Lebenslauf

Christoph Lüth

Anschrift Auf den Hornstücken 17
28359 Bremen
(0421) 2070498
cxl@informatik.uni-bremen.de
<http://www.informatik.uni-bremen.de/~cxl>

Geboren 24. Mai 1966 in Lübeck

Nationalität Deutsch

Familienstand Verheiratet

Wissenschaftliche Laufbahn

ab 1998 Wissenschaftlicher Assistent (C1) bei Prof. Bernd Krieg-Brückner, Universität Bremen.
Beurlaubung von 2003–2004 zur Teilnahme am *SafeRobotics* Project.

1996 – 1998 Wissenschaftlicher Mitarbeiter im BMBF-Projekt *UniForm*.

Ausbildung

1991–95 Promotion in Informatik, Laboratory for Foundations of Computer Science, University of Edinburgh. Dissertationsthema: *Categorical Term Rewriting: Monads and Modularity*. Betreuer: Prof. Donald Sannella; Mündl. Prüfung (ohne Benotung) 10. Dezember 1997; Gutachter: Prof. Samson Abramsky, Dr. David Rydeheard. Graduation am 9. Dezember 1998.

1986–91 Diplom der Informatik, TU Berlin (Gesamtnote sehr gut)

1987–91 Während des Studiums Programmierer bei der IFS-Berlin GmbH. Mein Aufgabengebiet umfaßte Entwicklung und Wartung von Software, Kundenbetreuung, Systeminstallation und Systemwartung.

1985–86 Wehrdienst.

1976–85 Abitur, Trave-Gymnasium Lübeck (Schnitt 1.7).

Wissenschaftlicher Werdegang

Mein Arbeitsgebiet umfaßt die Entwicklung korrekter Software, wobei ich bestrebt bin, sowohl grundlegende und theoretische als auch praktische Ansätze zu vertreten, und beide vorteilhaft zu kombinieren. Hierbei können Fortschritte in der Theorie die Praxis befruchten, während praktisch auftauchende Probleme neue theoretische Fragestellungen aufwerfen können.

Meine praktischen Kenntnisse reichen bis in mein Studium zurück (u.a. die Nebentätigkeit während des Studiums), während ich meine theoretischen Kenntnisse u.a. der Promotion an der Universität Edinburgh verdanke. In dieser Spannweite sehe ich persönlich meine Stärke.

Auf der theoretischen Seite wende ich Methoden der Logik, Algebra und Kategorientheorie in der Informatik an, insbesondere auf dem Gebiet der Termersetzungssysteme. Ein Schwerpunkt dieser Interessen sind Modularitätsprobleme, d.h. wie können Fragestellungen über große Systeme durch Reduktion auf Fragestellungen über kleinere Systeme gelöst werden.

Auf der praktischen Seite bin ich überzeugt, dass formale Methoden nur mit entsprechender Werkzeugunterstützung erfolgreich sein können werden. Aus diesem Grund entwickle ich auf der Basis moderner Theorembeweisertechnologie (insbesondere des Theorembeweisers *Isabelle*) Werkzeugunterstützung, zum Beispiel durch Einbettung von Spezifikationsformalismen in *Isabelle*, oder durch Entwicklung von Methoden zur Abstraktion und Wiederverwendung, aber auch auf der Ebene der Entwicklung adäquater Benutzerschnittstellen und Dokumentenpräsentation.

Wesentliche Arbeiten der letzten Jahre

- **Kategorielle Methoden der Termersetzung.**

In meiner Dissertation entwickelte ich eine neuartige Semantik für Termersetzungssysteme. Herkömmliche Semantiken für Termersetzungssysteme basieren entweder auf binäre Relationen ohne jedwede Struktur, oder auf Reduktionsordnungen auf Termen ohne jedwede Abstraktion. Die neue Semantik abstrahiert von der konkreten Termstruktur, kann aber dennoch Schlüsselkonzepte wie Variablen, Schichten und Substitution modellieren. Sie nutzt Konstrukte aus der Kategorientheorie, sogenannte Monaden, indem Termersetzungssysteme als gerichtete Gleichungen modelliert werden, und die Kombination von Termersetzungssystemen durch die Kombination dieser Monaden durch Kolimiten modelliert wird.

