Einleitung

Bevor ein reales System, wie ein Kaffeeautomat oder ein Flugzeugsteuerungssystem, gebaut wird, werden einerseits die Anforderungen an das System, andererseits das System selbst in irgendeiner Form *beschrieben*. Nun werden in aller Regel nicht alle Details eines solchen Systems in der Beschreibung erfasst: Vielmehr wird das System auf einer geeigneten Abstraktionsebene beschrieben. Eine formale Beschreibung eines Systems wird oft *Spezifikation* des Systems genannt.

Es gibt eine Reihe formaler Methoden, die es sowohl erlauben, ein System und die Anforderungen an das System zu spezifizieren, als auch zu prüfen, ob das System die geforderten Eigenschaften tatsächlich besitzt. Folglich wird bereits auf der Beschreibungsebene nachgewiesen, ob sich das System korrekt verhält. Dies wird *Verifikation* des Systems genannt.

Eine der vielen Methoden, die zur Spezifikation und Verifikation von Systemen dienen, ist die *Temporale Logik*. Ende der 70er Jahre wurde sie von Pnueli [83] als ein geeigneter Formalismus vorgeschlagen und hat sich seitdem zunehmend durchgesetzt. Ein wesentlicher Vorteil der Temporalen Logik als einer Spezifikationssprache ist der Folgende:

Sowohl das System als auch seine gewünschte Eigenschaft werden jeweils als eine temporallogische Formel spezifiziert. Formal wird also zwischen einem System und seiner Eigenschaft nicht unterschieden. Die Verifikation des Systems reduziert sich dann auf den Nachweis, dass aus der Spezifikation des Systems die Spezifikation der Eigenschaft logisch folgt. Für den Beweis der Implikation wird gewöhnlicherweise ein geeignetes, zu der Logik dazugehöriges Regelwerk verwendet.

Eine besondere Herausforderung für Spezifikationstechniken, auch für logikbasierte Methoden, auf die wir uns im Rahmen dieser Arbeit konzentrieren, stellen sog. *verteilte Systeme* dar. Ein verteiltes System besteht aus mehreren, räumlich verteilten Komponenten, die weitgehend unabhängig voneinander

ander arbeiten, und miteinander interagieren.

Ein verteiltes System wird meist nicht "als Ganzes" entwickelt. In der Regel werden die Komponenten des Systems zunächst einzeln entworfen, und dann zu einem Gesamtsystem zusammengesetzt. Später kann das Gesamtsystem weiterentwickelt werden, indem es um neue Komponenten ergänzt wird, oder einige seiner Komponenten gegen andere ausgetauscht werden.

Deshalb ist es notwendig, die Systemkomponenten auch einzeln zu spezifizieren. Aus dem Verhalten der Komponenten soll sich das Verhalten des Gesamtsystems ergeben. Spezifikationsmethoden, die solche komponentenweise Beschreibung des Systems erlauben, werden *kompositional* genannt.

Kompositionalen Spezifikationsmethoden bieten einen entscheidenden Vorteil: Einzeln beschriebene Systemkomponenten können unabhängig voneinander verifiziert werden. Die Korrektheit mehrerer Komponenten einzeln nachzuweisen ist in der Regel einfacher, als der Nachweis der Korrektheit für das meist viel größere Gesamtsystem. Aus der Korrektheit der Komponenten soll sich wiederum die Korrektheit des Gesamtsystems ergeben.

Folglich ist Kompositionalität einerseits ein erstrebenswertes Charakteristikum für eine logische Spezifikationstechnik, besonders eine solche, die für verteilte Systeme geeignet sein soll. Andererseits ist es nicht leicht, die Kompositionalität einer solchen Spezifikationsmethode zu garantieren. In einem der nachfolgenden Abschnitte geben wir einen Überblick über die Temporale Logiken, die zur Systemspezifikation entworfen wurden. Wie wir sehen werden, sind nur wenige davon kompositional. Alle uns bekannten kompositionalen Temporalen Logiken basieren auf Zustandssequenzen.

In dieser Arbeit entwickeln wir eine neue Temporale Logik, genannt *Temporal Logic of Distributed Actions* (kurz *TLDA*). TLDA ist eine kompositionale Temporale Logik, die zur Spezifikation und Verifikation insbesondere verteilter Systeme entworfen worden ist.[1] Der Logik liegt aber, im Gegenteil zu den anderen kompositionalen Temporalen Logiken, eine *halbgeordnete* semantische Struktur zugrunde, sog. *verteilte Abläufe*.

