Maschinelle Verifikation von parametrisierten Echtzeitsystemen*

Thomas Göthel

Technische Universität Berlin
thomas.goethel@tu-berlin.de

Abstract: Sicherheitskritische Echtzeitsysteme umfassen häufig nicht nur eine statische, sondern eine parametrisierte, beliebig große Anzahl von nebenläufigen Prozessen. Ein Echtzeitbetriebssystem z.B. verwaltet beliebig viele Threads. Solche Systeme enthalten prinzipiell zwei Quellen von Unendlichkeit: Es existieren einerseits unendlich viele Systeminstanzen, die andererseits durch die Betrachtung von Echtzeit jeweils einen unendlich großen Zustandsraum besitzen. Mit formaler Verifikation kann die Korrektheit in allen Systemabläufen nachgewiesen werden. Dies ist jedoch für derartige unendliche Systeme mit komplexen Systemstrukturen automatisch prinzipiell nicht möglich. Bisher fehlt es an maschinellen, teilautomatisierten Verifikationsansätzen, die weiterhin die Korrektheit von Beweisen selbst sicherstellen. Um dies zu ermöglichen, haben wir ein Rahmenwerk entwickelt, das die Verifikation in einem interaktiven Theorembeweiser erlaubt. Dadurch können Teilbeweise automatisiert werden und es wird sichergestellt, dass (möglicherweise kritische) Randfälle nicht übersehen werden können. Um den damit verbundenen interaktiven Verifikationsaufwand zu reduzieren, ermöglichen wir es weiterhin, Systeminstanzen automatisch zu validieren. Damit können eventuelle Gegenbeispiele vor der Verifikation im Theorembeweiser genutzt werden, um das parametrisierte Gesamtsystem entsprechend zu korrigieren.

1 Einleitung

Echtzeitsysteme müssen oftmals in der Lage sein, mit beliebig vielen (nebenläufigen) Komponenten umgehen zu können. Der Scheduler eines Echtzeitbetriebssystem z.B. verwaltet beliebig viele Threads. Solche Systeme fallen in die Klasse der parametrisierten Systeme, bei denen der Parameter der Anzahl der Komponenten darstellt. Deren maschinengestützte Verifikation ist besonders schwierig, da sie im allgemeinen nicht automatisch verifiziert werden können und weiterhin sichergestellt werden muss, dass Beweise für ihre Korrektheit lückenlos sind, d.h. möglicherweise kritische Randfälle nicht übersehen werden. Bisherige Ansätze für die automatische Verifikation spezieller Subklassen von parametrisierten Systemen schränken die Struktur der betrachteten Systeme stark ein, indem z.B. die einzelnen Komponenten selbst nicht von der Anzahl der Komponenten abhängen dürfen (z.B. [GL08]) oder sie können nicht mit Echtzeit umgehen (z.B. [BLW05]). Weiterhin stellen diese Ansätze nicht sicher, dass die Beweise selbst korrekt sind. Bisherige

*Englischer Titel der Dissertation: “Mechanical Verification of Parameterized Real-Time Systems”

Theorembeweiser-basierte Ansätze, die prinzipiell mit unendlichen Systemen umgehen können und die Korrektheit von Beweisen sicherstellen, betrachten entweder keine Echtzeit (z.B. [IR05]) oder stellen sehr eingeschränkte bzw. unvollständige Mechanisierungen dar (z.B. [WWB10]). Bisher fehlt es an Ansätzen für die umfassende und maschinelle Verifikation von parametrisierten Echtzeitsystemen mit komplexen Systemstrukturen.