Die erzielten Resultate [20,22,31] führten zu einer Einladung zu einem vierwöchigen Forschungsaufenthalt am ETL (Dr. de Vries; Tsukuba, Japan) im April 1998, und zu einer bis heute anhaltenden Zusammenarbeit mit Dr. Neil Ghani (University of Leicester) über kategorielle Methoden in der Termersetzung, deren Ergebnisse auf einem von mir mitorganisierten Workshop der Termersetzungskonferenz RTA'98 vorgestellt wurden.

Die Modellierung von Ersetzungssystemen durch Monaden wurde später auf anderen Formen der Termersetzung generalisiert, zum Beispiel unendliche Terme [4,5,6,26,27] und beliebige, durch Gleichungen definierte Strukturen [2]. Die Theorie der Kombination von Monaden reicht weit über die einfache Termersetzung hinaus, und fand auch Anwendung in der funktionalen Programmierung [13] und allgemeinen Modularitätsproblemen [14,8].

- **Transformationelle Programmentwicklung**

Während meine Arbeit an der Universität Bremen im Rahmen des UniForM-Projektes entwickelte ich in Zusammenarbeit mit Dr. Burkhard Wolff (jetzt ETH Zürich) das TAS-System für die Entwicklung durch Transformation im Theorembeweiser Isabelle [16,15,17,21]. Das System basiert auf dem Kalkül des *window inferencing*, und besitzt eine innovative graphische Benutzerschnittstelle, um die technischen Detail von Isabelle von dem Benutzer fernzuhalten. Es ist generisch und wurde für verschieden Formalismen instantiiert [16,19,18].

- **Wiederverwendung und Abstraktion.**

Die Wiederverwendung und systematische Verallgemeinerung zur Wiederverwendung (Abstraktion) ist ein großes Potential von formalen Programmentwicklungen. In Zusammenarbeit mit Prof. Krieg-Brückner habe ich das Projekt *AWE — Abstraktion und Wiederverwendung von formalen Entwicklungen* definiert und bei der DFG eine Sachmittelhilfe beantragt, die im Juni 2000 im Umfang von einem Mitarbeiter für zwei Jahre bewilligt wurde. Zusammen mit Dr. Einar Broch Johnsen (jetzt Universität Oslo), der auf dieser Stelle während des Jahres 2002 angestellt war, entwickelten wir eine Methode um Beweise im Theorembeweiser Isabelle zu abstrahieren und wiederzuverwenden [9], mit Anwendungen in der formalen Programmentwicklung [1,24].

- **Benutzerschnittstellen für Theorembeweiser.**

Der Entwurf und die Entwicklung einer grafischen Benutzerschnittstelle für Isabelle und das Transformationssystem TAS [15,17,21] führten zu einer generischen, wiederverwendbaren Benutzerschnittstelle für Theorembeweiser [7]. Dieser Entwurf wurde in einer Zusammenarbeit mit Dr. David Aspinall (University of Edinburgh) zu einer Synthese aus grafischen und text-basierten Benutzerschnittstellen weiterentwickelt [25]. In diesem Zusammenhang haben wir die Workshopreihe *International Workshop on User Interfaces for Theorem Provers (UITP)* für ein Treffen (*UITP'03*) im September 2003 in Rom [34] wiederbelebt. Das nächste Treffen (*UITP'05*) wird im April 2005 in Edinburgh stattfinden.

