

BestMasters

Malte Schmitz

Verteilte Laufzeit- verifikation auf eingebetteten Systemen

Logiken und Monitorkonstruktionen
für asynchrone Prozesse



Springer Vieweg

BestMasters

Weitere Informationen zu dieser Reihe finden Sie unter
<http://www.springer.com/series/13198>

Mit „BestMasters“ zeichnet Springer die besten Masterarbeiten aus, die an renommierten Hochschulen in Deutschland, Österreich und der Schweiz entstanden sind. Die mit Höchstnote ausgezeichneten Arbeiten wurden durch Gutachter zur Veröffentlichung empfohlen und behandeln aktuelle Themen aus unterschiedlichen Fachgebieten der Naturwissenschaften, Psychologie, Technik und Wirtschaftswissenschaften.

Die Reihe wendet sich an Praktiker und Wissenschaftler gleichermaßen und soll insbesondere auch Nachwuchswissenschaftlern Orientierung geben.

Malte Schmitz

Verteilte Laufzeitverifikation auf eingebetteten Systemen

Logiken und Monitorkonstruktionen
für asynchrone Prozesse

Mit einem Geleitwort von
Prof. Dr. Martin Leucker

 Springer Vieweg

Malte Schmitz
Lübeck, Deutschland

BestMasters

ISBN 978-3-658-12851-7

ISBN 978-3-658-12852-4 (eBook)

DOI 10.1007/978-3-658-12852-4

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Springer Vieweg

© Springer Fachmedien Wiesbaden 2016

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Der Verlag, die Autoren und die Herausgeber gehen davon aus, dass die Angaben und Informationen in diesem Werk zum Zeitpunkt der Veröffentlichung vollständig und korrekt sind. Weder der Verlag noch die Autoren oder die Herausgeber übernehmen, ausdrücklich oder implizit, Gewähr für den Inhalt des Werkes, etwaige Fehler oder Äußerungen.

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer Vieweg ist Teil von Springer Nature

Die eingetragene Gesellschaft ist Springer Fachmedien Wiesbaden GmbH

Geleitwort

Industrie 4.0 bedeutet die automatische integrierte und kosteneffiziente Erstellung von Gütern. Essentiell für die Industrie 4.0 ist die sichere und korrekte Steuerung von Industrieanlagen. Im Rahmen seiner Masterarbeit hat Herr Schmitz die Grundlagen für eine derartige Steuerung mit Hilfe von innovativen Überwachungskonzepten zur Laufzeit von Industrieanlagen gelegt. Er hat seine Konzepte modellhaft mit Hilfe von LEGO Mindstorms visualisiert, um neben den theoretischen Grundlagen insbesondere auch die praktische Anwendung transparent machen zu können.

Prof. Dr. Martin Leucker

Direktor des Instituts für Softwaretechnik und Programmiersprachen
an der Universität zu Lübeck

Institut für Softwaretechnik und Programmiersprachen

Das Institut für Softwaretechnik und Programmiersprachen (ISP) ist ein international ausgewiesenes Institut der Universität zu Lübeck und ist spezialisiert auf die systematische Entwicklung nachweisbar sicherer und zuverlässiger Systeme.

Daher wird neben allgemeinen programmier- und softwaretechnischen Aspekten ein Schwerpunkt auf Formale Methoden gelegt. Konkret arbeitet das ISP an Verifikationsverfahren für Softwaresysteme, wie Model-Checking, Testen und Runtime-Verification, aber auch Techniken zur Spezifikation von funktionalen und nicht-funktionalen Eigenschaften von Systemen, ihren Komponenten und deren Kommunikation.

Das ISP wird seit August 2010 von Prof. Dr. Martin Leucker geleitet und ist in zahlreichen Verifikations- und Testprojekten in den Domänen Medical, Automotive und Industry Automation involviert, in denen ein Schwerpunkt auf den zertifizierbaren Nachweis der Sicherheit von Systemen gelegt wurde.

