Regelungstheorie für ereignisdiskrete Systeme zur modellbasierten Berechnung von SPS-Programmen

Sebastian Perk, Thomas Moor, Klaus Schmidt
Universität Erlangen-Nürnberg, Lehrstuhl für Regelungstechnik,
Cauerstr. 7, 91058 Erlangen
E-Mail: sebastian.perk@rt.eei.uni-erlangen.de

Zusammenfassung

In diesem Beitrag soll die Regelungstheorie für ereignisdiskrete Systeme vorgestellt werden, welche die modellbasierte Berechnung von SPS-Programmen ermöglicht. Dazu wird als theoretische Grundlage die Supervisory Control Theory nach R.J. Ramadge und M. Wonham erläutert, welche eine formale Methode zur Berechnung von nachweislich korrektem Programmcode bereitstellt, sich jedoch aufgrund des erheblichen Rechenaufwands nicht für Systeme praxisrelevanter Größe eignet. Die Grundideen moderner Verfahren, die dieses Komplexitätsproblem auf verschiedene Arten angehen, werden im Anschluss vorgestellt. Der Beitrag schließt mit einem Einblick in weitere Arbeiten unserer Forschungsgruppe zu diesem Thema.

1. Einleitung

Zahlreiche technische Systeme, wie Telekommunikationsnetzwerke, Verkehrssysteme, Logistik, Fertigungsanlagen, Computerprogramme etc. weisen eine ereignisdiskrete Dynamik auf. Das heißt, diskrete Ereignisse wie Sensorsignale oder das Betätigen einer Taste bewirken eine ebenfalls diskrete Änderung des Systemzustands (z.B. Betriebszustand einer Maschine). Für viele derartige Automatisierungsaufgaben werden in der Industrie speicherprogrammierbare Steuerungen (SPS) eingesetzt, welche eine schnelle und zuverlässige Programmabarbeitung ermöglichen. Die für den Prozessablauf nötigen logischen und zeitlichen Bedingungen werden vom Entwickler vorgegeben und in ein SPS-Programm umgesetzt. Dem Entwickler fällt somit die anspruchsvolle Aufgabe zu, das gewünschte *gesteuerte* Prozessverhalten korrekt vorzugeben.
Mit wachsender Automatisierung technischer Prozesse steigt deren Komplexität jedoch drastisch an. Aufgrund zunehmender Unüberschaubarkeit für den Entwickler ist ein fehlerfreier bzw. optimierter Prozessablauf nicht mehr garantiert, mit Folgen wie suboptimaler Auslastung, Systemausfällen beispielsweise durch Blockierungen, bis hin zu sicherheitskritischem Fehlverhalten.
Daher wünscht man sich für ereignisdiskrete Systeme ein an die klassische Regelungstheorie angelehntes, modellbasiertes Entwurfsverfahren, welches ausgehend von einem Modell des *ungesteuerten* Prozesses (≅Regelstrecke) und einer formalen Spezifikation (≅Entwurfsziel) ein Steuerungsprogramm (≅Regler) bestimmt, so dass im gesteuerten Prozess (≅geschlossener Regelkreis) die Spezifikation *nachweislich* erfüllt wird.
Hierzu wurde in den späten 1980'er Jahren mit der Supervisory Control Theory (SCT) von R.J. Ramadge und M. Wonham ein systematisches Verfahren bereitgestellt, welches die automatische Synthese von Steuerungsprogrammen für ereignisdiskrete Systeme ermöglicht. In diesem Ansatz werden sowohl der Prozess wie auch die Spezifikation durch reguläre Sprachen in Form von Zustandsautomaten modelliert. In [3] wird ein Entwurfsalgorithmus vorgestellt, der ausgehend vom Prozessmodell und einer Spezifikation eine Steuerung, genannt Supervisor, berechnet, die Sicherheit (Einhaltung der Spezifikation) und Lebendigkeit (Erfüllung der Aufgaben) nachweislich *garantiert* und zugleich das Systemverhalten *minimal einschränkt*. Die resultierende Steuerung liegt ebenfalls als Zustandsautomat vor, welcher direkt z.B. in ein SPS-Programm übersetzt werden kann.
Dennoch ist der breite Einzug dieses Verfahrens in die industrielle Anwendung bislang ausgeblieben. Die Ursache für diesen vermeintlichen Rückstand der Anwendung gegenüber der verfügbaren Methodik liegt in der so genannten „Zustandsraumexplosion“: die Größe (Anzahl der Zustände) des Automatenmodells eines aus mehreren Komponenten bestehenden Prozesses (beispielsweise mehrere Module einer Fertigungslinie) hängt exponentiell von der Anzahl der Komponenten ab. Zahlenbeispiel: ein Prozess bestehe aus 10 Komponenten, die jeweils 20 verschiedene Zustände einnehmen können; dann weist das zusammengesetzte (monolithische) Modell $20^{10} \approx 10^{12}$ Zustände auf. Der Grund dafür liegt in der fehlenden Ausnutzung der Struktur des Prozesses: bestimmte Teilaufgaben der Spezifikation betreffen nur Teile des Gesamtprozesses, und übergeordnete Steuerungsaufgaben betreffen lediglich die Interaktion der Prozesskomponenten, wozu eine explizite Darstellung des Gesamtrozesses im Detail nicht erforderlich ist.
Seit rund 15 Jahren wird daher intensiv nach Entwurfsverfahren geforscht, die auf der SCT aufbauen, jedoch durch Nutzung der Prozessstruktur besser mit der Komponentenanzahl skalieren und so auf Systeme praxisrelevanter Größe anwendbar sind.
In diesem Beitrag wird mit Abschnitt 2 zunächst die SCT als Basis jener modernen Verfah-

ren in Grundzügen erläutert. Abschnitt 3 schildert die Grundideen strukturierter Ansätze zur Vermeidung des exponentiellen Rechenaufwands. Abschnitt 4 geht auf diesbezügliche Arbeiten unserer Forschungsgruppe ein.