- **Dokumentenverwaltung und Präsentation.**

Von 2001-2003 nahm ich am BMBF-Projekt MMiSS teil, in dem fünf deutsche Universitäten ein Multi-Media Curriculum entwickelten, welches alle Aspekte der Entwicklung von sicheren Systemen Besonderen Wert wurde auf semantisch gestützte Versions- und Konfigurationsverwaltung von Dokumenten gelegt [12,23]. Mein Beitrag war der Entwurf des zentralen Repositoriumsservers, welches die Verwaltung von Dokumenten in einem speziellen XML-Format erlaubt, sowie die Vorlesung *Praktische Informatik 3 — Einführung in die funktionale Programmierung*, welche Teil des Curriculums wurde, und als Fallstudie ins Englische und Französische übersetzt auch von ausländischen Partnern in Swansea und Kamerun wiederverwendet wurde.

- **Fallstudie: Sichere Robotik.**

Während einer einjährigen Beurlaubung von meiner Assistentenstelle nahm ich an dem Projekt *SafeRobotics* teil. Zusammen mit Dr. Paolo Torrini habe ich das Sicherheitsmodul für die Steuerungssoftware des autonomen Rollstuhls *Rolland* formal spezifiziert und entwickelt.

- **CASL Consistency Checker**

Zusammen mit Dr. Markus Roggenbach (University of Wales, Swansea) and Dr. Lutz Schröder (Universität Bremen) haben wir den *CASL Consistency Checker (CCC)* entwickelt und implementiert, mit dem die Konsistenz von Spezifikationen in der algebraischen Spezifikationssprache CASL bewiesen werden kann. Mein Beitrag war hierbei der Entwurf der Systemarchitektur und die Leitung der Implementation. Der CCC wurde auf dem *17th Workshop on Algebraic Development Techniques WADT'04* in Barcelona, März 2004, präsentiert.

Forschungsmittel

- *AWE: Abstraktion und Wiederverwendung von formalen Entwicklungen.*
Deutsche Forschungsgemeinschaft, 2001-2003, € 80.000,-.
- Reisekostenbeihilfen der DFG für die Konferenzen TPHOLs 2000 and ICFP 2002.
Deutsche Forschungsgemeinschaft, € 1.100,- and DM 3.000,-.
- Reisekostenbeihilfen für die Konferenzen FroCoS 2002, ETAPS 99, CMCS'97
Zentrale Forschungsförderung (FNK) Universität of Bremen, je € 500,-.
- Reisekosten- und Mittelbeihilfe für einen Werkzeugdemonstrationsstand auf der Konferenz FM'99. Senator für Häfen und Handel, Freie und Hansestadt Bremen, DM 3000,-.

Organisierte Workshops

- *International Workshop on User Interfaces for Theorem Provers UITP'03*, Satellitenworkshop der *European Joint Conferences on Theory and Practice of Software (ETAPS)*, Edinburgh, 9. April 2005, Edinburgh. Zusammen mit Dr. David Aspinall.
- *International Workshop on User Interfaces for Theorem Provers UITP'03*. Rom, 8. September 2003. Zusammen mit Dr. David Aspinall. Tagungsband [25] erschien in *Electronic Notes in Theoretical Computer Science* Vol. 103.
- *Evolutionary Formal Software Development EFSD'02*, assoziiert mit *Formal Methods Europe FME'02*, Kopenhagen, 21. Juni 2002. Zusammen mit Dr. Dieter Hutter, Dr. David Basin and Dr. Peter A. Lindsay.
- *Workshop on Categorical Term Rewriting* auf der *International Conference on Rewriting Techniques and Applications (RTA'98)*. Tsukuba, Japan, 31. März 1998. Zusammen mit Dr. Neil Ghani and Dr. Fer-Jan de Vries.

