits ist es auch möglich, Teile einer Modellierung, die in dieser Form implementiert sind, zu schützen.

- Erzeugen von ausführbarem Code aus einem SIMULINK-Modell

Ebenso wie einzelne Komponenten in Form von S-Funktionen implementiert werden können, ist es mit dem Real-Time-Workshop möglich, aus z. B. Subsystemen oder auch einem ganzen Simulationsmodell durch Code-Generierung (vgl. Kapitel 7.4) ein geschütztes Modell zu machen, was für unterschiedliche Zielsysteme kompiliert werden kann. Hierdurch kann ein batch-Betrieb z. B. für Parameterstudien (automatisierter Simulationsaufruf mit Parametervorgaben und Speichern der jeweiligen Ergebnisse) schneller ausführbar sein.

Nach dieser Einführung in die Modellbildung und rechnergeeignete Darstellung von kontinuierlichen dynamischen Systemen werden in den folgenden

Abschnitten mit ereignisdiskreten und hybriden Systemen weitere häufig vorkommende Systemklassen und rechnergeeignete Darstellungen vorgestellt.

3.2 Ereignisdiskrete Modellbildung

Neben der Regelung und Steuerung kontinuierlicher Systeme erfordert die Automatisierung technischer Systeme oftmals auch die Führung von Vorgängen, denen ein vorwiegend schrittweiser, d. h. diskreter Ablauf zugrunde liegt, vgl. Abb. 3.7.

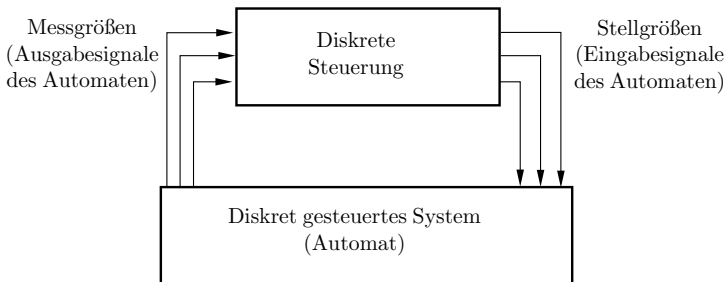


Abb. 3.7. Führung eines Systems durch eine diskrete Steuerung [16]

Im folgenden Abschnitt werden daher Möglichkeiten aufgezeigt, ereignisdiskrete Systeme¹ mit verschiedenen Beschreibungsmitteln und hier insbesondere den Zustandsgraphen, den *Statecharts* und mit Petrinetzen zu beschreiben. Der Schwerpunkt liegt im Rahmen dieses Buches auf den *Statecharts*, die auf D. Harel im Jahr 1987 zurückgehen [55].

3.2.1 Eigenschaften von Beschreibungsmitteln

Beschreibungsmittel zur Modellierung des Verhaltens und der Struktur einer automatisierungstechnischen Einrichtung sind textuelle, mathematisch symbolische oder grafische Formalismen oder Methoden. Beschreibungsmittel können auf die *Steuerstrecke* (das zu automatisierende System), auf die Steuereinrichtung und den Steueralgorithmus, oder auf das *Gesamtsystem*, auch Steuerung, angewandt werden. Der Begriff *System* wird in vielen Kontexten auch für das Gesamtsystem verwendet. Der Einsatzbereich dieser Beschreibungsmittel reicht von der Produktidee über die Spezifikation, den Entwurf, die Implementierung bis hin zur Anlagen- bzw. Produktdokumentation und zu Wartungshandbüchern. Die weiteren Ausführungen beschränken sich hierbei

¹ Discrete event (dynamic) systems (DES/DEDS)

auf Beschreibungsmittel, mit denen das Verhalten ereignisdiskreter Systeme modelliert werden kann.

Das Modell eines diskreten Systems soll *näherungsweise* beschreiben, wie sich Zustände und Ausgänge des Systems in Abhängigkeit von Anfangszustand und Eingängen ändern. Bei den meisten Beschreibungsmitteln wird hierbei den Zeitpunkten keine Aufmerksamkeit gewidmet, an denen sich die Zustände ändern und die Aktivierung einzelner Zustände wechselt. Stattdessen wird der Verlauf der Zustände als *logische Zustandsfolge* aufgefasst, wobei Zustandswechsel nur beim Auftritt von (bestimmten) Ereignissen stattfinden können. Deshalb ist die Folge der nacheinander erreichten Zustände, beginnend beim Anfangszustand, genau so gut durch die Folge der Ereignisse darzustellen, welche zu den jeweiligen Zustandsänderungen geführt haben.

Mit der Reduktion der diskreten Dynamik auf logische Zustandsfolgen und dem bewussten Verlust der Zeitinformation stellt sich die Frage, ob hiermit nicht ein bedeutsamer Teil des Systemverhaltens ignoriert wird. Die Antwort hierauf ist eng verknüpft mit der Entscheidung, ob ein System überhaupt sinnvoll diskret modelliert werden kann. Für Systeme, die diskrete Aspekte besitzen, aber trotzdem zeitbehaftet modelliert werden müssen, sei auf die hybride Modellbildung (Abschnitt 3.3) verwiesen.

Die *Repräsentationsform* ist ein wichtiges Kriterium für die Verständlichkeit und die Anschaulichkeit der Modelle. Es wird zwischen drei grundsätzlichen Arten der Darstellung unterschieden, die aber praktisch oft kombiniert werden. Das Verhalten des Systems kann erstens mit alphanumerischen Zeichen sprachlich-textuell wiedergegeben werden. Zum zweiten ist es möglich, das Systemverhalten mittels mathematisch-symbolischer Repräsentationsformen z. B. mittels Matrizenalgebra auszudrücken. Als dritte Variante verbleibt die Beschreibung anhand einer Grafik, die aus einzelnen Elementen und den Beziehungen zwischen diesen besteht.

Ereignisdiskrete Systembeschreibungen ermöglichen vielfach auch die Wiedergabe von nicht-deterministischem Verhalten. Systeme mit dieser Eigenschaft können auch bei gleichen Anfangsbedingungen unterschiedliches Verhalten aufweisen, gekennzeichnet durch die folgenden Systemzustände und Ausgangssignale. Insbesondere für die Analyse von ereignisdiskreten Systemen ist diese Möglichkeit von Interesse. Da kleine Fehler bei der Modellierung – bedingt durch die Diskretheit der Zustände – zu komplett unterschiedlichem Verhalten zwischen Modellsystem und realem System führen können und da kleine Fehler bei der Festlegung der Anfangsbedingungen des Systems ebenso zu gänzlich anderem Systemverhalten führen können, ist die nicht-deterministische Modellierung oft von Vorteil, um mehrere oder alle möglichen Verhaltensweisen im Modellsystem zu berücksichtigen.

Ein Beispiel für nicht-deterministisches Verhalten zeigt der Automatengraph Abb. 3.8a, in dem im Gegensatz zu Abb. 3.8b zwei verschiedene Zustände auf Zustand A folgen können. Welcher der beiden Zustände B oder C anschließend erreicht wird, hängt weder von den Anfangsbedingungen noch von den bekannten Eingangsgrößen des Systems ab. Mit einer



Abb. 3.8. Deterministischer und nicht-deterministischer Automatengraph

nicht-deterministischen Systembeschreibung ist es möglich, dieses Verhalten zu berücksichtigen.

Synchronisationsmechanismen beschreiben, wie Teilsysteme und Teilprozesse verkoppelt sind. Synchrone Teilprozesse sind getaktete Prozesse, die durch ein Taktsignal gestartet gleichzeitig ablaufen. Alle Komponenten führen gleichzeitig ihre Übergänge durch, meistens auf der Grundlage eines Taktes. Asynchrone Teilprozesse sind Prozesse ohne festen gemeinsamen Takt, gleichzeitige Übergänge müssen besonders gekennzeichnet werden. Konkurrierende Teilsysteme und deren Einzelschritte heißen *nebenläufig*, wenn sie voneinander kausal unabhängig durchgeführt werden – ob das gleichzeitig oder parallel geschieht, bleibt bei nebenläufigen Systemen offen.

Viele Beschreibungsmittel besitzen die Fähigkeit, *Hierarchien* abzubilden. In hierarchischen Systemen existieren über- bzw. untergeordnete Strukturen. Wird von einem Bottom-Up-Entwurf ausgegangen, in dem einzelne Teilsysteme mit einem Beschreibungsmittel dargestellt und anschließend zu einem Gesamtsystem zusammengesetzt werden, so spricht man von Komposition. Hingegen wird die Unterstützung des Top-Down-Entwurfs als Fähigkeit zur Dekomposition bezeichnet, da hier einzelne Elemente einer Ebene mit zusätzlichen Funktionalitäten und Eigenschaften verfeinert werden.

3.2.2 Graphen

Die Graphentheorie geht auf Leonhard Euler zurück [65]. Angeblich fragten sich die Bürger von Königsberg, dem heutigen Kaliningrad, ob es möglich sei, einen Weg über die sieben Brücken der Pregel zu finden, ohne eine der Brücken ein zweites Mal zu passieren. Euler formulierte dieses Problem wie folgt:

„Das Problem, das, wie ich glaube, wohlbekannt ist, lautet so: In der Stadt Königsberg in Preußen gibt es eine Insel A, genannt ‚Kneiphof‘, die von zwei Armen des Flusses Pregel umgeben ist (vgl. Abb. 3.9). Es gibt sieben Brücken a, b, c, d, e, f und g, die den Fluss an verschiedenen Stellen überqueren. Die Frage ist, ob man einen Spaziergang so planen kann, dass man jede dieser Brücken genau einmal passiert. Wie man mir erzählte, behaupten einige, dass

dies unmöglich sei; andere seien in Zweifel, und es gäbe niemanden, der von der Lösbarkeit überzeugt sei. Ausgehend von dieser Situation stellte ich mir das folgende Problem: Gegeben sei irgendeine Konfiguration des Flusses und der Gebiete, in die er das Land aufteilen möge sowie irgendeine Anordnung von Brücken. Man entscheide, ob es möglich ist, jede Brücke genau einmal zu überschreiten, oder nicht.“

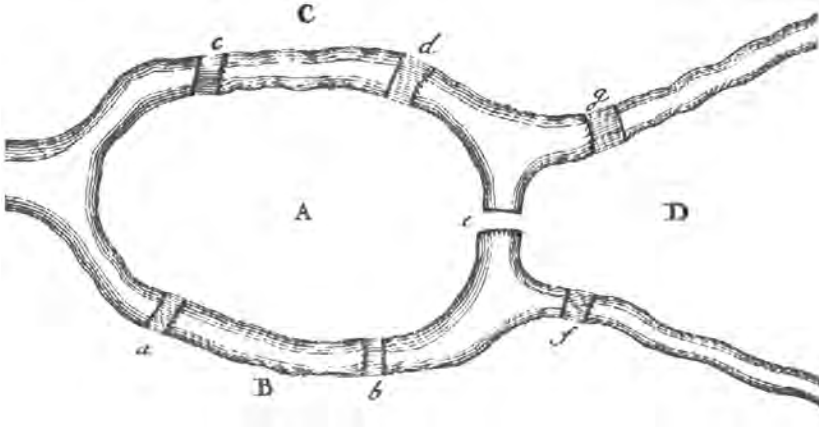


Abb. 3.9. Das Problem der sieben Brücken von Königsberg

Eine abstraktere Darstellung des Brückenproblems findet sich in Abb. 3.10. Die vier Landstücke A, B, C und D werden hier durch Punkte repräsentiert, welche durch Linien verbunden werden. Für jede Brücke, die es zwischen den Landstücken gibt, werden die entsprechenden Punkte mit genau einer Linie verbunden. Weiterhin werden die Linien mit den Buchstaben a bis f beschriftet.

Viele grafische Beschreibungsmittel nutzen solche *Graphen* zur Darstellung. Ein Graph G wird formal definiert als 2-Tupel (Paar) $G = (N, E)$. Die Menge N enthält die sogenannten *Knoten* (oder Ecken) des Graphen G , die Menge E enthält die Kanten. Jeder Kante e aus E werden hierbei zwei Knoten N_i und N_j aus N zugeordnet, die durch die Kante verbunden werden: $e = (N_i, N_j)$ mit $e \in N \times N$.

Der Graph G des Brückenproblems kann also folgendermaßen beschrieben werden:

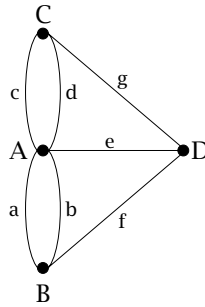


Abb. 3.10. Graph zum Brückenproblem in Abb. 3.9

$$\begin{aligned}
 G &= (N, E) \quad \text{mit} \\
 N &= \{A, B, C, D\}, \\
 E &= \{a, b, c, d, e, f, g\} \quad \text{und} \\
 a &= (A, B), b = (A, B), c = (A, C), \\
 d &= (A, C), e = (A, D), f = (B, D), g = (C, D).
 \end{aligned}
 \tag{3.47}$$

Graphen kann man bildlich darstellen, indem die Knoten als Punkte, Kreise, Rechtecke o.ä. repräsentiert werden. Werden zwei Knoten des Graphen durch eine Kante verbunden, zeichnet man eine Linie (Gerade, Kurve, ...) zwischen den zugehörigen Punkten, Kreisen etc.

Für den Graphen in Abb. 3.10 konnte Euler zeigen, dass es keine Möglichkeit gibt, einen der Aufgabenstellung entsprechenden Weg zu finden. Er bewies auch den allgemeinen Fall, dass es für eine beliebige Konfiguration von Landstücken und Brücken genau dann eine oder mehrere Lösungen gibt, wenn alle bis auf zwei Knoten einer geraden Anzahl von Kanten zugeordnet sind. Gehorchen bis zu zwei Knoten dieser Bedingung nicht, müssen diese als Anfangs- und Endpunkt eines sogenannten *eulerschen Weges* gewählt werden. Gehorchen alle Knoten der Bedingung einer geraden Zahl angrenzender Kanten – die Zahl angrenzender Kanten nennt man den Grad eines Knoten –, heißt der Graph *eulersch* oder *eulerscher Kreis*. Ein Spaziergang auf einem eulerschen Kreis kann auf jedem beliebigen Knoten begonnen werden und endet ebenfalls auf diesem Knoten.

Ein Beweis, dass man auf einem zusammenhängenden Graph G_0 mit Knoten geraden Grades einen solchen Spaziergang unternehmen kann, lässt sich mit folgender Idee führen:

- Jeder Knoten in G_0 besitzt geraden Grad. Also existiert mindestens ein Kreis in G_0 .
- Man wähle einen Kreis C_0 . Falls in diesem Kreis bereits alle Kanten enthalten sind, ist der Graph G_0 ein Eulerkreis. Anderenfalls setze man $i = 1$.