In [Göt12] präsentieren wir ein Rahmenwerk zur automatischen Validierung und maschinellen, umfassenden und semi-automatischen Verifikation parametrisierter Echtzeitsysteme mit komplexen Systemstrukturen. Den Kern unseres Rahmenwerks bildet der Prozesskalkül Timed CSP, mit dem funktionales und nicht-funktionales (zeitliches) Verhalten von nebenläufigen Systemen modelliert werden kann. Unsere wichtigsten Beiträge sind: Erstens, eine Formalisierung der operationalen Semantik von Timed CSP sowie Bisimulationsäquivalenzen im Theorembeweiser Isabelle/HOL. Bisimulationen eignen sich, um die Konformität zwischen Systemen zu beschreiben und zu verifizieren. Zweitens stellen wir eine in Isabelle/HOL mechanisierte zeitbehaftete Erweiterung der Hennessy-Milner Logik (HML) zur Verfügung, mit der zeitbehaftete Systemanforderungen beschrieben und maschinell verifiziert werden können. Da wir sowohl konformitäts- als auch eigenschaftsbasierte Verifikation unterstützen, ermöglichen wir insbesondere, das Verifikationsproblem in Teilprobleme zu zerlegen, was die Verifikation erheblich vereinfacht. Zusätzlich stellt der Theorembeweiser sicher, dass alle Beweise lückenlos und garantiert korrekt sind. Unser dritter wesentlicher Beitrag ist die Unterstützung des Verifikationsprozesses durch automatische Verifikationswerkzeuge zur Validierung. Dazu integrieren wir Transformationswerkzeuge in unseren Theorembeweiser-Ansatz, mit denen (endliche) Timed CSP Spezifikationen automatisch in einen diskreten Dialekt von CSP und in zeitbehaftete Automaten transformiert werden können. Somit können der Verfeinerungschecker FDR2 und der Model Checker UPPAAL eingesetzt werden, um Systeminstanzen automatisch zu simulieren und zu verifizieren. Dies hat den Vorteil, Designfehler schon vor dem relativ aufwändigen interaktiven Theorembeweisen aufdecken zu können. Wir wenden unser Rahmenwerk auf die Fallstudie des Schedulers eines Echtzeitbetriebssystems an. Damit demonstrieren wir einerseits den Gewinn durch unsere Transformationen von Timed CSP in automatisch analysierbare Sprachen und andererseits die Effektivität unseres Theorembeweiser-Ansatzes zur maschinellen Verifikation parametrisierter Echtzeitsysteme.

2 Integriertes Validierungs- und Verifikations-Rahmenwerk

In diesem Abschnitt präsentieren wir unser Rahmenwerk zur maschinellen Verifikation von parametrisierten Echtzeitsystemen. In [GG10b] haben wir eine frühere Version davon vorgestellt. Wir betrachten insbesondere parametrisierte Echtzeitsysteme, in denen ein ausgezeichneter Kontrollprozess C_n ein Netzwerk von beliebig vielen Netzwerkprozessen $P_{i,n}$ steuert. Das Verhalten sowohl des Kontrollprozesses als auch der einzelnen Netzwerkprozesse können dabei von der Gesamtgröße des Netzwerks n abhängen. Die generelle von uns betrachtete Struktur von parametrisierten Echtzeitsystemen ist:

$$C_n \otimes_0 (P_{1,n} \otimes P_{2,n} \otimes \cdots \otimes P_{n,n})$$

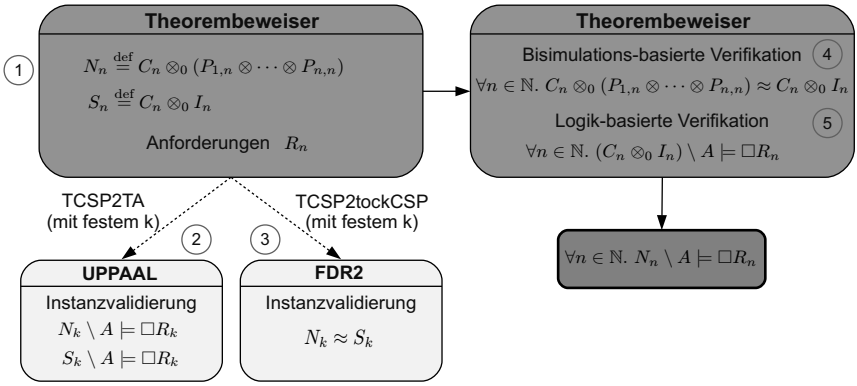

Abbildung 1: Übersicht unsres integrierten Validierungs- und Verifikations-Rahmenwerks

In diesem Kontext ist $\otimes_0$ ein Kompositionsoperator zum Verbinden des Kontrollprozesses mit dem Netzwerk. Dieser Operator beschreibt die Schnittstelle, über die der Kontrollprozess die Netzwerkprozesse steuern kann. Die Netzwerkprozesse werden über einen Kompositionsoperator $\otimes$ verbunden. Mittels dieses Operators wird zum Beispiel der Nachrichtenaustausch zwischen einzelnen Netzwerkprozessen beschrieben.

Unserer Rahmenwerk zielt darauf ab, den Verifikationsprozess für derartige Systeme mit unterschiedlichen Verifikationstechniken und -werkzeugen zu unterstützen. Der wesentliche Ablauf besteht aus einer Modellierungsphase, einer Validierungsphase und schließlich einer (umfassenden) Verifikationsphase. Für die Modellierung von parametrisierten Echtzeitsystemen setzen wir den Prozesskalkül Timed CSP [Sch99] innerhalb des Theorembeweisers Isabelle/HOL [NPW02] ein. Zur Validierung setzen wir die automatischen Verifikationswerkzeuge UPPAAL [BY04] und FDR2 [GRA05] ein. Schließlich führen wir die umfassende Verifikation in Isabelle/HOL durch.

