Spezifikation und Verifikation mobiler Realzeitsysteme

Andreas Schäfer

Department für Informatik
Carl von Ossietzky Universität Oldenburg
26111 Oldenburg
schaefer@informatik.uni-oldenburg.de

Abstract: In dieser Arbeit wird eine formale Methode, die Logik *Shape Calculus*, zur Beschreibung und Verifikation von Systemen vorgestellt, deren Beschreibung sowohl Aussagen über den zeitlichen Ablauf als auch die räumliche Konfiguration der Komponenten erfordert. Es wird gezeigt, dass die Logik im Allgemeinen unentscheidbar ist. Es werden zwei entscheidbare Teilklassen angeben. Eine Teilklasse wird durch Einschränkung auf endliche diskrete Räume gewonnen. Die zweite Teilklasse durch syntaktische Einschränkung der Formelklasse. Für die erste Teilklasse existiert ein automatisches Verifikationswerkzeug. Die Anwendung dieses Werkzeugs wird anhand eines Fallbeispiels demonstriert, das mit Methoden, die nur die zeitlichen Aspekte betrachten nicht untersucht werden kann.

1 Motivation

In vielen Bereichen des täglichen Lebens werden eingebettete Computer eingesetzt, angefangen von Waschmaschinen bis hin zu Steuerungssystemen von Flugzeugen und Autos. Mehr und mehr mechanische Systeme werden nun elektronisch gesteuert, wie zum Beispiel bei steer-by-wire oder brake-by-wire bei denen mechanische Steuerungs- und Bremssysteme durch Computersteuerungen ersetzt werden. Der Vorteil besteht darin, den Fahrer bei der Steuerung des Fahrzeuges elektronisch unterstüzen und so eine höhere Verkehrsicherheit erreichen zu können. Bei solchen Systemen muss jedoch sichergestellt sein, dass sie fehlerfrei arbeiten und keine Programmierfehler in der Software enthalten sind. Intensives Testen ist ein Ansatz, die Sicherheit des Systems zu erhöhen. Jedoch kann durch Testen nur die Anwesenheit von Fehlern aber nicht deren Abwesenheit nachgewiesen werden. Formale Methoden versuchen hingehen mittels eines mathematischen Beweises, die Sicherheit eines Systems zu garantieren. Dazu wird ein formales mathematisches Modell des Programms bzw. des Systems erstellt. Außerdem werden die gewünschten Systemeigenschaften ebenfalls mathematisch formal definiert. Mittels eines Beweises wird dann gezeigt, dass das Systemmodell die gewünschten Eigenschaften besitzt. Ein solcher Beweis kann für einige formale Methoden sogar vollautomatisch durch ein Computerprogramm [CGP00] auf Knopfdruck erbracht werden. Für andere Methoden ist diese Problem jedoch formal unentscheidbar, d.h. man kann zeigen, dass ein solcher Computerprogramm schon

prinzipiell nicht existieren kann. In diesen Fällen ist meist Interaktion mit einem Benutzer notwendig.

Diese Arbeit gehört zur Grundlagenforschung der Informatik. Es wird eine formale Methode, der *Shape Calculus*, entwickelt und untersucht mit der Systeme und Eigenschaften beschrieben werden können, die sowohl zeitliche Anforderungen – wie zum Beispiel garantierte Antwortzeiten – als auch räumliche Anforderungen – wie zum Beispiel minimale Abstände zweier Agenten – erfüllen müssen. Zu dieser Systemklasse gehören auch Steuerungssysteme für Eisenbahnen und Flugzeuge. Für die Beschreibung zeitlicher Aspekte existieren bereits eine Reihe von Methoden, wie zum Beispiel Realzeitautomaten [AD94], die jedoch räumliche Anforderungen an mobile Systeme nicht beschreiben können. Der in dieser Arbeit vorgestellte Ansatz geht von einer bewährten und gut untersuchten Methode für die Beschreibung von Realzeiteigenschaften, dem Duration Calculus [ZHR91, HZ04], aus und erweitert ihn konservativ, so dass auch räumliche Aspekte beschrieben werden können. Durch diese Erweiterung ändern sich grundlegende Eigenschaften des Formalismus.