- Sonst bilde man einen neuen Graphen G_i , indem man die Kanten in C_{i-1} von G_{i-1} abzieht. Knoten mit Grad Null werden aus G_i entfernt. Da G_0 zusammenhängend ist, besitzen G_i und C_{i-1} mindestens einen gemeinsamen Knoten N_i . Da beim Entfernen der Kanten eines Kreises der Grad jedes Knoten um Vielfache von 2 sinkt, da immer zwei Kanten des Weges an einen Knoten grenzen, besitzt jeder Knoten in G_i geraden Grad. Also gibt es einen Kreis C' in G_i , in dem N_i enthalten ist. Man wähle C_i zu C_{i-1} vereint mit C' . Falls in C_i alle Kanten des Graphen G_0 enthalten sind, ist dieser ein Eulerkreis. Anderenfalls erhöhe man i um 1 und wiederhole diesen Absatz.

Da die Anzahl der Landstücke und Brücken endlich ist, wird der oben angegebene Algorithmus terminieren und C_i ist gleich G_0 , während G_i keine Kanten und Knoten mehr enthält. Für einen zusammenhängenden ungerichteten Graphen sind die folgenden drei Aussagen äquivalent:

- An jeden Knoten grenzt eine gerade Anzahl von Kanten.
- Die Kantenmenge des Graphen lässt sich in Kreise zerlegen.
- Der Graph ist eulersch.

Besitzt ein zusammenhängender Graph nur Knoten geraden Grades und zwei Knoten mit ungeradem Grad, so muss man den Spaziergang an einem dieser beiden Knoten beginnen und am anderen beenden. Die Idee für einen Beweis hierzu lautet, die beiden Knoten mit einer Hilfskante zu verbinden, um so aus dem Graphen einen eulerschen Kreis zu machen – alle Knoten besitzen nun geraden Grad. Für diesen existiert ein Kreis C_0 , der an einem der beiden Knoten ungeraden Grades mit der Hilfskante beginnt, die zum zweiten Knoten ungeraden Grades führt. Entfernt man aus dem mit obigem Algorithmus gefundenen Weg C_i mit der ersten Kante die Hilfskante, verbleibt ein eulerscher Weg, der auf dem gewählten Knoten endet und auf dem zweiten Knoten beginnt.

In der Automatisierungstechnik werden vielfach *gerichtete* Graphen² verwendet. Hier werden die beiden Knoten N , die einer Kante E zugeordnet sind, in Ursprungs- und Zielknoten unterschieden. Typischerweise schreibt man für eine gerichtete Kante vom Knoten N_i zum Knoten N_j den Ausdruck (N_i, N_j) . Gerichtete Kanten werden i. d. R. durch (Einfach-) Pfeile dargestellt, mit der Pfeilspitze auf den Zielknoten zeigend. In Abb. 3.11 wird deutlich, dass sehr unterschiedliche Darstellungsarten von gerichteten Graphen existieren. In vielen Beschreibungsmitteln können den grafischen Elementen – Knoten und Kanten – des Graphen weitere Eigenschaften zugeschrieben werden. Sehr oft passiert das durch textuelle Annotationen, die Labels genannt werden. Diesbezüglich zeigt Abb. 3.11c, dass Knoten (oder auch Kanten) auch innerhalb eines Graphen mit unterschiedlichen Symbolen versehen werden können. Häufig wird das genutzt, um grundsätzlich unterschiedliche Bedeutungen der Knoten wiederzugeben.

² Directed Graph oder *Digraph*.

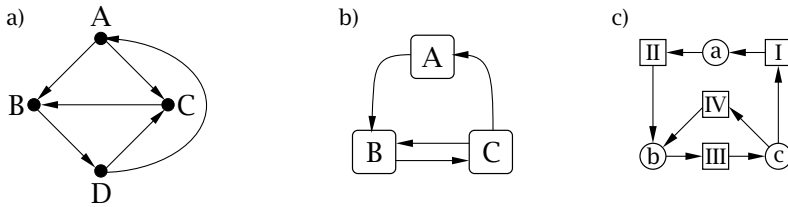


Abb. 3.11. Verschiedene gebräuchliche Darstellungsarten gerichteter Graphen

3.2.3 Statecharts (Harel-Graphen)

David Harel [4] entwickelte im Rahmen einer Beratertätigkeit für die Forschungs- und Entwicklungsabteilung der Israel Aircraft Industries eine grafische Notation für die Modellierung des Verhaltens komplexer Systeme mit Nebenläufigkeiten, den Formalismus der Statecharts. Der Grund bestand darin, dass es für viele spezielle Arten von Systemen einen Entwurfsmechanismus gab, komplexe Systeme jedoch mit bekannten Methoden nicht zugänglich waren. Ausgehend von den Automaten als eine Möglichkeit zur Modellierung von diskreten Systemen, wird im Folgenden die Idee der Statecharts und ihre Möglichkeiten erläutert.

Automaten

Die hier behandelten Systeme und deren Steuerungen können als Schaltwerke oder *Automaten* aufgefasst werden. Jedes diskrete technische System enthält nur endlich viele verschiedene Zustände und stellt somit einen endlichen Automaten $A = (X, Y, Z, F, G)$ dar, der durch

- eine endliche Eingabemenge $X = \{x_1, x_2, \dots\}$ (Menge der möglichen Eingaben),
- eine endliche Ausgabemenge $Y = \{y_1, y_2, \dots\}$ (Menge der möglichen Ausgaben),
- eine endliche Zustandsmenge Z (Menge der möglichen Zustände),
- eine Zustandsübergangsfunktion $Z_{neu} = F(X, Z_{alt})$ und
- eine Ausgabefunktion $Y = G(X, Z)$

charakterisiert wird. Ein Eingangssignal $x \in X$ löst in dem Automaten ein bestimmtes Ereignis aus, sofern die Schaltbedingung dafür vorliegt. Durch Eintritt dieses Ereignisses wird in dem Automaten der Übergang in einen neuen Zustand $z \in Z$ und gegebenenfalls die Ausgabe eines Signals $y \in Y$ vollzogen, was durch die Zustandsübergangsfunktion F festgelegt ist. Ein Automat beschreibt damit die zeitliche Ordnung von Zustandsübergängen in dem betrachteten System. Bei endlichen Automaten ist die Zahl der Zustände

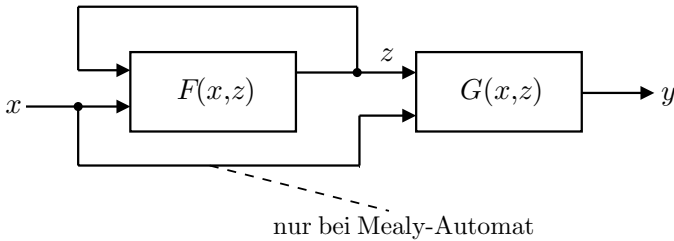


Abb. 3.12. Blockstruktur eines Automaten

begrenzt und das Verhalten wird durch die Zustandsübergangsfunktion F und die Ausgabefunktion G beschrieben.

Man unterscheidet:

- Mealy-Automat:
Die Ausgaben hängen von den momentanen Zuständen und den aktuellen Eingaben ab: $Y = G(X, Z)$
- Moore-Automat:
Die Ausgaben hängen nur von den momentanen Zuständen ab: $Y = G(Z)$

Mealy- und Moore- Automaten sind ineinander überführbar. Bei gleichem Verhalten hat der Moore-Automat mehr Zustände.

Man unterscheidet bei Automaten folgende Verhaltensweisen:

- synchrones Verhalten:
Zustandswechsel nur zu vorgegebenen Zeitpunkten möglich (extern vorgegebener Takt).
- asynchrones Verhalten:
Zustandswechsel jederzeit möglich. Fehlverhalten durch Laufzeitunterschiede möglich.

Endliche Automaten können durch so genannte *Zustandsgraphen* dargestellt werden, die einer Auflistung aller in dem betrachteten System möglichen Zustände (= Knoten, Anfangszustand hervorgehoben) und Zustandsübergänge (= Kanten) entsprechen. Die Kanten werden mit den Ereignissen oder Übergangsbedingungen beschriftet, die die durch die jeweiligen Kanten beschriebenen Zustandsübergänge hervorufen. Beim Mealy-Automaten steht die Ausgabefunktion durch einen Strich getrennt unterhalb der Übergangsbedingung. Die Abhängigkeit der Ausgaben von den Eingaben bei gleichem Zustandswechsel kann durch unterschiedliche Zustandsübergänge dargestellt werden. Beim Moore-Automaten wird die Ausgabefunktion in den Kreis des zugehörigen Zustands geschrieben. Ein Strich trennt Zustandsname von Ausgabefunktion.

Neben den Zustandsgraphen können Automaten in äquivalente Zustandsübergangstabellen abgebildet werden. Beim Moore-Automaten wird eine Ausgabetable ergänzt, die jedem Zustand eine Ausgabefunktion zuordnet.

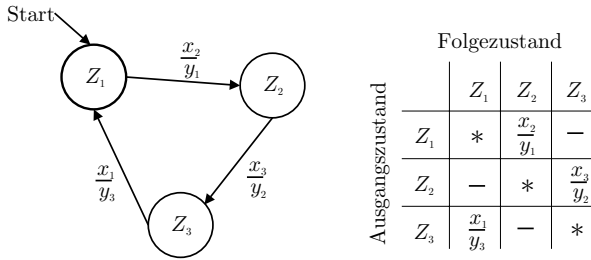


Abb. 3.13. Mealy-Automat

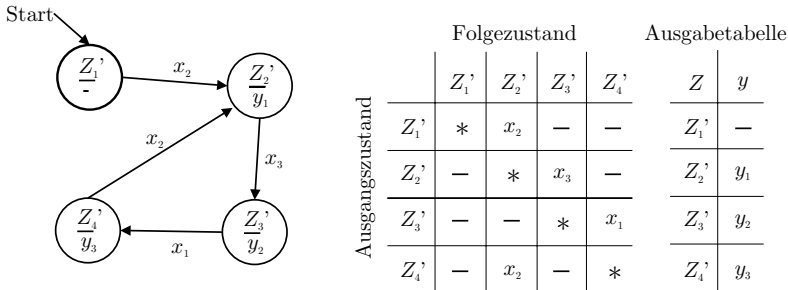


Abb. 3.14. Moore-Automat

Ist die Konstruktion des Zustandsgraphen für einen Prozess mit rein *sequentiellen* Abläufen noch recht einfach, so gestaltet sich diese jedoch bald als unüberschaubar, wenn zeitlich parallele, d. h. *nebenläufige* Vorgänge auftreten. Diese Aussage wird im Folgenden anhand Abb. 3.15 erläutert.

Betrachtet werden als Erstes die Zustandsgraphen zweier sequentieller Prozesse, die jeweils drei Zustände annehmen können und zunächst ohne gegenseitige Kopplungen verlaufen (Teil a). Die fehlenden Kopplungen zwischen den asynchronen Prozessen führen zu einer Trennung der Zustandsgraphen. Zur Beschreibung des Gesamtsystems können diese zu einem globalen Zustandsgraphen zusammengefasst werden, der den gleichen Informationsgehalt besitzt (Teil b). Die auftretenden Nebenläufigkeiten führen offenbar zu einem wesentlich umfangreicheren Zustandsgraphen mit einer Knotenzahl, die dem Produkt der Zahl der Zustände der Teilsysteme entspricht.

Ist die Beschreibung paralleler asynchroner Prozesse durch einen globalen Zustandsgraphen als Alternative zu den getrennten auch wenig sinnvoll, so wird dies jedoch unumgänglich, wenn die Teilprozesse synchronisiert werden sollen. Wird z. B. verlangt, dass die Ereignisse e₃ und e₆ nur gleichzeitig auftreten können, so kann dies durch eine Modifikation des globalen Zustandsgraphen ausgedrückt werden (Teil f).

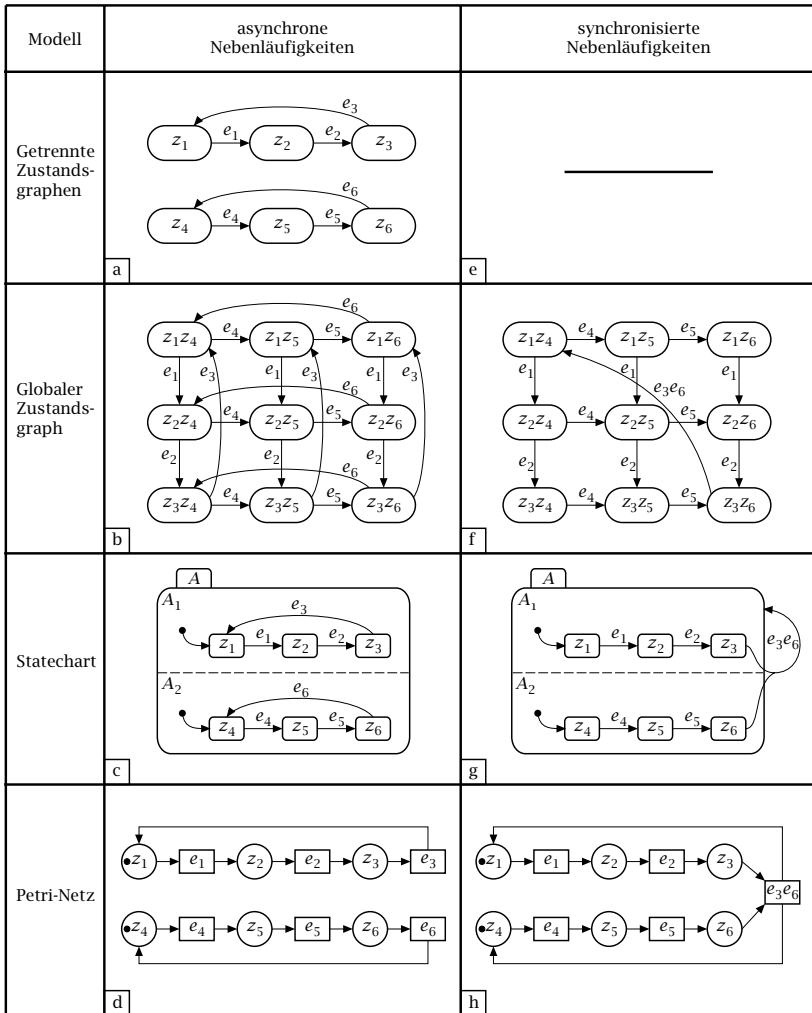


Abb. 3.15. Modelle von Systemen mit Nebenläufigkeiten

Statecharts und Stateflow

Für die Modellierung realer Systeme sind globale Zustandsgraphen durch die Zustandsexplosion und die damit verbundene extreme Unübersichtlichkeit nicht geeignet. Aus diesem Grund wurden die *Statecharts* entwickelt, bei denen es sich um eine Erweiterung der Zustandsgraphen mit Möglichkeiten zur Darstellung von Nebenläufigkeiten, von Broadcast-Kommunikation (Synchronisation von Nebenläufigkeiten) und von hierarchischen Strukturen handelt.