2. Modellbasierter Reglerentwurf für ereignisdiskrete Systeme

Die automatische Berechnung einer Steuerung eines ereignisdiskreten Prozessablaufs wird nach den theoretischen Grundlagen von P.J.Ramadge und W.M.Wonham [3] als regelungstechnisches Problem aufgefasst. Ausgangspunkt dafür ist die modellhafte Beschreibung des ungesteuerten Prozesses als ereignisdiskretes System, kurz DES (Discrete Event System). Mit diesem Modell und einer Spezifikation, d.h. einer Beschreibung des Wunschverhaltens, lässt sich ein so genannter Supervisor berechnen, der die Steuerung entsprechend der Spezifikation übernimmt. Der Supervisor kann algorithmisch bestimmt und ebenso wie das Streckenmodell als DES dargestellt werden. Die Implementierung des Supervisors erfolgt z.B. als Programm auf einer SPS.
Es folgt zunächst eine Einführung in die Klasse der ereignisdiskreten Systeme sowie die SCT. Eine ausführliche Beschreibung bietet das Standardwerk [1], sowie die im Internet verfügbare Dissertation [4].

Ereignisdiskrete Systeme.

Ein ereignisdiskretes System ist ein System, welches nur diskrete Zustände annehmen kann. Sobald ein ebenfalls diskretes Ereignis auftritt, wechselt das System vom aktuellen in den nachfolgenden Zustand. Dabei spielt die Zeit, die zwischen zwei Ereignissen vergeht, keine Rolle, es wird nur die Tatsache und die Reihenfolge des Auftretens betrachtet. Die Menge aller im Prozess auftretenden Ereignisse wird als Alphabet Σ bezeichnet. Das Verhalten eines DES lässt sich mit der Menge aller möglichen Sequenzen von Ereignissen, der Sprache L, beschreiben.
Eine mögliche Darstellungsform einer solchen Sprache ist der endliche Zustandsautomat. Ein Automat G besteht aus einem 5-Tupel von Zuständen Q, Alphabet Σ, der Transitionenrelation δ, dem Startzustand q_0 und markierten Zuständen Q_m:

$$G := (Q, \Sigma, \delta, q_0, Q_m)$$

Q ist die Menge aller Zustände des Automaten. Das Alphabet Σ ist die Menge aller Ereignisse, die im Automaten auftreten können. Die Transitionenrelation δ beschreibt die möglichen Zustandsübergänge, die durch Ereignisse getriggert werden. Ausgangspunkt ist der Startzustand q_0, und markierte Zustände Q_m sind gewünschte Zustände, die z.B. das erfolgreiche Abschließen eines Produktionsschrittes anzeigen.
Ein Automat lässt sich durch einen gerichteten Graph visualisieren. In der folgenden Abbildung ist das Automatenmodell G eines Förderbandes dargestellt, welches über ein Laufband Werkstücke nach links oder rechts transportieren kann. Werkstücke können mit einem mittig platzierten Sensor detektiert werden.

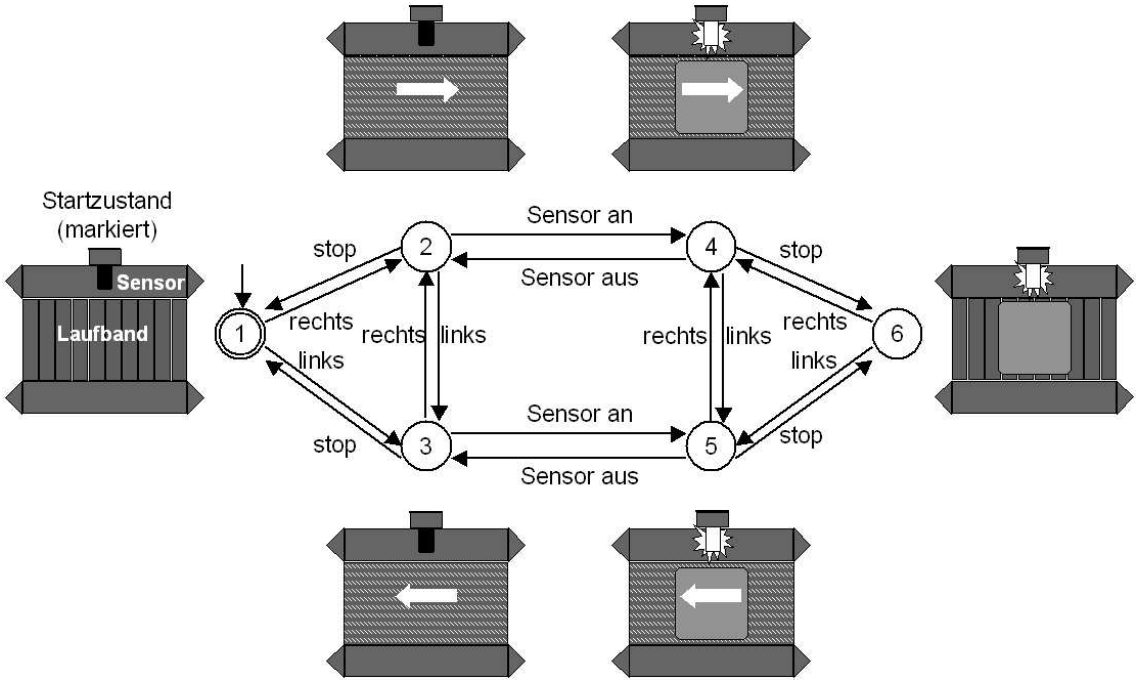

Abbildung 1: Automatenmodell eines Förderbandes

Die grafische Darstellung der Zustände erfolgt durch Kreise, in deren Inneren sich die jeweilige Bezeichnung des Zustands, z.B eine Nummer, befindet. Die Transitionen werden durch Pfeile zwischen den Zuständen angezeigt. Dabei zeigt der Pfeil vom Ausgangszustand zum Zielzustand und ist mit dem Ereignis beschriftet, welches auftreten muss, um diese Transition zu initiieren. Beispielsweise führt in der Abbildung eine Transition mit dem Ereignis „rechts“ von Zustand 1 nach Zustand 2. Der Startzustand q_0 ist durch einen Pfeil ohne Ausgangszustand gekennzeichnet. Die Wahl dieses Anfangszustands hängt vom gewünschten Initialzustand des realen Systems ab, hier wurde das stillstehende Förderband ohne Werkstück (Zustand 1) gewählt. Markierte Zustände sind durch einen Doppelkreis gekennzeichnet. Beim Förderband wurde als einziger markierter Zustand der Startzustand gewählt, da die Bereitschaft zur Beförderung des nächsten Werkstücks zugleich den Abschluss der Beförderung des vorangegangenen Werkstücks darstellt.