Dieser Aufsatz gliedert sich wie folgt: Zuerst wird das zu lösende Problem anhand eines Fallbeispiels aus der Praxis illustriert. Nachfolgend wird der Shape Calculus formal eingeführt und dessen Anwendung illustriert. Danach werden die technischen Resultate zum Shape Calculus vorgestellt. Es wird gezeigt, dass im Allgemeinen die Frage, ob ein System eine gewünschte Eigenschaft besitzt für den Shape Calculus unentscheidbar ist. Nachfolgend werden zwei Teilklassen vorgestellt, für diese Frage jedoch automatische entschieden werden kann. Für eine dieser Teilklassen wird gezeigt, wie dieses theoretische Resultat zu einem praktische einsetzbarem Verifikationswerkzeug führt. Beispielhaft wird die Anwendung an einer kleinen Fallstudie demonstriert.

2 Sicherung eingleisiger Streckenstücke

Die folgende Fallstudie stammt aus dem UniForm Projekt [KBPOB99] und wurde von dem Industriepartner ElPro bereitgestellt. Das Szenario der Fallstudie ist in Abbildung 1 dargestellt: Aufgrund von Wartungsarbeiten ist bei einer normalerweise zweigleisigen Straßenbahnstrecke ein Abschnitt auf einer Seite gesperrt, so dass der gesamte Straßenbahnverkehr für beide Fahrtrichtungen über ein gemeinsam genutztes Streckenstück geleitet werden muss. Durch Signalanlagen muss sichergestellt werden, dass es innerhalb des Streckenstücks zu keinen Kollisionen kommt. Hinzu kommen noch folgende Anforderungen:

1. Es dürfen mehrere Straßenbahnen hintereinander in gleicher Richtung das Steckenstück durchfahren, ohne dass das entsprechende Signal zwischendurch rot ist.

2. Eine Straßenbahn darf die Richtung innerhalb des Streckenstücks ändern, sofern sich keine Straßenbahn dahinter befindet. Dies wird durch den Fahrer durch fahren auf Sicht festgestellt.

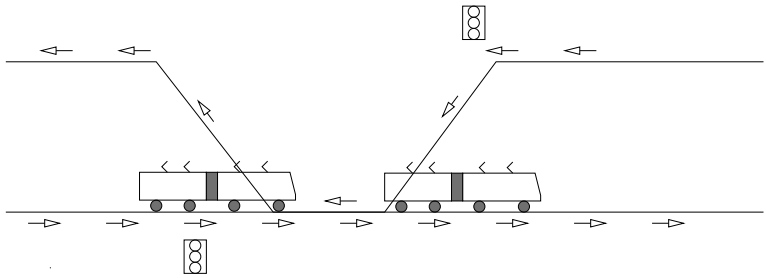

Abbildung 1: Fallstudie: Sicherung eingleisiger Streckenabschnitte

Durch diese zusätzlichen Anforderungen ist klar, dass bereits für die Formulierung der Sicherheitseigenschaft die räumliche Konfiguration der Straßenbahnen eine Rolle spielt. So reicht es aufgrund der ersten Anforderung nicht aus, zu verlangen, dass sich jeweils nur eine Straßenbahn im kritischen, gemeinsam genutzten Abschnitt befindet. Wegen der Anforderung zwei reicht es ebenfalls nicht aus, zu verlangen, dass falls sich zwei Straßenbahnen in diesem Abschnitt befinden, diese auch unterschiedliche Fahrtrichtungen haben.

Im Rahmen des UniForm-Projektes war es nicht möglich die allgemeine Sicherheitseigenschaft der Kollisionsfreiheit zu formulieren. Ebenfalls nicht formalisiert werden konnte das sinnvolle Verhalten des Fahrers wie in der zweiten Anforderungen angegeben. Von diesen Eigenschaften musste bei der Analyse abstrahiert werden [Die99].