Unser Validierungs- und Verifikationsrahmenwerk ist in Abbildung 1 dargestellt. Die aufeinanderfolgenden Schritte sind durch die Nummerierung gekennzeichnet.

① Ein Entwickler erstellt zunächst eine formale Beschreibung eines konkreten parametrisierten Echtzeitsystems N_n mit einem Netzwerk bestehend aus beliebig vielen Netzwerkprozessen. Weiterhin legt er Anforderungen R_n fest, die alle Instanzen des parametrisierten Systems erfüllen sollen. Um die Verifikation der Anforderungen zu erleichtern bietet unser Rahmenwerk weiterhin die Möglichkeit ein abstraktes parametrisiertes System S_n anzugeben, das zum konkreten Modell konform (im Sinne von *schwacher zeitbehafteter Bisimilarität*) ist. Es dient dazu die parallele Komplexität des konkreten Systems zu reduzieren. Sowohl das konkrete als auch das abstrakte System werden in Timed CSP beschrieben und in unserer Isabelle/HOL Formalisierung mechanisiert. Zur uniformen Beschreibung der Anforderungen bieten wir eine zeitbehaftete Erweiterung von HML an, die wir ebenfalls in Isabelle/HOL mechanisiert haben. Das übergeordnete Verifikationsziel ist, zu zeigen, dass das konkrete parametrisierte Echtzeitsystem mit gegebenenfalls versteckten Ereignissen $N_n \setminus A$ für alle möglichen Netzwerkgrößen die (lokalen) Anforderungen R_n in allen jeweils erreichbaren Zuständen erfüllt ($\Box R_n$).

② Im ersten Validierungsschritt nutzt der Entwickler unsere automatische Transformation *TCSP2TA* von endlichen Timed CSP Prozessen in zeitbehaftete Automaten, um Instanzen des Systems ($N_k \setminus A$ and $S_k \setminus A$) für eine festgelegte Netzwerkgröße k zu übersetzen. Dadurch können diese simuliert und bezüglich der gegebenen Anforderungen unter Verwendung des UPPAAL Model Checkers überprüft werden.

③ Der zweite Validierungsschritt dient der Überprüfung semantischer Konformität zwischen Systeminstanzen N_k und S_k für eine festgelegte Netzwerkgröße k. Dazu transformiert der Entwickler entsprechende Instanzen in eine diskret zeitbehaftete Variante von CSP (tock CSP) mit unserer automatischen Transformation *TCSP2tockCSP*. Unter anschließender Verwendung des Verfeinerungschecker FDR2, kann semantische Konformität bezüglich des timed traces Modells automatisch nachgewiesen werden.

Wenn Checks in einem der beiden Validierungsschritte nicht erfolgreich sind, können generierte Gegenbeispiele von UPPAAL oder FDR2 genutzt werden, um die ursprünglichen parametrisierten Systeme in Timed CSP entsprechend zu korrigieren. Wenn die Checks hingegen erfolgreich sind, besteht eine gute Evidenz dafür, dass das konkrete parametrisierte System sich auch *für alle* Instanzen korrekt verhält. Der formale Nachweis dafür wird in der nachfolgenden umfassenden Verifikationsphase erbracht.

④ Im ersten Verifikationsschritt wird die semantische Konformität *aller* Systeminstanzen N_n und S_n nachgewiesen. Dies geschieht unter Verwendung der Beweistechnik von (schwachen zeitbehafteten) Bisimulationen. Dafür gibt der Entwickler eine entsprechende Bisimulationsrelation an, formalisiert diese in Isabelle/HOL und zeigt mit maschineller Unterstützung, dass die üblichen Beweisverpflichtungen für Bisimulationen erfüllt sind.

⑤ Die zweite Verifikationsschritt besteht darin, maschinell nachzuweisen, dass das abstrakte System $S_n \setminus A$ für alle Instanzen die gegebenen (lokalen) Anforderungen R_n in jeweils allen erreichbaren Zuständen erfüllt. Für diesen Nachweis, gibt der Entwickler, ähnlich wie im Fall der Bisimulationen, eine Invarianzmenge an, die den Initialzustand des Systems beinhaltet. Anschließend wird gezeigt, dass jeder Zustand des Systems die (lokalen) Anforderungen R_n erfüllt. Weiterhin muss nachgewiesen werden, dass jeder Prozess der Invarianzmenge über Schritte der operationalen Semantik von Timed CSP wieder nur Prozesse der Invarianzmenge erreichen kann.

