Technische Universität München, Fakultät für Informatik

Hauptseminar „Design hybrider, eingebetteter Systeme“

Teil 4:

Hybride Automaten

Richard Atterer
⟨richard§atterer.net⟩

25. Juni 2001

Inhaltsverzeichnis

1 Einleitung

Hybride, eingebettete Systeme sind Systeme, die heute in vielen Zweigen der Industrie, von der Luftfahrt über Kontroll- und Regelungssysteme bis hin zu Haushaltsgeräten, eine immer größere Rolle spielen.

Die Rechensysteme werden als *hybride* bezeichnet, da sie im Gegensatz zu herkömmlichen Computern nicht ausschließlich mit digitalen Größen arbeiten, sondern auch mit ihrer Umwelt interagieren, die durch analoge Größen wie z.B. Temperatur, Entfernung oder Neigungswinkel gekennzeichnet ist. Diese kontinuierlichen Eingabewerte nehmen nicht jeweils einen Wert aus einer Reihe von möglichen Werten an, sondern verändern sich stufenlos, was eine gesonderte Behandlung erfordert.

Der überwiegende Teil von Systemen in diesem Anwendungsgebiet ist *eingebettet* (*embedded*), d.h. die Rechner sind Kleinstcomputer, die mit nur einem oder wenigen Chips realisiert werden und so in ein Produkt integriert sind, daß sie nur als Teil des Produkts genutzt werden können. Im allgemeinen steht die Interaktion mit der physikalischen Umwelt des Systems im Vordergrund, weniger dagegen die Interaktion mit anderen Rechnern. Eingebettete Systeme sind heute weit verbreitet (ihre Zahl überschreitet die Zahl „herkömmlicher“ Arbeitsplatz-Computer bereits beträchtlich), ebenso sind sie für die Funktionalität der Produkte immer wichtiger. Die Gründe für diese Entwicklung sind die niedrigen Stückkosten bei gleichzeitig hoher Flexibilität.

In diesem Beitrag zum Hauptseminar „Design hybrider, eingebetteter Systeme“ soll ein Überblick über einen wichtigen Aspekt beim Entwurf dieser Systeme gegeben werden: Dem Konzept der *hybriden Automaten*, mit deren Hilfe hybride Systeme beschrieben werden können. Zunächst wird das Konzept und die mathematischen Grundlagen von hybriden Automaten vorgestellt, bevor anhand des praktischen Beispiels „elektronische Höhenregelung eines Fahrzeugs“ die Vor- und Nachteile der Modellierung von Systemen durch hybride Automaten und die Fähigkeiten existierender Software-Tools für diese Aufgabe betrachtet werden.

2 Formales Modell

Hybride Automaten stellen eine Erweiterung des Konzeptes der *endlichen Automaten* dar, mit dem sich beliebige diskrete Systeme modellieren lassen und das z.B. in [Broy98, S.226] vorgestellt wird.

Die grundlegende Idee bei endlichen Automaten, die im Entwurf für hybride Automaten übernommen wurde, ist es, davon auszugehen, daß ein System zu jedem Zeitpunkt jeweils in einem Zustand aus einer Menge von Zuständen ist, die beim Entwurf des Systems festgelegt wurden. Die Zustände symbolisieren hierbei, an welcher Stelle der Automat bei der Bearbeitung einer Aufgabe angelangt ist – bei einer Ampelsteuerung könnte z.B. jede der Phasen „grün“, „gelb“, „rot“, „rot-gelb“ durch einen Zustand modelliert werden. Unter bestimmten, zusätzlich anzugebenden Bedingungen ist ein Übergang von einem Zustand in einem anderen möglich.

Der Unterschied zwischen endlichen Automaten und hybriden Automaten besteht hauptsächlich darin, daß bei den ersteren keine Betrachtungen darüber angestellt werden, was vor sich geht, solange keine Zustandsänderung stattfindet. Bei den letzteren dagegen „passiert auch

etwas“, solange der Automat in einem Zustand bleibt: Zusätzliche Variablen aus $\mathbb{R}$ verändern sich entsprechend vorgegebener Differentialgleichungen.

Eine naheliegende Formalisierung, die auch für die Visualisierung von Automaten nützlich ist, ist die Darstellung als gerichteter Graph. Hierbei werden die Zustände des Automaten als die Knoten des Graphen dargestellt, die Übergänge sind durch dessen Kanten symbolisiert.

2.1 Definition

Nach [MS00, S.3] ist ein hybrider Automat A durch ein 10-Tupel

$$(X, V, inv, init, flow, E, upd, jump, L, sync)$$

gegeben, wobei die einzelnen Elemente wie nachfolgend beschrieben definiert sind. Als Beispiel für einen Automaten sei auf Abbildung 1 verwiesen.

X ist eine endliche, geordnete Menge $\{x_1, \ldots, x_n\}$ von Variablen aus $\mathbb{R}$.
Eine Belegung (engl. *valuation*) ist ein Punkt in $\mathbb{R}^n$.
Die i-te Komponente einer Belegung s wird mit s_i bezeichnet.
Für ein Prädikat Φ über die Variablen aus X schreiben wir $[\![\Phi]\!]$, um die Menge aller Belegungen zu bezeichnen, für die $\Phi[X := s] = true$ gilt.

V ist eine endliche Menge von Orten (engl. *locations*). Für endliche Automaten sind die Begriffe Ort und Zustand synonym, bei hybriden Automaten ist der Zustand dagegen durch ein Tupel (v, s) bestehend aus einem Ort $v \in V$ und einer Belegung s gegeben.
Eine Menge von Zuständen bezeichnen wir als Region (engl. *region*).

inv stellt die Invarianten dar, die für einen Ort gelten müssen. Es handelt sich hierbei um eine Abbildung $V \to \Phi$, d.h. jedem Ort wird ein Prädikat über die Variablen in X zugeordnet. Solange der aktuelle Ort eines Automaten der Ort v ist, sind nur Belegungen $s \in [\![inv(v)]\!]$ erlaubt.
In der graphischen Darstellung wird die Invariante „*true*“ meist weggelassen.

init legt die Anfangsbelegung der Variablen aus X und den oder die Anfangsort(e) fest. Genau wie *inv* ist *init* eine Abbildung $V \to \Phi$ von Orten auf Prädikate, die als Anfangsbedingung (*initial condition*) bezeichnet werden. Ein Zustand (v, s) ist Anfangszustand des Automaten, wenn $init(v)$ und $inv(v)$ für die Belegung s erfüllt sind.
In der graphischen Darstellung werden Orte, für die $init(v) \neq false$ gilt, durch einen eingehenden Pfeil symbolisiert, mit dem zusammen die Anfangsbedingung angegeben wird.

flow beschreibt für jeden Ort aus V, wie sich die Variablen aus X verändern, solange sich der Automat in dem jeweiligen Ort befindet.
flow ist eine Abbildung von V auf Prädikate über $X \cup \dot{X}$, wobei $\dot{X} = \{\dot{x}_1, \ldots, \dot{x}_n\}$, d.h. es kann sich um Differentialgleichungen erster Ordnung handeln.