3 Shape Calculus

In diesem Abschnitt wird die Logik *Shape Calculus* in einer vereinfachten Fassung formal eingeführt. Es werden Syntax und Semantik der Formeln sowie einige gebräuchliche Abkürzungen definiert.

Das mobile Realzeitsystem wird dabei durch eine Menge von Observablen beschreiben, die das beobachtbare Systemverhalten modellieren. Für jeden Punkt im Raum und Moment in der Zeit kann eine solche Observable entweder wahr oder falsch sein. Je nach Anwendungsgebiet können Raum und Zeit entweder beide als diskret oder als kontinuierlich angenommen werden.

Eine Observable Train kann beispielsweise beschreiben, an welchen Punkten im Raum sich zu welchen Zeitpunkten ein Zug befindet. Die Semantik einer Observablen wird durch eine Funktion $\mathcal{I}$ beschrieben, die für jeden Raum- und Zeitpunkt den Wahrheitswert der Observablen liefert.

3.1 Zustandsausdrücke

Observablen können mit den bekannten aussagenlogischen Verknüpfungen zu komplexeren *Zustandsausdrücken* verknüpft werden. Auf diese Weise kann beschrieben werden, dass sich an einem Punkt im Raum und Moment in der Zeit Zug A befindet, aber nicht außerdem noch Zug B. Formal ist die Menge der Zustandsausdrücke durch folgende Grammatik in Backus-Naur-Form gegeben:

$$\pi ::= X \mid \neg\pi_1 \mid \pi_1 \wedge \pi_2,$$

wobei X eine Observable ist. Die übrigen aussagenlogischen Verknüpfungen wie $\vee, \rightarrow$, ... können wie üblich als Abkürzungen definiert werden. Die Semantik eines Zustandsausdrucks ist dann wie in der Aussagenlogik definiert. Ein Ausdruck $\neg\pi_1$ ist also für einen Zeit- und Raumpunkt wahr genau dann wenn π_1 für diesen Punkt falsch ist und der Ausdruck $\pi_1 \wedge \pi_2$ ist wahr genau dann wenn sowohl π_1 als auch π_2 wahr sind.

3.2 Formeln

Formeln erlauben nun Aussagen nicht über einzelne Punkte im Raum und in der Zeit, sondern über Zeit- und Raumbereiche, genauer über konvexe Polyeder. Somit kann zum Beispiel formalisiert werden, dass im betrachteten Raum und Zeitbereich niemals zwei Züge zusammen den gleichen Punkt belegen, also niemals kollidieren.

Zu diesem Zweck wird zum einen der Everywhere-Operator $\lceil\pi\rceil$ eingeführt, der beschreibt, dass überall im betrachteten Zeit- und Raumbereich (Polyeder) der Zustandsausdruck π für alle Punkte wahr ist. Ein weiterer Operator, genannt *Chop*-Operator ermöglicht es, das aktuelle Beobachtungspolyder durch Schnitt mit einer Hyperebene in zwei Teile zu teilen. Dieser Operator ist, wie in Abbildung 2 illustriert, mit der Richtung in die der Polyeder geteilt werden soll indiziert. Zudem ist es möglich, Aussagen über die Größe des aktuell betrachteten Polyeders zu machen. Dazu kann die Größe $\ell_{\vec{e}_i}$ des Polyeders in Richtung i mit einer Konstanten verglichen werden. Aus Gründen der Einfachheit der Darstellung werden nachfolgend statt allgemeiner konvexer Polyeder Hyperwürfel betrachtet und Chop-Operatoren nur entlang einer Koordinatenachse zugelassen.