Wir haben maschinell nachgewiesen, dass Formeln unserer zeitbehafteten HML von Bisimulationen bewahrt werden. Dadurch folgt mit den beiden Verifikationsschritten, dass das konkrete parametrisierte Echtzeitsystem $N_n \setminus A$ die Anforderungen in allen erreichbaren Zuständen erfüllt, womit schließlich das übergeordnete Verifikationsziel gezeigt ist.

Die Vorteile unseres Rahmenwerks sind, dass konformitäts- und eigenschaftsbasierte Korrektheitsaussagen maschinell im Theorembeweiser nachgewiesen werden können. Dadurch wird sichergestellt, dass möglicherweise kritische Randfälle bei der Verifikation der betrachteten parametrisierten Echtzeitsysteme nicht übersehen werden können. Weiterhin können Teilbeweise durch die Benutzung von Taktiken in Isabelle automatisiert werden. Da das maschinelle Beweisen in einem interaktiven Theorembeweiser jedoch trotzdem relativ zeitaufwendig ist, setzen wir automatische Verifikationwerkzeuge zur Validierung ein. Dies ermöglicht das Aufdecken und Beheben von möglichen Design-Fehlern *vor* der nachfolgenden umfassenden Verifikationsphase im Theorembeweiser.

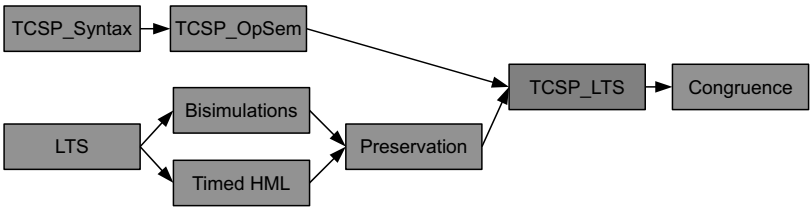

Abbildung 2: Abhängigkeiten zwischen unseren Isabelle-Theorien

3 Mechanisierung in Isabelle/HOL

Im folgenden geben wir einen Überblick unserer Formalisierungen von Timed CSP, einer zeitbehafteten HML und Bisimulationen im Theorembeweiser Isabelle/HOL. Auf dieser Grundlage ist es möglich, (parametrisierte) Echtzeitsysteme in einer mechanisierten Umgebung zu beschreiben sowie Eigenschaften und die semantische Konformität kompositional mit maschineller Unterstützung nachzuweisen. Teile unserer Formalisierungen haben wir in [GG10a] vorgestellt.

In Abbildung 2 zeigen wir die Abhängigkeiten zwischen unseren Isabelle-Theorien. Unabhängig voneinander haben wir zum einen Timed CSP und zum anderen die zeitbehaftete HML sowie Bisimulationen formalisiert und jeweils grundlegende Eigenschaften nachgewiesen. Die Theorie *LTS* definiert den grundlegenden Begriff der zeitbehafteten beschrifteten Transitionssysteme, die im wesentlichen einen Zustandsraum sowie diskrete und zeitbehaftete Transitionen zwischen Zuständen umfassen. Auf dieser Ebene haben wir gezeigt, dass Formeln in unserer zeitbehafteten HML von Bisimulationen bewahrt werden. Durch den anschließenden Nachweis, dass (die operationale Semantik von) Timed CSP als zeitbehaftetes beschriftetes Transitionssystem interpretiert werden kann, werden alle Theorien und Eigenschaften auf den Kontext von Timed CSP übertragen. Schließlich haben wir gezeigt, dass die von uns betrachteten Ausprägungen von Bisimulationen Kongruenzrelationen sind, was die kompositionale Verifikation von Timed CSP Prozessen ermöglicht.

Für den Aufbau der Isabelle/HOL Theorien haben wir das Konzept der *Locales* verwendet. Dies hat den wesentlichen Vorteil, dass die Teiltheorien hochgradig wiederverwendbar sind. Zum Beispiel wäre es möglich unsere Formalisierungen von Bisimulationen und der zeitbehafteten HML für andere Prozesssprachen neben Timed CSP wiederzuverwenden.