Schriftenverzeichnis

Zeitschriftenartikel

1. Einar Broch Johnsen and Christoph Lüth: Abstracting Refinements for Transformation. *Nordic Journal of Computing*, 10: 316– 336, 2003.
2. Neil Ghani and Christoph Lüth. Rewriting via Coinserters. *Nordic Journal of Computing*, 10: 290– 312, 2003.
3. Christoph Lüth. Haskell in Space — An Interactive Game as a Functional Programming Exercise. *Educational Pearl, Journal of Functional Programming* **14**(6): 1077– 1085, November 2003.
4. Neil Ghani, Christoph Lüth and Federico de Marchi: Monads of Coalgebras: Rational Terms and Term Graphs. Angenommen für *Mathematical Structures in Computer Science*.
5. Federico de Marchi, Neil Ghani and Christoph Lüth. Solving Algebraic Equations using Coalgebra. *Theoretical Informatics and Applications*, 37: 301– 314, 2003.
6. Neil Ghani, Christoph Lüth, Federico de Marchi and John Power. Dualizing initial algebras. *Mathematical Structures in Computer Science*, **13**(2):349– 370, 2003.
7. Christoph Lüth and Burkhardt Wolff. Functional design and implementation of graphical user interfaces for theorem provers. *Journal of Functional Programming*, **9**(2):167– 189, March 1999.

Konferenzbeiträge

8. Micheal Abbott, Neil Ghani and Christoph Lüth: Abstract Modularity. Angenommen für *Rewriting Techniques and Applications, RTA'05*.
9. Einar Broch Johnsen and Christoph Lüth. Theorem Reuse by Proof Term Transformation. In K. Slind, A. Bunker and G. Gopalakrishnan (eds), *International Conference on Theorem Proving in Higher-Order Logic TPHOLs 2004*, Lecture Notes in Computer Science 3223, p. 152– 167. Springer, 2004.
10. Christoph Lüth, Markus Roggenbach and Lutz Schröder. CCC — The CASL Consistency Checker. In José Fiadeiro (ed.), *Recent Trends in Algebraic Development Techniques WADT 2004*, Lecture Notes in Computer Science 3423, p. 94–105. Springer, 2004. ,
11. Lutz Schröder, Till Mossakowski and Christoph Lüth. Type class polymorphism in an institutional framework. In José Fiadeiro (ed.), *Recent Trends in Algebraic Development Techniques WADT 2004*, Lecture Notes in Computer Science 3423, p. 234–248. Springer, 2004.

5

12. Bernd Krieg-Brückner, Dieter Hutter, Arne Lindow, Christoph Lüth, Achim Mahnke, Erica Melis, Philipp Meier, Arnd Poetzsch-Heffter, Markus Roggenbach, George Russell, Jan-Georg Smaus and Martin Wirsing. MultiMedia Instruction in Safe and Secure Systems. *Recent Trends in Algebraic Development Techniques, 16th International Workshop WADT 2002*. Lecture Notes in Computer Science 2755, p. 82 – 117. Springer, 2002.
13. Christoph Lüth and Neil Ghani. Composing Monads Using Coproducts. In *Proc. Seventh ACM SIGPLAN International Conference on Functional Programming ICFP'02*, p. 133–144. ACM Press, 2002.
14. Christoph Lüth and Neil Ghani. Monads and modularity. In A. Armando (ed.), *Frontiers of Combining Systems FroCos 2002, 4th International Workshop*, Lecture Notes in Artificial Intelligence 2309, p. 18–32. Springer, 2002.
15. Christoph Lüth and Burkhart Wolff. More about TAS and IsaWin: Tools for formal program development. In T. Maibaum (ed.), *Fundamental Approaches to Software Engineering FASE 2000. Joint European Conferences on Theory and Practice of Software ETAPS 2000*, Lecture Notes in Computer Science 1783, p. 367– 370. Springer, 2000.
16. Christoph Lüth and Burkhart Wolff. TAS — a generic window inference system. In J. Harrison and M. Aagaard (ed.), *Theorem Proving in Higher Order Logics: 13th International Conference TPHOLs 2000*, Lecture Notes in Computer Science 1869, p. 406–423. Springer, 2000.
17. Christoph Lüth, Haykal Tej, Kolyang, and Bernd Krieg-Brückner. TAS and IsaWin: Tools for transformational program development and theorem proving. In J.-P. Finance (ed.), *Fundamental Approaches to Software Engineering FASE'99. Joint European Conferences on Theory and Practice of Software ETAPS'99*, Lecture Notes in Computer Science 1577, p. 239– 243. Springer, 1999.
18. Christoph Lüth, Einar W. Karlsen, Kolyang, Stefan Westmeier, and Burkhart Wolff. Tool integration in the UniForM workbench. In *Tool Support for System Specification, Development, and Verification*, Advances in Computing Science, p. 160–173. Springer-Verlag Wien New York, 1999.
19. Christoph Lüth, Einar W. Karlsen, Kolyang, Stefan Westmeier, and Burkhart Wolff. Hol-Z in the UniForM-workbench – a case study in tool integration for Z. In J. P. Bowen, A. Fett, and M. G. Hinchey (ed.), *ZUM'98: The Z Formal Specification Notation, 11th International Conference of Z Users*, Lecture Notes in Computer Science 1493, p. 116–134. Springer, 1998.
20. Christoph Lüth and Neil Ghani. Monads and modular term rewriting. In E. Moggi and G. Rosolini (ed.), *Category Theory and Computer Science CTCS 97*, Lecture Notes in Computer Science 1290, p. 69– 86. Springer, 1997.