Darüberhinaus bietet das ISP Vorlesungen, Seminare und Praktika in den Bereichen Softwaretechnik und Verifikation an. Auf diese Weise fließen aktuelle Forschungsergebnisse direkt in die Ausbildung der Studenten ein.

Kurzfassung

In dieser Arbeit werden Varianten der linearen Temporallogik (LTL) für den Einsatz zur Laufzeitverifikation verteilter, asynchroner, eingebetteter Systeme und die zugehörigen Monitorkonstruktionen untersucht und implementiert. Dazu wird eine verteilte Temporallogik mit Vergangenheitsoperatoren (ptDTL) und eine auf dreiwertiger Temporallogik (LTL_3) basierende, verteilte Temporallogik (fDTL) sowie eine synchronisierte Variante dieser Logik (fSDTL) verwendet. Für die dreiwertigen Logiken werden neue Automatenmodelle entwickelt und mit bekannten Monitorkonstruktionen verglichen. Als eingebettete Systeme kommen dabei Roboter von LEGO Mindstorms zum Einsatz, die in dem C-Dialekt NXC programmiert werden. Im Rahmen dieser Arbeit wurde ein Scala-Programm zur Monitorinjektion über Programmtransformation entwickelt. Das Programm liest den C-Quelltext mehrerer Roboter ein, parst die in Kommentaren enthaltenen Annotationen und ergänzt die für die verteilte Laufzeitverifikation notwendigen Routinen im Quelltext. Anhand dieser Software werden die verschiedenen Monitorkonstruktionen und Techniken der Zustandsgenerierung im praktischen Einsatz verglichen.

Inhaltsverzeichnis

Abkürzungsverzeichnis	xv
Tabellenverzeichnis	xvii
Abbildungsverzeichnis	xix
Definitions- und Theoremverzeichnis	xxi
1 Einleitung	1
1.1 Verwandte Arbeiten	4
1.2 Aufbau der Arbeit	6
2 Logiken	7
2.1 Grundlagen	7
2.1.1 Verbände	8
2.1.2 Alphabet	9
2.1.3 Wort	10
2.1.4 Sprache	11
2.2 Lineare Temporallogik (LTL)	11
2.3 Lineare Temporallogik mit Vergangenheitsoperatoren (ptLTL)	16
2.4 Unvoreingenommene (impartial) Antizipation	19
2.4.1 Unvoreingenommene (impartial) Semantikfunktionen	19
2.4.2 Antizipierende (anticipatory) Semantikfunktionen ...	20
2.5 LTL mit unvoreingenommener Antizipation (LTL ₃)	21
2.6 Monitorbarkeit	23

2.6.1	Hierarchie temporaler Sprachen	25
2.7	Lineare Temporallogik für verteilte Systeme	30
2.7.1	Verteilte Lineare Temporallogik (DTL)	35
2.7.2	DTL-Semantik	36
2.8	LTL ₃ mit dreiwertigen Propositionen (fDTL)	38
2.8.1	fDTL _ω -Semantik	40
2.8.2	fDTL-Semantik	43
2.9	fDTL mit Synchronisation (fSDDL)	45
2.9.1	fSDDL _ω -Semantik	48
2.9.2	fSDDL-Semantik	49
3	Monitore	51
3.1	Monitorkonstruktion für ptLTL	51
3.2	Monitorkonstruktion für LTL ₃	55
3.2.1	Automatenmodell für LTL	55
3.2.2	Automatenmodell für LTL ₃	61
3.3	Monitorkonstruktion für DTL	66
3.4	Monitorkonstruktion für fDTL	71
3.4.1	Automatenmodell für fDTL _ω	72
3.4.2	Automatenmodell für fDTL	76
3.5	Monitorkonstruktion für fSDDL	77
3.5.1	Approximatives Automatenmodell für LTL ₃	78
3.5.2	Automatenmodell für fSDDL _ω	84
3.5.3	Approximatives Automatenmodell für fSDDL	90
4	Implementierung	97
4.1	Not eXactly C (NXC)	100
4.1.1	Kommunikation und Knowledge-Vektoren	101
4.2	Monitorannotationen	103
4.3	Belegungen für Propositionen	107
4.3.1	Entfernte Propositionen	107
4.3.2	Explizite C-Funktion	108
4.3.3	Explizite Umschaltung	109
4.3.4	Programmtransformation	110
4.4	Zustandswechsel	111
4.4.1	Expliziter Schritt	112
4.4.2	Zu bestimmten Ereignissen	112
4.4.3	Nach fester Zeit	115
4.4.4	Öffentliche Propositionen	115
4.5	Monitorgenerierung in Scala	116