3.1 Timed CSP

Timed CSP ist eine Erweiterung des Prozesskalküls CSP [Sch99]. Zur Beschreibung von Echtzeitsystemen wurden zeitbehaftete Operatoren definiert und mit einer kontinuierlichen zeitbehafteten Semantik versehen. Timed CSP eignet sich, um möglicherweise unendliche Systeme kompositional zu beschreiben und zu verifizieren. Der Fokus liegt auf der Interaktion von nebenläufigen Prozessen, die über Nachrichten miteinander kommunizieren.

```
inductive_set evstep :: (('v,'a)Process , 'a eventplus)lts
  ...
| Hiding_step1: ⟦ (P , ev a , P') ∈ evstep ; a∈A ; a∈obs ⟧
                ⟹ (P\A , τ_a , P'\A) ∈ evstep
| Hiding_step2: ⟦ (P , ev a , P') ∈ evstep ; a∈A ; a∉obs ⟧
                ⟹ (P\A , τ , P'\A) ∈ evstep
| Hiding_step3: ⟦ (P , ev a , P') ∈ evstep ; a∉A ⟧
                ⟹ (P\A , ev a , P'\A) ∈ evstep
| Hiding_step4: ⟦ (P , e , P') ∈ evstep ; e=√ ∨ internal e ⟧
                ⟹ (P\A , e , P'\A) ∈ evstep
  ...

inductive_set tstep :: (('v,'a)Process , 'a eventplus)lts
  ...
| Hiding_step1: ⟦ (P , time t , P') ∈ tstep ;
                 ∀ a∈A. ¬(∃ Q. (P , ev a , Q) ∈ evstep) ⟧
                ⟹ (P\A , time t , P'\A) ∈ tstep
  ...
```

Abbildung 3: Mechanisierung der Operationalen Semantik von Timed CSP (Auszug)

Syntax Wir repräsentieren die abstrakte Syntax von Timed CSP mit Hilfe eines Isabelle-Datentyps. Weiterhin nutzen wir die Möglichkeit, konkrete Syntax für Prozessterme anzugeben. Damit ist es nicht nur möglich Prozesse formal zu erfassen, sondern diese auch annähernd wie in der Timed CSP Literatur innerhalb von Isabelle darzustellen.

Semantik In der ursprünglichen operationalen Semantik von Timed CSP haben lediglich interne τ Ereignisse, die für umgebende Prozesse unsichtbar und damit nicht beeinflussbar sind, Priorität über zeitbehafteten Schritten. Dies hat zur Konsequenz, dass sichtbare Ereignisse bis zum nächsten möglichen internen Schritt zeitlich verzögert werden können. Um geschlossene Systeme zu modellieren wird der *Hiding* Operator verwendet, was bedeutet dass alle versteckten Ereignisse nun nicht mehr von einer Umgebung beeinflusst und demnach auch nicht mehr zeitlich verzögert werden können. Konsequenterweise sind versteckte Ereignisse nach außen nicht mehr sichtbar und werden in der Semantik in τ Ereignisse transformiert. Das Problem mit dieser Behandlung ist jedoch, dass das interne Verhalten von Prozessen nicht analysiert werden kann, da die operationale Semantik lediglich ununterscheidbare τ Ereignisse nach außen propagiert.

Um auch das interne Verhalten analysierbar zu machen, haben wir die Semantik leicht angepasst, indem *Hiding* zwar weiterhin interne Ereignisse nicht-verzögerbar macht, aber diese zum Teil über die operationale Semantik unterscheidbar gemacht werden. Um dies zu erreichen parametrisieren wir die operationale Semantik mit einer globale Menge obs von sichtbaren Ereignissen, die sichtbar bleiben selbst nachdem sie versteckt wurden. Der Vorteil ist, dass das Transitionssystem der ursprünglichen Semantik im wesentlichen erhalten bleibt. Lediglich originale τ Kanten werden gegebenenfalls durch τ_a ersetzt. Im Falle, dass $obs = \emptyset$ sind beide Transitionssysteme sogar identisch. Diese Art der konservativen Erweiterung, erlaubt es die komfortablen kompositionalen Eigenschaften von Timed CSP zu erhalten.

Zur Formalisierung der angepassten operationalen Semantik in Isabelle/HOL nutzen wir induktive Mengendefinitionen für diskrete Ereignisschritte und kontinuierliche zeitbehaftete Schritte. Wir zeigen exemplarisch die semantischen Regeln für *Hiding* in Abbildung 3. Die vollständige operationale Semantik ist die Vereinigung von Ereignisschritten und zeitbehafteten Schritten. Sie beschreibt ein zeitbehaftetes beschriftetes Transitionssystem, auf dem wir grundlegende Eigenschaften von Timed CSP maschinell nachgewiesen haben.