Mithilfe dieser Abbildung können wir Zustandsänderungen beschreiben, bei denen der aktuelle Ort des Automaten gleich bleibt, die Variablen sich aber ändern. Für die entsprechende „Zeitschritt-Relation“ $\xrightarrow{\delta}$ existiert der Übergang $(v, s) \xrightarrow{\delta} (v, s')$ genau dann, wenn

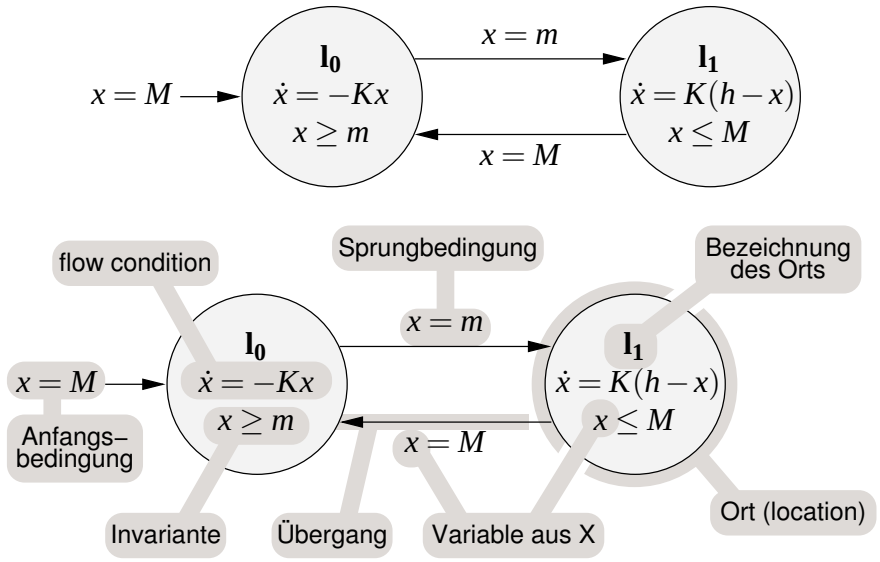

Abbildung 1: *Hybrider Automat für einen Thermostat [ACH+95, S.4]*

es eine differenzierbare Funktion $\rho : [0,\delta] \to \mathbb{R}^n$ gibt und folgende Bedingungen erfüllt sind:

- $\rho(0) = s$ und $\rho(\delta) = s'$
- die Invariante von v ist erfüllt: $\rho(t) \in [\![inv(v)]\!]$ für $t \in [0,\delta]$
- das Prädikat aus *flow* ist erfüllt (*flow condition*):
  $flow(v)[X, \dot{X} := \rho(t), \dot{\rho}(t)]$ gilt für $t \in [0,\delta]$ mit $\dot{\rho}(t) = (\dot{\rho}_1(t), \ldots, \dot{\rho}_n(t))$.

E stellt die Übergänge des Automaten dar, von denen jeder durch Anfangs- und End-Ort gekennzeichnet ist. $E \subseteq V \times V$, für $(v,v') \in E$ existiert also ein Übergang von v nach v'.
Es ist möglich, daß mehr als ein Übergang zwischen zwei Orten vorhanden ist.

upd legt für jeden Übergang die „Update-Menge“ (engl. *update set*) fest, d.h. diejenigen Variablen aus X, die sich verändern können, wenn der Übergang benutzt wird. Formal handelt es sich also um eine Abbildung von E auf eine Menge von Elementen aus X.
In der graphischen Darstellung wird die Update-Menge nicht explizit angegeben, kann aber leicht bestimmt werden, da in den Sprungbedingungen der Operator „:=“ benutzt wird, wenn bei einer Gleichung wie „$x := 5$“ die Variable x in der Update-Menge ist.

jump, eine Abbildung von E auf Sprungbedingungen (engl. *jump conditions*), beschreibt, unter welchen Bedingungen ein Übergang stattfinden kann und welchen der Variablen dabei neue Werte zugewiesen werden.
Die Sprungbedingung $jump(e)$ eines Übergangs e ist ein Prädikat über die Variablen aus $X \cup upd'(e)$, wobei $upd'(e) = \{y'_1, \ldots, y'_k\}$ für $upd(e) = \{y_1, \ldots, y_k\}$. Dabei bezeichnet y'_i den neuen Wert der Variablen y_i, nachdem der Übergang stattgefunden hat.

Während ein Übergang benutzt wird, vergeht keine Zeit.
In der „Übergangsschritt-Relation“ $\xrightarrow{e}$ existiert der Übergang $(v,s) \xrightarrow{e} (v',s')$ genau dann, wenn gilt:

- $e = (v,v')$
- die Invarianten sind erfüllt: $s \in [\![inv(v)]\!]$ und $s' \in [\![inv(v')]\!]$
- die Variablen in $upd(e)$ ändern sich in Übereinstimmung mit der Sprungbedingung: $jump(e)[X, upd'(e) = s, s'[upd(e)]] = true$, wobei $s'[upd(e)]$ diejenigen Variablen aus s' bezeichnet, die in $upd(e)$ enthalten sind.
- die Variablen, die nicht in $upd(e)$ enthalten sind, bleiben unverändert: $s_i = s'_i$ für alle $x_i \in (X \setminus upd(e))$

Bei graphischer Darstellung der Automaten schreiben wir für ein Prädikat Φ, eine Update-Menge $\{y_i\}$ und eine Sprungbedingung $\Phi \wedge y'_i = c$ kurz: $\Phi \rightarrow y_i := c$.
Falls $\Phi = true$, schreiben wir: $y_i := c$.

L ist eine endliche Menge von Synchronisations-Bezeichnungen (engl. *synchronization labels*). Bei der Parallelschaltung von hybriden Automaten müssen Übergänge, die bei zwei oder mehr Automaten die gleiche Bezeichnung haben, bei allen diesen Automaten jeweils zur gleichen Zeit benutzt werden. Übergänge mit verschiedenen Bezeichnungen können dagegen zu unterschiedlichen Zeitpunkten stattfinden.

sync ordnet jedem Übergang eines Automaten eine Synchronisations-Bezeichnung zu.
sync ist eine Abbildung $E \rightarrow L \cup \{\tau_A\}$. Die spezielle Bezeichnung τ_A darf nur für Übergänge des Automaten A benutzt werden und kennzeichnet Übergänge, für die keine Synchronisation mit anderen Automaten stattfinden soll.
Bei graphischer Darstellung von Automaten stehen an der Kante für Übergang e die Werte von $sync(e)$ und $jump(e)$, wobei τ_A weggelassen wird.

2.2 Beispiel: Thermostat

Als Beispiel für die Umsetzung einer Problemstellung in einen hybriden Automaten modellieren wir die Heizungs-Regelung in einem Raum.

Die Temperatatur x des Raumes wird von einem Thermostat durch einen Sensor gemessen und die Heizung je nach Bedarf ein- und ausgeschaltet. Wenn die Heizung ausgeschaltet ist, folgt die Raumtemperatur der Funktion

$$x(t) = \theta e^{-Kt},$$

wobei t die Zeit, θ die Anfangstemperatur und K eine vom Raum abhängige Konstante ist. Ist die Heizung angeschaltet, folgt die Temperatur der Funktion

$$x(t) = \theta e^{-Kt} + h(1 - e^{-Kt})$$

für eine Konstante h, deren Wert von der Heizungsleistung abhängt. Die Temperatur soll stets zwischen den Werten m und M bleiben.

Abbildung 1 zeigt einen hybriden Automaten, mit dem sich dieses System darstellen läßt. Im Ort l_0 ist dabei die Heizung ausgeschaltet, im Ort l_1 eingeschaltet.

2.3 Weitere Begriffe

Im Zusammenhang mit der Klassifizierung verschiedener hybrider Automaten haben sich eine Reihe von Bezeichnungen für Automaten mit bestimmten Eigenschaften eingebürgert. Nachfolgend werden die wichtigsten dieser Begriffe vorgestellt.

Linearer hybrider Automat Ein hybrider Automat wird als linearer hybrider Automat bezeichnet, wenn folgende Bedingungen erfüllt sind [MS00, S.6]:

- Alle Invarianten aus *inv* sind konvexe lineare Prädikate über X.
- Alle Anfangsbedingungen aus *init* sind konvexe lineare Prädikate über X.
- Alle *flow conditions* aus *flow* sind konvexe lineare Prädikate über $\dot{X}$.
- Alle Sprungbedingungen aus *jump* sind konvexe lineare Prädikate über $X \cup X'$, wobei $X' = \{x'_1, \ldots, x'_n\}$.

Ein **lineares Prädikat** über Y ist eine Gleichung oder Ungleichung zweier linearer Terme über Y, d.h. zweier Linearkombination der Variablen aus Y mit Koeffizienten aus $\mathbb{R}$. Für $Y = \{y_1, y_2, \ldots, y_n\}$ ist z.B. die Gleichung $y_1 + 0.4711y_3 = 42y_2$ ein lineares Prädikat.

Ein **konvexes lineares Prädikat** ist eine endliche Konjunktion von linearen Prädikaten, d.h. es hat die Form $\Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_n$.