21. Kolyang, Christoph Lüth, Thomas Meier, and Burkhart Wolff. TAS and IsaWin: Generic interfaces for transformational program development and theorem proving. In M. Bidoit and M. Dauchet (ed.), *TAPSOFT '97: Theory and Practice of Software Development*, Lecture Notes in Computer Science 1214, p. 855– 859. Springer, 1997.
22. Christoph Lüth. Compositional term rewriting: An algebraic proof of Toyama’s theorem. In H. Ganzinger (ed.), *Rewriting Techniques and Applications RTA’96*, Lecture Notes in Computer Science 1103, p. 261– 275. Springer, 1996.

Begutachtete Workshops

23. Bernd Krieg-Brückner, Arne Lindow, Christoph Lüth, Achim Mahnke and George Russell. Semantic Interrelation of Documents via an Ontology. In G. Engels and S. Seehusen (eds.), *DeLFI 2004, Tagungsband der 2. Deutschen e-Learning Fachtagung Informatik*. Lecture Notes in Informatics P-52, p. 271– 282. Springer, 2004.
24. Einar Broch Johnsen, Christoph Lüth and Maksym Bortin. An Approach to Transformational Development in Logical Frameworks. In *Proc. International Symposium on Logic-based Program Synthesis and Transformation, LOPSTR 2004*, Verona, Italy, August 2004.
25. David Aspinall and Christoph Lüth. Proof General meets IsaWin: Combining Text-Based And Graphical User Interfaces. In David Aspinall and Christoph Lüth (ed.), *User Interfaces for Theorem Provers, International Workshop UITP’03*, Rome, Italy, September 2003. Electronic Notes in Theoretical Computer Science 103, p. 3–26.
26. Neil Ghani, Christoph Lüth, and Federico de Marchi. Coalgebraic monads. In L. Moss (ed.), *Coalgebraic Methods in Computer Science CMCS’02*, Electronic Notes in Theoretical Computer Science 65.1, 2002.
27. Neil Ghani, Christoph Lüth, Federico de Marchi, and John Power. Algebras, coalgebras, monads and comonads. In U. Montanari (ed.), *Coalgebraic Methods in Computer Science CMCS’01*, Electronic Notes in Theoretical Computer Science 44.1, 2001.
28. Christoph Lüth. Transformational program development in the UniForM workbench. In M. Haveran and O. Owe (ed.), *Selected papers from the 8th Nordic Workshop on Programming Theory*. Oslo University, 1996.
29. Kolyang, Christoph Lüth, Thomas Meier, and Burkhart Wolff. Generating graphical user-interfaces in a functional setting. In N. Merriam (ed.), *User Interfaces for Theorem Provers UITP ’96*, Technical Report, p. 59– 66. University of York, 1996.
30. Kolyang, Christoph Lüth, Thomas Meier, and Burkhart Wolff. Generic interfaces for transformation systems and interactive theorem provers. In R. Berghammer, B. Buth, and J. Pelleska (ed.), *International Workshop on Tool Support for Validation and Verification*, BISS Monograph 1. Shaker Verlag, 1998.