Die asynchronen Prozesse (Teil c) können hiermit durch einen sog. „Superstate“ dargestellt werden, der in zwei parallele Zustände, symbolisiert durch die gestrichelte Linie, aufgeteilt wird. In dieser sogenannten AND-Zerlegung sind beide Zustände gleichzeitig aktiv. Die beiden Zustände werden dann mit den getrennten Zustandsgraphen aus (Teil a) verfeinert. Die kurzen Kanten mit Punkt geben den Anfangszustand beim Betreten des Superstates an. Die mit e_3 und e_6 beschriftete Kante führt von beiden Zuständen z_3 und z_6 zum Superstate A. Beim Ausführen dieser Transition wird A reinitialisiert, so dass die (Anfangs-)Zustände z_1 und z_4 erneut aktiv sind. (Teil g) zeigt das entsprechende Statechart für die synchronisierten Prozesse.

Im Bereich der Programmierung von Steuereinrichtungen haben die Zustandsgraphen in Form des Tools HiGRAPH der Firma Siemens Eingang gefunden, bei dem prinzipiell Statecharts mit reduziertem Funktionsumfang zum Einsatz kommen. Das Tool ermöglicht eine graphisch sehr anschauliche und vor allem auch prozessnahe Entwicklung von Steuerungen. Des Weiteren kommen Statecharts zunehmend auch zur Modellierung beliebiger ereignisdiskreter/reaktiver Systeme zum Einsatz. Als Tools seien hier STATEMATE der Firma i-logix, welches z. B. in der Automobilindustrie zum Einsatz kommt, und STATEFLOW der Firma The MathWorks genannt, bei dem es sich um eine Erweiterung von SIMULINK zur Integration von Statecharts in kontinuierliche Modelle handelt. Hierdurch wird es möglich, mit SIMULINK auch hybride Systeme vollständig und korrekt zu modellieren. Hierauf wird in Abschnitt 3.3.1 weiter eingegangen.

Eine Hierarchiebildung ermöglicht hierbei eine Aufteilung, welche oft in enger Anlehnung zum realen Prozess gebildet werden kann. Durch eine Übersetzung in eine s-function wird der Rechenzeitbedarf während der Ausführung gering gehalten. Die Grundzüge von STATEFLOW werden im Folgenden kurz vorgestellt.

Die Erzeugung der Statecharts erfolgt in STATEFLOW mit einem graphischen Editor. Die Charts müssen in ein SIMULINK-Modell eingebettet sein, welches den Aufruf und den Ablauf des Statecharts steuert.

Mit der Werkzeugleiste des Chart-Editors auf der linken Seite können die verschiedenen Chart-Elemente wie Zustände (States) und Transitionen ausgewählt und mit der Maus auf der Arbeitsfläche platziert werden. Innerhalb des Zustandes steht sein Label als Fließtext, welches zwingend vergeben werden muss. Ist noch kein Label eingegeben, erscheint stattdessen ein Fragezeichen.

Der Name des Zustands muss eindeutig (innerhalb einer Hierarchieebene) vergeben werden. Die weiteren Elemente des Labels sind optional und bezeichnen Aktionen, welche durch diesen Zustand angestoßen werden. Die Syntax für die Festlegung der Aktionen ist die *Action Language*, eine Mischung aus Elementen zur Beschreibung des zeitlichen Ablaufs, der aus MATLAB bekannten Syntax und C. Folgende Schlüsselwörter legen dabei fest, dass die Aktion ausgeführt wird,

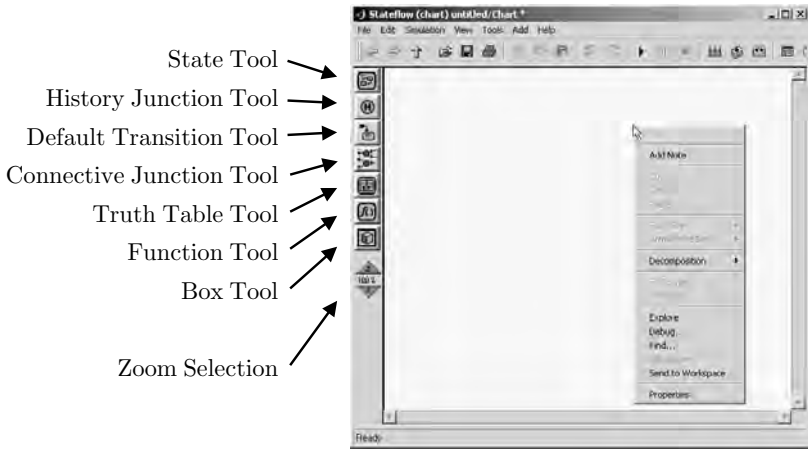


Abb. 3.16. Graphischer STATEFLOW-Editor

```

Name1/
entry: aktion1;
during: aktion2;
exit: aktion3;
on event1: aktion4;

```

Abb. 3.17. Zustand mit komplettem Label

- **entry**: wenn der Zustand aktiviert wird,
- **during**: wenn der Zustand aktiv ist (und das Chart ausgeführt wird),
- **exit**: wenn der Zustand verlassen wird,
- **on event**: wenn der Zustand aktiv ist und das angegebene Ereignis auftritt.

Eine Transition für einen Zustandsübergang erzeugt man, indem man auf den Rand des Ausgangszustandes klickt und die Maus bis zum Rand des folgenden Zustandes zieht und dort loslässt. Dabei entsteht ein die Transition symbolisierender Pfeil, dessen Form und Lage durch Ziehen verändert werden kann.

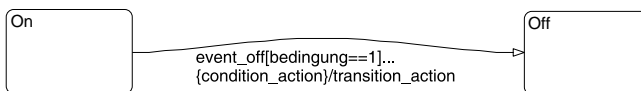


Abb. 3.18. Transition mit komplettem Label

Ebenso wie der Zustand besitzt die Transition ein Label, welches hier aber nicht zwingend vergeben werden muss. Ein Transitionslabel besitzt die Syntax

`event[condition]{conditionAction}/transitionAction`

Alle Komponenten des Labels sind optional und können beliebig kombiniert werden. Eine Transition ist *gültig*, falls

- der Ausgangszustand aktiv ist,
- das Ereignis `event` auftritt oder kein Ereignis angegeben ist und
- die angegebene Bedingung `condition` wahr ist oder keine Bedingung gestellt wurde.

Sobald eine Transition gültig ist, wird die `conditionAction` ausgeführt. Die `transitionAction` wird beim Zustandsübergang ausgeführt. Für einfache Transitionen besteht zwischen den beiden Aktionen, abgesehen von der Reihenfolge, kein Unterschied.

Die *Standardtransition* (engl. *default transition*) legt fest, welcher Zustand bei erstmaliger Ausführung eines Charts aktiv ist.

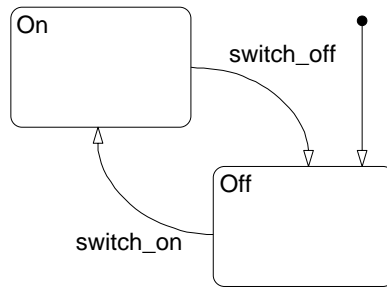


Abb. 3.19. Chart mit Standardtransition

In dem in Abb. 3.19 dargestellten Beispiel wird bei erstmaliger Ausführung des Charts der Zustand `Off` aktiv. In der dargestellten OR-Zerlegung, in der nur ein Zustand aktiv sein kann, wird entsprechend ein Zustandsübergang bei gültiger Transition – Umlegen des Schalters – stattfinden.

Ein weiteres Element ist der *Verbindungspunkt* (engl. *connective junction*). Mit Verbindungspunkten lassen sich komplexere Transitionen durch Zusammenführung und Verzweigung erzeugen. Damit lassen sich Verzweigungen, Fallunterscheidungen, zustandsfreie Flussdiagramme, verschiedene Schleifen, Selbstschleifen u. v. m. erzeugen.

Ist der STATEFLOW-Editor während der Simulationszeit offen, wird der Ablauf der Zustandsaktivierungen graphisch animiert. Zusätzliche Verzögerungen zur besseren Sichtbarkeit können im Debugger (Menü `Tools - Debugger`) konfiguriert werden. Hier kann die Animation auch gänzlich abgeschaltet werden. Viele weitere Funktionen unterstützen dort den Anwender

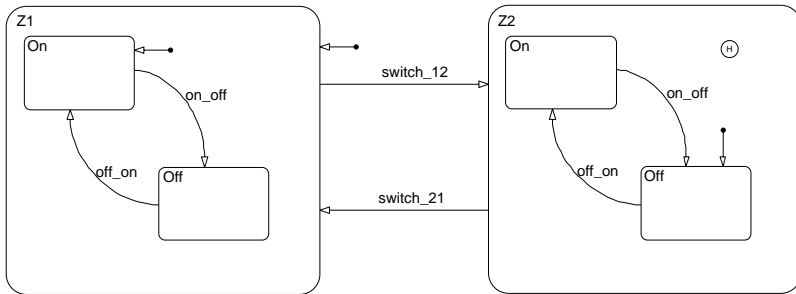


Abb. 3.20. Chart mit zwei Superstates. Der rechte Superstate Z2 besitzt eine History Junction.

bei der Fehlersuche. So können *Breakpoints* an verschiedenen Stellen gesetzt werden, automatisch Fehler (Zustandsinkonsistenzen, Konflikte, Mehrdeutigkeiten, Gültigkeitsbereiche, Zyklen) gesucht werden, zur Laufzeit Werte von Variablen und die Aktivität von Zuständen angezeigt werden.

Eine wichtige Eigenschaft von Statecharts ist die Möglichkeit, Zustandshierarchien bilden zu können. Zustandshierarchien erlauben verschiedene Grade der Detaillierung und lassen in bestimmter Hinsicht zusammengehörige Zustände leicht als solche erkennen. Mit ihrer Hilfe kann somit eine Strukturierung des Zustandsraumes erreicht werden, die für ein klareres Verständnis komplexer Verhaltensbeschreibung unerlässlich ist. In STATEFLOW lassen sich mit *Superstates* zusammengehörige Zustände zu einem Oberzustand zusammenfassen, indem ein neuer Zustand um die zusammenzufassenden Zustände gelegt wird.

Für das Innere eines solchen Superstates gelten dieselben Regeln wie für die oberste Ebene (Initialisierung, Anzahl aktiver Zustände etc.). Ein Superstate kann wie ein herkömmlicher Zustand Ziel und Ausgangspunkt von Transitionen sein. Er wird dann aktiviert, wenn er selbst oder einer seiner Unterzustände das Ziel einer gültigen Transition ist. Ist er selbst das Ziel, wird i. d. R. die Standardtransition ausgeführt. Ist ein Superstate nicht aktiviert, ist auch keiner seiner Unterzustände aktiv. Die Unterzustände können dabei als eigenständiges Chart innerhalb des Superstates aufgefasst werden.

Verliert ein Superstate durch eine Transition seine Aktivität, werden auch alle Unterzustände passiv. Bei einer neuerlichen Aktivierung des Superstate entscheidet erneut die Standardtransition, welcher Unterzustand die Aktivierung erhält. Dies kann man dadurch verhindern, indem man in den Zustand aus der Werkzeugleiste eine *History Junction* setzt. Nun erhält derjenige Unterzustand, welcher vor dem Verlust aktiviert war, die Aktivierung zurück (vgl. Abb. 3.20).

In einem Superstate kann eine weitere Transitionstyp eingesetzt werden, die *Innere Transition*. Sie wird erzeugt, in dem eine Transition vom Rand des

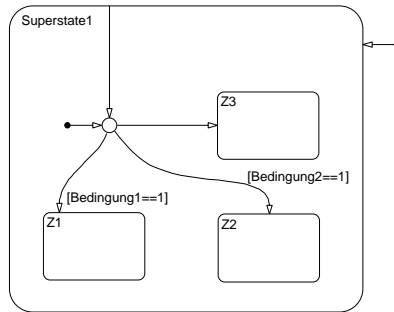


Abb. 3.21. Superstate mit einer Inneren Transition

Superstates zu einem seiner Unterobjekte mit der Maus gezogen wird. Diese Transition wird immer auf Gültigkeit überprüft, solange der Superstate aktiv ist, unabhängig von seinen Unterzuständen. Ein Beispiel für die Verwendung einer Inneren Transition ist die Vereinigung einer verzweigten Zustandskette. Anstatt Transitionen von allen Enden dieser Kette einzuzichnen, genügt oft der Einsatz einer Inneren Transition (vgl. Abb. 3.21).

Je nach Beschriftung der Transitionen durch die Labels kann es zu Fällen kommen, bei denen zwei unterschiedliche Transitionen prinzipiell gleichzeitig gültig werden. Solche Mehrdeutigkeiten sollte man zwar bei der Charterstellung unbedingt vermeiden, jedoch gibt es für eine Auflösung dieses Konflikts feste Regeln.

So wird diejenige Transition ausgeführt, welche (in der angegebenen Reihenfolge)

- zu einem Zielzustand höherer Hierarchie führt,
- ein Label mit Ereignis- und Bedingungsabfrage besitzt,
- ein Label mit Ereignisabfrage besitzt,
- ein Label mit Bedingungsabfrage besitzt,

Ist mit diesen Kriterien noch keine eindeutige Entscheidung möglich, wird die Reihenfolge graphisch bestimmt: Es wird diejenige Transition zuerst ausgeführt, deren Abgangspunkt der linken oberen Ecke eines Zustandes (bzw. der 12-Uhr-Stelle eines Verbindungspunktes) im Uhrzeigersinn als nächster folgt.

Bisher bezogen sich alle Ausführungen auf die exklusive Anordnung (Oder-Anordnung) von Zuständen, bei der immer genau ein Zustand aktiviert ist. Neben dieser Anordnung gibt es die *parallele Und-Anordnung*, bei der alle Zustände einer Hierarchieebene gleichzeitig aktiv sind und nacheinander abgearbeitet werden.

Innerhalb jeder Hierarchieebene (Chart, Superstate) kann die Art der Anordnung getrennt festgelegt werden. Dies geschieht im Kontextmenü eines zugehörigen Zustandes unter **Decomposition**. Wird auf **Parallel (AND)** um-



Abb. 3.22. Chart mit parallelen Zuständen

geschaltet, ändert sich die Umrandung der betroffenen Zustände zu einer gestrichelten Linie und es wird eine Zahl in der rechten oberen Ecke eingeblendet (vgl. Abb. 3.22). Diese Zahl gibt die Ausführungsreihenfolge der Zustände an. Die parallelen Zustände werden nacheinander von oben nach unten und von links nach rechts ausgeführt.

Mit parallelen Zuständen können Systeme modelliert werden, welche parallel ablaufende Teilprozesse besitzen (vgl. Abb. 3.15 auf Seite 103).

Statecharts sind neben der graphischen auch einer formalen, mathematischen Beschreibung zugänglich. Hierdurch wird es möglich, ein Statechart mit Hilfe von Werkzeugen auf bestimmte Eigenschaften hin zu überprüfen, die vorher in eine mathematisch-logische Darstellung gebracht worden sind (z. B. dass ein bestimmter Zustand des Systems nie erreicht wird). Dieser Vorgang wird als *Verifikation* bezeichnet, da der Nachweis in einigen Fällen mit mathematischen Mitteln in Form eines Beweises durchgeführt werden kann.