Verteilte Abläufe, die ein System modellieren, bringen einen zweifachen Vorteil mit sich: Zum einen lassen sie exakt und intuitiv manche Phänomene beschreiben, die typischerweise in verteilten Systemen auftreten, aber durch andere Verhaltensmodelle, wie *Zustandssequenzen* beispielsweise, nicht

[1] Nichtverteilte Systeme können als ein Spezialfall von verteilten Systeme aufgefasst werden. Die in der Arbeit entwickelten Techniken können daher ebenso auf solche Systeme angewendet werden.

erfasst werden. Zum anderen eignen sich verteilte Abläufe ausgezeichnet zur Repräsentation komponierter Systeme: Aus den Abläufen der Systemkomponenten lassen sich auf "natürliche" Weise die Abläufe des Gesamtsystems gewinnen.

In dieser Arbeit entwickeln wir eine kompositionale Methode zur Spezifikation von Systemen: Wir unterteilen ein System in Komponenten, die unabhängig voneinander arbeiten. Im ersten Schritt betrachten wir also ein System, in dem diese Komponenten *parallel* ausgeführt werden. Dies lässt sich mit verteilten Abläufen besonders einfach und anschaulich realisieren. Im zweiten Schritt addieren wir eine *Synchronisationsbedingung* hinzu, die das Zusammenspiel der Komponenten beschreibt. Im Extremfall, wie wir am Beispiel der Uhr sehen werden, können dadurch die Abläufe der Komponenten völlig sequentialisiert werden.

Sind wir an einer Eigenschaft des Systems interessiert, die "lokal" feststellbar ist, die also eine Komponente erfüllt, dann kann die Eigenschaft in der betreffenden Komponente verifiziert werden, bevor die Komponenten durch die Synchronisationsbedingung in das meistens größere Gesamtsystem integriert wird. Wir garantieren dabei, dass diese Eigenschaft im komponierten System ebenfalls erfüllt wird.

Diese Spezifikationsmethode kann genau so gut eingesetzt werden, wenn das Gesamtsystem noch nicht vorliegt, sondern aus einzelnen Bausteinen sukzessiv zusammengesetzt wird. Denn wir benötigen zur Komposition immer nur "lokales" Wissen über die Komponenten, die gerade durch die Synchronisation zusammengefasst werden. Solche synchronisierten Komponenten verhalten sich nämlich genauso, wie jede andere Komponente, und können weiter zur Komposition verwendet werden.

Verteilte Abläufe erlauben es, sowohl die parallele Ausführung von Systemkomponenten als auch ihre Synchronisation adäquat zu beschreiben. Jetzt ist eine entsprechende logische Beschreibung dieser Abläufe nötig, die eine solche kompositionale Vorgehensweise ermöglicht. Dafür zeichnen wir eine spezielle Klasse der Formeln in TLDA aus, die wir *umgebungsinvariant* nennen. Wir weisen in der Arbeit nach, dass sich diese Formeln aufgrund ihrer Eigenschaften insbesondere zur Spezifikation der Komponenten (und ihrer Synchronisation) sehr gut eignen.

Syntaktisch ist TLDA aus der bereits etablierten *Temporal Logic of Actions* (kurz *TLA*, [1, 8, 73, 54], vgl. auch [57]) von Lamport entstanden. Darauf deutet auch die Namensähnlichkeit hin.

TLA zeichnet sich durch eine einfache Syntax aus. Darüberhinaus lassen

sich in TLA die Sachverhalte “Ein System erfüllt eine Eigenschaft” und “Ein System ist komponiert aus den Systemen A und B” elegant als Implikation bzw. Konjunktion beschreiben. Diese Vorteile haben wir in TLDA beibehalten.

Wir haben aber eine gänzlich unterschiedliche semantische Struktur gewählt: Während TLDA auf verteilten Abläufen basiert, verwendet TLA eine spezielle Art der Zustandssequenzen, sog. *Stottersequenzen*. Diese bedingen wiederum die Kompositionalität von TLA. Obgleich das Prinzip des *Stotterns*, das wir in Abschnitt 7.3 genauer erläutern werden, ein recht einfacher und auch beliebter Lösungsansatz für die Kompositionalität einer Logik ist, hat es seinen Preis. In einem System können nämlich Aktionen, die keine bleibenden Veränderungen hinterlassen, wie beispielsweise das *Lesen* von Variablen, nicht erkannt und demzufolge auch nicht beschrieben werden. Dieses Problem tritt in verteilten Abläufen nicht auf. In Abschnitt 7.3 weisen wir auf einige weitere Phänomene hin, die in verteilten Systemen auftreten, sich in TLDA leicht beschreiben lassen, aber in TLA, meist wegen der Stottersequenzen, nicht ausdrückbar sind. Daher sehen wir in verteilten Abläufen eine bessere Alternative für das semantische Modell einer kompositionalen Logik.