Verschiedenes

31. Neil Ghani, Christoph Lüth and Stefan Kahrs. Rewriting the conditions in conditional rewriting. Technical Report 2000/20, Dept. Mathematics and Computer Science, University of Leicester, 2000.
32. Christoph Lüth, Stefan Westmeier and Burkhart Wolff. sml.tk: Functional programming for graphical user interfaces. Technischer Bericht 8/96, FB 3, Universität Bremen, 1996.
33. Christoph Lüth. *Categorical Term Rewriting: Monads and Modularity*. PhD thesis, University of Edinburgh, 1998.

Herausgegeben

34. David Aspinall and Christoph Lüth (eds.), *Proc. User Interfaces for Theorem Provers, International Workshop UITP'03*, Rome 2003. *Electronic Notes in Theoretical Computer Sciences* 103.

Wissenschaftliche Vorträge

1. Theorem Reuse by Proof Term Transformation. *Theorem Proving in Higher-Order Logics TPHOLs 2004*. Park City, Utah, USA, September 2004.
2. The CASL Consistency Checker. *Recent Trends in Algebraic Development Techniques WADT 2004*. Barcelona, Spanien, April 2004.
3. Formal Development of a Safe Wheelchair Control Program. Department of Computer Science, University of Wales at Swansea, Großbritannien, März 2004; Institutt for informatikk, Universitetet i Oslo, Norwegen, Juni 2004.
4. IsaWin meets Proof General. *User Interfaces for Theorem Provers UITP'03*. Rom, Italien, September 2003.
5. The MMiSS repository. Teil des MMiSS Tutorials auf der *European Joint Conferences on Theory and Practice of Software ETAPS'03*. Warschau, Polen, April 2003.
6. TAS: A Generic Transformation System. Institut für Softwaretechnik und Programmiersprachen, Universität Lübeck, Dezember 2002.
7. Composing Monads using Coproducts. *International Conference on Functional Programming ICFP 2002*. Pittsburgh, USA, September 2002.
8. Haskell in Space — An Interactive Game as a Functional Programming Exercise. *Functional and Declarative Programming in Education FDPE'02*. Pittsburgh, USA, September 2002.
9. The MMiSS Repository, Development Graph and Authoring Tools. *Workshop on Algebraic Development Techniques WADT 2002*. Frauenchiemsee, September 2002.
10. Monads and Modularity. *Frontiers of Combining Systems FroCoS 2002*. Santa Margherita Ligure, Italien, April 2002.
11. Algebras, coalgebras, monads and comonads. *Coalgebraic Methods in Computer Science CMCS'01*. Genua, Italien, März 2001.
12. TAS — a generic window inference system. Werkzeugdemonstration, *13th International Conference on Theorem Proving in Higher Order Logics: TPHOLs 2000*. Portland, USA, August 2000.
13. More about TAS and IsaWin. *Joint European Conferences on Theory and Practice of Software ETAPS 2000*. Berlin, März 2000.
14. An Introduction to Coalgebraic Methods and Comonads. Norddeutsches Kategorienseminar *NDKS 2000* (North-German Category Seminar). Dötlingen, Februar 2000.
15. Transformational Development and Proof with TAS and IsaWin. Werkzeugdemonstration auf der *FM'99 World Congress on Formal Methods*. Toulouse, September 1999.