Durch diese Definitionen wird deutlich, daß nicht alle linearen Differentialgleichungen auch in Elemente von *flow* umgewandelt werden können; dies wäre nur möglich, wenn *flow* konvexe lineare Prädikate über $X \cup \dot{X}$ und nicht nur über $\dot{X}$ enthalten könnte. Somit ist auch der Thermostat aus Abbildung 1 kein linearer hybrider Automat.

Ein linearer Automat wird **einfach** (engl. *simple*) genannt, wenn sämtliche darin vorkommenden linearen Prädikate Φ_i einfach sind – für eine Variable $x \in X$ und eine Konstante $k \in \mathbb{Z}$ bedeutet dies, daß es sich um Ungleichungen der Form $x \leq k$ oder $k \leq x$ handelt. (Insbesondere verbietet diese Einschränkung bei den weiter unten vorgestellten Multirate-Automaten den Vergleich von *skewed clocks*, die mit unterschiedlicher Geschwindigkeit laufen.)

Abbildung 2 zeigt ein Beispiel für einen linearen hybriden Automaten. Modelliert wird ein System, das durch Ein- und Ausschalten einer Pumpe den Wasserpegel in einem Tank kontrolliert. Ist die Pumpe ausgeschaltet, sinkt der Wasserspiegel y um 2cm pro Sekunde, ist sie angeschaltet, steigt er um 1cm pro Sekunde. Als Anfangsbedingung wird angenommen, daß der Wasserpegel bei 1cm liegt und die Pumpe eingeschaltet ist. Der Pegel soll zwischen 1cm und 12cm gehalten werden.

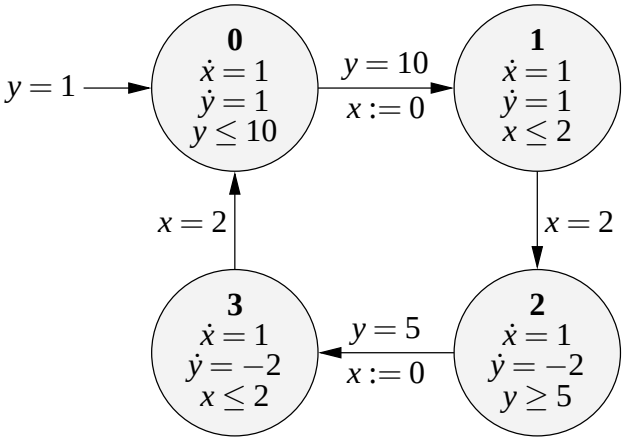

Abbildung 2: *Linearer hybrider Automat für einen Wasserpegel-Regler [ACH+95, S.7]*

Es ist noch eine zusätzliche Einschränkung gegeben, wonach es nach An-/Abschalten der Pumpe 2 Sekunden dauert, bis die entsprechende Wirkung eintritt. Der Automat muß die Pumpe also anschalten, noch bevor ein Pegel von 1cm erreicht ist, bzw. abschalten, noch bevor 12cm erreicht sind.

Im Automaten aus Abbildung 2 kann man diese Eigenschaften des Systems wiederfinden: Ein Zustand mit sinkendem Wasserspiegel führt zu der *flow condition* $\dot{y} = -2$, bei steigendem Wasser ist sie dagegen $\dot{y} = 1$. Die Uhr x mißt kontinuierlich die Zeit seit dem letzten An-/Abschalten – da die Pumpe in den Orten 0 und 3 angeschaltet, in den Orten 1 und 2 ausgeschaltet sein soll, wird x jeweils beim Übergang von 0 nach 1 sowie von 2 nach 3 mit „$x := 0$“ zurückgesetzt. Die Invarianten wie z.B. $x \leq 10$ resultieren aus den Wasserständen, die beim An-/Abschalten erreicht sein müssen.

Diskreter Automat Ein diskreter Automat ist ein Automat, dessen Variablen alle diskret sind [ACH+95, S.5]. Von einer **diskreten Variable** (engl. *discrete variable*) sprechen wir, wenn sich die Variable zu keinem Zeitpunkt ändert, solange der Automat in einem Ort bleibt, sondern nur bei Übergängen von einem Ort in einen anderen.

Zeit-Automat Ein hybrider Automat ist ein Zeit-Automat (engl. *timed automaton*), wenn zwei Bedingungen erfüllt sind:

- Für jede Variable gilt: der Wert jeder Variablen ist stets entweder 0 oder 1, oder die Variable ist eine Uhr.
- Die Gleichungen oder Ungleichungen in den konvexen linearen Prädikaten können lediglich die Form $x_1 \circ c$ oder $x_1 - x_2 \circ c$ annehmen, mit $x_1, x_2 \in X$, einem konstanten $c \in \mathbb{Z}$ und einem Operator $\circ \in \{<, \leq, =, \geq, >\}$.

Als **Uhr** (engl. *clock*) wird eine Variable x bezeichnet, deren Wert für alle Orte des Automaten mit der stets gleichen Geschwindigkeit 1 zunimmt (d.h., die *flow condition* für x ist immer: $\dot{x} = 1$).

Als einzige weitere Änderung der Variable ist das Zurücksetzen auf 0 gestattet. Somit kann eine Uhr dafür verwendet werden, die Zeitspanne seit dem Eintreten eines bestimmten Ereignisses zu messen. Ist die Geschwindigkeit einer Uhr ungleich 1 (d.h. $\dot{x} = k$ mit $k \in \mathbb{Z}\backslash\{0,1\}$), wird der Begriff **skewed clock** verwendet.

Zeit-Automaten lassen sich anhand der Uhren, die sie enthalten, noch genauer klassifizieren: Wenn für jede Variable gilt, daß sie entweder immer 0 oder 1 ist, oder daß sie eine *skewed clock* ist, handelt es sich um einen **multirate** timed automaton. Verändern sich die Uhren eines Automaten mit n verschiedenen Geschwindigkeiten, nennen wir den Automaten auch **n-rate** timed automaton.

Integrator-Automat (engl. *integrator system*) Diese Art von Automat ist eine Erweiterung von Multirate-Automaten in der Hinsicht, daß die Uhren des Automaten (genannt Integratoren oder engl. *integrators*) nicht nur linear weitergezählt und auf 0 zurückgesetzt werden können, sondern auch zeitweise angehalten werden können. Formal bedeutet dies, daß *flow* für einen Integrator x zusätzlich auch *flow conditions* der Form $\dot{x} = 0$ enthalten darf. Dadurch eignet sich ein Integrator zum Akkumulieren von Zeitspannen.

Parametrisierter Automat Die obigen Definitionen verlangen in einigen Fällen, daß an bestimmten Stellen nur Konstanten und keine Variablen verwendet werden. Um einen Automaten übersichtlicher und leichter verständlich zu machen, wird diese Einschränkung mit der Einführung von parametrisierten Automaten (engl. *parameterized automata*) dahingehend aufgeweicht, daß Konstanten durch symbolische Konstanten (Parameter) ersetzt werden dürfen. Da alle Vorkommen einen Parameters in einem Automaten jederzeit durch dessen Wert ersetzt werden können, hat dies keine Auswirkungen auf die Mächtigkeit eines Automatentyps.

2.4 Eigenschaften von hybriden Automaten

2.4.1 Parallelkomposition

Wie bereits bei der Definition in Abschnitt 2.1 erwähnt können hybride Automaten parallelgeschaltet werden. Die Kommunikation zwischen den einzelnen Automaten erfolgt dabei durch gemeinsame Variablen und die Synchronisations-Bezeichnungen aus L, mit denen Übergänge gekennzeichnet werden, die von allen Automaten, die einen Übergang mit der jeweiligen Bezeichnung besitzen, gleichzeitig durchgeführt werden müssen.