Hierzu wird die formale Spezifikation, z. B. eine Beschreibung der Systemeigenschaften mit Mitteln der gewöhnlichen oder temporalen Logik, mit dem gesamten dem Modell möglichen Verhalten verglichen.

In Bezug auf den Entwurf von Steuerungen ist es auf diese Weise möglich, die Korrektheit einer in Statecharts entworfenen Steuerung ansatzweise zu überprüfen, z. B. ob bei einer Ampelsteuerung „niemals alle Ampeln gleichzeitig auf grün stehen“. Die Firma Siemens bietet hierfür ein Verifikationsprogramm zu dem oben angesprochenen Tool HiGRAPH an.

3.2.4 Petrinetze

Nebenläufigkeiten

In Abb. 3.15 (Teil f) wird deutlich, dass die Synchronisation zweier paralleler Zustandsgraphen sehr unübersichtlich ist. Die Vereinigung des Transitionspeils³ in Abb. 3.15 (Teil g) zur Transitionssynchronisation ist eine Möglichkeit zur Lösung des Problems. Dieses wird auch das Problem der Zustandsexplosion genannt, da die Anzahl der Knoten eines Automatengraphen exponentiell mit der Anzahl der nebenläufigen Prozesse ansteigt. Ein anderes

³ Die Unified Modelling Language definiert hierzu auch Synchronisationsbalken für Statecharts.

Beschreibungsmittel, dass aufgrund der guten Analysemöglichkeiten den wissenschaftlichen Bereich der Automatisierungstechnik im Besonderen prägt, sind die Petrinetze, die auf Arbeiten Carl Adam Petris in den sechziger Jahren des vergangenen Jahrhunderts zurückgehen [16]. Petris Motivation bei der Schaffung der Netztheorie war vor allem der angesprochene Punkt der Modellierung von Nebenläufigkeiten. Aus diesem Grunde soll auch im Folgenden kurz in die Modellierung mit Petrinetzen eingeführt werden.

Aufbau

Eine sehr elegante Art, Systeme mit Nebenläufigkeiten zu beschreiben, ist die Modellierung mit *Petrinetzen*. Ein Petrinetz kann ebenfalls als erweiterter Zustandsgraph aufgefasst werden und ist ein gerichteter Graph, dessen Knoten in zwei Arten unterschieden werden, in Stellen und Transitionen. Die Kanten, die nur Knoten unterschiedlichen Typs verbinden können, stellen logische bzw. kausale Verknüpfungen zwischen den Knoten dar. Durch eine gegenüber den Zustandsgraphen geänderte Betrachtungsweise, die im Gegensatz zur Abbildung möglicher Zustandsübergänge näher an der Abbildung möglicher Ereignisse orientiert ist, gelingt mit Petrinetzen die Beschreibung auch solcher Systeme, die umfangreichere Automaten darstellen. Dazu wird eine Interpretation der unterschiedlichen Knotentypen erforderlich, wobei

- mit den durch Kreise symbolisierten *Stellen* (und deren *Markierungen*, die hier durch dicke Punkte gekennzeichnet sind) Teilzustände im System und damit Bedingungen für das Eintreten bestimmter Ereignisse wiedergegeben werden und
- mit den *Transitionen*, die als Rechtecke dargestellt sind, Ereignisse abgebildet werden, die im System auftreten können und damit zu Zustandsübergängen führen.

Dieser Zuordnung folgend können neben sequenziellen und alternativen Prozessen mit Petrinetzen auch nebenläufige Prozesse und deren Synchronisation einfach und elegant modelliert werden. Alternativen werden dabei durch Verzweigungen an Stellen und Nebenläufigkeiten durch Verzweigungen an Transitionen erzeugt. Unter Berücksichtigung der Richtungsinne der Kanten sind somit vier elementare Verknüpfungen zu unterscheiden, die in Abb. 3.23 zusammengestellt sind.

Durch Verwendung der in Abb. 3.23 gezeigten Netzelemente können sequenzielle Teilprozesse, die unabhängig voneinander laufen, in Verbindung gebracht werden. Die Kopplung über gemeinsame Stellen bewirkt dabei eine *wechselseitige Koordination* der asynchronen Teilprozesse. In diesem Fall kann immer nur einer der beiden Teilprozesse ausgeführt werden, wobei durch das Netz nicht festgelegt wird, in welcher Weise sich diese abwechseln. Dagegen ermöglicht die Kopplung über gemeinsame Transitionen eine *Synchronisation* der Teilprozesse. Das Schalten der vorderen Transition bewirkt den Start

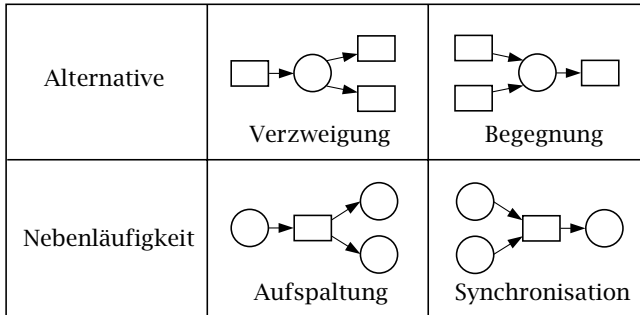


Abb. 3.23. Elementare Verknüpfungen in Petrinetzen

beider parallel ablaufender Teilprozesse, die an der hinteren Transition aufeinander warten.

Beispielsweise wird die in Abb. 3.15 geforderte Synchronisation durch ein Petrinetz (Teil h) wiedergegeben, welches aus einer nur geringfügigen Modifikation des Netzes der asynchronen Prozesse (Teil d) entstanden ist.

Die bisher beschriebene Netzstruktur eines Petrinetzes bildet nur die statischen Eigenschaften des modellierten Systems ab. Der aktuelle Zustand und die dynamischen Abläufe im System ergeben sich erst aus der Markierung der Stellen bzw. dem Markenfluss. Hierbei liegt ein durch eine Stelle repräsentierter Teilzustand im System genau dann vor, wenn die Stelle eine (oder mehrere) Marken trägt. Der Gesamtzustand des modellierten Systems ergibt sich dann – als „Summe aller Teilzustände“ – aus der aktuellen Markierung im Netz.

Der Markenfluss im Netz wird durch Schalten der aktiven Netzelemente, der Transitionen, erzeugt. Eine Transition wird im Netz immer genau dann aktiviert, d. h. schaltfähig, wenn alle Stellen in ihrem *Vorbereich* (zuführende Kanten) mit einer ausreichenden Anzahl Marken besetzt und alle Stellen im *Nachbereich* (auslaufende Kanten) leer sind bzw. hinreichend freie Kapazitäten besitzen. Eine Stelle kann zu jedem Zeitpunkt maximal so viele Marken enthalten, wie ihre *Kapazität* zulässt. Die unter einer Markierung aktivierten Transitionen zeigen Ereignisse auf, die ausgehend von diesem Zustand eintreten können. Das Schalten einer Transition führt dann zum eigentlichen Markenfluss vom Vor- zum Nachbereich und damit zum Übergang des Systems in einen neuen Zustand. An einer Kante können Zahlen als *Gewicht* notiert werden um anzuzeigen, dass über diese Kante genau die entsprechende Anzahl von Marken hinzugefügt oder entfernt wird. Kantengewichte von 1 werden vereinbarungsgemäß nicht dargestellt.

Die eingeführten Begriffe können anschaulicher an einem einfachem Beispiel erläutert werden, welches auch weiter unten wieder aufgegriffen werden soll. Als Beispielprozess diene die in Abb. 3.24 skizzierte Fertigungszelle, in

der die Roboter I und II Werkstücke vom Typ A und B in unterschiedlicher Reihenfolge bearbeiten.

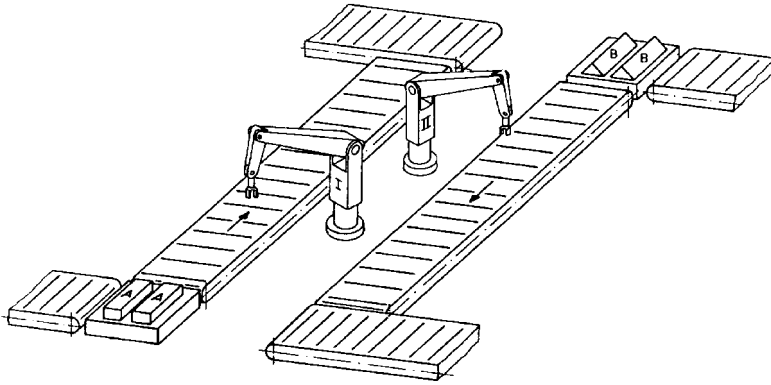


Abb. 3.24. Fertigungszelle als Beispielprozess

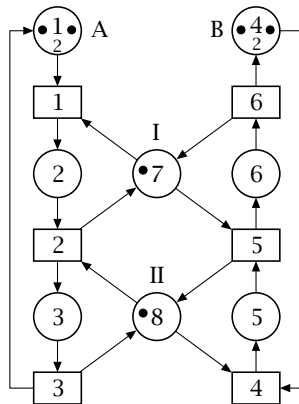


Abb. 3.25. Petrinetz-Modell der Fertigungszelle aus Abb. 3.24

Mit Abb. 3.25 wird ein einfaches Petrinetz-Modell des Beispielprozesses wiedergegeben, wobei die Werkstücke hier der Einfachheit halber nach der Bearbeitung wieder zurückgeführt werden, um die Fertigungszelle getrennt von ihrer Umgebung betrachten zu können. In dem Petrinetz ist bereits die

gewünschte Bearbeitungsfolge enthalten, was als Abbild einer ersten, intuitiv entworfenen Steuerung angesehen werden kann. Die Anfangsmarkierung gibt wieder, dass jeweils zwei Werkstücke von Typ A und B auf den Eingangspuffern vorhanden und dass beide Roboter verfügbar sind. Das Schalten der Transition t_1 bzw. t_4 entspricht somit dem Start des betreffenden Bearbeitungsganges.

Neben der graphischen Darstellung ist ein Petrinetz auch einer mathematischen Beschreibung zugänglich. Diese erfolgt vorteilhaft in Form der so genannten *Netzmatrix* \mathbf{N} , die Verknüpfungen zwischen den Stellen (s_i) und den Transitionen (t_j) eines Petrinetzes enthält. Wie aus Abb. 3.26 hervorgeht, werden Kanten, die von einer Stelle zu einer Transition verlaufen, negativ und die umgekehrt orientierten positiv eingetragen. Ergänzt wird diese Netzmatrix durch den *Anfangsmarkierungsvektor* \mathbf{m}_0 , der die Anzahl der Marken auf den Stellen des Petrinetzes wiedergibt.

	Transitionen					
Stellen	1	2	3	4	5	6
1	-1	0	1	0	0	0
2	1	-1	0	0	0	0
3	0	1	-1	0	0	0
4	0	0	0	-1	0	1
5	0	0	0	1	-1	0
6	0	0	0	0	1	-1
7	-1	1	0	0	-1	1
8	0	-1	1	-1	1	0

$N:$

Marken
2
0
0
2
0
0
1
1

$m_0:$

Abb. 3.26. Mathematische Beschreibung des Petrinetzes aus Abb. 3.25

Die Definition der Netzmatrix \mathbf{N} lässt eine ebenfalls mathematische Formulierung der Schaltregel zu, die die Grundlage für weitergehende Analyseverfahren bildet. So ergibt sich ausgehend von der Anfangsmarkierung \mathbf{m}_0 die Folgemarkierung \mathbf{m} nach dem Schalten einer Transition t_j durch die vektorielle Addition der j -ten Spalte der Netzmatrix zur Ausgangsmarkierung

$$\mathbf{m} = \mathbf{m}_0 + \mathbf{t}_j \quad . \tag{3.48}$$

Fasst man mehrere Schaltvorgänge zusammen, erhält man den Vektor der Schalthäufigkeiten \mathbf{v} , dessen Elemente v_i angeben, wie oft die Transition t_i schaltet:

$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{|T|} \end{pmatrix} . \quad (3.49)$$

Hiermit lässt sich Gl.3.48 allgemein für beliebig viele schaltende Transitionen formulieren zu

$$\Delta \mathbf{m} = \mathbf{N} \cdot \mathbf{v} . \quad (3.50)$$

Angewandt auf das Beispielnetz führt das Schalten der Transition t_4 zu der Folgemarkierung

$$\mathbf{m} = \mathbf{m}_0 + \mathbf{N} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \mathbf{m}_0 + \mathbf{t}_4 = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad (3.51)$$

welches dem Abzug jeweils einer Marke von den Stellen s_4 und s_8 sowie dem Transfer einer Marke auf die Stelle s_5 entspricht.

Ausgehend von der mathematischen Beschreibung eines Petrinetzes kann nun mit Hilfe rechnergestützter Analysewerkzeuge Einblick in das dynamische Verhalten gewonnen werden. Das Einfachste dieser Verfahren, die Simulation, überlässt nach einer Aktiviertheitsüberprüfung dem Bediener (oder einem Zufallsgenerator) die Auswahl einer schaltenden Transition und führt daraufhin den gewünschten Schaltvorgang gemäß Gl.(3.48) aus. Auf weitergehende Analyseverfahren, welche graphentheoretische oder algebraische Methoden nutzen, um Einblick in das dynamische Verhalten des betreffenden Petrinetzes zu gewinnen, soll hier jedoch aus Komplexitätsgründen nicht eingegangen werden.

3.2.5 Weitere Beschreibungsmittel

In der Automatisierungstechnik sind neben Statecharts und Petrinetzen viele weitere Beschreibungsmittel üblich, die sich für bestimmte Anwendungsbereiche auszeichnen und dort vornehmlich eingesetzt werden. Mit den Sequenzdiagrammen soll hier eines kurz vorgestellt werden, dass bei der Entwicklung von ereignisdiskreten Systemen nicht direkt zur Implementation sondern allenfalls implementierungsnah eingesetzt werden kann. Daran schließt sich eine Übersicht zu weiteren wichtigen Beschreibungsmitteln an.

Sequenzdiagramme

Sequenzdiagramme zeigen exemplarisch den Kommunikationsablauf zwischen den Beteiligten am modellierten Prozess. Sequenzdiagramme besitzen zwei Dimensionen: auf der vertikalen Achse wird die Zeit nach unten fortlaufend aufgetragen, wobei die Skalierung der Achse festgelegt sein kann, aber nicht muss; die horizontale Achse repräsentiert die verschiedenen beteiligten Instanzen, die Reihenfolge der Instanzen ist hierbei unerheblich.

Sequenzdiagramme sind eine Ableitung der in der Telekommunikationsbranche üblichen *Message Sequence Charts* (MSC). Sie sind ein Diagrammtyp, mit dem die Interaktion mehrerer Objekte beschrieben wird. Bewusst wird hierbei auf die Beschreibung von Strukturen verzichtet, wie sie etwa Statecharts bieten. So werden die internen Abläufe der einzelnen Instanzen nicht genauer dargestellt, sondern lediglich, ob eine Instanz aktiv ist oder nicht. Der dargestellte Kommunikationsablauf der Sequenzdiagramme ist lediglich exemplarisch, da keine Alternativen im Ablauf möglich sind. Tritt zum Beispiel ein Fehler auf, muss der hierfür geltende Ablauf in einem weiteren Sequenzdiagramm dargestellt werden. Das Gleiche gilt für Verzweigungen im Programm oder zu Beginn, so dass ein mittels eines Sequenzdiagramms dargestellter Kommunikationsablauf nur unter festgelegten Umständen eintritt.