3.2 Zeitbehaftete Hennessy-Milner Logik

Mit Hennessy-Milner Logik [HM80] können Eigenschaften von Zuständen (auch Prozesse genannt) in beschrifteten Transitionssystemen beschrieben werden. Um über Echtzeitsysteme argumentieren zu können, haben wir eine zeitbehaftete HML basierend auf zeitbehafteten beschrifteten Transitionssystemen entwickelt und in Isabelle/HOL mechanisiert.

Im folgenden sei p ein Prädikat auf zeitbehafteten Ereignissen (t, a). Weiterhin schreiben wir $(t, a)_p$, wenn (t, a) das Prädikat p erfüllt. Wir definieren, dass ein Prozess P mit dem zeitbehafteten Ereignis (t, a) den Prozess P'' erreichen kann, wenn P in t Zeiteinheiten zu einem P' übergehen kann und P' mit Ereignis a zu P'' übergehen kann.

Definition[Syntax der zeitbehafteten Hennessy-Milner Logik] Die Formeln unserer zeitbehafteten HML sind wie folgt rekursiv definiert: $\phi := tt \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle\langle p \rangle\rangle\phi \mid [[p]]\phi$

Die Semantik der letzten beiden Operatoren ist am interessantesten: $P \models \langle\langle p \rangle\rangle\phi$ ist erfüllt, wenn ein zeitbehaftetes Ereignis $(t, a)_p$ existiert, so dass P mit $(t, a)_p$ einen Prozess P'' erreichen kann und P'' die Formel ϕ erfüllt. $P \models [[p]]\phi$ ist erfüllt, wenn *alle* Prozesse P'', die von P mit einem zeitbehafteten Ereignis $(t, a)_p$ erreicht werden können, ϕ erfüllen. Über abgeleitete Operatoren können Formeln wie z.B. $[[a]]_{\leq t} ff$ beschrieben werden, die ausdrückt, dass a nicht innerhalb der nächsten t Zeiteinheiten kommuniziert werden kann.

3.3 Bisimulationen

Bisimulationen eignen sich zur Beschreibung und Verifikation semantischer Konformität zwischen Prozessen. Im wesentlichen sind zwei Prozesse bisimilar, wenn sie ihre beschrifteten Transitionen gegenseitig nachahmen können und dabei jeweils wieder bisimilare Prozesse erreichen. Wir haben Bisimulationen abstrakt definiert und erhalten verschiedene Ausprägungen (stark, schwach, schwach zeitbehaftet) durch jeweilige Instantiierung. Eigenschaften auf der abstrakten Ebene, wie Reflexivität, Symmetrie und Transitivität, werden dadurch vererbt und müssen nicht jeweils erneut bewiesen werden. Damit ist unsere Bisimulationstheorie sehr modular aufgebaut und dadurch äußerst wiederverwendbar.

Unsere Mechanisierungen erlauben es parametrisierte Echtzeitsysteme interaktiv und kompositional im Theorembeweiser zu verifizieren. Wenn Design-Fehler jedoch erst spät im Beweis gefunden werden, muss dieser u.U. komplett überarbeitet werden. Um diese Gefahr zu reduzieren, integrieren wir automatische Beweiswerkzeuge in unser Rahmenwerk.

4 Automatische Validierung

Um automatische Simulations- und Verifikationswerkzeuge wie UPPAAL [BY04] und FDR2 [GRA05] nutzbar zu machen, haben wir Transformationen von Timed CSP in zeitbehaftete Automaten und nach tock CSP entwickelt. Diese Transformationsansätze basieren auf den Arbeiten aus [DHQ+08] und [Oua01]. Da diese Ansätze jedoch nicht alle Timed CSP Operatoren unterstützen und zum Teil die Semantik von Timed CSP inkorrekt behandeln, haben wir diese korrigiert, adaptiert, erweitert und schließlich implementiert. Weiterhin haben wir gezeigt, wie typische Eigenschaften in unserer zeitbehafteten HML in UPPAAL codiert werden können.

Wir verwenden den Echtzeit-Model Checker UPPAAL, um Eigenschaften automatisch auf Instanzen der gegebenen parametrisierten Echtzeitsysteme nachzuweisen. Um die Konformität zwischen Instanzen des konkreten und abstrakten parametrisierten Systems automatisch nachzuweisen verwenden wir den FDR2 Verfeinerungschecker, der über Möglichkeiten verfügt, interne τ Ereignisse gegenüber $tock$ Ereignisse zu priorisieren, um die Semantik von Timed CSP korrekt zu berücksichtigen (interne Ereignisse können nicht verzögert werden). Damit bieten wir die Möglichkeit, beide Verifikationsschritte in unserem Rahmenwerk durch die Verwendung automatischer Verifikationswerkzeuge vorzubereiten.