Formal kann nach [ACH+95, S.4] die Parallelkomposition zweier Automaten A_1 und A_2 als die Bildung eines Kreuzprodukts $A_1 \times A_2$ aufgefaßt werden. Das Ergebnis ist wieder ein hybrider Automat. Seien

$$A_1 = (X, V_1, inv_1, init_1, flow_1, E_1, upd_1, jump_1, L_1, sync_1)$$
$$A_2 = (X, V_2, inv_2, init_2, flow_2, E_2, upd_2, jump_2, L_2, sync_2)$$

zwei hybride Automaten. Dann ist das Kreuzprodukt definiert als

$$A_1 \times A_2 = (X, V_1 \times V_2, inv, init, flow, E, upd, jump, L_1 \cup L_2, sync),$$

mit folgenden weiteren Einschränkungen:

- Ein Übergang ist im neuen Automaten enthalten, wenn die zugehörigen Übergänge in den Ausgangsautomaten vorhanden sind. Außerdem werden die Orte, die den Übergang benutzen, $v_1 \in V_1$ und $v_2 \in V_2$, zu einem Ort $v_{12} \in V_1 \times V_2$ verschmolzen:

  $$(v_{12}, v'_{12}) \in E \quad \Leftrightarrow \quad (v_1, v'_1) \in E_1 \wedge (v_2, v'_2) \in E_2$$

- Die Bezeichnung a des Übergangs, $a = sync((v_{12}, v'_{12}))$, ist entweder die gemeinsame Synchronisations-Bezeichnung, d.h.

  $$a = a_1 = a_2 \;\text{ mit }\; a_1 = sync((v_1, v'_1)),\; a_2 = sync((v_2, v'_2)) \,,$$

  oder die Bezeichnung des einen Übergangs, wenn der andere mit τ_{A_i} als „nicht synchronisierend“ deklariert wurde:

  $$\begin{aligned} a_2 \notin L_1 \wedge a_1 = \tau_{A_1} \quad &\Rightarrow \quad a = a_2 \;\text{ bzw.} \\ a_1 \notin L_2 \wedge a_2 = \tau_{A_2} \quad &\Rightarrow \quad a = a_1 \end{aligned}$$

- Für die *flow condition* des neuen Orts v_{12} gilt: $flow(v_{12}) = flow_1(v_1) \cup flow_2(v_2)$.
  Analoges gilt für den Ort v'_{12}.

- Im neuen Ort v_{12} müssen die Invarianten der Orte v_1 und v_2 gleichzeitig erfüllt sein, aus denen er entsteht: $inv(v_{12}) = inv_1(v_1) \wedge inv_2(v_2)$.
  $inv(v'_{12})$ entsteht wiederum in analoger Weise.

Der Automat, der als Produkt zweier Ausgangsautomaten entsteht, führt somit Abläufe durch, in denen alle Abläufe oder eine Untermenge der Abläufe jedes der Ausgangsautomaten enthalten sind.

Durch die Möglichkeit der Parallelkomposition von hybriden Automaten ist ein Mittel gegeben, um komplexere Aufgabenstellungen zu bearbeiten, da das System durch eine Reihe von einfacheren Automaten repräsentiert werden kann, die über Synchronisations-Bezeichnungen miteinander kommunizieren.

2.4.2 Erreichbarkeits-Analyse

Ein zentrales Problem bei der Analyse von hybriden Automaten ist es zu bestimmen, welche Zustände von welchen anderen Zuständen aus erreicht werden können bzw. welche Zustände niemals erreicht werden können. Aufgrund der Formalisierung des Systems als Automat sind durch die Erreichbarkeits-Analyse beweisbare Aussagen über Eigenschaften des Automaten möglich – ein wesentlicher Vorteil von hybriden Automaten gegenüber anderen, weniger formalen Designmethoden.

Wenn σ und σ' Zustände des Automaten A sind, bezeichnen wir den Zustand σ' als von σ aus **erreichbar** (auch: $\sigma \mapsto^* \sigma'$), wenn es einen Ablauf von A gibt, der in σ beginnt und in σ' endet. Das Erreichbarkeitsproblem bei einem Automaten besteht darin herauszufinden, ob $\sigma \mapsto^* \sigma'$ für zwei gegebene Zustände des Automaten gilt.

Je nach den Eigenschaften des zu untersuchenden Automaten lassen sich unterschiedliche Aussagen darüber treffen, ob das Erreichbarkeitsproblem entscheidbar ist. Für die formalen Beweise der Aussagen sei auf [ACH+95, S.11] verwiesen.

- Für Zeit-Automaten ist das Erreichbarkeitsproblem entscheidbar.
- Für einfache Multirate-Zeit-Automaten ist das Erreichbarkeitsproblem entscheidbar.
  Diese Aussage läßt sich auf die erste Aussage zurückführen, da ein solcher Automat in einen Zeit-Automat umgewandelt werden kann.
- Für 2-rate-Zeit-Automaten ist das Erreichbarkeitsproblem *nicht* entscheidbar.
  Dies kann bewiesen werden, indem gezeigt wird, daß die Aufgabenstellung äquivalent zum Halteproblem für einen nichtdeterministischen 2-Zähler-Automat ist.
- Für einfache Integrator-Automaten ist das Erreichbarkeitsproblem *nicht* entscheidbar.

Leider muß also festgestellt werden, daß Erreichbarkeitsaussagen nur für die einfachsten Klassen von linearen hybriden Automaten möglich sind. Aus diesem Grund können in der Praxis auch nur sie für die Modellierung von Systemen verwendet werden. In Abschnitt 3.2.1 werden wir noch Methoden kennenlernen, um kompliziertere Aufgaben zumindest näherungsweise durch Zeit-Automaten oder einfache Multirate-Zeit-Automaten zu beschreiben.

2.4.3 Überprüfen von Invarianten

Über die Untersuchung von Erreichbarkeitseigenschaften hinaus macht es auch Sinn, noch weitere Aussagen über das Verhalten eines Automaten auf ihre Korrektheit hin zu überprüfen. So könnte es von Interesse sein, Annahmen über das Verhalten des Systems zu verifizieren, z.B.

> „Solange Variable x einen bestimmten Wertebereich nicht verläßt, wird Ort v niemals erreicht."

oder

> „Ein bestimmter Zustand kann nach dem Start des Systems frühestens nach einer gegebenen Zeitspanne eintreten."

Es sollte ersichtlich sein, daß es sich hierbei um generellere Probleme handelt als die Frage nach der Erreichbarkeit von Zuständen.

Für die Formulierung dieser Anforderungen bedient man sich einer „Echtzeit-Logik" (engl. *real-time temporal logic*, siehe [ACH+95, S.22]), die den Namen TCTL (**timed computation tree logic**) trägt und für die Beschreibung von Aussagen wie in den obigen Beispielen besonders gut geeignet ist.

TCTL erlaubt die Beschreibung von System-Zuständen mithilfe einer Reihe von Konstrukten:

- Ein **Zustandsprädikat** (engl. *state predicate*) ist ein lineares Prädikat über die Variablen eines Automaten.
- Eine **TCTL-Formel** besteht aus einem oder mehreren Zustandsprädikaten (Φ_1, Φ_2 etc.), die über die bekannten booleschen Operatoren $\neg$, $\vee$, $\wedge$ sowie über die zusätzlichen TCTL-Operatoren in der Form $\Phi_1 \exists\mathscr{U} \Phi_2$ oder $\Phi_1 \forall\mathscr{U} \Phi_2$ und $z.\Phi$ miteinander verknüpft sind.
- Die **temporalen Operatoren** $\exists\mathscr{U}$ und $\forall\mathscr{U}$ erlauben Prädikate über Zustandsübergänge.

  Wenn für einen Zustand σ die TCTL-Formel $\Phi_1 \exists\mathscr{U} \Phi_2$ erfüllt ist, bedeutet dies, daß für den Automaten *irgendein* Ablauf $\sigma \mapsto^* \sigma'$ existiert, wobei zu jedem Zeitpunkt des Ablaufs das Prädikat $\Phi_1 \vee \Phi_2$ erfüllt ist und im Zustand σ' das Prädikat Φ_2 erfüllt ist.

  Analog gilt: Wenn die TCTL-Formel $\Phi_1 \forall\mathscr{U} \Phi_2$ erfüllt ist, bedeutet dies, daß für *alle* möglichen Abläufe $\sigma \mapsto^* \sigma'$ zu jedem Zeitpunkt des Ablaufs das Prädikat $\Phi_1 \vee \Phi_2$ erfüllt ist und im Zustand σ' das Prädikat Φ_2 erfüllt ist.
- Mithilfe des **Rücksetz-Quantors** $z.\Phi$ können Zeitbeschränkungen spezifiziert werden. So fordert z.B. die TCTL-Formel $z.(\mathit{true} \exists\mathscr{U} (\Phi \wedge z \leq 5))$, daß ein Ablauf existiert, für den Φ innerhalb von 5 Zeiteinheiten erfüllt ist. z ist dabei eine „externe Uhr“, die nur zum Zweck der Analyse existiert und z.B. die Zeit seit dem Start des Systems mißt.
  Wenn für $z.\Phi$ alle weiteren Vorkommen der Variable z auf Φ beschränkt sind, nennen wir Φ eine geschlossene (engl. *closed*) Formel.