Supervisorentwurf.

Hat man sich ein Modell des ungesteuerten Prozesses, also des *möglichen* Verhaltens verschafft, so liegt der nächste Schritt in der Berechnung einer Steuerung, die in der SCT Supervisor genannt wird. Von einem Supervisor wünscht man sich folgende Eigenschaften:

1. Einhaltung einer Spezifikation (Sicherheit): das gesteuerte System soll nur *gewünschtes* Verhalten aufweisen.

2. Blockierungsfreiheit (Lebendigkeit): das gesteuerte System soll gewünschte Abläufe stets zum Abschluss bringen können.

3. Minimalrestriktion: unter den Bedingungen 1. und 2. soll das Systemverhalten minimal eingeschränkt werden.

zu 1. Der erste Schritt des Supervisorentwurfs liegt demnach in der Formulierung einer Spezifikation, also des *gewünschten* Verhaltens. So könnte man sich für das Förderbandbeispiel in

Abbildung 1 wünschen, dass das Band zur Schonung des Antriebes vor jedem Wechsel der Laufrichtung gestoppt wird. Eine ruckartige Richtungsumkehr, die durch die direkten Transitionen zwischen den Zuständen 2 und 3 bzw. 4 und 5 gekennzeichnet ist, möchte man also vermeiden. Eine entsprechende Spezifikation lässt sich ebenfalls als Automatenmodell formulieren, siehe folgende Abbildung.

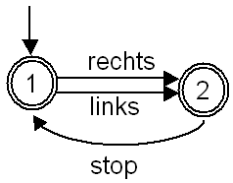

Abbildung 2: Spezifikation für das Förderband

Das Automatenmodell der Spezifikation für das Förderband beschreibt also die Forderung, dass sich die Bewegungsereignisse „rechts“ und "links“ mit dem „stop“-Ereignis abwechseln sollen. Die offensichtlich simple Struktur der Spezifikation ist darin begründet, dass man lediglich ausdrücken muss, *was* man von der zu berechnenden Steuerung fordert, nicht aber, *wie* man für die Einhaltung der Spezifikation im gesteuerten Prozess sorgt.
Um einen Supervisor zu entwerfen, der das System so steuert, dass die Spezifikation eingehalten wird, muss zwischen steuerbaren und nichtsteuerbaren Ereignissen unterschieden werden. Steuerbar sind Ereignisse, die vom Supervisor unmittelbar beeinflusst werden können, z.B. das Ein- und Ausschalten von Aktoren. Nicht steuerbar sind Ereignisse, die nicht unmittelbar beeinflußbar sind, wie z.B. Sensorsignale. Im Förderbandbeispiel sind die Aktorereignisse „rechts“, „links“ und „stop“ steuerbar, entsprechend sind die Sensorereignisse „Sensor an“ und „Sensor aus“ nicht steuerbar.
Die Idee der Supervisory Cotrol Theory ist nun, dass der Supervisor das ungesteuerte Verhalten des Systems soweit einschränkt, dass die Spezifikation nicht verletzt wird. Es ergibt sich eine geschlossene Rückführstruktur wie nachfolgend abgebildet.

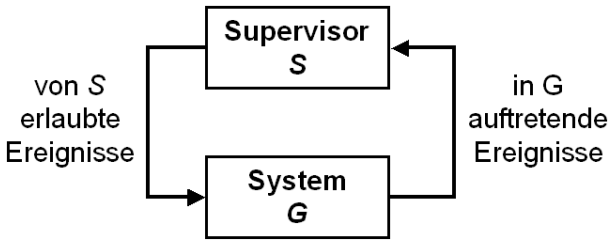

Abbildung 3: Rückführstruktur aus System und Supervisor

Der Supervisor beobachtet alle auftretenden Ereignisse und verhindert gegebenenfalls Ereignisse so, dass die vorgegebene Spezifikation erfüllt ist (*Sicherheit*). Damit der Supervisor realisierbar ist, darf er ausschließlich steuerbare Ereignisse verhindern.
zu 2. Des Weiteren muss der Supervisor so ausgelegt sein, dass das Verhalten des gesteuerten Systems blockierungsfrei ist. Blockierungsfreiheit bedeutet, dass ein Automat stets durch eine Folge von Ereignissen aus jedem Zustand einen markierten, d.h. einen gewünschten Zustand erreichen kann. Wäre dies nicht der Fall, so könnte eine gewünschte Aufgabe nicht zum Abschluss gebracht werden. Folgende Abbildung zeigt mögliche Arten von Blockierungen.

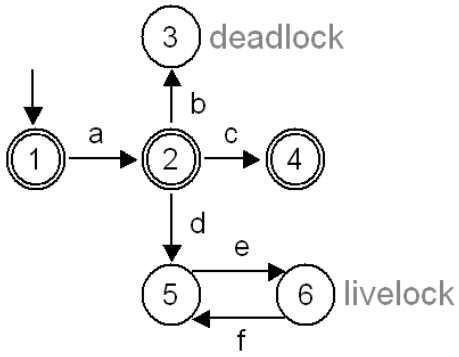

Abbildung 4: Arten von Blockierungen: deadlock und livelock

Die Blockierung in Zustand 3 ist dadurch gekennzeichnet, dass das System sich in einem nicht markierten Zustand befindet (also eine bestimmte Aufgabe noch nicht abgeschlossen hat) und kein weiteres Ereignis mehr möglich ist. Im Falle der Livelock (Zustände 5 und 6) hingegen sind stets weitere Ereignisse möglich, jedoch existiert ebenso kein Pfad zu einem markierten Zustand. Das System in einer Schleife gefangen. Das Systemverhalten muss also auch soweit eingeschränkt werden, dass keine Blockierungen mehr vorhanden sind. Dies muss wie in 1. durch Verhinderung steuerbarer Ereignisse geschehen.

zu 3. Ist der Supervisor minimal restriktiv, so werden nicht mehr Ereignisse unterbunden, als zur Einhaltung von Sicherheit und Lebendigkeit gerade nötig ist. Dies gelingt dann, wenn die Menge aller Lösungen für den Supervisor, welche die Bedingungen 1. und 2. erfüllen, ein *eindeutiges* Supremum besitzt. Wie Ramadge und Wonham bewiesen haben [3], ist dies stets der Fall.