5 Fallstudie: Verifikation eines Echtzeitbetriebssystem-Schedulers

Um die Anwendbarkeit unseres Rahmenwerks nachzuweisen, haben wir die Fallstudie des Echtzeitbetriebsystem-Schedulers von BOSS [MBK06] betrachtet. Dieser Scheduler kann beliebig viele Threads verwalten und ist somit ein parametrisiertes Echtzeitsystem in unserem Sinn. Wir haben eine vereinfachtes Modell davon in Timed CSP entwickelt, das einen prioritäts- und rundenbasierten präemptiven Scheduler realisiert. Weiterhin nutzen wir die Eigenschaft aus, dass maximal ein Thread zu einem bestimmten Zeitpunkt aktiv sein kann. Dies hat es uns ermöglicht, ein zusätzliches abstraktes Modell zu entwickeln, in dem das konkrete Netzwerk von Threads durch einen sequentiellen Prozess dargestellt wird und somit die parallele Komplexität deutlich reduziert ist. Die Anforderungen, die wir exemplarisch betrachtet haben sind, dass ein Thread weder zu kurz noch zu lang die Kontrolle erhält und dass der Scheduler die Prioritäten der Threads korrekt berücksichtigt.

Zunächst haben wir relativ kleine Instanzen des Systems automatisch mit UPPAAL und FDR2 validiert. Dabei sind uns bei der Entwicklung z.B. kleinere Fehler in unserem Modell bezüglich der inkorrekten Behandlung von Prioritäten aufgefallen, die wir mit Hilfe der generierten Gegenbeispiele simulieren, analysieren und entsprechend beheben konnten. In der anschließenden umfassenden Verifikationsphase in Isabelle/HOL haben wir über Konformitätsverifikation sowie Eigenschaftsverifikation schließlich gezeigt, dass das konkrete Scheduler-Modell die gegebenen Anforderungen erfüllt. Die interaktive Beweiserstellung hat ca. 35 Stunden in Anspruch genommen. Wir sind uns jedoch sicher, dass dieser Verifikationsaufwand deutlich höher ausgefallen wäre, wenn die vorherige Validierungsphase nicht gegeben gewesen wäre.

6 Fazit und Ausblick

In dieser Arbeit haben wir einen integrierten Valdierungs- und maschinellen Verifikationsansatz für parametrisierte Echtzeitsysteme entwickelt. Damit ist es möglich, unendliche Echtzeitsysteme maschinell im Theorembeweiser Isabelle/HOL zu verifizieren und somit auszuschließen dass Randfälle übersehen werden können. Dabei unterstützen wir sowohl eigenschaftsbasierte Verifikation mittels unserer zeitbehafteten HML sowie konformitätsbasierte Verifikation mittels verschiedener Ausprägungen von Bisimulationen. Die unterliegenden Isabelle-Theorien sind möglichst generisch gehalten, um unsere Beiträge z.B. auch für andere Prozesssprachen verfügbar zu machen. Weiterhin können vor der relativ aufwendigen interaktiven Verifikation im Theorembeweiser Instanzen validiert werden, um den maschinellen Verifikationsaufwand möglichst gering zu halten. Dazu stellen wir Transformationen von Timed CSP in zeitbehaftete Automaten und nach tock CSP zur Verfügung, um automatische Verifikationswerkzeuge wie z.B. UPPAAL und FDR2 nutzbar zu machen. Damit stellen wir also einen umfassenden maschinellen Verifikationsansatz sowie einen integrierten Validierungsansatz für unendliche Echtzeitsysteme bereit.

In weiterführenden Arbeiten wollen wir unseren Ansatz auf weitere Fallstudien wie z.B. Bussysteme und Multicore-Architekturen anwenden. Weiterhin wäre es interessant zu untersuchen, inwieweit sich unsere Isabelle/HOL-Mechanisierungen und unsere Transformationsansätze auf andere Prozesssprachen wie Timed CCS übertragen lassen. Um schwächere Konfomitätsbegriffe zu betrachten, könnten Bisimulationen durch z.B. Verfeinerungsbegriffe ersetzt werden und untersucht werden, welche Anteile unseres Rahmenwerks wie übernommen werden können bzw. geändert werden müssen. Schließlich wäre es interessant zu untersuchen, wie automatische Verifikationsergebnisse im Theorembeweiser wiederverwendet werden könnten, ohne die Theorembeweiser-Sicherheit zu verlieren. Es wäre z.B. möglich SMT-basierte Techniken für die Verifikation von Teilprozessen einzusetzen und automatisch gefundene Beweise zurück in den Theorembeweiser zu transferieren. Damit wäre es möglich, diese automatisch und maschinell im Theorembeweiser zu prüfen und somit, trotz der Verwendung potentiell unsicherer Implementierungen automatischer Verifikationswerkzeuge, einen lückenlosen maschinellen Beweis zu erhalten.