Für häufig vorkommende TCTL-Terme gibt es Abkürzungen, so z.B.

$\forall\Diamond\Phi$	anstelle von	$\mathit{true} \forall\mathscr{U} \Phi$
$\exists\Diamond\Phi$	anstelle von	$\mathit{true} \exists\mathscr{U} \Phi$
$\exists\Box\Phi$	anstelle von	$\neg\forall\Diamond\neg\Phi$
$\forall\Box\Phi$	anstelle von	$\neg\exists\Diamond\neg\Phi$
$\exists\Diamond_{<5}\Phi$	anstelle von	$z.(\exists\Diamond(\Phi \wedge z < 5))$

In [ACH+95, S.23] wird ein Algorithmus für lineare hybride Automaten vorgestellt, mit dessen Hilfe sich TCTL-Formeln validieren lassen. Die Terminierung des Algorithmus ist jedoch nur für Multirate-Zeit-Automaten garantiert.

3 Praktische Anwendung

Nachdem die theoretischen Grundlagen von hybriden Automaten vorgestellt wurden, soll nun demonstriert werden, wie sie in der Praxis zur Lösung eines Problems eingesetzt werden können. Dabei wird nicht nur betrachtet, wie gut sich die vorgegebene Aufgabe mit den vorhandenen Mitteln modellieren läßt, sondern es wird auch untersucht, welche Eigenschaften des Systems mit den angesprochenen Algorithmen bewiesen werden können.

Im Rahmen der Fallstudie in [MS00], auf die sich dieser Abschnitt stützt, gehen die Autoren außerdem auf ihre Erfahrungen mit den zur Zeit vorhandenen Software-Tools für die Verifikation von hybriden Automaten ein und sind so in der Lage zu beurteilen, inwiefern die im vorherigen Kapitel gemachten Aussagen sich auch auf die Praxis übertragen lassen.

3.1 Szenario: Elektronische Höhenregelung

Bei dem System, das modelliert werden soll, handelt es sich um die elektronische Höhenregelung (EHC, *electronic height control*) des Fahrgestells eines Autos. Je nach Fahrweise und Beschaffenheit des Untergrunds werden dabei die vier Radaufhängungen unabhängig voneinander pneumatisch nach oben oder unten verschoben. Ziel ist es, den Fahrkomfort zu erhöhen, die Scheinwerfer unabhängig von der Beladung immer gleich ausgerichtet zu lassen und den Abstand des Fahrgestells vom Boden bei Fahrten auf unebenen Wegen und Straßen zu vergrößern.

Zu jedem Zeitpunkt wird durch Sensoren die Abweichung der gewünschten Höhe von der tatsächlichen gemessen. Wie Abbildung 3 zeigt, wird dabei unterschieden, ob sich die Abweichung vom gewünschten Wert (als *sp* bzw. *set point* bezeichnet) noch innerhalb oder bereits außerhalb eines bestimmten Toleranzbereiches bewegt. Bei der Ermittlung der Abweichung werden höherfrequente Änderungen der Radhöhe, wie sie z.B. durch Schlaglöcher oder Kopfsteinpflaster entstehen, herausgefiltert.

Sobald die Abweichung den äußeren Toleranzbereich (*oto, outer tolerance interval*) verläßt, wird die entsprechende Korrekturmaßnahme angestoßen und die Radaufhängung solange verschoben, bis die Abweichung wieder innerhalb des inneren Toleranzbereichs (*iti, inner tolerance interval*) liegt. Zusätzlich existiert ein Sensor, der anzeigt, ob das Auto gerade durch eine Kurve fährt; solange dies der Fall ist, soll die Fahrgestell-Höhe unverändert bleiben.

Das Anheben des Fahrgestells geschieht durch Einschalten eines Kompressors. Die Geschwindigkeit, mit der angehoben wird, soll im Intervall $[cp_{\min}, cp_{\max}]$ liegen, wobei $cp_{\min}$ und $cp_{\max}$ vorgegebene Parameter sind (*cp* steht für *compressor*). Analog soll das Absenken des Fahrgestells mit einer Geschwindigkeit im Intervall $[ev_{\min}, ev_{\max}]$ durch Öffnen eines Ventils erfolgen (*ev* steht für *escape valve*). Es gilt: $ev_{\max} < 0 < cp_{\min}$.

Es sollte erwähnt werden, daß die Aufgabenstellung in dieser Form bereits eine Reihe von Vereinfachungen enthält. So ist das individuelle Verschieben der einzelnen Räder in der Praxis komplex, da für alle vier Räder nur ein einziger Kompressor existiert, der Luft in die Stoßdämpfer pumpen kann – um die gewünschte Wirkung zu erzielen, muß neben dem Kompressor auch ein

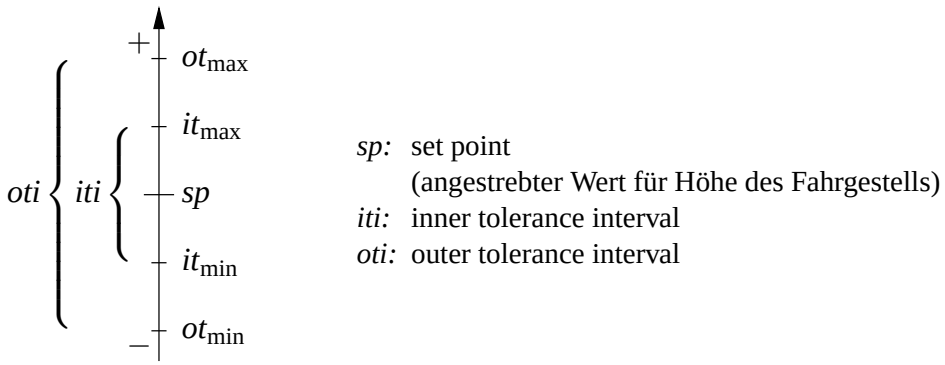

Abbildung 3: *Toleranzbereiche für die Abweichung der Fahrgestellhöhe vom Idealwert.*

System von Ventilen korrekt angesteuert werden. In unserem Modell beschränken wir uns jedoch auf allgemeine Impulse „Rad heben“ bzw. „Rad senken“, ohne dies zu beachten. Überdies wird nur ein einzelnes Rad modelliert.

Eine weitere Vereinfachung gibt es bezüglich der verschiedenen möglichen diskreten Zustände: Während im Modell lediglich die Zustände „fährt“ und „Kurve“ unterschieden werden, existieren in Wirklichkeit noch weitere (z.B. „Fahrzeug steht, Motor aus“), und die Werte von *sp*, *iti* und *oti* verändern sich abhängig vom Zustand.

3.2 Beschreibung der Komponenten

Die elektronische Höhenregelung kann in mehrere Teile aufgeteilt werden, die jeweils durch einen hybriden Automaten spezifiziert werden: Ein Filter, ein Zeitgeber, die restlichen Sensoren (in [MS00] *plant*, „Anlage“, genannt) und der Controller. Wie wir in den folgenden Abschnitten sehen werden, haben Filter, Zeitgeber und restliche Sensoren eine eher untergeordnete Rolle und versorgen hauptsächlich den Controller mit Eingabedaten. Die Automaten der Komponenten kommunizieren dabei über gemeinsame Variablen und Synchronisationsbezeichnungen. Wie Abbildung 4 zeigt, beeinflußt der Controller durch die Ansteuerung von Kompressor und Ventilen die Fahrgestellhöhe, die wiederum als gefilterte Größe die Entscheidungen des Controllers mitbestimmt – es findet also eine für Regel-Aufgaben typische Rückkopplung statt.