Formal ist die Syntax der Shape Calculus Formeln durch folgende Grammatik in Backup-Naur-Form gegeben:

$$F ::= \lceil\pi\rceil \mid \neg F_1 \mid F_1 \wedge F_2 \mid F_1\langle\vec{e}_i\rangle F_2 \mid F_1\langle\vec{e}_t\rangle F_2 \mid \ell_{\vec{e}_i} \sim d \mid \ell_{\vec{e}_t} \sim d,$$

wobei π ein Zustandsausdruck ist und $\vec{e}_i$ einen Einheitsvektor in Richtung der i-ten Raumdimension und $\vec{e}_t$ den Einheitsvektor in Richtung der Zeitdimension angibt. Eine Formel

$$F\langle\vec{e}_i\rangle G$$

ist für ein Beobachtungspolyeder im Raum und in der Zeit wahr, genau dann wenn eine Hyperebene mit Normalenvektor $\vec{e}_i$ durch den Polyeder gelegt werden kann, so dass auf

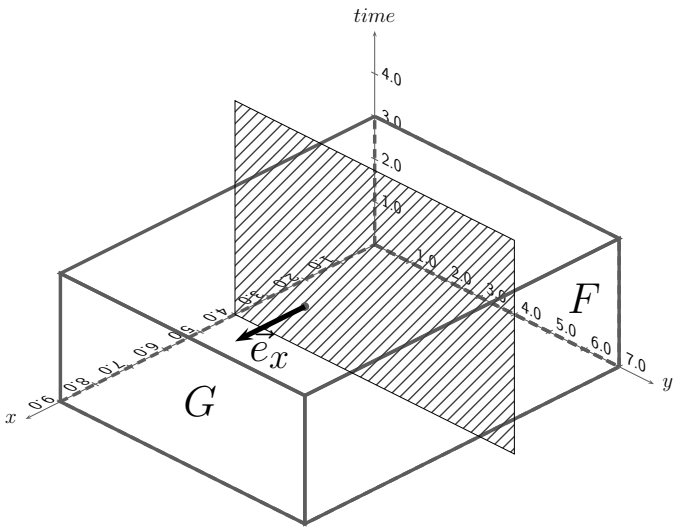

Abbildung 2: Der Chop-Operator ($F\langle\vec{e}_x\rangle G$)

einem entstehenden Teilpolyeder F gilt und auf dem anderen G gilt. Entsprechend analog ist der zeitliche Chop-Operator definiert. Eine Formel $F\langle\vec{e}_t\rangle G$ ist wahr genau dann wenn das Beobachtungspolyeder in zeitlicher Richtung geteilt werden kann, so dass zuerst F gilt und danach G gilt. Zur besseren Übersichtlichkeit werden auch $\vec{e}_x, \vec{e}_y, \ldots$ zur Bezeichnung der Raumrichtungen verwendet. Eine Formel heißt erfüllbar, genau dann wenn es eine Interpretation und einen konvexen Polyeder gibt für die Formeln wahr ist. Sie heißt allgmeingültig, genau dann wenn sie für alle Interpretationen und konvexen Polyeder wahr ist.

Mit Hilfe des Chop-Operators können aus der Modallogik bekannte Operatoren als Abkürzung definiert werden.

Der irgendwann Operator definiert durch

$$\Diamond_{\vec{e}_t} F := true\langle\vec{e}_t\rangle F\langle\vec{e}_t true$$

beschreibt, dass die Formel F auf einem temporalen Teilintervall irgendwo gilt. Analog kann der irgendwo Operator für eine Koordinatenachse definiert werden.

Beispiel Für die Modellierung des in der Einführung beschriebenen Fallbeispiels können zwei Observablen Train_1 und Train_2 verwendet werden. Die Sicherheitseigenschaft kann dann beschrieben werden durch den Ausdruck

$$\neg\Diamond_{\vec{e}_x}\Diamond_{\vec{e}_t}\lceil Train1 \wedge Train2\rceil$$

der angibt, dass für alle Punkte im Raum und Momente in der Zeit Train_1 oder Train_2 diesen Raumpunkt nicht belegt.