Literatur

[BLW05] Ahmed Bouajjani, Axel Legay und Pierre Wolper. Handling Liveness Properties in (ω-) Regular Model Checking. *Electronic Notes in Theoretical Computer Science*, 138:101–115, 2005.

[BY04] Johan Bengtsson und Wang Yi. Timed Automata: Semantics, Algorithms and Tools. In *Lectures on Concurrency and Preti Nets*, Seiten 87–124. Springer, 2004.

[DHQ+08] Jin Song Dong, Ping Hao, Shengchao Qin, Jun Sun und Wang Yi. Timed Automata Patterns. *IEEE Transactions on Software Engineering*, 34:844–859, 2008.

[GG10a] Thomas Göthel und Sabine Glesner. An Approach for Machine-Assisted Verification of Timed CSP Specifications. *Innovations in Systems and Software Engineering - A NASA Journal*, 7:181–193, 2010.

[GG10b] Thomas Göthel und Sabine Glesner. Towards the Semi-Automatic Verification of Parameterized Real-Time Systems using Network Invariants. In *Proceedings of the 8th IEEE International Conference on Software Engineering and Formal Methods*, Seiten 310–314. IEEE Computer Society, 2010.

[GL08] Olga Grinchtein und Martin Leucker. Network Invariants for Real-Time Systems. *Formal Aspects of Computing*, 20(6):619–635, 2008.

[Göt12] Thomas Göthel. *Mechanical Verification of Parameterized Real-Time Systems*. Dissertation, Technische Universität Berlin, 2012.

[GRA05] Michael Goldsmith, Bill Roscoe und Philip Armstrong. Failures-Divergence Refinement - FDR2 User Manual, 2005.

[HM80] Matthew Hennessy und Robin Milner. On Observing Nondeterminism and Concurrency. In *Proceedings of the 7th Colloquium on Automata, Languages and Programming (ICALP)*, Seiten 299–309. Springer, 1980.

[IR05] Yoshinao Isobe und Markus Roggenbach. A Generic Theorem Prover of CSP Refinement. In *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, Seiten 108–123. Springer, 2005.

[MBK06] S. Montenegro, K. Briess und H. Kayal. Dependable Software (BOSS) for the BEESAT Pico Satellite. In *DASIA 2006*, Data Systems In Aerospace - DASIA 2006, May 2006, Berlin. ESTEC, 2006.

[NPW02] Tobias Nipkow, Lawrence C. Paulson und Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, Jgg. 2283 of *LNCS*. Springer, 2002.

[Oua01] Joël Ouaknine. *Discrete Analysis of Continuous Behaviour in Real-Time Concurrent Systems*. Dissertation, Oxford University, 2001.

[Sch99] Steve Schneider. *Concurrent and Real Time Systems: The CSP Approach*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[WWB10] Kun Wei, Jim Woodcock und Alan Burns. Embedding the Timed Circus in PVS. Bericht, University of York, 2010.

Thomas Göthel wurde am 09. Februar 1984 in Radebeul geboren. Von 2002 bis 2007 studierte er Informatik an der Technischen Universität Berlin und schloss sein Studium „mit Auszeichnung“ ab. Im Anschluss arbeitete er als wissenschaftlicher Mitarbeiter im DFG-geförderten Projekt VATES (Verification and Transformation of Embedded Systems) und betreute Lehrveranstaltungen im Bachelor- und im Master-Studium der Informatik. Seine Promotion zum „Doktor der Ingenieurwissenschaften“ schloss er 2012 am Fachgebiet „Programmierung eingebetteter Systeme“ der Technischen Universität Berlin „mit Auszeichnung“ ab. Seine Forschungsinteressen liegen u.a. auf den Gebieten der formalen maschinellen Verifikation, der Echtzeitsysteme und der Kombination von automatischen und interaktiven Verifikationstechniken. Seit seiner Promotion arbeitet er als Postdoktorand am Fachgebiet „Programmierung eingebetteter Systeme“. Seit Mai 2013 forscht er, gefördert durch ein DAAD-Stipendium, am Department of Computer Science an der University of Oxford.