Für die Verifikation von hybriden Automaten existieren eine Reihe von Verifizierungsprogrammen. Unter ihnen sind HYTECH, KRONOS und UPPAAL am bekanntesten. Laut [MS00, S.7] ist jedoch nur HYTECH in der Lage, Spezifikationen für lineare hybride Automaten zu verarbeiten, die anderen Programme beschränken sich auf eingeschränkte Modelle, z.B. im Fall von UPPAAL Zeit-Automaten. Darüber hinaus unterstützt HYTECH die Angabe von Parametern anstelle von Konstanten sowie den Einsatz von „Monitor-Automaten“ für das Überprüfen komplexerer Anforderungen.

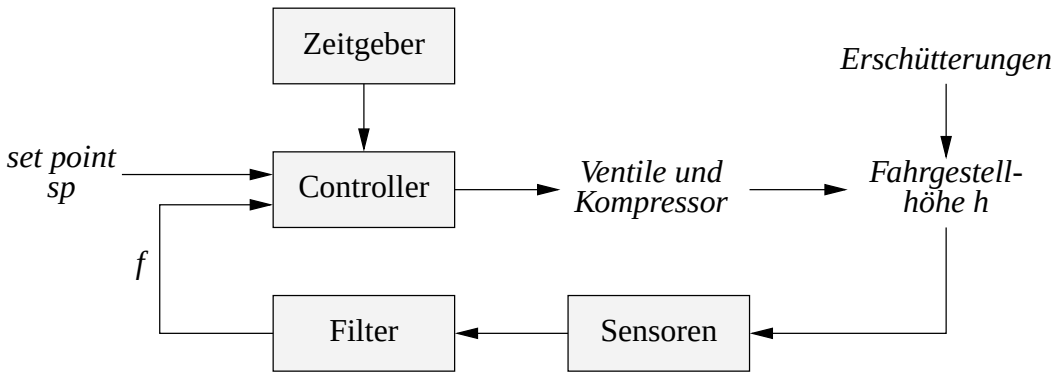

Abbildung 4: *Es findet eine Rückkopplung der Steuerimpulse des Controllers statt.*

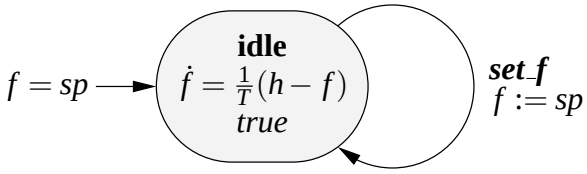

Abbildung 5: *Der Filter liest die tatsächliche Fahrgestell-Höhe h und gibt die gefilterte Höhe f aus.*

3.2.1 Filter

Der Filter verarbeitet Sensordaten über die aktuelle Höhe h des Fahrgestells, bevor sie als gefilterte Größe f an den Controller weitergegeben werden. Wie Abbildung 5 zeigt, ist es außerdem möglich, den Filter über die Synchronisationsbezeichnung *set_f* in seinen Ausgangszustand zurückzuversetzen – das Zurücksetzen wird durch den Controller veranlaßt, sobald nach einer Korrektur der Fahrgestellhöhe der innere Toleranzbereich *iti* erreicht wurde.

Die notwendige Dämpfung von hochfrequenten Schwingungen im Eingangssignal h ist durch den Filter gewährleistet, da er das Ausgangssignal anhand der Differentialgleichung $\dot{f} = \frac{1}{T}(h - f)$ berechnet. T ist die Zeitkonstante des Filters.

Unglücklicherweise ist der Filter aufgrund der gewählten Berechnungsformel kein linearer hybrider Automat: Zwar handelt es sich um eine lineare Differentialgleichung, nicht aber um ein konvexes, lineares Prädikat. Um das Verhalten in einer durch HYTECH verständlichen Form darzustellen, muß der Filter durch einen anderen Automat approximiert werden, wofür verschiedene Techniken existieren. In [MS00] wird *linear phase-portrait approximation*, also die stückweise lineare Annäherung an die ursprüngliche Funktion, gewählt. Dabei wird der Ort, in dem die Differentialgleichung gelten soll, in mehrere Orte aufgeteilt, von denen jeder für einen Teil des Zustandsraums des ursprünglichen Orts verantwortlich ist (siehe Abbildung 6). Die Orte sind so miteinander verbunden, daß sie einen vollständigen Graph darstellen, d.h. es können jederzeit Übergänge von jedem Ort zu jedem anderen stattfinden. Auch Übergänge vom ursprünglichen

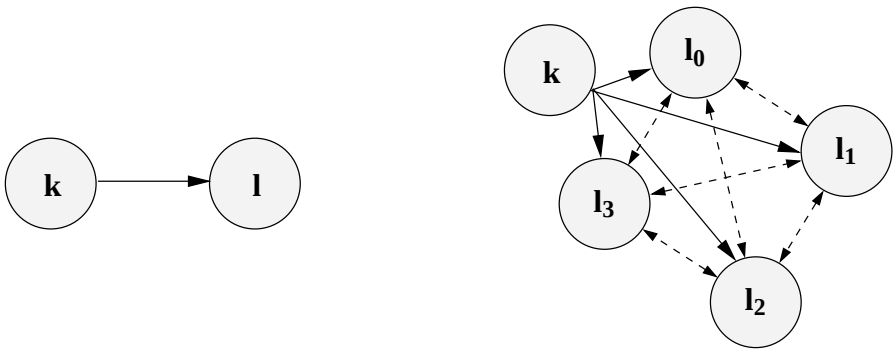

Abbildung 6: *Schema für die* linear phase-portrait approximation*: Der Ort l im ursprünglichen Automaten (links) wird in die vier Orte l_0, l_1, l_2, l_3 aufgeteilt.*

Ort zu anderen Orten des Automaten werden für jeden der neuen Orte repliziert.

Für jeden der Orte wird eine neue *flow condition* vorgegeben, die für die zugehörige Zustandsraum-Partition nahe an der ursprünglichen Funktion liegt, aber ein konvexes lineares Prädikat ist. Dies kann z.B. durch Berechnen der Extremwerte der Originalfunktion für den jeweiligen Bereich geschehen. Im Fall des Filters ergeben sich für eine Zeitkonstante $T = 2$ und eine Aufteilung Werteraums von $h - f$ in sechs Partitionen folgende Invarianten und Differentialgleichungen (siehe [MS00, S.9]):

Ort	**Invariante**	**Flow condition**
$idle_0$	$h - f \in (-\infty, -10]$	$\dot{f} \in (-\infty, -5]$
$idle_1$	$h - f \in [-10, -6]$	$\dot{f} \in [-5, -3]$
$idle_2$	$h - f \in [-6, 0]$	$\dot{f} \in [-3, 0]$
$idle_3$	$h - f \in [0, 6]$	$\dot{f} \in [0, 3]$
$idle_4$	$h - f \in [6, 20]$	$\dot{f} \in [6, 10]$
$idle_5$	$h - f \in [20, \infty)$	$\dot{f} \in [10, \infty)$

Mithilfe dieser Approximationsmethode kann der Filter nun durch einen linearen hybriden Automaten mit sechs Zuständen dargestellt werden. Ein Nachteil der vorgestellten Methode ist aber, daß die Partitionierung des Werteraums von Hand erfolgen muß, wobei zwischen einer geringen Gesamtzahl von Zuständen auf der einen Seite und einer genauen Annäherung an die Funktion auf der anderen Seite abgewogen werden muß. Ebenso müssen die Funktionen, deren Wertebereich partitioniert wird, sorgfältig ausgewählt werden. Im vorliegenden Fall hätten h und f z.B. auch jeweils für sich partitioniert werden können. Bei gleicher Qualität des Ergebnisses wären dafür aber $6 \cdot 6 = 36$ Orte vonnöten gewesen.

3.2.2 Zeitgeber

Der hybride lineare Automat für den Zeitgeber ist in Abbildung 7 zu sehen. Er ist so aufgebaut, daß sein Übergang alle *t_sample* Zeiteinheiten benutzt wird. Die Synchronisationsbezeichnung *nc* wird auch von den meisten Übergängen des Controllers benutzt, so daß der Controller durch diesen Automaten „getaktet“ wird.