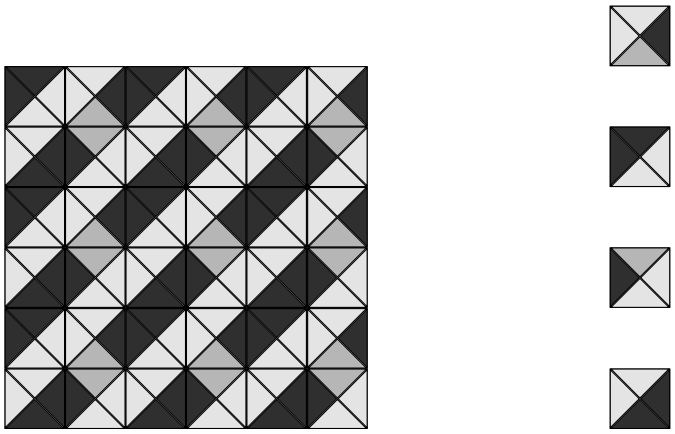

Abbildung 3: Das Domino-Problem

4 Technische Resultate

Grundlegende Fragen zu einem solchen Formalismus betreffen Entscheidbarkeit und Axiomatisierbarkeit. Bei Entscheidbarkeit geht es darum, ob es prinzipiell ein Computerprogramm geben kann, das für eine gegebene Formel entscheidet, ob diese Formel erfüllbar ist oder nicht. Die Frage der Axiomatisierbarkeit untersucht, ob es möglich ist, eine Menge von Axiomen und Regeln anzugeben, so dass jede gültige Formel aus den Axiomen durch Anwendung der Regeln abgeleitet werden kann.

4.1 Unentscheidbarkeit

Entscheidbarkeit einer Logik ist Voraussetzung für die automatische Verifikation auf Knopfdruck, während Axiomatisierbarkeit für die Entwicklung halbautomatischer Verfahren von Bedeutung ist. In [Sch05, Sch07] wird gezeigt, dass der Shape Calculus weder entscheidbar noch axiomatisierbar ist. Dazu wird gezeigt, dass es “einfacher” ist, das bekanntermaßen unentscheidbare Domino-Problem zu entscheiden. Bei diesem Problem geht man, wie in Abbildung 3 dargestellt, von einer Menge von Fliesentypen aus und untersucht, ob es möglich ist, eine Fläche zu parkettieren, so dass zwei benachbarte Fliesen jeweils passende Farben an den Grenzflächen haben.

4.2 Entscheidbare Teilklassen

Trotz dieser fundamentalen Unentscheidbarkeit ergibt sich die praktische Anwendbarkeit der formalen Methode durch entscheidbare Teilklassen. In [Sch07] werden zwei solcher Klassen identifiziert.

Endlicher diskreter Raum, unendliche Zeit Auf der einen Seite ist es möglich Entscheidbarkeit zu erreichen, indem man von diskreter Zeit und endlichem diskreten Raum ausgeht. Diese Klasse liegt an der Grenzlinie zur Entscheidbarkeit, da die Logik für diskrete und unendliche Zeit und diskreten und unendlichen Raum bereits unentscheidbar ist. Für diese Teilklasse wurde auch das Werkzeug MoDiShCa [QS06] entwickelt, mit dem Erfüllbarkeit und Allgemeingültigkeit automatisch auf Knopfdruck überprüft werden können. Diese Einschränkung auf endlichen Raum ist in den meisten Fällen nicht gravierend, da auch das reale System in einem endlichen Raum operiert. Gleichzeitig erlaubt die weiterhin unendliche Zeit auch die Beschreibung von Sicherheitseigenschaften. Würde für die Zeit nur ein endlicher Bereich betrachtet, könnte nicht beschrieben werden, dass der gefährliche unerwünschte Zustand nie eintreten kann, sondern nur, dass er in einem bestimmten Zeitraum nicht auftritt. Unter Umständen war das Zeitintervall aber zu kurz gewählt, als dass das fehlerhafte Verhalten auftreten könnte.

Formeln ohne Alternierung des Chop Operators Die zweite entscheidbare Teilklasse kann man dadurch erhalten, dass man beliebiges Alternieren des Chop-Operators verbietet. Für das Entscheidungsverfahren können dann Ideen aus aktuellen Forschungen zur Kombination von Modallogiken benutzt werden [GKWZ03].