Eine mögliche Realisierung eines Supervisors, der alle drei Bedingungen erfüllt, ist das Automatenmodell des gesteuerten Verhaltens, welches mit dem Algorithmus von Ramadge und Wonham berechnet werden kann. Für die Spezifikation aus Abbildung 2 ergibt sich folgende Lösung.

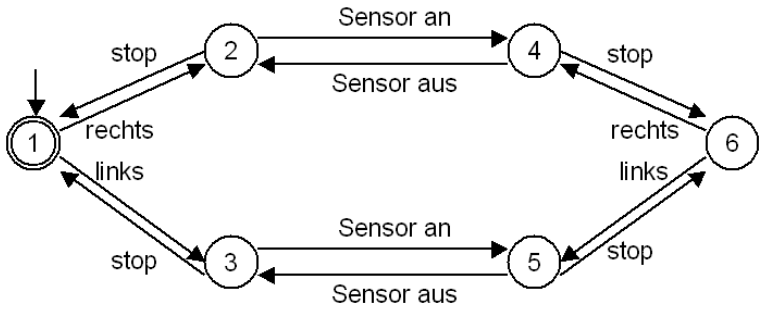

Abbildung 5: Gesteuertes Verhalten des Förderbandes

Im Vergleich zum ungesteuerten Verhalten in Abbildung 1 sieht man, dass der Supervisor die per Spezifikation verbotenen Richtungswechsel in den Zuständen 2, 3, 4 und 5 verhindert. Forderung 1. ist also erfüllt. Außerdem existiert von jedem Zustand aus ein Pfad in den markierten Zustand 1, das Ergebnis ist also gemäß Forderung 2. blockierungsfrei. Minimalrestriktion nach Forderung 3. erkennt man in diesem Beispiel daran, dass jedes in den Zuständen 2, 3, 4 und 5 verbotene Ereignis direkt zu einer Verletzung der Spezifikation führen würde.

Erzeugung von SPS-Code.

Die Implementierung des Supervisors als SPS-Programm besteht in der direkten Übersetzung des Automatenmodells aus Abbildung 5. Der aktuelle Zustand wird als Integerwert gespeichert. Der SPS-Programmzyklus beginnt mit dem Einlesen der Sensorwerte (z.B. Sensorwert des Förderbandes). Abhängig vom aktuellen Zustand springt das Programm zu Programmteilen (gekennzeichnet durch Sprungmarken), die jeweils den Zuständen des Automaten entsprechen. Hier wird nun ausgewertet, ob nichtsteuerbare Ereignisse (Änderungen der Sensorwerte) eingetreten oder steuerbare Ereignisse (Änderungen der Aktorwerte) auszuführen sind. Anschließend wird gegebenenfalls zur Sprungmarke des Folgezustands gesprungen und der Integerwert des Zustands aktualisiert.

Algorithmische Komplexität.

Der Algorithmus zur Berechnung des Supervisors skaliert in der Rechenzeit an sich gut mit der Zahl der Zustände von System und Spezifikation. Die Problematik liegt jedoch darin, dass der Algorithmus für die Berechnung ein zusammengesetztes, monolithisches Modell des Gesamtsystems benötigt. Möchte man beispielsweise die Steuerung für zwei benachbarte Förderbänder berechnen, so kann man diese zwar getrennt voneinander mit zwei Automaten zu je 6 Zuständen modellieren; zur Berechnung des Supervisors muss man aber das Gesamtverhalten in einem einzigen Automaten darstellen. Da in jedem der 6 Zustände des einen Förderbandes das zweite Förderband jeweils 6 verschiedene Zustände annehmen kann, ergeben sich im monolithischen Modell des Verhaltens zweier Förderbänder $6^2 = 36$ Zustände. Dieser als „Zustandsraumexplosion“ bekannte exponentielle Anstieg der Modellkomplexität führt dazu, dass der monolithische Supervisorentwurf bei Systemen praxisrelevanter Größe am nicht beherrschbaren Rechenaufwand scheitert.
Dass in diesem Abschnitt beschriebene Förderband ist Teil einer Versuchsanlage des Lehrstuhls für Regelungstechnik der Universität Erlangen-Nürnberg. Diese besteht aus einem Fischertechnik-Modell einer industriellen Fertigungsanlage, gesteuert von einer SPS (Siemens Simatic S7-300).