3.2.3 Weitere Sensoren

Mithilfe des Automaten in Abbildung 8 werden die Umwelteinflüsse modelliert, die von außen auf das System einwirken und dadurch die Werte von Variablen ändern.

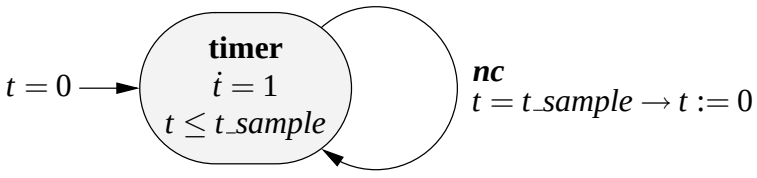

Abbildung 7: *Der Zeitgeber benutzt alle t_sample Zeiteinheiten den nc-Übergang.*

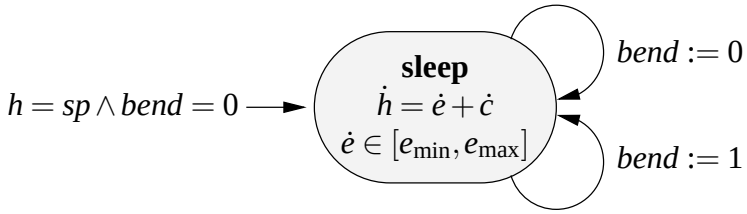

Abbildung 8: *Die restlichen Sensoren des Systems messen, ob sich das Fahrzeug in einer Kurve befindet und wie groß die Erschütterungen sind.*

Durch die beiden Übergänge, bei deren Benutzung die Variable *bend* verändert wird, ist angedeutet, daß das Auto zu jedem Zeitpunkt in eine Kurve fahren kann. Übergänge im Controller, die den Wert von *bend* überprüfen, sorgen dafür, daß die Höhenregelung ausgesetzt wird, solange $bend = 1$ gilt.

Ebenfalls von diesem Automaten werden Veränderungen des Werts von h spezifiziert, also der ungefilterten Höhe des Fahrzeuggestells. Wie aus der Abbildung ersichtlich setzt sich $\dot{h}$ aus zwei weiteren Variablen zusammen, nämlich den Erschütterungen des Fahrzeuggestells $\dot{e}$ und den Höhenveränderungen $\dot{c}$, die der Controller durch sein Anheben bzw. Absenken des Rades verursacht.

Damit bei der Analyse des gesamten Systems untere und obere Schranken für h im Fall von Erschütterungen ermittelt werden können, muß der Wert von e einer zusätzlichen Einschränkung unterworfen sein, und zwar in der Weise, daß die größtmögliche Erschütterung immer noch in derselben Zeiteinheit durch ein Anheben bzw. Senken des Rades kompensiert werden kann. Genauer ausgedrückt bedeutet das: $e_{\min} \geq ev_{\max} \wedge e_{\max} \leq cp_{\min}$ – wie bereits auf Seite 12 erwähnt sind $ev_{\max}$ und $cp_{\min}$ die betragsmäßig kleinstmöglichen Geschwindigkeiten, mit der das Rad abgesenkt bzw. angehoben werden kann. Leider entspricht diese Einschränkung in keiner Hinsicht der Wirklichkeit, wo es ohne weiteres möglich ist, daß Erschütterungen auftreten, die nicht sofort korrigiert werden können.

3.2.4 Controller

Beim Controller handelt es sich um die zentrale Einheit, die die gefilterte Fahrgestellhöhe f, die Variable für die Kurvenerkennung *bend* sowie Zeittakte über den *nc*-Übergang des Zeitgebers verarbeitet. Sie initiiert ihrerseits durch Verändern der Variable $\dot{c}$ ein Anheben ($\dot{c} > 0$) bzw. Absenken ($\dot{c} < 0$) des Fahrgestells, denn $\dot{c}$ fließt zusammen mit den Erschütterungen $\dot{e}$ in die tatsächliche Fahrgestellhöhe h ein, aus der der Filter wiederum f errechnet.

Der hybride Automat für den Controller ist in Abbildung 9 zu sehen. Sein Ziel ist es zu reagieren, sobald der Wert von f den äußeren Toleranzbereich $[ot_{\min}, ot_{\max}]$ verläßt, und die Fahrgestellhöhe dann solange zu verändern (außer wenn sich das Fahrzeug in einer Kurve befindet), bis f wieder innerhalb des inneren Toleranzbereichs $[it_{\min}, it_{\max}]$ liegt.

Die Ausführung beginnt im Ort *in_tolerance*. Wird f zu klein oder zu groß, findet beim

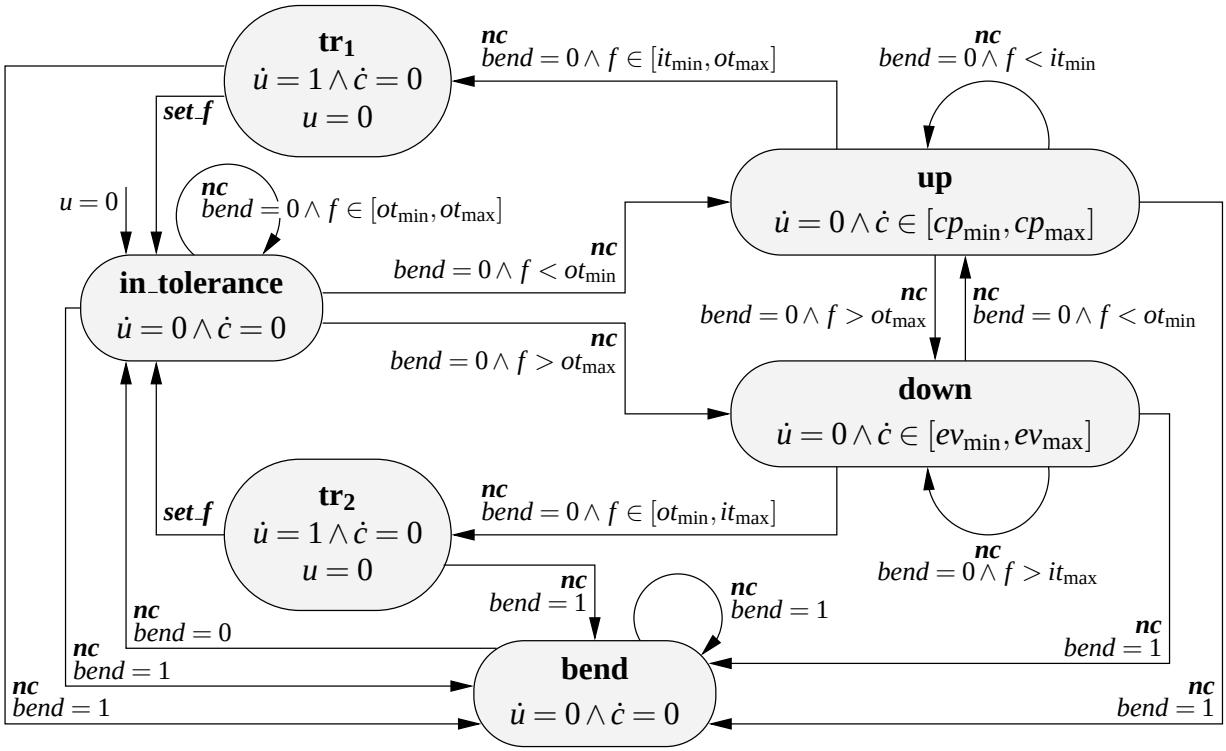

Abbildung 9: *Der Controller steuert über die Variable c die Fahrgestell-Höhe*

nächsten *nc*-Impuls (und wenn $bend = 0$) ein Übergang zum Ort *up* bzw. *down* statt. Dort wird c so lange verändert, bis der innere Toleranzbereich erreicht ist. Sobald dieses Ziel erreicht ist, wird der Übergang zu tr_1 bzw. tr_2 benutzt; dort wird das weitere Anheben bzw. Absenken beendet ($\dot{c} = 0$). Nur im Fall von sehr starken Erschütterungen kann alternativ der direkte Übergang von *up* nach *down* oder umgekehrt stattfinden.