16. TAS and IsaWin. *Joint European Conferences on Theory and Practice of Software ETAPS'99*. Amsterdam, März 1999.
17. Hol-Z in the UniForM-workbench – a case study in tool integration for Z. *ZUM'98: The Z Formal Specification Notation, 11th International Conference of Z Users*. Berlin, September 1998.
18. Tool integration in the UniForM workbench. *International Workshop on Tool Support for System Specification, Development and Verification TOOLS'98*. Malente, Mai 1998.
19. A monadic proof of Toyama's theorem. *Workshop on Categorical Term Rewriting, 9th International Conference on Rewriting Techniques and Applications RTA'98*. Tsukuba, März 1998.
20. Monads and modular term rewriting. *Category Theory and Computer Science CTCS'97*. Santa Margherita Ligure, Italien. September 1997.
21. TAS and Isawin: Generic Interfaces for Transformational Program Development and Theorem Proving. Werkzeugdemonstration, *Formal Methods Europe FME'97*, Graz, Österreich, September 1997.
22. Using enriched monads to model term rewriting systems. *64th Meeting of the Peripatetic Seminar on Sheaves and Logic*. Braunschweig, Mai 1997.
23. TAS and IsaWin: Generic interfaces for transformational program development and theorem proving. Talk and tool demonstration, *TAPSOFT 97': Theory and Practice of Software Development*, Lille, Frankreich, April 1997.
24. Transformational program development in the UniForM workbench. *8th Nordic Workshop on Programming Theory*. Oslo, Norwegen, Dezember 1996.
25. Efficient Second Order Matching (in Vertretung von Dr. Shi Hui). *7th International Conference on Rewriting Techniques and Applications RTA'96*, New Brunswick, USA, Juli 1996.
26. Compositional term rewriting: An algebraic proof of Toyama's theorem. *7th International Conference on Rewriting Techniques and Applications RTA'96*, New Brunswick, USA, Juli 1996.
27. Generic interfaces for transformation systems and interactive theorem provers. *2nd International Workshop on Tool Support for System Specification, Development and Verification*, Bremen, Mai 1996.
28. Generating graphical user-interfaces in a functional setting. *Workshop User Interfaces for Theorem Provers (UITP '96)*, Universität York, England, April 1996.
29. Using Monads to model term rewriting. *Compass Workshop*, Sintra, Portugal, Februar 1995.
30. Explorations in Categorical Term Rewriting. *Compass Workshop*, Dresden, September 1993.

Lehre

Die Lehrmaterialien für die folgenden Lehrveranstaltungen wurden im Anhang zusammengestellt.

Lehrveranstaltungen

Praktische Informatik 3

Vorlesung, 4 SWS, Wintersemester 2000/01, 2001/02, 2002/03, 2004/05.

In dieser Grundstudiumsveranstaltung (3. Semester) werden Studenten, die bereits die objektorientierte und funktionale Programmierung kennengelernt haben, in die funktionale Programmierung eingeführt. Als Programmiersprache kommt die nicht-strikte rein funktionale Programmiersprache Haskell zum Einsatz. Die Veranstaltung stützt sich auf zwei Lehrbücher, *Haskell — The Craft of Programming* von Simon J. Thompson, und *The Haskell School of Expression* von Paul Hudak.

Die Veranstaltung gliedert sich in zwei Teile: während des ersten Teils werden die Grundkonzepte der funktionalen Programmierung wie Funktionsdefinition, Typisierung, Basisdatentypen, Funktionen höherer Ordnung, Polymorphie, algebraische Datentypen und abstrakte Datentypen (ADTs) systematisch vorgestellt, und während des zweiten Teils werden diese Konzepte im Rahmen von Fallstudien angewandt. Fokus der Veranstaltung ist die Demonstration der unmittelbaren Anwendbarkeit und Nützlichkeit dieser Konzepte, und damit die Übertragbarkeit auf andere Situationen und Programmierparadigmen.

Eine Fallstudie ist interaktive Grafikprogrammierung. In diesem Rahmen wurde eine Übungsaufgabe entwickelt, in welcher die Studenten einen interaktiven Weltraumsimulator entwickeln, und die als *Educational Pearl* veröffentlicht wurde [3].

Forgeschrittene Techniken der Funktionalen Programmierung

Kurs, Wintersemester 2004/05

Diese Veranstaltung wurde im wesentlichen für die Teilnehmer des studentischen Projektes *Bordeaux* konzipiert, fand aber auch bei anderen Studenten Interesse. Themen sind über die *Praktische Informatik 3* hinausgehende Techniken der funktionalen Programmierung, wie Ein/Ausgabe, Effizienzaspekte, Sprachinteroperabilität, Fehler- und Ausnahmebehandlung, aber auch Themenfelder wie Netzwerkprogrammierung, nebenläufige und reaktive Programmierung, grafische Benutzerschnittstellen, sowie fortgeschrittene Aspekte des Typsystems wie existentielle Typen, *Arrows* und abhängige Typen.