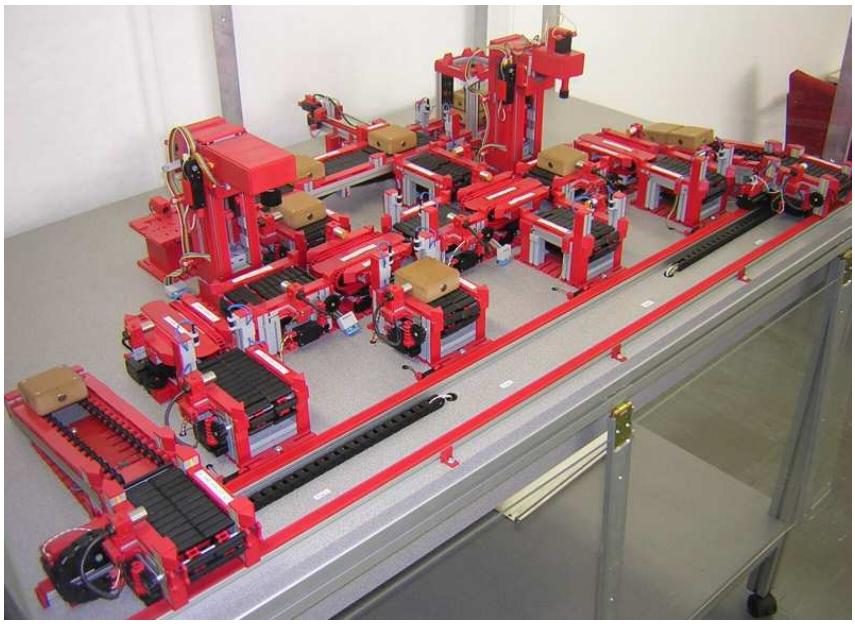

Abbildung 6: Fischertechnikmodell einer Fertigungsanlage

Die Anlage kann symbolhaft als Holzblöcke ausgeführte Werkstücke aus einem Stapellager zwei verschiedenen Bearbeitungsstationen zuführen und anschließend über ein Schienentransportsystem auf zwei Rollenbahnen abladen. Das System beinhaltet allein 16 Förderbänder der oben beschriebenen Art, ein monolithisches Gesamtmodell der Anlage ergäbe einen Automaten mit Schätzungsweise 10^{30} Zuständen, was einen monotlithischen Supervisorentwurf auch hier ausschließt.
Die Weiterentwicklung des hier beschriebenen Supervisor-Entwurfs hin zu modernen strukturierten Methoden hat jedoch dazu geführt, dass Steuerungsprogramme inzwischen auch für komplexe Systeme wie diese Fertigungsanlage automatisch berechnet werden können, wobei nach wie vor die Einhaltung der Spezifikation sowie Blockierungsfreiheit *nachweislich* garantiert werden können. Die Grundideen solcher strukturierten Ansätze werden im nachfolgenden Abschnitt erläutert.

3. Strukturierte Entwurfsverfahren

Die Ideen zur Reduktion des Rechenaufwands beim Supervisorentwurf orientieren sich an der Vorgehensweise erfahrener Entwickler von Steuerungsprogrammen, die, abgesehen von den eingangs erwähnten Nachteilen, auch komplexe industrielle Anlagen erfolgreich automatisieren. Die strukturierte Vorgehensweise dieser Entwickler zu formalisieren, um geeignete Entwurfsmethoden abzuleiten, welche besser mit der Anzahl der Systemkomponenten skalieren, war und ist Gegenstand der Forschung seit den vergangenen 15 Jahren. Zahlreiche literarische Beiträge zum Thema ereignisdiskrete Systeme belegen den wissenschaftlichen Fortschritt, auf eine detaillierte Auflistung wird hier aus Gründen der Übersichtlichkeit verzichtet. Stattdessen möchten wir hier die dabei entstandenen Grundideen vermitteln.
Naheliegend war zunächst die Beobachtung, dass die Spezifikation beim Steuerungsentwurf als Summe von Teilaufgaben betrachtet wird. So kann man die Komplexität des Entwurfs reduzieren, indem man nicht ein Programm entwickelt, welches alle spezifizierten Anforderungen gleichzeitig erfüllen muss; stattdessen kann man die Gesamtaufgabe aufteilen auf mehrere Unterprogramme, welche jeweils die Erfüllung von Teilanforderungen gewährleisten. Aus diesem Gedanken heraus haben sich die modularen Verfahren zur Steuerung ereignisdiskreter Systeme entwickelt.

Modulare Vefahren.

Bei den modularen Verfahren werden zu verschiedenen Teilanforderung entsprechend verschiedene Spezifikation vorgegeben und für jede Spezifikation monolithische Supervisors entworfen.

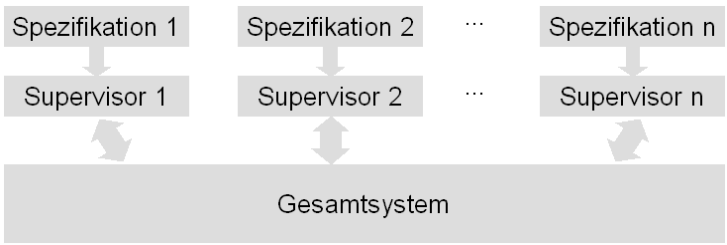

Abbildung 7: Modulare Steuerungsarchitektur

Die Verringerung des Rechenaufwands ergibt sich aus der geringeren Komplexität der Teilspezifikationen. Es verbleibt jedoch weiterhin die Notwendigkeit der monolithischen Darstellung

des Gesamtsystems. Ferner beeinflussen die Supervisors sich gegenseitig, was zu so genannten *Konflikten* führen kann, das heißt, die Supervisors blockieren sich gegenseitig. Der Test des resultierenden gesteuerten Gesamtsystems auf Blockierungsfreiheit macht oft die zuvor gewonnene Einsparung an Rechenaufwand zunichte.
Eine andere Idee ist die Berücksichtigung der funktionalen Struktur des zu steuernden Prozesses. Gerade bei komplexen Systemen liegt der zu steuernde Prozess als Gesamtheit aus mehreren kleineren funktionalen Einheiten vor; man spricht dann von einem *zusammengesetzten* ereignisdiskreten System. Diese Beobachtung führte zu den dezentralen Entwurfsverfahren.

Dezentrale Vefahren.

Bei den dezentralen Verfahren wird die Gesamtaufgabe von mehreren Supervisors erledigt, die jeweils nur einen bestimmten Teil des Gesamtprozesses steuern. Die Aufteilung des Gesamtprozesses erfolgt im Sinne Funktionaler Einheiten, z.B. Förderbänder und Fertigungszellen. Für jede Einheit wird je ein Supervisor zur Umsetzung der gegebenen Gesamtspezifikation entworfen.