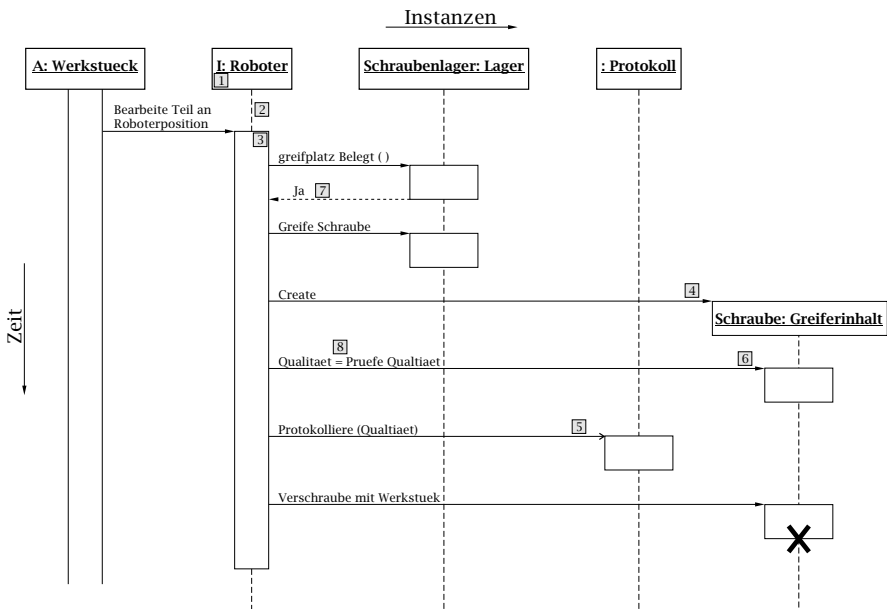


Abb. 3.27. Sequenzdiagramm zur Verschraubung eines Werkstückes durch einen Roboter

Jede Objektinstanz auf der horizontalen Achse wird wie in Klassendiagrammen als rechteckiger Kasten [1] symbolisiert, vgl. Abb. 3.27. Unterhalb einer Instanz findet sich die *Lifeline* [2], die sich in Zeiten der Aktivität zu einem *Aktivitätsblock* [3] verbreitert.

Das wesentliche Element der Sequenzdiagramme – die Darstellung der Kommunikation – wird durch Pfeile symbolisiert. Hierdurch können synchrone und asynchrone Kommunikation hergestellt werden, auch ist es möglich, Objektinstanzen neu zu erzeugen oder zu zerstören. In Abb. 3.27 beginnt die Lebenslinie des Objekts `Schraube:Greiferinhalt` erst nach der `create`-Nachricht [4] durch den `I:Roboter` und der anschließenden Erzeugung der Instanz. Bei asynchroner Kommunikation (Pfeil mit offener Spitze [5]) wartet der Sender nicht, bis der Empfänger die Bearbeitung der Nachricht beendet hat. Bei synchroner Kommunikation (durchgezogene Linie und geschlossene Pfeilspitze [6]) ist der Ablauf der aufrufenden Instanz so lange unterbrochen, bis die aufgerufene Instanz die Bearbeitung beendet hat. Es ist möglich, dass die aufgerufene Instanz bei Beendigung einen Rückgabewert an die aufrufende Instanz liefert (gestrichelte Linie und geschlossene Pfeilspitze [7]) oder bei Methodenaufruf [8]).

UML-Diagramme

Die *Unified Modelling Language* (UML) umfasst mehrere Diagrammtypen zur Darstellung von Struktur und Dynamik eines objektorientierten Modells. Neben den Statecharts in Kap. 3.2.3 und den Sequenzdiagrammen werden in der UML weitere Diagrammtypen definiert, von denen vor allem Klassendiagramme und Anwendungsfalldiagramme gebräuchlich sind.

Die Unified Modelling Language lässt den Anwendern der Diagrammtypen und den Entwicklern von Modellierungssoftware viele Freiheiten und bleibt in einigen Punkten bewusst offen, um in unterschiedlichen Anwendungsgebieten einsetzbar zu sein. Vielfach liefert sie sogar mit der Definition von *Variationen* mögliche, aber nicht zwingende Erweiterungen im Text mit. Die UML darf aus diesem Grunde nicht als fester Industriestandard wie ein Dokument von DIN/ISO/IEC o. ä. verstanden werden. Die freie Interpretierbarkeit bietet den Vorteil, dass z. B. Sequenzdiagramme bereits frühzeitig bei Diskussionen skizziert werden können, um bestimmte Kommunikationsschemata festzulegen, ohne die eigentliche Implementation vorwegnehmen zu müssen. Erst mit der Festlegung auf ein bestimmtes Entwicklungswerkzeug legt man sich auf den Diagramm-,Dialekt‘ fest. Die folgenden Diagrammartentypen sind ebenfalls in der UML definiert:

- Klassendiagramme (Class diagrams)
Darstellung von Methoden und Attributen einer Objektklasse und Einbeziehung von Vererbung und Assoziationen,
- Anwendungsfalldiagramme (Use cases)
Repräsentation von wesentlichen Funktionalitäten eines Systems bei Angabe der beteiligten Akteure,

- Aktivitätsdiagramme (Activity diagrams)
Ähnlichkeiten zu Petrinetzen und Datenflussdiagrammen erlauben, auch nebenläufige Prozesse darzustellen,
- Komponentendiagramme (Component diagrams)
Beschreibung von Abhängigkeiten zwischen einzelnen Softwarekomponenten wie Quelltexten, ausführbarem Code oder Skripts.
- Verteilungsdiagramme (Deployment Diagrams)
Darstellung der Konfiguration/Verteilung von Softwarekomponenten z. B. auf Prozessoren oder Rechnernetzwerke.

DIN/EN 61131-3

Bei der Programmierung *Speicherprogrammierbarer Steuerungen* (SPS) finden hauptsächlich die Programmiersprachen nach DIN/EN 61131-3⁴ Anwendung. Unter diesen finden sich neben der Ablaufsprache (AS), engl. Sequential Function Chart (SFC), die von den Petrinetzen abgeleitet ist, die vier Sprachen (siehe auch Abb. 3.28)

- Anweisungsliste (AWL), Instruction List (IL)
Hardwarenahe und Assembler ähnliche Text-Sprache,
- Funktionsbausteinsprache (FBS), Function Block Diagram (FBD)
Grafische Programmiersprache, die sehr gut zur Darstellung Boolescher Verknüpfungen geeignet ist,
- Kontaktplan (KOP), Ladder Diagram (LD)
grafische Repräsentation von Relaissteuerungen mit Stromschienen und -pfaden,
- Strukturierter Text (ST), Structured Text (ST)
Textuelle, an Pascal angelehnte Hochsprache.

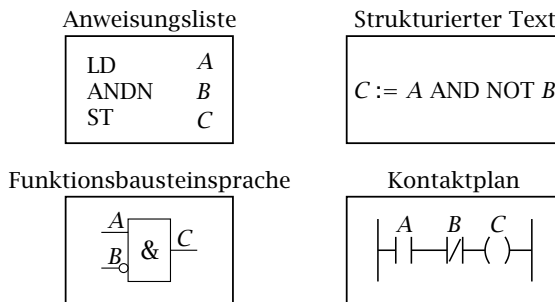


Abb. 3.28. Darstellung von $C = A \wedge (\neg B)$ in AWL, FBS, KOP und ST aus DIN/EN 61131-3

⁴ Die DIN/EN 61131-3 entspricht der IEC 1131-3

IEC 61499

In der IEC 61499 werden Funktionsblöcke für verteilte Systeme definiert. Wesentlich für diese Funktionsblöcke ist, dass sie hybrid sind und aus einer einfachen grafischen Automatendarstellung (Zustandsdiagramm, Abb. 3.30) und kontinuierlichen Algorithmen bestehen. Die Programmierung der Algorithmen erfolgt nach DIN/EN 61131-3 unter Einbindung von Funktionsblöcken nach IEC 61499. Bei den Funktionsblöcken wird auch grafisch durch die Aufteilung der Ein- und Ausgangsschnittstellen in Ereignisports (oben) und Datenports (unten) in einen ereignisdiskreten Teil und einen kontinuierlichen Modellteil unterschieden (siehe Abb. 3.29), innerhalb dessen sich eine Hierarchisierung durch Einbindung weiterer hybrider Funktionsblöcke realisieren lässt (Abb. 3.31). Grundlagen und Details der Modellierung hybrider Systeme werden im folgenden Abschnitt behandelt.

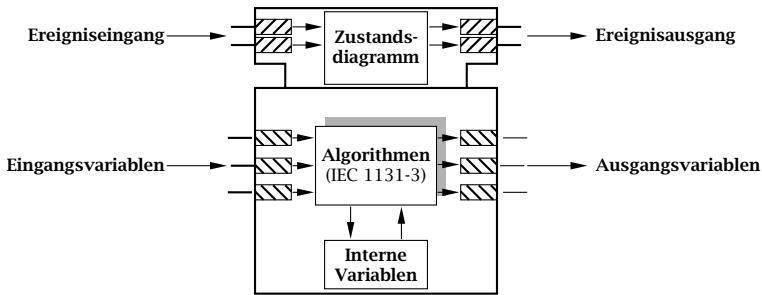


Abb. 3.29. Funktionsblock nach IEC 61499

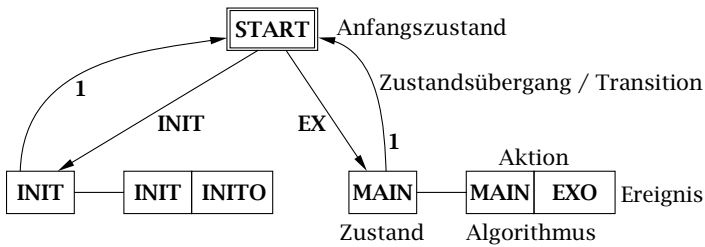


Abb. 3.30. Zustandsdiagramm eines Funktionsblocks nach IEC 61499

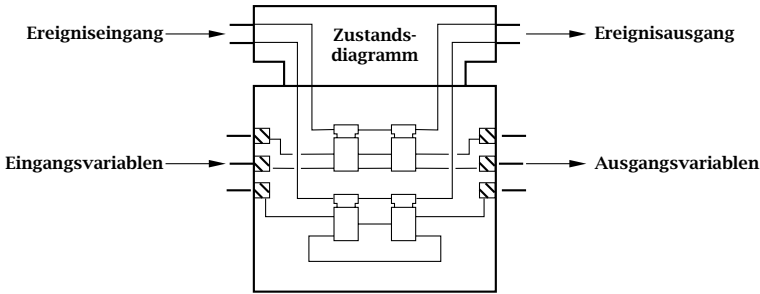


Abb. 3.31. Hierarchischer Aufbau eines Funktionsblocks nach IEC 61499

3.3 Hybride Modellbildung

In der Automatisierungstechnik haben sich in den vergangenen Jahrzehnten zwei deutlich unterschiedliche Sichtweisen auf technische Systeme herausgebildet. Einerseits sind dies die kontinuierlichen Systeme (Kap. 3.1), für die eine umfangreiche Systemtheorie und eine große Zahl von Analyse- und Synthesemethoden zur Auslegung kontinuierlicher Regelkreise entwickelt wurde (siehe Kap. 5.2). Andererseits werden viele Systeme als schrittweise ablaufende Folge einzelner Zustände interpretiert, deren Gesamtzustand sich beim Auftreten von Ereignissen ändert. Die Menge diskreter Zustände kann jeweils z. B. kontinuierliches Verhalten repräsentieren, aber die Wirkungen der Teilprozesse aufeinander durch Ereignisse sind das wesentliche Charakteristikum für diese ereignisdiskreten Prozesse. Die Entwicklung der Systemtheorie des Ereignisdiskreten und zugehöriger Analyseverfahren ist ebenfalls weit fortgeschritten bei einer Vielzahl existierender Modellformen (Kap. 3.2). Für die Synthese von Steuerkreisen gibt es in vielen Fällen Syntheseverfahren, die aber nicht allgemeingültig sind (Kap. 5.5).

Reale physikalische Prozesse sind von Natur aus *hybrid*.⁵ im Sinne eines wertekontinuierlichen *und* ereignisdiskreten Verhaltens [62]. So wird bei der Erwärmung von Flüssigkeiten, die in der Nähe des Arbeitspunktes durch eine lineare Differentialgleichung beschrieben werden kann, bei Erreichen einer bestimmten Temperatur die Verdampfung beginnen; bei einem Abkühlvorgang kann die Flüssigkeit erstarren; ein fallender Ball ändert seine Richtung bei Kontakt; ein Behälter unter zu großem Druck kann zerstört werden etc. In vielen technischen Systemen ist Steuerungslogik implementiert, deren Quelltexte Fallunterscheidungen wie *if-then-else*-Verzweigungen enthalten. Trotzdem können viele technische Systeme mit der Theorie kontinuierlicher oder diskreter Systeme behandelt werden. Innerhalb dieses Abschnitts sollen Systeme behandelt werden, für die das nicht gilt:

⁵ Der Begriff *hybrid* bedeutet: *gemischt; von zweierlei Herkunft; aus Verschiedenem zusammengesetzt.*

Ein *hybrides System* ist ein dynamisches System, das weder mit den Methoden der kontinuierlichen Systemtheorie noch mit den Methoden der diskreten Systemtheorie mit ausreichender Genauigkeit dargestellt und analysiert werden kann.

Das Interesse an der Entwicklung der Theorie der *hybriden* Systeme hat sich im letzten Jahrzehnt erheblich verstärkt. Als die *diskreten* Systeme in den 80er Jahren des vergangenen Jahrhunderts in den in den Interessensbereich größerer Teile der wissenschaftlichen Gemeinschaft gerieten, wurden neue Werkzeuge zur Modellierung und Analyse diskreter Systeme entwickelt. Hiermit konnten auch die diskreten Aspekte von hybriden Systemen besser verstanden und modelliert werden.

Die Anforderungen an ein Beschreibungsmittel, das allgemeine hybride Systeme abbilden kann, sind dadurch gekennzeichnet, dass folgende Eigenschaften wiedergegeben werden müssen:

- diskrete und kontinuierliche Zustandsvariablen,
- diskrete und kontinuierliche Ein- und Ausgangssignale,
- diskrete Ereignisse, die von der kontinuierlichen Zustandstrajektorie ausgelöst werden,
- diskontinuierliche Änderungen der kontinuierlichen Zustandstrajektorie, die durch diskrete Ereignisse ausgelöst werden,
- Änderung von Modellordnung und -struktur durch diskrete Ereignisse.