4.5.1	Architektur der Applikation	118
4.5.2	Datenstruktur der Annotationen	120
4.5.3	ptLTL-Monitorgenerierung	126
4.5.4	fDTL-Monitorgenerierung	128
4.5.5	Minimierung der Transitionen mit Quine-McCluskey	132
4.5.6	Caching mit db4o	133
4.6	Beispiele	135
4.6.1	Taktstraße	135
4.6.2	Ampelanlage	142
4.7	Benchmark-Tests	146
5	Zusammenfassung und Ausblick	153
	Literaturverzeichnis	157
A	Anhang	161
A.1	Klassifikationen temporaler Eigenschaften	161
A.2	Paket- und Dateistruktur der Software	163

Abkürzungsverzeichnis

ABA	alternierender Büchi-Automat, engl. <i>alternating Büchi automaton</i>
AFA	alternierender endlicher Automat, engl. <i>alternating finite automaton</i>
BA	Büchi-Automat, engl. <i>Büchi automaton</i>
BNF	Normalform kontextfreier Grammatiken, engl. <i>Backus–Naur form</i>
DFA	endlicher Automat, engl. <i>deterministic finite automaton</i>
DTL	verteilte Linearzeit-Temporallogik, engl. <i>distributed temporal logic</i>
fDTL	verteilte Linearzeit-Temporallogik mit Zukunftsoperatoren, engl. <i>future distributed linear temporal logic</i>
fDTL _ω	verteilte Linearzeit-Temporallogik mit Zukunftsoperatoren auf unendlichen Worten, engl. <i>future distributed linear temporal logic on words</i>
FLTL	lineare Temporallogik auf endlichen Worten, engl. <i>finite linear temporal logic</i>
FLTL ₃	dreiwertige lineare Temporallogik auf endlichen Worten, engl. <i>3-valued finite linear temporal logic</i>
FLTL ₄	vierwertige lineare Temporallogik auf endlichen Worten, engl. <i>4-valued finite linear temporal logic</i>
fSDTL	synchronisierte verteilte Linearzeit-Temporallogik mit Zukunftsoperatoren, engl. <i>future synchronized distributed linear temporal logic</i>
fSDTL _ω	synchronisierte verteilte Linearzeit-Temporallogik mit Zukunftsoperatoren auf unendlichen Worten, engl. <i>future synchronized distributed linear temporal logic on words</i>

LTL	lineare Temporallogik, engl. <i>linear temporal logic</i>
LTL ₃	dreiwertige lineare Temporallogik, engl. <i>3-valued linear temporal logic</i>
NFA	nichtdeterministischer endlicher Automat, engl. <i>nondeterministic finite automaton</i>
NNF	Negationsnormalform, engl. <i>negation normal form</i>
NXC	C-artige Programmiersprache, engl. <i>not exactly C</i>
ptDTL	verteilte Linearzeit-Temporallogik mit Vergangenheitsoperatoren, engl. <i>past-time distributed linear temporal logic</i>
ptLTL	lineare Temporallogik mit Vergangenheitsoperatoren, engl. <i>past-time linear temporal logic</i>
RLTL	reguläre Linearzeit-Temporallogik, engl. <i>regular linear temporal logic</i>
TL	beliebige temporale Logik, engl. <i>temporal logic</i>

Tabellenverzeichnis

3.1	Beispiel eines indizierten Laufes	89
3.2	Beispiel der indizierten Läufe ρ_φ und $\rho_{\neg\varphi}$	95
4.1	Darstellung der in den Formeln verwendeten mathematischen Symbole mit dem ASCII-Zeichensatz.....	105