5 Anwendungen

Um den Shape Calculus für die praktische Verifikation einsetzen zu können, wurde der prototypische Model-Checker MoDiShCa [QS06] entwickelt, mit dessen Hilfe entschieden werden kann, ob eine Formel des Shape Calculus erfüllbar oder allgemeingültig ist. Dazu wird die im technischen Teil als entscheidbar erkannte Teilklasse angenommen, die von einem endlichen Raum ausgeht.

Für die Verifikation der Zugangssteuerung [Sch06] für das eingleisige Streckenstück werden unter anderem folgende Eigenschaften des der Züge modelliert:

1. Wenn sich ein Zug an einer Stelle befindet, dann nach einem Zeitschritt um ν Felder entfernt in Abhängigkeit von der aktuellen Fahrtrichtung. Für ν wird die Maximalgeschwindigkeit eingesetzt.
2. Ein Zug fährt nur in den kritischen Abschnitt ein, wenn das Signal grün zeigt.
3. Ein Zug ändert nur dann die Richtung, wenn sich kein anderer Zug im kritischen Abschnitt dahinter befindet.

Zusätzlich werden folgende Anforderungen an die Signalsteuerung beschrieben:

1. Beide Signale zeigen nicht gleichzeitig grün.
2. Ein Signal wechselt nur dann, wenn der kritische Abschnitt leer ist.

Beispielhaft sei nur die Beschreibung der letzten Eigenschaft in der Syntax des Werkzeugs dargestellt.

$$\begin{aligned}\neg\Diamond_{\vec{e}_t}((((\lceil SIG1\rceil\vec{e}_t\lceil\neg SIG1\rceil) \wedge \neg\lceil\neg Train1 \wedge \neg Train2\rceil) \vee \\ ((\lceil SIG2\rceil\langle\vec{e}_t\rangle\lceil\neg SIG2\rceil) \wedge \neg\lceil\neg Train1 \wedge \neg Train2\rceil) \vee \\ ((\lceil\neg SIG1\rceil\langle\vec{e}_t\rangle\lceil SIG1\rceil) \wedge \neg\lceil\neg Train1 \wedge \neg Train2\rceil) \vee \\ ((\lceil\neg SIG2\rceil\langle\vec{e}_t\rangle\lceil SIG2\rceil) \wedge \neg\lceil\neg Train1 \wedge \neg Train2\rceil))\end{aligned}$$

Diese Formel beschreibt in der ersten Zeile, dass es nicht möglich ist, dass irgendwann in der Zeit $\Diamond_{\vec{e}_t}$ zuerst $SIG1$ und dann nicht $SIG1$ gilt und nicht die ganze Zeit über das Streckenstück nicht beide Züge enthält. Die anderen Zeilen beschreiben die entsprechenden anderen Fälle.

Mit dieser formalen Spezifikation kann dann automatisch auf Knopfdruck gezeigt werden, dass daraus die Sicherheitseigenschaft

$$\neg\Diamond_{\vec{e}_x}\Diamond_{\vec{e}_t}\lceil Train1 \wedge Train2\rceil$$

folgt, die beschreibt, dass nirgendwo im Raum und nirgendwann in der Zeit ein Streckenstück gleichzeitig von zwei Zügen belegt ist. Die beiden Eigenschaften betreffen nur den Controller der Signalanlage und können dann mit Methoden für Realsysteme, wie PLC-Automaten [Die99], beschrieben und zur automatischen Code-Generierung genutzt werden.

Damit ist gezeigt, dass prinzipiell eine Anwendung auf sinnvolle Fallbeispiele möglich ist. Das Verfahren hat jedoch nicht-elementaren Aufwand in der Größe der Formeln und polynomiellen Aufwand in der Größe des betrachteten Raumes. Daraus ergeben sich eine Reihe von noch offenen Forschungsfragen, wie beispielsweise die Möglichkeit des Einsatzes von Abstraktionstechniken, um die Überprüfung komplexerer Systeme zu ermöglichen.