Beschreibungsmittel, die diese Forderungen zumindest zum großen Teil erfüllen, wurden erst in den letzten Jahren entwickelt, wobei die Entwicklung eher noch am Anfang steht. Prinzipiell bieten sich zwei grundverschiedene Vorgehensweisen an, um ein solches Beschreibungsmittel zu entwickeln - die Erweiterung bzw. Integration bereits vorhandener Beschreibungsmittel (Kap. 3.3.2) oder die Definition eines völlig neuen Modellansatzes.

Ein erstes einfaches Modell für hybride Systeme, bei denen im Wesentlichen kontinuierliche Aspekte im Vordergrund stehen, erhält man, indem ein normales kontinuierliches Modell durch Umschaltungen der Systemdynamik erweitert wird. Ein solches Modell besteht aus den kontinuierlichen Zustandsvariablen und einer Menge diskreter Zustände, die vielfach als Modi bezeichnet werden. In jedem Modus wird das System durch einen anderen Satz von Differentialgleichungen beschrieben. Erreicht nun die Zustandstrajektorie in einem Modus einen vorbestimmten Zustand, wird in einen neuen Modus umgeschaltet.

Das dynamische Verhalten eines hybriden Systems und damit der Verlauf der hybriden Trajektorie wird wesentlich durch die Wechselwirkungen zwischen den kontinuierlichen und den diskreten Systemteilen bestimmt. Nach dem Start bewegt sich das hybride System, ausgehend von der Anfangsbedingung und dem Anfangszustand, mit einer bestimmten kontinuierlichen Dynamik im Zustandsraum. Tritt nun ein Ereignis auf, kann dieses im kontinuierlichen Systemteil unmittelbar zu einer Änderung der Systemdynamik, zu

einer geänderten Systemordnung oder -struktur oder zu einer diskontinuierlichen Änderung im Verlauf der Systemtrajektorie führen. Allerdings kann ein Ereignis auch nur Auswirkungen auf den diskreten Systemteil oder gar keine Auswirkungen haben. Es können auch mehrere Ereignisse zu einem Zeitpunkt stattfinden oder sich gegenseitig auslösen. Auch kann die kontinuierliche Zustandsänderung des Systems das Auftreten von Ereignissen bewirken, die dadurch eintreten, dass Zustandsgrößen bestimmte Werte annehmen. Dies ist z. B. der Fall, wenn ein Behälter beginnt überzulaufen oder wenn eine Masse eines Zweimassenschwingers auf eine Begrenzung trifft. Ereignisse können auch durch die ereignisdiskreten Teilsysteme oder von außen ausgelöst werden – z. B. durch eine Steuerung oder das Umlegen eines Schalters von Hand. Je nachdem, ob bei Veränderungen des kontinuierlichen Systemverhaltens das auslösende Ereignis eine innere Systemeigenschaft ist oder von außen kommt, spricht man auch von autonomem und von gesteuertem Umschalten bzw. von autonomen und von gesteuerten Sprüngen. Da das Verhalten und die Abläufe im hybriden System in beiden Fällen jedoch auf den gleichen Mechanismen beruhen, ist die Unterscheidung in autonom und gesteuert nicht unbedingt erforderlich.

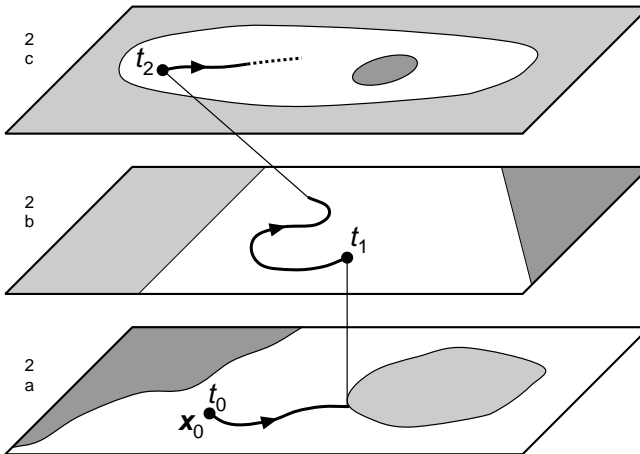


Abb. 3.32. Beispiel für den Verlauf einer Trajektorie eines hybriden Systems

Der prinzipielle Verlauf einer hybriden Trajektorie als Folge der Interaktion von kontinuierlichen und ereignisdiskreten Teilsystemen ist in Abb. 3.32 zu sehen. Das System startet zum Zeitpunkt t_0 im Punkt x_0 und verläuft entsprechend der eingezeichneten Trajektorie. Erreicht die Trajektorie, wie zum Zeitpunkt t_1 , den Rand eines der grauschattierten Bereiche, lösen die Zustandsgrößen ein Ereignis aus. Durch das Ereignis geht das hybride Sys-

tem in einen neuen diskreten Zustand über mit veränderter kontinuierlicher Dynamik und entsprechend auch mit neuen Übergangsbereichen im kontinuierlichen Zustandsraum (graue Bereiche). Nach dem Ereignis läuft die Trajektorie entsprechend den nun völlig neuen Verhältnissen weiter. Zum Zeitpunkt t_2 wird von außen ein Ereignis ausgelöst. Dieses führt neben einer erneuten Veränderung der kontinuierlichen Dynamik auch zu einer Reinitialisierung der Zustandsgrößen. Danach verläuft die Trajektorie entsprechend der neuen Dynamik bis das nächste Ereignis eintritt, usw.

3.3.1 Getrennte Modellierung am Beispiel Stateflow

Mit der Kenntnis der Modellierungsmöglichkeiten kontinuierlicher (siehe Abschnitt 3.1) und ereignisdiskreter (Abschnitt 3.2) Systeme ist es naheliegend, ein hybrides System in einen kontinuierlichen und einen diskreten Teil aufzuteilen (*Dekomposition*). Beide Teilsysteme lassen sich unabhängig voneinander mit Methoden ihrer Domäne entwerfen und überprüfen, um bestimmte Eigenschaften der Teilsysteme sicherzustellen. Bei der *Komposition* zu einem Gesamtsystem nach Abb. 3.33 müssen außerdem noch Schnittstellen hinzugefügt werden, die es ermöglichen, dass diskrete bzw. digitale Signale in kontinuierliche bzw. analoge umgewandelt werden und umgekehrt. Diese Schnittstellen heißen *Injektor* und *Quantisierer*. Neben dem hybriden Signalkreis besitzt ein solches System auch kontinuierliche und diskrete Ein- und Ausgänge.

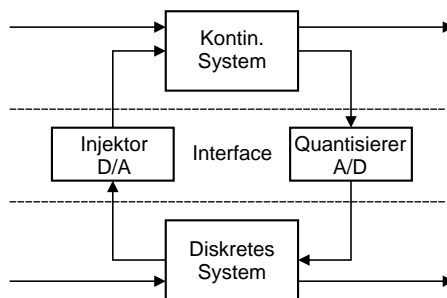


Abb. 3.33. Getrennte Modellierung kontinuierlicher und diskreter Teilsysteme

Solche Erweiterungen sind für Automaten, Statecharts, Petrinetze etc. üblich. So spricht man von einer *steuerungstechnischen* oder *signaltechnischen Interpretation* von Petrinetzen, wenn diese um Ein- und Ausgänge erweitert werden, um als Modell für eine Steuerung zu dienen.

Ein typisches Beispiel für dieses Konzept ist die Integration von Statecharts unter SIMULINK. Innerhalb dieses Abschnitts werden deshalb am Beispiel von STATEFLOW – stellvertretend für die getrennt modellierten hybriden Systeme – die noch fehlenden Schnittstellen erläutert.

The screenshot shows the STATEFLOW-Explorer window titled 'Exploring - aufzug_sol2/Chart'. The interface is divided into two main panes. The left pane, 'ObjectHierarchy', shows a tree structure of objects: 'aufzug_sol2' (parent), 'Chart' (child), 'Tuer' (child of Chart), 'Auf' (child of Tuer), 'Zu' (child of Tuer), 'Position' (child of Zu), 'StackZwe' (child of Position), 'StackEin' (child of Position), 'EG' (child of Position), 'UG' (child of Position), 'Fahrbetrieb' (child of Tuer), 'Aus' (child of Fahrbetrieb), 'Ein' (child of Fahrbetrieb), 'Zu_niedrig' (child of Ein), 'Bereit' (child of Ein), and 'Zu_hoch' (child of Ein). The right pane, 'Contents of: (chart) aufzug_sol2/Chart', displays a table of variables and signals.

Name	Scope	Trigger Type	Type	Size	Min	Max	InitVal
hoch	Local						
oeffnen	Local						
runter	Local						
schliessen	Local						
Z	Local		double				0
Wunsch	Input(1)		double				
Einschalter	Input(2)		double				
S	Output(1)		double				0

At the bottom of the right pane, there is a status bar showing 'events(4) data(4) targets(0) 8 [1:8]'.

Abb. 3.34. Übersicht über die internen und externen Ereignisse und Signale im STATEFLOW-Explorer

In hybriden Systemen existieren allgemein die beiden Variablentypen von Signalen und Ereignissen. Vor der Verwendung müssen diese Variablen im *Data Dictionary* deklariert werden. Mit dem STATEFLOW-Explorer kann das Data Dictionary bearbeitet werden und es lassen sich die in einem Statechart definierten Variablen ansehen, umbenennen und entfernen, oder es können auch neue Variablen deklariert und mit Eigenschaften versehen werden. Im linken Teil befindet sich in Abb. 3.34 ein Baum mit der Objekthierarchie. Als Objekte werden unter STATEFLOW die einzelnen Statecharts, die Zustandsmaschine als deren Elternobjekt und alle Zustände betrachtet, wobei Substates Kindobjekte des zugehörigen Superstates sind. Ein Objekt kennt dabei neben seinen eigenen Variablen auch die aller Elternobjekte. Die höheren Ebenen der Elternobjekte sind jeweils eine Ebene weiter links als die Kindobjekte angeordnet. In Abb. 3.34 sind auf Diagramm- oder Chartebene vier lokale Ereignisse, eine lokale Variable vom Typ *double*, zwei Signaleingänge und ein Signalausgang definiert. Ereignisse werden im STATEFLOW-Explorer durch einen Blitz links vom Variablennamen gekennzeichnet, Signale durch ein Matrix-Icon.

Ein- und Ausgänge zu SIMULINK können nur auf Chart-Ebene definiert werden, lokale Variablen, *events* und *data*, hingegen auf allen Ebenen. Wurde ein Ereignis oder ein Datum deklariert und mit dem *scope* festgelegt, ob es lokal oder ein Eingang oder Ausgang zu SIMULINK sein soll, können und müssen je nach Variable weitere Eigenschaften festgelegt werden.

Grundsätzlich kann der Injektor in Abb. 3.33 auf zwei Arten modelliert werden. Interessiert an einem kontinuierlichen Signal in SIMULINK der Über-

gang von einem Wertebereich in einen zweiten, kann dieser Übergang als Ereignis aufgefasst werden, so dass ein Ereigniseingang adäquat erscheint. Soll hingegen bei der Überprüfung der Gültigkeit einer Transition nur getestet werden, ob sich das Signal in einem bestimmten Bereich bestimmt, und wird der Zeitpunkt der Transitionsausführung nicht ausschließlich durch den Wechsel des Wertebereichs festgelegt, sollte das Signal über einen Dateneingang in das Statechart gelangen.

Neben diesen beiden Möglichkeiten der Injektion kann ein Statechart auch zur Abbildung kontinuierlichen geschalteten Verhaltens genutzt werden. Besitzt ein Eingangssignal keinen Einfluss auf die Abläufe im Statechart, sondern wird nur abhängig vom (diskreten) Zustand des Statecharts manipuliert und wieder nach außen gegeben, enthält das Statechart auch Teile, die in Abb. 3.33 im kontinuierlichen Teilsystem enthalten sind.

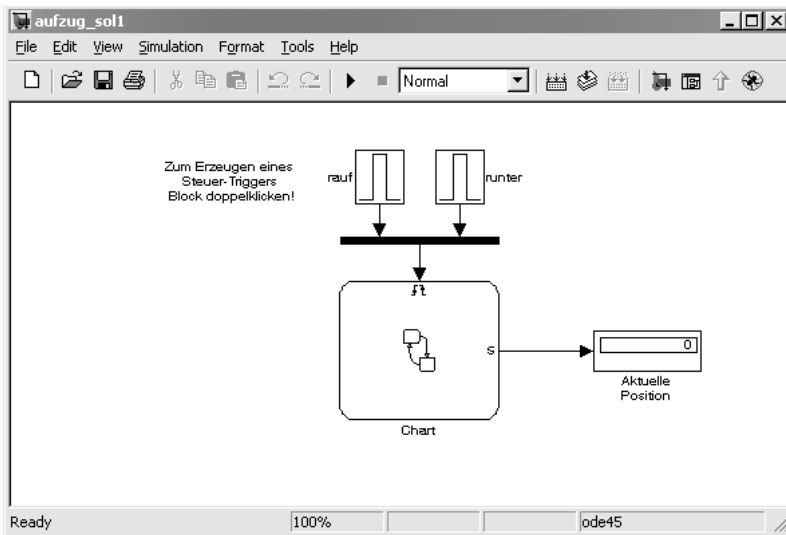


Abb. 3.35. Ereigniseingänge eines STATEFLOW-Diagramms unter SIMULINK

Ereignisse: Nach der Deklaration von Ereigniseingängen wird unter SIMULINK dem entsprechenden STATEFLOW-Block ein Triggereingang hinzugefügt. Dieser wird bei mehreren deklarierten Eingangseingängen als Vektor aufgefasst, wie in Abb. 3.35 zu sehen ist, die Zuordnung der Vektorelemente zu den Events geschieht über das *Index*-Feld in den Ereignisseigenschaften. Dieser Index wird im STATEFLOW-Explorer auch unter Scope angezeigt: *Input(n)*. Dort kann im Feld *Trigger* auch ausgewählt werden, ob bei einem Nulldurchgang des SIMULINK-Signals nur die steigende, nur die fallende oder ob beide Flanken als Ereignis interpretiert werden sollen. Für den Fall, dass ein Ereignis von einem Statechart zu einem zweiten Statechart übertragen

werden soll, ist der Triggertyp *function call* definiert, mit dem das aufgerufene Statechart verzögerungsfrei ausgeführt wird. Ereignisausgänge werden an einem Statechart einzeln und nicht vektoriell abgegriffen, im Gegensatz zu Ereigniseingängen stehen hier nur die Triggertypen *either edge* – d. h. es wird abwechselnd eine steigende und eine fallende Flanke mit Nulldurchgang erzeugt – und *function call* zur Verfügung.