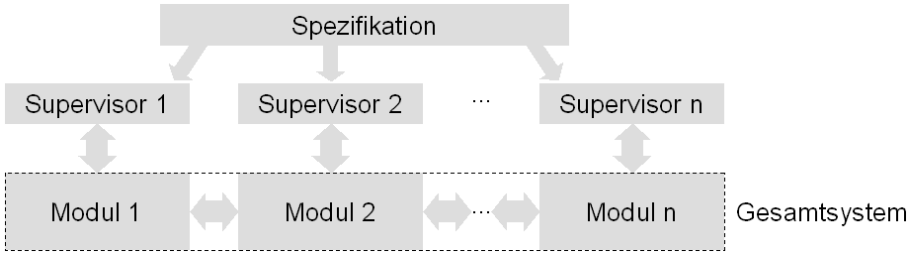

Abbildung 8: Dezentrale Steuerungsarchitektur

Da jeder Supervisor nur einen Teil des Gesamtsystems beobachtet, kann die Spezifikation im resultierenden Gesamtverhalten noch verletzt sein. Ebenso ist die Lebendigkeit des gesteuerten Gesamtsystems nicht garantiert. Genügen die einzelnen Teilsysteme jedoch bestimmten strukturellen Bedingungen, so treten diese beiden Probleme nachweislich nicht auf. Diese Bedingungen können sich aber als zu restriktiv erweisen, oder die Überprüfung der Bedingung erfordert wiederum zu hohen Rechenaufwand.
Ein weiterer Weg zur Umgehung des Komplexitätsproblems besteht darin, den Supervisor zwar für das Gesamtmodell zu berechnen, dies jedoch basierend auf einer übergeordneten, weniger detaillierten Sichtweise des tatsächlichen Modells.

Hierarchische Vefahren.

Die hierarchischen Ansätze führen in Schichten geordnete Modellabstraktionen unterschiedlicher Detailtreue ein. Dabei werden zu zugehörigen Spezifikationen Steuerungen entworfen, die virtuell das abstrahierte Modell steuern. Durch eine geeignete Übersetzung wird der Supervisor für das tatsächliche System in der detaillierteren darunter liegenden Schicht der Hierarchie implementiert.

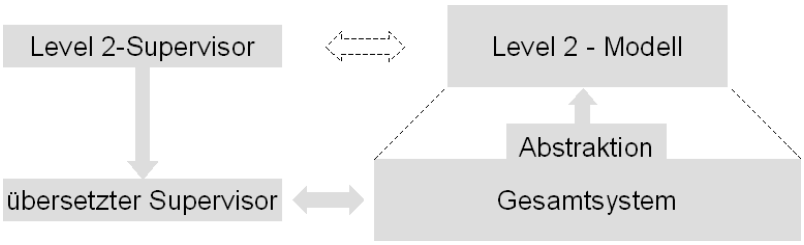

Abbildung 9: Hierarchische Steuerungsarchitektur

Eine Verringerung des Rechenaufwands erwartet man aufgrund einer feinen Abstimmung der Modellierungsgenauigkeit mit der jeweiligen Spezifikation. Während die Erfüllung der Spezifikation am tatsächlichen System meist direkt anhand der Monotonie der zur Abstraktion herangezogenen Operatoren nachgewiesen werden kann, unterscheiden sich die bekannten Verfahren methodisch in der Untersuchung bzw. Sicherstellung von Lebendigkeit. Hier hat sich der Begriff der *hierarchischen Konsistenz* etabliert, durch den gefordert wird, dass die Implementierung einer Steuerung auf der unteren Schicht das in der darüber liegenden Schicht geforderte Verhalten exakt widerspiegelt. Als Kehrseite dieser Methodik führt eine zu grobe Abstraktion entweder zur Verletzung gewünschter Eigenschaften oder zu einem übermäßig restriktiven Supervisor. Ferner bleibt problematisch, dass sowohl für das Ermitteln der Abstraktion als auch für die Implementierung der Steuerung auf der jeweils darunter liegenden Schicht das vollständige detaillierte Modell herangezogen werden muss.

Kombinierte Vefahren.

Jüngste Ansätze zum modellbasierten Steuerungsentwurf für ereignisdiskrete Systeme setzen sich mit der Zusammenführung der obigen Verfahren auseinander, um die verschiedenen Vorteile zu kombinieren. Dabei wird die Gesamtspezifikation *modular* in lokale Spezifikationen, die jeweils auf einzelne Teilsysteme des Prozesse bezogen sind, und globale Spezifikationen, die sich übergeordnet auf mehrere Teilsysteme beziehen, aufgeteilt. Zu jeder lokalen Spezifikationen werden *dezentrale* lokale Supervisors entworfen, die das entsprechende Teilsystem steuern. Das Gesamtverhalten der gesteuerten Systeme wird auf ein übergeordnetes abstraktes Modell abgebildet, für welches dann zur globalen Spezifikation ein *hierarchischer* Supervisor berechnet wird. Die Implementierung dieser übergeordneten Steuerung in der unteren Schicht erfolgt durch zusätzliche Einschränkung des Verhaltens der lokalen Supervisors.

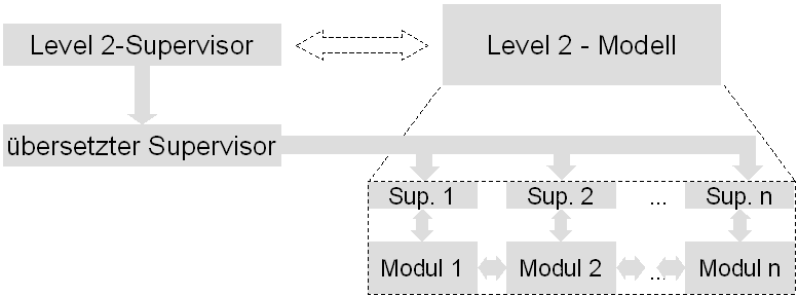

Abbildung 10: Kombinierte Steuerungsarchitektur

Mit diesen Ansätzen ist es erstmals möglich, sichere und lebendige Steuerungen für komplexe Anwendungsszenarien zu entwerfen. Der Erhalt von Sicherheit und Lebendigkeit gelingt durch