Die Variable u dient ausschließlich dem Zweck, dafür zu sorgen, daß tr_1 und tr_2 nach Ankunft in dem Ort sofort wieder verlassen werden (*tr* steht für *transient*) – bei der Ankunft kann die Invariante $u = 0$ durch Setzen der Variable auf diesen Wert erfüllt werden, aber da die *flow condition* $\dot{u} = 1$ verlangt, muß der Ort dann *sofort* (ohne daß Zeit vergeht) über den Übergang zu *in_tolerance* wieder verlassen werden. Der Grund für dieses Vorgehen ist, daß eigentlich ein Übergang benötigt würde, der mit zwei Bezeichnungen gleichzeitig synchronisieren kann: Mit *nc*, da der Controller stets mit dem Zeitgeber synchronisiert ist, und mit *set_f*, um den Filter vor der Rückkehr nach *in_tolerance* zurückzusetzen. Unser Modell erlaubt jedoch nur *eine* Synchronisationsbezeichnung pro Übergang.

Von allen Orten aus (mit Ausnahme von tr_1 und tr_2, für die es keinen Sinn macht) existieren zusätzlich noch Übergänge zum Ort *bend*, der das Verhalten beim Durchfahren einer Kurve modelliert. Er sorgt dafür, daß die Höhe des Fahrgestells während dieser Zeit nicht verändert wird, d.h. $\dot{c} = 0$. Erst bei $bend = 0$ wird der Übergang zu *in_tolerance* benutzt, von dem aus die Höhenregelung erneut angestoßen werden kann.

3.3 Ergebnisse der Fallstudie

Im Anschluß an die Modellierung des Szenarios „elektronische Höhenregelung“ geben die Autoren in [MS00, S.13] einen Überblick über die Erfahrungen, die sie bei der Durchführung des Projekts gemacht haben.

Formales Modell „hybrider Automat“ Folgende Aspekte des für die Modellierung verwendeten Modells der hybriden Automaten werden als besonders positiv erachtet:

- Hybride Automaten sind ein mächtiges Ausdrucksmittel, mit dem die diskreten und kontinuierlichen Teile eines Systems in einer Spezifikation vereinigt werden können oder voneinander getrennt werden können, je nachdem, welche Möglichkeit vorteilhafter erscheint.
- Andere Spezifikationsansätze, bei denen die diskreten und kontinuierlichen Anteile voneinander getrennt sind, reichen nicht aus, um Elemente wie z.B. den Filter zu modellieren, der eine diskrete Rücksetz-Operation besitzt.
- Im Gegensatz zu anderen Ansätzen existiert bei hybriden Automaten eine gemeinsame semantische Grundlage für die diskreten und kontinuierlichen Teile einer Spezifikation.

Allerdings werden auch eine Reihe von Schwächen bei der Systemspezifikation mittels hybrider Automaten bemängelt:

- Hybride Automaten sind nicht modular in der Hinsicht, daß das Resultat der Parallelschaltung zweier Automaten A und B eine andere Menge erreichbarer Zustände haben kann als das der Parallelschaltung der Automaten A' und B, auch wenn A' die gleiche Menge erreichbarer Zustände hat wie A.
- Die Verifikation der einzelnen Module jeweils für sich ist nicht möglich, d.h. vor der Verifikation müssen die Teilautomaten zu einem großen Automaten vereinigt werden, was sich sehr ungünstig auf die Komplexität der Lösungssuche auswirkt.
- Bislang existiert keine Grundlage zur Definition hierarchischer Strukturen, also z.B. die Möglichkeit, einen einzigen Ort bei einer Verfeinerung des Designs durch mehrere Orte zu ersetzen. Dieser Nachteil wiegt jedoch nicht schwer, da sich dies allein durch eine Erweiterung der Spezifikations-Syntax bewerkstelligen läßt.
- Übergänge hybrider Automaten können nur eine einzige Synchronisationsbezeichnung haben. Im Fall des Controllers wurden wir gezwungen, dieses Problem durch Einführen zusätzlicher Zustände zu lösen – eine Erweiterung des Modells ist aber problematisch.
- Es kann mitunter schwer sein, Fehler in der Spezifikation zu entdecken, die dazu führen, daß der Automat angehalten wird, so z.B., wenn eine Invariante spezifiziert, daß der Automat sich höchstens für eine bestimmte Zeitspanne in einem Ort befinden darf, es aber nicht möglich ist, den Ort noch innerhalb dieser Zeitspanne zu verlassen. Beim Aufstellen der Invarianten muß deswegen sorgfältig vorgegangen werden.

Mächtigkeit der verfügbaren Software-Tools Die Autoren von [MS00] überprüften mit HYTECH einige Eigenschaften des erstellten Systems, so z.B. unter anderem, daß die Höhe des Fahrgestells den äußeren Toleranzbereich für bestimmte Parameter niemals länger als einen bestimmten Zeitraum verläßt, oder daß die Räder in Kurven nie angehoben oder abgesenkt werden. Die automatische Verifikation solcher Aussagen benötigte in manchen Fällen auf der verwendeten Hardware (Sun Sparcstation 20) mehr als eine Stunde Zeit.

Die Tatsache, daß die Verifikation mithilfe der Tools vollständig automatisierbar ist, wird als großer Vorteil angeführt. In der Praxis überwiegen allerdings die Nachteile, die teils prinzipbedingt scheinen, teils aber auch nur zeigen, daß die Software-Lösungen noch nicht ausgereift sind:

- Aufgrund mangelnder Effizienz der Tools (Speicherbedarf und benötigte Rechenzeit sind sehr groß) müssen die Modelle oft stark vereinfacht werden, um die automatisierte Analyse durchführen zu können. Verbesserungen durch effizientere Algorithmen und Datenstrukturen sind hier jedoch noch möglich und auch dringend nötig.

- Die Approximation nichtlinearer hybrider Automaten, wie z.B. im Fall des Filters, erhöht die Komplexität des entsprechenden linearen Automaten leider stark.

- Die benutzte Version von HYTECH litt unter Problemen bei der Nutzung von Standard-Libraries, die sich in häufigen arithmetischen Überläufen äußerten.

4 Zusammenfassung

Bei hybriden Automaten handelt es sich um eine interessante und flexible Erweiterung von endlichen Automaten, nicht nur, weil das Konzept viel mächtiger ist als diese, sondern vor allem auch, weil hybride Automaten besser geeignet zur Modellierung von Aufgaben sind, wie wir sie heute in vielen Bereichen der Informatik antreffen, so auch bei eingebetteten Systemen. Gerade für eingebettete Systeme, die zunehmend auch Funktionen übernehmen, in denen sie lebenswichtige Entscheidungen treffen, ist auch die Eigenschaft hybrider Automaten von Vorteil, daß eine automatische, formale Überprüfung der Modelle möglich ist.

Wie wir am Fallbeispiel der elektronischen Höhenregelung gesehen haben, erlauben heute existierende Software-Tools bereits die Analyse recht komplexer Automaten. Allerdings treten bei der praktischen Nutzung auch eine Reihe von Problemen zutage, wobei vor allem die geringe Effizienz bei der Verifikation der Automaten bedenklich ist. Aufgrund verbesserter Algorithmen auf der einen und stetig wachsender Prozessorleistung auf der anderen Seite kann man jedoch davon ausgehen, daß hybride Automaten beim Design von Systemen in Zukunft vermehrt bei Fällen zum Einsatz kommen werden, in denen die Fehlerfreiheit des Endprodukts gewährleistet sein muß.

Literatur

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine: The Algorithmic Analysis of Hybrid Systems. *Theoretical Computer Science*, 138:3–34, 1995.

[Broy98] Manfred Broy: Informatik, eine grundlegende Einführung. Teil 2: Systemstrukturen und theoretische Informatik. Springer-Verlag, 1998.

[MS00] O. Müller and T. Stauner: Modelling and Verification using linear hybrid automata – a case study. *Mathematical and Computer Modelling of Dynamical Systems*, 6(1):71–89, 2000.

[SMF97] T. Stauner, O. Müller, and M. Fuchs: Using HYTECH to verify an automotive control system. In *Proc. Int. Workshop on Hybrid and Real-Time Systems (HART'97)*, LNCS 1201, pp. 139–153. Springer-Verlag, 1997.