6 Conclusion

In diesem Aufsatz wird eine formale Methode zur Beschreibung der Verhaltens mobiler Realzeitsystem und von Anforderungen an diese Systeme vorgestellt. Grundlegende technische Eigenschaften wie Entscheidbarkeit und Axiomatisierbarkeit werden erklärt und die Beweisideen illustriert. Da der Formalismus im Allgemeinen unentscheidbar ist, werden für die praktische Verifikation Teilklassen untersucht. Für eine dieser Teilklassen existiert das automatisches Verifikationswerkzeug MoDiShCa. Damit wird auch die erfolgreiche Anwendung auf bisher nicht handhabbare Problemfelder gezeigt. Offene Forschungsfragen bestehen beispielsweise in der Verbesserung der Anwendbarkeit für Nicht-Logiker und in der Verbesserung der automatischen und halbautomatischen Verifikationsverfahren, um größere Fallstudien handhaben zu können.

Literatur

[AD94] R. Alur und D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[CGP00] E.M. Clarke, O. Grumberg und D Peled. *Model Checking*. MIT Press, 2000.

[Die99] H. Dierks. *Specification and Verification of Polling Real-Time Systems*. Dissertation, University of Oldenburg, Juli 1999.

[GKWZ03] D. Gabbay, A. Kurucz, F. Wolter und M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Elsevier, 2003.

[HZ04] M. R. Hansen und Zhou Chaochen. *Duration Calculus: A Formal Approach to Real-Time Systems*. EATCS: Monographs in Theoretical Computer Science. Springer, 2004.

[KBPOB99] B. Krieg-Brückner, J. Peleska, E.-R. Olderog und A. Baer. The UniForM Workbench, a Universal Development Environment for Formal Methods. In J.M. Wing, J. Woodcock und J. Davies, Hrsg., *FM'99 – Formal Methods*, Jgg. 1709 of *LNCS*, Seiten 1186–1205. Springer, 1999.

[QS06] J.-D. Quesel und A. Schäfer. Spatio-Temporal Model Checking for Mobile Real-Time Systems. In K. Barkaoui, A. Cavalcanti und A. Cerone, Hrsg., *Theoretical Aspects of Computing, ICTAC 2006*, Jgg. 4281 of *LNCS*, Seiten 347–361. Springer, 2006.

[Sch05] A. Schäfer. A Calculus for Shapes in Time and Space. In Z. Liu und K. Araki, Hrsg., *Theoretical Aspects of Computing, ICTAC 2004*, Jgg. 3407 of *LNCS*, Seiten 463–478. Springer, 2005.

[Sch06] A. Schäfer. *Specification and Verification of Mobile Real-Time Systems*. Dissertation, University of Oldenburg, December 2006.

[Sch07] A. Schäfer. Axiomatisation and Decidability of Multi-Dimensional Duration Calculus. *Information and Computation*, 205(1):25–64, 2007.

[ZHR91] Zhou Chaochen, C.A.R. Hoare und A.P. Ravn. A calculus of durations. *IPL*, 40(5):269–276, 1991.

Andreas Schäfer wurde am 7. November in Witten 1977 geboren. Nach dem Abitur mit Durchschnittsnote 1.0 hat er parallel zum Zivildienstes im Marienkrankenhaus Papenburg zwei Semester Informatik an der Fernuniversität Hagen studiert. Nach Ableistung des Zivildienstes hat er das Studium in Oldenburg fortgesetzt und sich auf Theoretische Informatik spezialisiert, insbesondere auf Formale Methoden. In der Diplomarbeit "Fehlerbaumanalyse und Model-Checking" entwickelte er ein Verfahren zur Kombination der Fehlerbaumanalyse mit einer formalen Verifikationsmethode. Nachdem Abschluss des Diploms mit Auszeichnung arbeitete Andreas Schäfer als Wissenschaftlicher Mitarbeiter in der Arbeitsgruppe von Prof. Dr. Ernst-Rüdiger Olderog. Neben der Mitarbeit in der Lehre der Abteilung hat er auf dem Gebiet der Verifikation von Realzeitsystemen und mobiler Systeme geforscht. Die Verteidigung der Dissertation "Specification and Verification of Mobile Real-Time Systems" fand im Dezember 2006 statt. Seit April 2007 ist Andreas Schäfer beim Europäischen Patentamt tätig.