Static analysis of unbounded structures in object-oriented programs

Grabe, I.

Static Analysis of Unbounded Structures in Object-Oriented Programs

Proefschrift

ter verkrijging van

de graad van Doctor aan de Universiteit Leiden

op gezag van Rector Magnificus prof. mr. P.F. van der Heijden,

volgens besluit van het College voor Promoties

te verdedigen op woensdag 19 december 2012

klokke 10:00 uur

door

Immo Grabe

geboren te Kiel, Duitsland, in 1979

Promotiecommissie

Promotoren:	Prof. Dr. F.S. de Boer	Universiteit Leiden
	Prof. Dr. M. Steffen	Universitetet i Oslo
Overige Leden:	Prof. Dr. E. Broch Johnsen	Universitetet i Oslo
	Prof. Dr. F. Arbab	Universiteit Leiden
	Prof. Dr. J.N. Kok	Universiteit Leiden
	Dr. M. Bonsangue	Universiteit Leiden

The work in this thesis has been carried out at the Christian-Albrechts--Universität zu Kiel, the Centrum Wiskunde & Informatica (CWI), and the Universiteit Leiden. The research was partially funded by the EU-project IST-33826 *Credo: Modeling and analysis of evolutionary structures for distributed services*, the EU-project FP7-231620 *HATS: Highly Adaptable and Trustworthy Software using Formal Methods*, and the German-Norwegian DAAD-NWO exchange project *Avabi* (Automated validation for behavioral interfaces of asynchronous active objects).

Cover design by Immo Grabe.
Printed by: Proefschriftmaken.nl || Uitgeverij BOXPress
Published by: Uitgeverij BOXPress, Oisterwijk
ISBN: 978-90-8891-154-75

Contents

1 Introduction **1**

I Object Creation **5**

2 Abstract Object Creation **7**
2.1 Introduction 7
2.2 Dynamic Logic 9
2.2.1 Syntax 10
2.2.2 Semantics 12
2.3 Axiomatization 14
2.3.1 Sequent Calculus 14
2.3.2 Application of General Updates 16
2.3.3 Contextual Application of Object Creation 18
2.4 Symbolic Execution 21
2.4.1 Simultaneous Updates for Symbolic State Representation 21
2.4.2 Symbolic Execution and Abstract Object Creation . . . 22
2.5 Discussion 24
2.5.1 Object Creation vs. Object Activation 24
2.5.2 Expressiveness 24
2.6 Conclusion 25

II Multi-threading **27**

3 Deadlock Detection **29**
3.1 Introduction 29
3.2 Synchronized Multithreaded Programs 32
3.2.1 Syntax 32

3.2.2 Operational Semantics 33
3.3 Thread Automata 34
3.4 CFL-Reachability 38
3.5 Soundness and Completeness of CFL-Reachability 41
3.6 Conclusion 43

III Active Objects **45**

4 Futures and Promises **47**
4.1 Introduction 48
4.2 Calculus 59
4.2.1 Syntax 59
4.2.2 Type system 64
4.2.3 Operational semantics 72
4.3 Interface behavior 88
4.3.1 Legal traces system 89
4.4 Conclusion 97

5 Termination Detection **99**
5.1 Introduction 99
5.2 Active Objects 101
5.2.1 Syntax 102
5.2.2 Operational Semantics 104
5.2.3 Finite Representation of Creol 109
5.3 Linda 114
5.3.1 Syntax 115
5.3.2 Semantics 115
5.3.3 Expressivness 117
5.4 Translation of Configurations 118
5.4.1 Translation of a Single Method 119
5.4.2 Translation of a Single Object 121
5.4.3 Translation of a Creol model 123
5.4.4 Translation of a Creol configuration 123
5.5 Termination 126
5.6 Conclusion 128

A Proofs **131**
A.1 Wellformedness of generated sequences 131
A.2 Existence of derivation 133
A.3 Soundness of lock handling 134

Abstract **137**

Samenvatting **139**

Curriculum Vitae **141**

List of Figures

2.1 Grammar rules for the simple WHILE-language 10
2.2 Sequent rules - first-order logic rules 15
2.3 Sequent rules - dynamic logic rules 16
2.4 Application of Updates - standard cases 17
2.5 Object activation style proof 25

3.1 Grammar rule for the simple multithreaded language 32
3.2 Operational rules 34

4.1 Claiming a future (busy wait) 57
4.2 Abstract syntax 60
4.3 Claiming a future 64
4.4 Typing (component level) 66
4.5 Typing (objects and threads) 69
4.6 Typing (objects and threads) 71
4.7 Internal steps 74
4.8 Structural congruence 75
4.9 Reduction modulo congruence 75
4.10 Labels 82
4.11 Typechecking labels 84
4.12 Scenarios 86
4.13 External steps 89
4.14 Legal traces 90

5.1 Linda operational semantics (symmetric rules omitted) 116
5.2 Message sending – ordered semantics 116
5.3 Message sending – unordered semantics 117
5.4 Congruence rules 127
5.5 Simulation steps 127