In den folgenden Abschnitten stellen wir intuitiv die Grundkonzepte der Arbeit am Beispiel einer *Uhr* vor. Dieses Beispiel haben wir von Lamport [55] übernommen. Wir beschreiben dann kurz den Aufbau der Arbeit. Zum Schluss geben wir einen Überblick über die Forschung in diesem Bereich.

System

Wir definieren ein System als eine Menge verteilter Abläufe. Um dies genauer zu erklären, betrachten wir ein einfaches System, nämlich eine *Stundenuhr*, die auf ihrem elektronischen Display nur Stunden anzeigt. Die Funktionsweise einer Stundenuhr kann leicht mit einer *Variablen*, hr, beschrieben werden: hr hat am Anfang einen Wert aus $\{0, \ldots, 23\}$. Bei jedem *lokalen Übergang* der Stundenuhr wird die Variable hr *aktualisiert*, indem hr der nächste Wert, $suc(hr)$, zugewiesen wird. Dabei wird $suc(hr)$ erwartungsgemäß wie folgt berechnet: $suc(23) = 0$ und $suc(hr) = hr + 1$ für $hr \neq 23$. Folglich befindet sich hr zu einem gegebenen Zeitpunkt in einem der *lokalen Zustände* 22 23 00 01 ... Eine solche Folge lokaler Zustände, d.h. eine Sequenz von Werten, die hr nacheinander zugewiesen werden, heißt *Historie von* hr.

Ein *Ablauf* eines Systems besteht aus Historien einiger relevanter Variablen und lokalen Übergängen, die zwischen lokalen Zuständen dieser Varia-

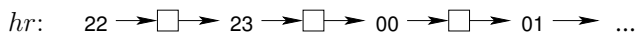

Abbildung 1: Ein Ablauf der Stundenuhr

blen stattfinden. Lokale Übergänge werden graphisch als Quadrate dargestellt. Durch Pfeile wird gekennzeichnet, welche Variablen in einen lokalen Übergang *involviert* sind. Im Falle einer Stundenuhr ist hr die einzige in die lokalen Übergänge involvierte Variable. Abb. 1 zeigt ein Beispiel für einen Ablauf der Stundenuhr. Eine Stundenuhr ist also die Menge aller solchen Abläufe.

Die lokalen Übergänge in einer Stundenuhr sind offenbar zeitlich *geordnet*. Somit ist eine Stundenuhr ein typisches sequentielles System. In verteilten Systemen dagegen sind lokale Übergänge meistens zeitlich nicht geordnet, d.h. *nebenläufig*. Folglich wird in einem Ablauf eines Systems eine *Halbordnung* auf lokalen Übergängen definiert. Somit sind auch lokale Zustände verschiedener Variablen halbgeordnet. Eine Menge lokaler Zustände, in denen sich Variablen gleichzeitig befinden können, beschreiben einen *globalen Zustand*. Demzufolge sind auch globale Zustände in einem Ablauf halbgeordnet. Betrachtet man *alle* lokalen Übergänge, die in einem globalen Zustand stattfinden können, dann kann für einen solchen globalen Zustand der globale *Folgezustand* eindeutig festgestellt werden. Ein globaler Zustand und sein Folgezustand samt den (nebenläufigen) lokalen Übergängen dazwischen bilden einen *Schritt*. Somit betrachten wir stets *maximale* Schritte in einem Ablauf.

Sowohl das Konzept eines Ablaufs als auch seine graphische Darstellung sind stark an ein *Kausalnetz* angelehnt (vgl. [16, 89, 95]), das aus der Theorie der *Petrinetze* wohl bekannt ist.

Temporallogische Spezifikation

Nun spezifizieren wir die Stundenuhr mit einer temporallogischen Formel. Eine solche Formel beschreibt die Schritte eines Ablaufs.