moderate Anforderungen an die Struktur der Teilprozesse und durch die besondere Form der Abstraktion sowie der Implementierung des Level-2-Supervisors in der unteren Schicht.
Beim so genannten *hierarchisch interface-basierten Ansatz* [2] erfolgen Abstraktion und Implementierung über spezielle Zwischenschichten, deren besondere Struktur zusammen mit der Struktur der beteiligten Systeme die gewünschte Sicherheit und Lebendigkeit erzielt. Die geforderten Strukturen stellen eine gewisse Einschränkung dar, jedoch belegen erfolgreiche Anwendungsstudien wie [7] Praxistauglichkeit. Eine Erweiterung dieses Ansatzes auf mehrere überlagerte Abstraktionsschichten ist angedacht.
Aufgrund der im Prinzip beliebigen Wiederholbarkeit des Abstraktions- sowie des Implementierungsschrittes erlaubt der *hierarchisch-dezentrale Ansatz* [4] eine Vielzahl überlagerter Hierarchieebenen, unter Garantie von Lebendigkeit und Sicherheit. Dabei können lokal gesteuerte Teilsysteme zu mehreren Gruppen zusammengefasst und jeweils die Gruppe abstrahiert werden. Für das abstrakte Modell jeder Gruppe kann auf der nächsten Ebene eine weitere übergeordnete Steuerung entworfen werden. Durch Abwechseln von Steuerungsentwurf, Gruppierung und Abstraktion wird die mehrstufige Hierarchie aufgebaut. Es resultiert eine Systemstruktur wie im nachfolgend abgebildeten Beispiel.

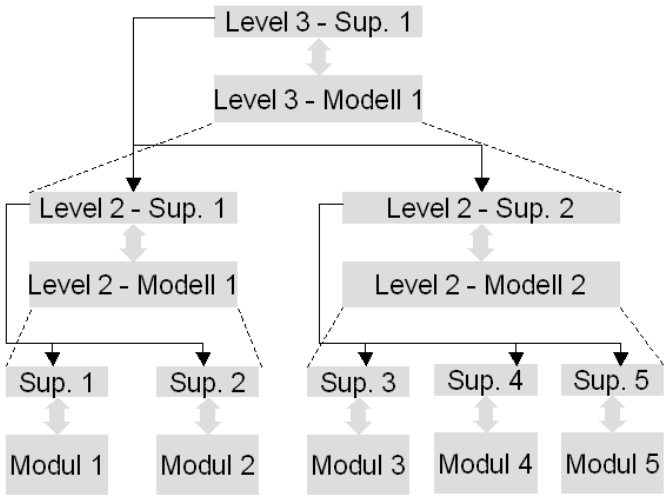

Abbildung 11: Hierarchisch-dezentraler Ansatz [4]

Die Implementierung der Supervisors der höheren Ebene durch zusätzliche Einschränkung der Supervisors in der Ebene darunter ist durch abwärts gerichtete Pfeile angedeutet. Die Tauglichkeit für komplexe Szenarien in der Praxis wurde durch die Anwendung auf das in Abbildung 6 gezeigte Fischertechnikmodell belegt.

Ein-/Ausgangsbasierter Ansatz.

In [6] haben die Autoren dieses Beitrags eine Ein-Ausgangs-(E/A-) basierte System- und Regelungstheorie für ereignisdiskrete Systeme vorgestellt. Diese ist angelehnt an die so genannte Behavioural Systems Theory nach J.C. Willems [8], welche grundlegende Eigenschaften dynamischer Systeme, wie die Charakteristik von Ein- und Ausgängen, verhaltensorientiert und so allgemeingültig beschreibt, dass sie auch zur E/A-basierten Beschreibung von ereignisdiskreten Systemen herangezogen werden konnte. Die E/A-basierte Modellierung der einzelnen Prozesskomponenten erfolgt zunächst umgebungsunabhängig (d.h. unabhängig von Nachbarkomponenten oder der Steuerung). Dies bewirkt ihre Wiederverwendbarkeit innerhalb unter-

schiedlicher Anordnungen. Jedes E/A-basierte Modell der lokalen Teilsysteme verfügt über zwei E/A-Ports, einer zum Anschluss einer Steuerung, der andere zum Anschluss eines dynamischen Umgebungsmodells, welches jeweils die Interaktion einer Gruppe von Teilsystemen untereinander beschreibt.
Gemäß lokaler von der Umgebung unabhängiger Spezifikationen werden für die einzelnen Prozesskomponenten zunächst lokale Steuerungen basierend auf der SCT entworfen. Der Nachweis von Sicherheit und Lebendigkeit des geschlossenen Regelkreises gelingt infolge der ein-/ausgangsbasierten Systembeschreibung. Auf der nächsthöheren Stufe der Hierarchie werden jeweils mehrere Prozesskomponenten zusammengefasst und ihre Interaktion durch ein dynamisches Umgebungsmodell modelliert. Der Entwurf überlagerter Steuerungen für Gruppen von Komponenten greift nicht auf die detaillierte Beschreibung der lokal gesteuerten Prozessmodelle, sondern auf eine *Abstraktion* derselben auf Grundlage der lokalen Spezifikationen zurück, was den Rechenaufwand wirkungsvoll begrenzt. Die Zulässigkeit des Abstraktionsschrittes wurde formal bewiesen. Das heißt, die anhand der Abstraktion entworfene Steuerung steuert auch das tatsächliche System nachweislich korrekt. Durch gezieltes Abwechseln der Abstraktionsschritte, der Beschreibung der Interaktion durch Umgebungsmodelle und der Überlagerung von Steuerungen lässt sich ein Gesamtsystem entwickeln, welches mit der Anzahl der Prozesskomponenten gut skaliert.