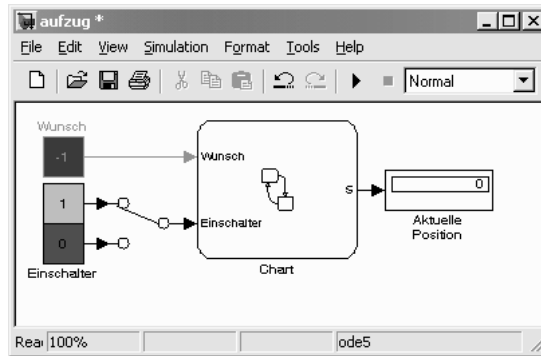


Abb. 3.36. Dateneingänge eines STATEFLOW-Diagramms unter SIMULINK

Daten: Für Daten ist neben den bisherigen Werten für den *scope* auch die Möglichkeit vorgesehen, ein Datum als Konstante, die nicht durch Ausdrücke in der *Action Language* verändert werden kann, oder als temporäre Variable zu definieren, die bei Beginn der Ausführung des Elternobjekts erzeugt und bei Beendigung zerstört wird. Im Eigenschaftsfenster eines Datums kann neben dem Variablentyp, der auf verschiedene Fließkomma-, Festkomma-, Ganzzahl- und Binärformate festgelegt werden kann, auch definiert werden, ob die Variable ein Skalar, ein Vektor oder eine Matrix sein soll. Der Wert unter *size* ist allerdings nicht dynamisch, sondern muss explizit angegeben werden. Lokale Variablen können mit einer Konstanten (*InitVal*) oder vom SIMULINK-Workspace des Charts (*FrWS*) initialisiert werden. Der letzte aktuelle Wert einer Variable kann auch nach Ausführung in den Workspace eines Modells geschrieben werden (*ToWS*). Während des Debuggens ist es möglich, auf die Einhaltung eines bestimmten Wertebereiches zwischen den mit *min* und *max* angegebenen Werten zu achten, oder bei Wahl von *watch* jede Veränderung des Datums angezeigt zu bekommen. Für jedes Eingangs- oder Ausgangsdatum wird ein Port unter SIMULINK erzeugt, dessen Reihenfolge über den *Index*-Eintrag festgelegt wird. Datenausgänge befinden sich bei nicht gedrehter Platzierung des Statecharts unter SIMULINK oberhalb von den Ereignisausgängen.

STATEFLOW-Diagramme werden anders als normale SIMULINK-Blöcke nicht zu jedem Zeitschritt der Simulation aufgerufen. Stattdessen kann im Ei-



Abb. 3.37. Eigenschaftsseite eines STATEFLOW-Diagramms

genschaftendialog eines Diagramms eingestellt werden, mit welcher Methode das Statechart aktiviert werden soll. Diese Eigenschaften können für jedes Statechart unterschiedlich sein. In Abb. 3.37 kann die Einstellung *Update Method* auf einen der folgenden Werte gestellt werden:

- Triggered or Inherited,
- Sampled,
- Continuous.

Hierbei bedeuten:

- Triggered
Enthält das *data dictionary* eines STATEFLOW-Diagramms mindestens einen Ereigniseingang, wird das Chart nur beim Auftreten externer Ereignisse durch Nulldurchgänge der kontinuierlichen Signale am Triggereingang „aufgeweckt“.
- Inherited, Sampled, Continuous
Gilt nur für Charts, für die *keine* Ereigniseingänge definiert wurden:
 - Inherited
Das Chart wird zu jedem Simulationszeitpunkt eines der Eingangssignale an den Dateneingängen ausgeführt, indem zu jedem dieser Zeitpunkte sogenannte implizite Ereignisse generiert werden. Diese Update-Methode heißt *Inherited*, da das Statechart die Abtastzeitpunkte aller eingehenden Signale „erbt“.

- **Sampled**
Bei dieser Update-Methode werden implizite Ereignisse mit der unter *Sample Time* eingegebenen Rate generiert, so dass das Chart mit dieser und nur mit dieser Rate periodisch aktiviert wird. Analog zur Einstellung *fixed-step* in den SIMULINK-Solver-Optionen.
- **Continuous**
Das Statechart wird bei Berechnung jedes Zeitschritts (major step) und auch bei Berechnung von Zwischenschritten (minor step) aufgerufen, falls der gewählte SIMULINK-Solver diese unterstützt. Analog zur Einstellung *variable-step* in den SIMULINK-Solver-Optionen.

3.3.2 Erweiterungen von Beschreibungsmitteln

Beispielhaft werden in diesem Abschnitt ein kontinuierliches und ein diskretes Beschreibungsmittel vorgestellt, die zu hybriden Beschreibungsmitteln erweitert wurden. Als Beschreibungsmittel des Kontinuierlichen wurde das Zustandsraummodell gewählt, das um Möglichkeiten des Umschaltens zwischen mehreren Dynamiken erweitert wurde. Als klassisches diskretes Beschreibungsmittel werden Automaten mittels Zuordnung von Differentialgleichungssystemen zu hybriden Automaten erweitert.

Erweitertes Zustandsraummodell

Ein (zeitkontinuierliches) hybrides System kann durch die Erweiterung des Zustandsraummodells gewonnen werden (siehe auch Kap. 2.10). Das gelingt, indem der kontinuierlichen Systembeschreibung mit n Zuständen, p Eingängen und q Ausgängen die Menge der diskreten Zustände \mathbb{M} , die Menge der diskreten Eingänge \mathbb{I} und die Menge der diskreten Ausgänge \mathbb{O} hinzugefügt werden. Hiermit ergibt sich ein hybrides System zu einem 7-Tupel \mathcal{H} .

- $\mathcal{H} = (\mathbb{R}^n \times \mathbb{M}, \mathbb{R}^p \times \mathbb{I}, \mathbb{R}^q \times \mathbb{O}, f, \phi, g, \gamma)$,
- mit dem hybriden Zustandsvektor $\mathbf{h}(t) = (\mathbf{x}^T(t), \mathbf{m}^T(t))^T$, gebildet aus dem kontinuierlichen Zustandsvektor $\mathbf{x}(t)$ und dem diskreten Zustandsvektor $\mathbf{m}(t)$,
- aus der nichtleeren Menge $\mathbb{H} = \mathbb{R}^n \times \mathbb{M}$ als hybridem Zustandsraum mit $\mathbf{h}(t) \in \mathbb{H}$,
- die Menge $\mathbb{R}^p \times \mathbb{I}$ als Raum der hybriden Eingänge von \mathcal{H} ,
- die Menge $\mathbb{R}^q \times \mathbb{O}$ als Raum der hybriden Ausgänge von \mathcal{H} ,
- der kontinuierlichen Übergangsfunktion $f : D_f \rightarrow \mathbb{R}^n$ mit $D_f \subseteq \mathbb{R}^n \times \mathbb{M} \times \mathbb{R}^p$
- der diskreten Übergangsfunktion $\phi : D_\phi \rightarrow \mathbb{M}$ mit $D_\phi \subseteq \mathbb{R}^n \times \mathbb{M} \times \mathbb{R}^p \times \mathbb{I}$
- der kontinuierlichen Ausgangsfunktion $g : D_g \rightarrow \mathbb{R}^q$ mit $D_g \subseteq \mathbb{R}^n \times \mathbb{M} \times \mathbb{R}^p$
- der diskreten Ausgangsfunktion $\gamma : D_\gamma \rightarrow \mathbb{O}$ mit $D_\gamma \subseteq \mathbb{R}^n \times \mathbb{M} \times \mathbb{R}^p \times \mathbb{I}$

Die Systemdarstellung nach Gl.(3.52) lässt sich in einem Wirkungsplan allgemein auch wie in Abb. 3.38 darstellen.

$$\begin{aligned}
 \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{m}(t), \mathbf{u}(t)), \\
 \mathbf{m}^+(t) &= \phi(\mathbf{x}(t), \mathbf{m}(t), \mathbf{u}(t), \mathbf{i}(t)), \\
 \mathbf{y}(t) &= g(\mathbf{x}(t), \mathbf{m}(t), \mathbf{u}(t)), \\
 \mathbf{o}^+(t) &= \gamma(\mathbf{x}(t), \mathbf{m}(t), \mathbf{u}(t), \mathbf{i}(t)).
 \end{aligned}
 \tag{3.52}$$

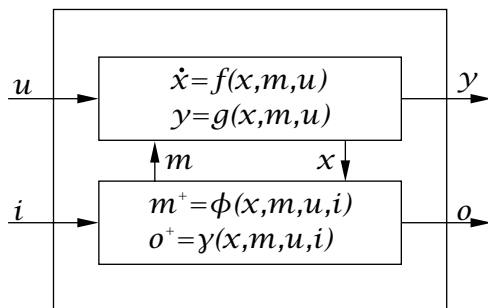


Abb. 3.38. Wirkungsplandarstellung eines erweiterten Zustandsraummodells

In Gl.(3.53) wird ein einfaches System zweier geschalteter Differentialgleichungen mit zwei Modi a und b gezeigt ($\mathbb{M} = \{a, b\}$), das vollkommen autonom ist, so dass $\mathbb{R}^p, \mathbb{R}^q, \mathbb{I}$ und \mathbb{O} leere Mengen sind.

$$\dot{\mathbf{x}}^a = \begin{pmatrix} -1 & -100 \\ 10 & -1 \end{pmatrix} \cdot \mathbf{x}^a \quad \text{und} \quad \dot{\mathbf{x}}^b = \begin{pmatrix} -1 & 10 \\ -100 & -1 \end{pmatrix} \cdot \mathbf{x}^b
 \tag{3.53}$$

$$\phi = \left\{ \begin{aligned} m_{a \rightarrow b} &= \{ \mathbf{x} \in \mathbb{R}^2 \mid x_2 = -0.2 \cdot x_1 \}, \\ m_{b \rightarrow a} &= \{ \mathbf{x} \in \mathbb{R}^2 \mid x_2 = 5 \cdot x_1 \} \end{aligned} \right\}
 \tag{3.54}$$

Wichtige Fragen wie z. B. die nach der Stabilität für derartige geschaltete lineare Systeme lassen sich nicht mehr einfach beantworten. Schon bei relativ übersichtlichen Systemen können Umschaltungen sehr schnell zu unerwartetem Verhalten führen, wie das Beispiel zweier linearer Systeme zeigt, die in Zustandsraumdarstellung durch Gl.(3.53) und gegeben sind, wenn Gl.(3.54) zur Bestimmung der Umschaltunkte zwischen den Systemen benutzt wird. Die Systeme haben mit $\lambda_{1,2} = -1 \pm j \cdot \sqrt{1000}$ die gleichen Eigenwerte und

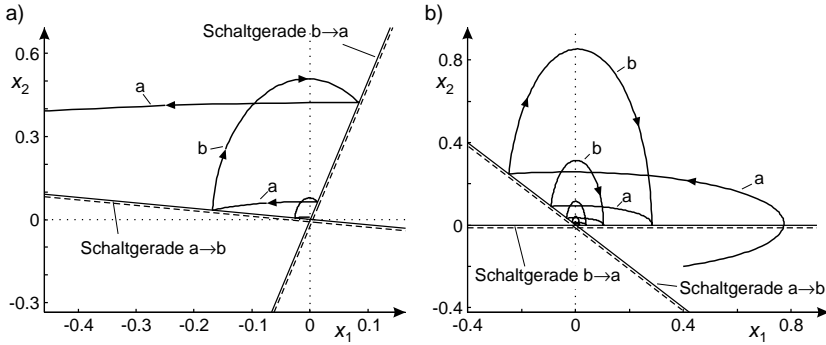


Abb. 3.39. Beispiel für die Modellierung mit geschalteten Differentialgleichungen – a) Graph zu Gl.(3.53)/(3.54), b) Graph zu Gl.(3.55)/(3.56)

sind beide stabil. Da die Eigenwerte zudem weit vom Stabilitätsrand entfernt liegen, könnte man erwarten, dass auch eine Verschaltung der beiden Systeme stabil sein wird. In Abb. 3.39a ist der Verlauf der beiden Zustandsgrößen für die Anfangsbedingung $\mathbf{x}_0 = (0, 0.01)^T$ dargestellt.

Ausgehend von der Anfangsbedingung ist zunächst die Systemdynamik a gültig. Umgeschaltet von der Systemdynamik a zur Systemdynamik b wird immer dann, wenn die Systemtrajektorie von a auf die Schaltgerade $x_b = 0.2 \cdot x_a$ trifft. Umgekehrt wird immer von der Systemdynamik b nach a gewechselt, wenn die Trajektorie von b auf die Schaltgerade $x_b = 5 \cdot x_a$ trifft. Durch diese Schaltstrategie kommt es zu einem ständigen Wechsel zwischen den beiden Systemdynamiken und letztlich zu instabilem Verhalten.

Umgekehrt kann man für zwei gleichermaßen nach Gl.(3.55) und (3.56) aufgebaute instabile Systeme mit den Eigenwerten $\lambda_{1,2} = 1 \pm j \cdot \sqrt{1000}$ zeigen, dass deren Verschaltung bei korrekter Wahl der Umschaltunkte ein stabiles hybrides Gesamtsystem ergibt, dessen Trajektorie für alle Startwerte in den Ursprung konvergiert (Abb. 3.39b).

$$\dot{\mathbf{x}}^a = \begin{pmatrix} 1 & -100 \\ 10 & 1 \end{pmatrix} \cdot \mathbf{x}^a \text{ und } \dot{\mathbf{x}}^b = \begin{pmatrix} 1 & 10 \\ -100 & 1 \end{pmatrix} \cdot \mathbf{x}^b \quad (3.55)$$

$$\phi = \left\{ \begin{aligned} m_{a \rightarrow b} &= \{ \mathbf{x} \in \mathbb{R}^2 \mid x_2 = -x_1 \}, \\ m_{b \rightarrow a} &= \{ \mathbf{x} \in \mathbb{R}^2 \mid x_2 = 0 \} \end{aligned} \right\} \quad (3.56)$$

Erweiterte Automaten

Ohne formalen Anforderungen zu genügen, soll in diesem Abschnitt der umgekehrte Weg zur hybriden Systemdarstellung gegangen werden: die Erweiterung

von Automaten zu hybriden Automaten, deren Grundelemente in Abb. 3.40 dargestellt werden. Mittels der festen Zuordnung von Differentialgleichungssystemen $F_z(\dot{\mathbf{x}}, \mathbf{x}, t)$ zu jedem Zustand z eines Automaten steht jeder Zustand für ein bestimmtes Verhalten des Systems – deshalb heißen die Zustände von hybriden Automaten auch *Modi*. Jeder Zustandsübergang aus der Menge der Zustandsübergänge E kann nun zusätzlich zur Auslösung durch Ereignisse von den Zustandsvariablen abhängen und es kann eine Initialisierung der Zustandsvariablen während des Übergangs erfolgen. Dies wird durch die Sprungfunktion des hybriden Automaten beschrieben, die jedem Zustandsübergang e_i eine Bedingungsfunktion g_i und eine Initialisierungsfunktion h_i zuweist. Ein Zustandsübergang kann somit nur stattfinden, wenn die Übergangsbedingung erfüllt ist, er muss aber nicht. Um einen Zustandsübergang auch erzwingen zu können, wird jedem Zustand z des hybriden Automaten eine Invariante INV_z zugewiesen. Die Invariante muss erfüllt sein, wenn der Automat in den Zustand übergeht und solange er sich in diesem befindet. Schließlich kann jedem möglichen Anfangszustand des Automaten noch eine Anfangsbedingung zugewiesen werden, die grafisch durch eine Kante ohne Quellknoten dargestellt wird. Der hybride Gesamtzustand eines hybriden Automaten ergibt sich aus dem Modus, in dem sich der Automat gerade befindet, und den aktuellen Werten der Zustandsvariablen. Zustandsänderungen finden statt durch die kontinuierliche Änderung der Zustandsvariablen, während sich der Automat in einem Modus aufhält, oder durch einen diskreten Wechsel des Modus.