Der Wert von hr in einem globalen Zustand wird direkt mithilfe der Variable hr beschrieben, z.B. $hr = 22$. Der Wert von hr im Folgezustand wird, wie in vielen Temporalen Logiken (vgl. [68, 54, 69]), mithilfe der gestrichenen Variante der Variable hr repräsentiert, also $hr' = 23$. Folglich gilt in *jedem* Schritt eines Ablaufs der Stundenuhr $HR_{next} \triangleq hr' = suc(hr)$. Um

dies auszudrücken, verwenden wir den Temporaloperator $\Box$. Somit ist

$$Hr \triangleq \Box\, HR_{next}$$

die erste naive Spezifikation der Stundenuhr. (Dabei ignorieren wir der Einfachheit halber die Anfangswerte von *hr*.) Damit gehören alle Abläufe, in denen *Hr* gilt, zur Stundenuhr.

Komponiertes System

Schließlich soll die Stundenuhr mit einem ähnlichen System, nämlich einer *Minutenuhr* zu einer *Uhr*, komponiert werden. Für die Minutenuhr nehmen wir lediglich an, dass sie, analog zu der Stundenuhr, nur Minuten auf ihrem Display anzeigt und mit einer Formel *Min* spezifiziert wird, die eine einzige Variable *min* verwendet.

Die Stunden- und Minutenuhr arbeiten in der Uhr parallel zueinander; die Komposition der beiden Komponenten besteht aus geeigneter *Synchronisation*. Demnach soll die Variable *hr* in der Uhr genauso aktualisiert werden wie in der Stundenuhr. Ebenso soll die Variable *min* in der Uhr so verändert werden wie in der Minutenuhr. Auf der anderen Seite wird aber *min* in einer Uhr bekanntlich häufiger aktualisiert als *hr*. Deshalb muss das Verhaltensmodell für die Systeme so gewählt werden, dass es möglich ist, eine solche Sequenz lokaler Zustände von *min* "zwischen" zwei benachbarte lokale Zustände von *hr* einzufügen.

Gewöhnliche Zustandssequenzen erlauben das nicht. Denn beispielsweise kann aus einer Zustandssequenz der Stundenuhr $hr:\ 22\ \ 23\ \ 0\ \ 1\ \ 2\ \ \ldots$ und einer Zustandssequenz der Minutenuhr $min:\ 58\ \ 59\ \ 0\ \ 1\ \ 2\ \ \ldots$ niemals eine Zustandssequenz der Uhr

$$\begin{array}{rccccccc} hr: & 22 & 22 & 23 & 23 & 23 & \ldots \\ min: & 58 & 59 & 0 & 1 & 2 & \ldots \end{array}$$

gewonnen werden. Daher müssen Zustandssequenzen zu Stottersequenzen ([52]) erweitert werden, um Komponenten und komponierte Systeme adäquat modellieren zu können.

Verteilte Abläufe leisten das Verlangte "natürlicherweise": Da die lokalen Übergänge explizit dargestellt werden, werden im Falle der Uhr zwei Arten lokaler Übergänge modelliert: In die meisten, wie z.B. in den Übergang von 22:58 auf 22:59, ist nur die Variable *min* involviert. Immer, wenn die Minute

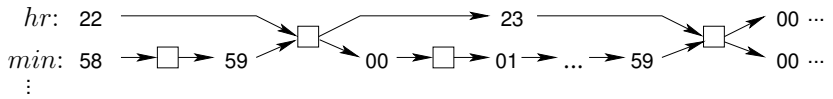

Abbildung 2: Ein Ablauf der Uhr.

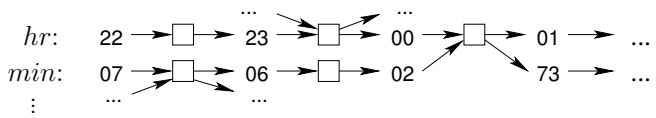

Abbildung 3: Ein Ablauf der Stundenuhr.

den Wert 59 erreicht hat, findet ein lokaler Übergang statt, in den sowohl *min* als auch *hr* involviert sind, wie z.B. der Übergang von 22:59 auf 23:00. Auf diese Art und Weise wird die Variable *hr* in der Uhr genauso häufig aktualisiert, wie in der Stundenuhr. Dies kann man am Ablaufbeispiel der Uhr aus Abb. 2 deutlich erkennen.