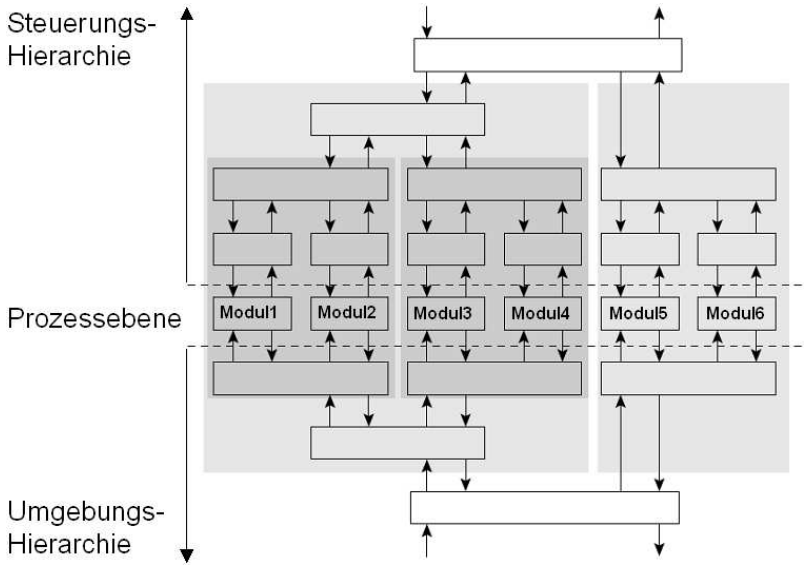

Abbildung 12: Ein-/Ausgangsbasierter Ansatz [6]

Der in der Theorie vorhergesagte geringe Rechenaufwand ergab sich auch bei der Anwendung auf das virtuelle Beispiel einer Kette von Transporteinheiten. Derzeit erfolgt die Umsetzung für die Fischertechnikanlage, um die Praxistauglichkeit zu belegen.
Zum Abschluss möchten wir kurz auf weitere Arbeitsgebiete der Forschungsgruppe ereignisdiskrete Systeme des Lehrstuhls für Regelungstechnik der Universität Erlangen-Nürnberg eingehen.

4. Weitere Arbeiten der Forschungsgruppe Ereignisdiskrete Systeme

Die SCT sowie darauf aufbauende Ansätze zum ereignisdiskreten Steuerungsentwurf erfordert

eine Vielzahl an Computeralgorithmen zur Manipulation von regulären Sprachen und Automaten, welche in der umfassenden C++ Bibliothek `libfaudes` (Friedrich-Alexander University Discrete Event Systems Library) implementiert wurden. Es ist geplant, diese kostenlos unter der GNU Lesser General Public License (LGPL) öffentlich zur Verfügung zu stellen.
Aufbauend auf dieser Bibliothek wurde das Softwaretool `DESolution` entwickelt, welches eine grafische Oberfläche zum Entwurf ereignisdiskreter Steuerungen bietet. Automatenmodelle können mit einem grafischen Editor entworfen werden. Die Software unterstützt den Anwender beim Entwurf der Steuerung mit zahlreichen Funktionen. Die Steuerung kann anschließend per Knopfdruck über den eingebauten Codegenerator z.B. in SPS-Code übersetzt werden.
Ein weiteres Forschungsgebiet sind *verteilte ereignisdiskrete Systeme*. Hier wird berücksichtigt, dass Prozess-Steuerungen zunehmend nicht mehr auf einer zentralen SPS ablaufen, sondern auf mehrere, an getrennten Orten angebrachte SPSen verteilt werden, welche über ein Netzwerk kommunizieren. Die Vernachlässigung der Verzögerung des Steuereingriffs durch Kommunikation ist nur bei vergleichsweise langsamen technischen Prozessen zulässig. Andernfalls muss der Kommunikationsaufwand beim Entwurf berücksichtigt werden. Dies geschieht in der Praxis oft durch stochastische Simulationen, deren Ergebnisse jedoch keine *garantierten* Aussagen zur fehlerfreien Kommunikation liefern. Dieses Problem wird in jüngsten Arbeiten zu verteilten ereignisdiskrete Systemen angegangen:
In [5] wird ein innerhalb der Forschungsgruppe entwickeltes Verfahren zur verteilten Implementierung der nach dem hierarchisch-dezentralen Ansatz entworfenen Supervisors vorgeschlagen. Die dabei verwendete Netzwerkarchitektur eignet sich beispielsweise für eine Ethernet-Implementierung. Im Gegensatz zu stochastischen Analyseverfahren wird die notwendige Kommunikation der Supervisors mit einem *deterministischen* ereignisdiskreten Kommunikationsmodell beschrieben. Eine Scheduling-Strategie zum Versenden von Nachrichten zwischen den Supervisors *garantiert nachweislich* den korrekten Ablauf der Supervisors bei ausreichender Netzwerkgeschwindigkeit. In Folgearbeiten wurde zudem eine Methode zur Bestimmung der zum korrekten Betrieb der Supervisors erforderlichen Mindestgeschwindigkeit des Netzwerks entwickelt.

Dieser Beitrag sollte einen Überblick über die Steuerung ereignisdiskreter Systeme verschaffen. Dazu wurde die zugrunde liegende Supervisory Control Theory und ihre Anwendung in modernen, praxistauglichen Verfahren erläutert. Aktuelle Forschungsarbeiten, die sich zunehmend an der praktischen Umsetzung ereignisdiskreter Steuerungen orientieren, belegen die wachsende Relevanz dieser Thematik für die industrielle Anwendung.

Literatur

[1] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[2] R.J. Leduc. Hierarchical interface based supervisory control. *PhD thesis, Department of Electrical and Computer Engineering, University of Toronto*, 2002.

[3] P.J. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77:81–98, 1989.

[4] K. Schmidt. Hierarchical control of decentralized discrete event systems: Theory and application. *Dissertation, Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg*, 2005. Download: http://www.rt.eei.uni-erlangen.de/FGdes/publications.html.

[5] K. Schmidt, E. Schmidt und J. Zaddach. A shared-medium communication architecture for distributed discrete event systems. *Mediterranean Conference on Control and Automation*, 2007.

[6] S. Perk, T. Moor und K. Schmidt. Hierarchical discrete event systems with inputs and outputs. *Workshop on Discrete Event Systems*, 2006. Download: http://www.rt.eei.uni-erlangen.de/FGdes/publications.html.

[7] F. Wenck. Modellbildung, Analyse und Steuerungsentwurf für gekoppelte ereignisdiskrete Systeme. *Dissertation, Lehrstuhl für Automatisierungstechnik und Prozessinformatik, Ruhr-Universität Bochum*, 2006.

[8] J.C. Willems. Paradigms and puzzles in the theory of dynamic systems. *IEEE Transactions on Automatic Control*, 36:258–294, 1991.