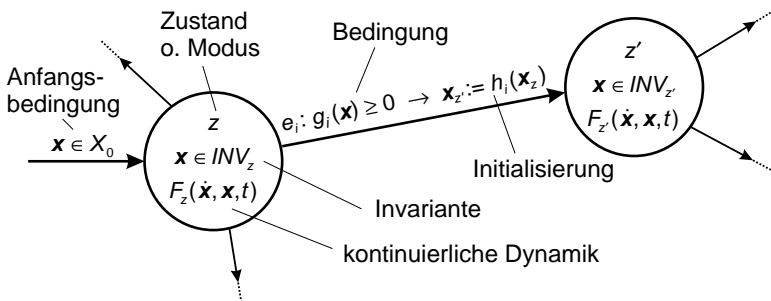


Abb. 3.40. Elemente eines hybriden Automaten

Zur Veranschaulichung zeigt Abb. 3.41 das Beispiel eines Billardtisches mit einer schwarzen und einer weißen Kugel. Der Tisch ist h Längeneinheiten breit und l Längeneinheiten lang und die schwarze Kugel befindet sich anfangs an der Position (x_s, y_s) . Die schwarze Kugel wird angestoßen und bewegt sich mit der konstanten Geschwindigkeit v . Trifft die Kugel nun auf eine Bande, kommt es zu einem voll elastischen Stoß und die entsprechende Geschwindigkeitskomponente ändert ihr Vorzeichen. Abbildung 3.41 zeigt

rechts den hybriden Automaten für die Bewegung der schwarzen Kugel. Jede der vier möglichen Vorzeichenkombination der beiden Geschwindigkeitskomponenten wird durch einen eigenen Zustand im Automaten dargestellt. Die Zustandsübergänge des Automaten korrespondieren mit der Berührung einer der Banden durch die Kugel. Mit den Invarianten wird dabei der Übergang exakt zum Zeitpunkt der Bandenberührung erzwungen.

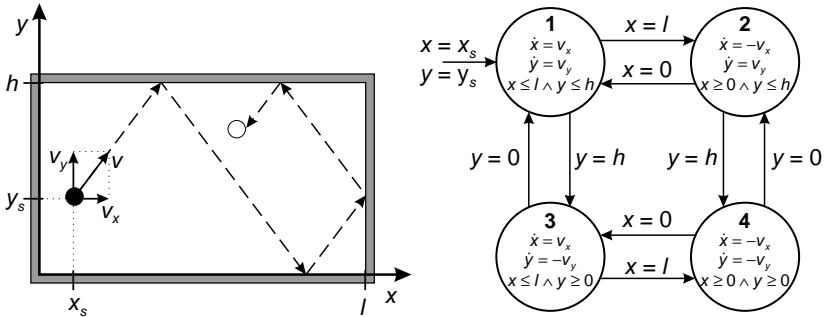


Abb. 3.41. Beispiel eines Billardtisches mit hybridem Automaten zur Darstellung der Bewegung der schwarzen Kugel

Ein wesentliches Problem bei der Modellierung von Systemen mit hybriden Automaten wird deutlich, wenn im Billardtischbeispiel die schwarze Kugel auf die weiße trifft und sich beide Kugeln in unterschiedlicher Richtung weiter bewegen. Die Anzahl der darzustellenden Zustände des Automaten wächst exponentiell mit der Zahl nebenläufiger Prozesse (Zustandsexplosion). Abhilfe bietet hier die Modellierung des Systems durch mehrere miteinander kommunizierende Automaten oder durch ein Hierarchiekonzept, wie es z. B. in den Statecharts umgesetzt ist.

3.4 Modellabstraktionen

Bei den in Kap. 3.3 beschriebenen Ansätzen zur Modellierung hybrider Systeme wird ein kontinuierliches oder diskretes Beschreibungsmittel um Methoden zur Darstellung diskreter bzw. kontinuierlicher Dynamik erweitert. Im Gegensatz dazu wird bei den im Folgenden vorgestellten Modellansätzen versucht, durch geeignete Approximations- bzw. Abstraktionsverfahren die kontinuierlichen oder diskreten Teile des Verhaltens hybrider Systeme mit einem diskreten bzw. kontinuierlichen Modell zu beschreiben. Das anschließend in einer der beiden bekannten Domänen beschriebene Verhalten des hybriden Systems kann auf diese Weise mit vorhandenen, diskreten oder kontinuierlichen Beschreibungsmitteln dargestellt werden. Modifikationen im Modell oder

der Theorie des Beschreibungsmittels sind nicht notwendig. Die Aufgabe bei diesen Ansätzen besteht vielmehr darin, eine geeignete Abstraktion des ursprünglich hybrid modellierten Systems zu finden. Die gefundene Abstraktion genügt dann der Definition in Kap. 3.3 naturgemäß nicht mehr.

3.4.1 Diskrete Abstraktionen

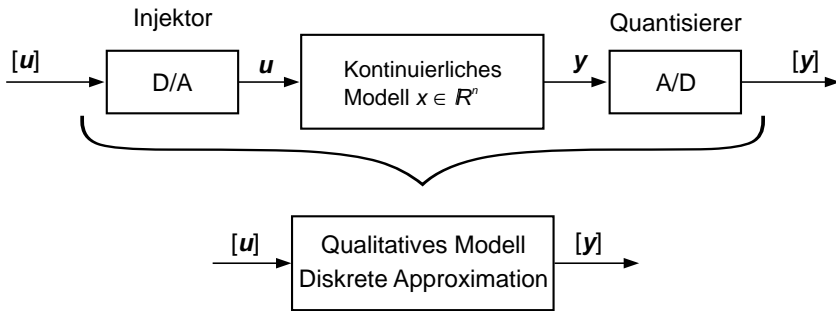


Abb. 3.42. Kontinuierlich-diskretes Originalsystem und seine diskrete Approximation

Ausgangspunkt für die qualitative Modellbildung ist die in Abb. 3.42 dargestellte Situation: Eine endliche Menge U diskreter Stellsignale $[u]$ wird über eine eindeutige Abbildung (Injektion) in ein reellwertiges Signal u umgesetzt. Dieses wirkt auf ein wertekontinuierliches System, dessen Zustand sich im \mathbb{R}^n bewegt. Als Messinformation steht eine quantisierte Version $[y]$ der reellwertigen Ausgangsgröße y aus der Menge Y zur Verfügung. Aufgabe ist also, den durch das wertekontinuierliche System, die Injektion und den Quantisierer vermittelten Zusammenhang zwischen $[u]$ und $[y]$ durch ein qualitatives Modell zu approximieren.

Damit die Approximation des Originalsystems vollständig ist, muss sie für alle relevanten Kombinationen von Ein- und Ausgangssignalen des Originalsystems $\mathcal{B}_C \subseteq U \times Y$ aus U und Y definiert sein. Das Verhalten \mathcal{B}_A des approximierten Systems darf durchaus in weiteren Bereichen definiert sein, muss aber der Bedingung $\mathcal{B}_C \subseteq \mathcal{B}_A$ gehorchen. Je kleiner die Differenzmenge $\mathcal{B}_A \setminus \mathcal{B}_C$ ist, desto genauer ist die Approximation. Wird die Approximation zu ungenau, die Menge \mathcal{B}_A also zu groß, können die zu betrachtenden Fragestellungen vielfach nicht mehr oder nur falsch beantwortet werden, da das Modell Möglichkeiten des Systemverhaltens bietet, die das Originalsystem nicht besitzt. Je kleiner die Differenzmenge $\mathcal{B}_A \setminus \mathcal{B}_C$ ist, umso weniger nicht zutreffende Lösungen werden erzeugt. Mit einer Erhöhung der Genauigkeit steigt aber im Gegenzug der Rechenaufwand und die Modellgröße stark an, so dass wie bei jeder allgemeinen Modellbildung ein geeigneter Kompromiss zwischen Genauigkeit und Komplexität gefunden werden muss.

Bei der Anwendung qualitativer Modelle ist ein weiterer entscheidender Aspekt zu beachten: Qualitative Modelle sind in aller Regel nicht-deterministisch. Der Nicht-Determinismus resultiert aus der unvollständigen Kenntnis des exakten Anfangszustandes des wertekontinuierlichen Systems. Bei gleicher Folge von Ein- und Ausgängen kann das System bei qualitativ gleichem Anfangszustand (der eine Menge wertekontinuierlicher Zustände umfasst) unterschiedliche Trajektorien erzeugen. Der Nicht-Determinismus kann entsprechend auch nicht durch ein beliebig genaues Modell verhindert werden.

Ein typischer Ansatz bei der diskreten Abstraktion hybriden Verhaltens ist, neben der Quantisierung von Ein- und Ausgangssignalen auch den Zustandsraum diskret zu abstrahieren und ihn durch eine Menge qualitativer Zustände $[x]$ darzustellen. Der Zustandsraum wird also in einzelne Zellen aufgeteilt, die sich normalerweise nicht überlappen und den physikalisch sinnvollen Bereich des Zustandsraumes (Wertebereich, den die Zustandsgrößen im physikalischen System annehmen können) abdecken. Diese qualitativen Zustände bilden die diskrete Zustandsmenge des qualitativen Modells. In einem zweiten Schritt werden dann alle möglichen Zustandsübergänge zwischen den qualitativen Zuständen ermittelt.

Abbildung 3.43 zeigt einen Ausschnitt aus dem Trajektorienfeld eines kontinuierlichen Systems mit zwei Zustandsgrößen $x_1(t)$ und $x_2(t)$ bei konstanter Eingangsgröße. Der Zustandsraum wurde in vier Bereiche aufgeteilt, für die rechteckige Zellen innerhalb des physikalisch sinnvollen Bereichs $x_1 \in [x_{1min}, x_{1max}]$ und $x_2 \in [x_{2min}, x_{2max}]$ der Zustandsgrößen gewählt wurden. Jede der Zellen beschreibt einen qualitativen Zustand des Systems. Zwei benachbarte Zustände werden über eine Transition verbunden, wenn im System ein Übergang von dem einen qualitativen Zustand in den anderen möglich ist. Die Transitionen werden mit der zugehörigen Bedingung an die kontinuierlichen Variablen beschriftet. Dasselbe Vorgehen muss nun ebenfalls für alle Möglichkeiten des diskret abstrahierten Eingangssignals durchgeführt werden – dieser Vorgang ist in Abb. 3.43 nicht dargestellt. Um ein vollständiges qualitatives Modell zu erhalten, müssen *sämtliche* Kombinationen von qualitativen Zuständen mit den in diesen Zuständen sinnvollen Eingangsgrößen berücksichtigt werden.

3.4.2 Kontinuierliche Abstraktionen

Ein sehr schwer wiegendes Problem bei der Modellierung diskreter Systeme ist das der *Zustandsexplosion*. Eine Möglichkeit der Vereinfachung ist es, diskrete Teile eines hybriden Systems als kontinuierlich zu abstrahieren. Diese Vereinfachungs- oder Relaxationstechnik wird vom Menschen im Grunde alltäglich verwendet – die Physik lehrt, dass alle Materie aus kleinsten Teilchen aufgebaut ist, elektrische Energie nur als Quantum auftreten kann etc. Trotzdem wäre es unpraktisch, sich diesen Charakter bei der großen Menge zählbarer Einheiten dauernd zu vergegenwärtigen, so dass bei Materie vom

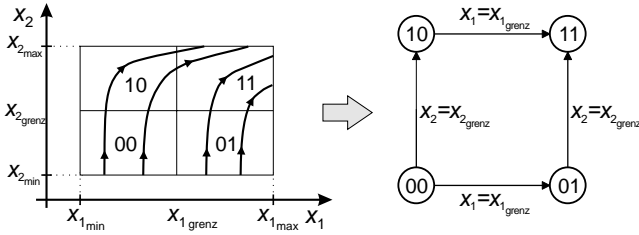


Abb. 3.43. Beispiel einer diskreten Approximation als Automat

Kontinuum gesprochen wird oder elektrische Größen als reelle Zahlen mit Dimension hingenommen werden. Aber auch bei deutlich kleineren diskreten Mengen lohnt sich die kontinuierliche Abstraktion des Problems, etwa bei der Untersuchung eines großen Computernetzwerks, in dem zahlreiche Rechner (diskrete) Daten anfordern und liefern. Die Idee zu einem auf Petrinetzen basierenden Ansatz soll im folgenden kurz vermittelt werden.

Fasst man die Markierung einer Petrinetzstelle als nichtnegativen kontinuierlichen Wert $m \in \mathbb{R}_0^+$ auf und fasst die Änderung der Markierung in Gl.(3.50) als kontinuierlich auf – also als Ableitung der Markierung –, so ergibt sich die Grundgleichung der *kontinuierlichen Petrinetze*

$$\dot{m} = N \cdot f \quad . \tag{3.57}$$

Der Schalthäufigkeitsvektor v wurde hierin durch den Flussvektor f ersetzt, der eine Funktion der Markierung ist. Hierbei kann der Fluss durch eine Transition von begrenzter Geschwindigkeit $f_{t_i}(t) \leq \lambda(t_i)$ eingeschränkt sein, so dass eine Transition mit einer konstanten Geschwindigkeit schaltet, solange die Markierung einer Stelle im Vorbereich nicht Null wird. Durch die Differentiation stellen sich aufgrund des schnellen Geschwindigkeitswechsels zwischen Null und $\lambda(t_i)$ auch Zwischenwerte ein, vergleichbar etwa mit einem nichtlinearen 2-Punkt-Glied, das ‚im ‚Umschaltpunkt betrieben wird und durch unendlich häufiges und schnelles Schalten quasi Werte zwischen den beiden Punkten erzeugt.

Eine weitere Möglichkeit ist es, den Fluss durch eine Transition nicht zu begrenzen, sondern proportional von der Markierung des Vorbereichs abhängig zu machen; auch hier ist gewährleistet, dass die Markierung einer Stelle nicht kleiner als Null wird, da der Fluss durch eine Transition im Nachbereich einer nicht markierten Stelle ebenfalls Null ist.

Durch die Wahl von f können stückweise lineare Gleichungssysteme oder stückweise lineare Differentialgleichungssysteme aufgebaut werden, deren Analyse vorteilhaft jeweils zum Teil mit Methoden der Netztheorie *und* mit Methoden der kontinuierlichen Systemtheorie durchgeführt werden kann.

Lässt man bei diesem Modellierungsansatz auch diskrete Transitionen zu, bilden die hybriden Petrinetze ein mächtiges Modellierungswerkzeug, in dem kontinuierliche und diskrete Modellierung in einer Darstellungsform integriert sind, dessen Analyse- und Syntheseverfahren allerdings bei weitem noch nicht ausreichend erforscht sind.