Um ein komponiertes System, wie die Uhr, einfach und elegant definieren zu können, nehmen wir an (wie auch TLA), dass jedes System über eine unendliche Menge aller Variablen $\mathcal{V}ar$ verfügt. So besitzt beispielsweise die Stundenuhr, neben *hr*, die wir zur Unterscheidung als *Systemvariable* der Stundenuhr bezeichnen, unendlich viele Variablen, insbesondere *min*. Dies wird in einem Ablauf (siehe z.B. Abb. 2) durch drei übereinander angeordnete Punkte angedeutet. Das Verhalten der Systemvariable *hr* ist in der Stundenuhr festgelegt. Für alle anderen Variablen aus $\mathcal{V}ar$, insbesondere *min*, wird angenommen, dass sie in der Stundenuhr beliebig aktualisiert werden. Demnach sieht ein Ablauf der Stundenuhr tatsächlich so aus wie in Abb. 3. Die Menge aller Abläufe dieser Form definiert die Stundenuhr. Ein Ablauf der Minutenuhr wird in Abb. 4 dargestellt.

Ein System, das aus der *Komposition* der Stundenuhr und Minutenuhr entstanden ist, kann jetzt als *Durchschnitt* der beiden Komponenten definiert werden.

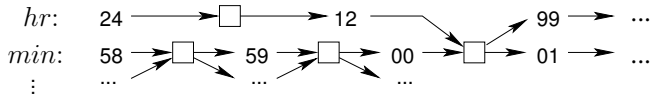

Abbildung 4: Ein Ablauf der Minutenuhr.

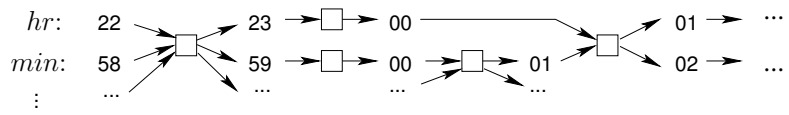

Abbildung 5: Ein Ablauf der Stunden- und Minutenuhr.

Nun enthält der Durchschnitt der Stunden- und Minutenuhr auch Abläufe wie den aus Abb. 5, in denen die beiden Komponenten beliebig synchronisiert werden. Offenbar ist dies kein korrekter Ablauf der Uhr. Genau genommen ist also die Uhr keine Komposition der Stunden- und Minutenuhr, sondern vielmehr eine geeignet *synchronisierte Komposition*.

Komposition temporallogischer Spezifikationen

Nun ist nach der Definition der Komposition jeder Ablauf der Uhr insbesondere ein Ablauf der Stundenuhr und ebenso ein Ablauf der Minutenuhr. Demzufolge erfüllt ein Ablauf der Uhr sowohl die Spezifikation *Hr* der Stundenuhr, die Aktualisierungen von *hr* beschreibt, als auch die Spezifikation *Min* der Minutenuhr, die Aktualisierungen von *min* festlegt. Folglich wird die Uhr als $Hr \wedge Min \wedge SYNC$ spezifiziert, wobei $SYNC$ eine geeignete Synchronisationsbedingung ist. Die Idee, die *Komposition* von Systemkomponenten als *Konjunktion* der Komponentenspezifikationen zu beschreiben, findet sich in den meisten logikbasierten Spezifikationsprachen, z.B. [9, 5, 6, 55, 19, 45] (vgl. auch [100] für einen allgemeineren Ansatz).

Jetzt stellen wir aber fest, dass die naive Spezifikation $Hr \triangleq \Box\, HR_{next}$ der Stundenuhr nicht im Ablauf der Uhr aus Abb. 2 gilt. Denn einige Schritte in diesem Ablauf, wie z.B. von 22:58 auf 22:59, aktualisieren nur die Variable *min*, während *hr* nicht involviert ist. Die Formel $\Box\, HR_{next}$ verlangt dagegen nach der Definition des Temporaloperators $\Box$, dass *hr* in jedem Schritt des Ablaufs gemäß HR_{next} verändert wird. Da dies nicht der Fall ist, gilt *Hr* nicht im Ablauf der Uhr aus Abb. 2.

Also muss die Spezifikation der Stundenuhr liberaler formuliert werden. In jedem Schritt eines Ablaufs der Stundenuhr soll Folgendes gelten: Wenn die Variable *hr* in einen lokalen Übergang involviert ist, dann wird sie gemäß HR_{next} aktualisiert. Die Involviertheit von Variablen in einen lokalen Übergang wird in TLDA mithilfe von $\sim$-Variablen ausgedrückt. Die Spezifikation