Techniken zur Entwicklung korrekter Software II

Kurs, 4 SWS, zusammen mit L. Schröder, M. Roggenbach, und H. Shi im Sommersemester 2002

Aufbauend auf in der Einführungsveranstaltung “Techniken zur Entwicklung korrekter Software I” erworbenen Kenntnisse werden in diesem Kurs fortgeschrittene Methoden der formalen Softwareentwicklung vermittelt. Themen sind fortgeschrittene Aspekte der algebraischen Spezifikation, insbesondere partielle Funktionen, Subtypen, term erzeugte Datentypen und strukturierte Spezifi-

kation, Verfeinerungsbeweise für CASL-Spezifikationen unter Verwendung der in Isabelle/HOL implementierten Beweisunterstützung für CASL, schrittweise Programmentwicklung aus Anforderungsspezifikationen, sowie praktische Übungen in CSP.

Internet-Programmierung

Kurs, 4 SWS, zusammen mit Shi Hui im Sommersemester 2001.

Eine Einführung in nebenläufige und verteilte Programmierung. Es wurden Themen von den Grundlagen der Nebenläufigkeit wie Synchronisationstechniken bis hin zur Netzwerkprogrammierung (Sockets, *remote method invocation*) in Haskell und Java behandelt.

Abstrakte Systemprogrammierung I und II

Kurs, 4 SWS, Wintersemester 1999/00 und Sommersemester 2000.

Systemprogrammierung in deklarativen Sprachen für fortgeschrittene Studenten. Am Beispiel der funktionalen Sprache Haskell wurden Themen wie Nebenläufigkeit (*lightweight concurrency*), Netzwerkprogrammierung, Webprogrammierung, Programmierung grafischer Benutzerschnittstellen, Sprachinteroperabilität und komponentenbasierte Programmierung dargestellt. Außerdem wurde auf Erweiterungen von Haskell wie Monaden, Ausnahmebehandlung und existentielle Typen eingegangen.

Im zweiten Teil des Kurses bearbeiteten die Studenten in Gruppen ein größeres Projekt in Haskell (wie z.B. eine CORBA-Anbindung).

Korrekte Software durch Formale Methoden

Kurs 4 SWS im Wintersemester 1998/99

Eine Einführung in formale Methoden für Hauptstudiumsstudenten. Die Studenten wurden über die operationale und denotationale Semantik je einer einfachen imperativen und funktionalen Sprache an den Korrektheitsbegriff herangeführt. Danach werden Verifikationsmethoden wie der Hoare-Kalkül und formale Programmentwicklung durch Transformation eingeführt. Zur Verifikation und Entwicklung von nebenläufigen und reaktiven Programmen werden Prozeßkalküle wie CSP und der π -Kalkül vorgestellt.

Grundlage der Beweisarbeit in diesem Kurs ist der Theorembeweiser Isabelle, und der Modelchecker FDR2.

Studentische Projekte

ProSecCo — Proven Security and Correctness.

Betreut in Zusammenarbeit mit Bernd Krieg-Brückner, Lutz Schröder und Markus Roggenbach, Wintersemester 2001/02 bis Sommersemester 2003.

In diesem studentischen Projekt ist ein Dateisystem entwickelt worden mit dem Ziel, Sicherheit sowohl hinsichtlich Datensicherheit als auch hinsichtlich funktionaler Korrektheit zu garantieren (was naturgemäß nur teilweise zu erreichen war). Basierend auf einer Mehrschichtenarchitektur ist

ein Client-Server-System mit verschlüsselter Datenübertragung implementiert worden. Modulinterfaces sowie die Anforderungen an einzelne Module wurden dabei in CASL spezifiziert; für eine Abstraktion des Kommunikationsprotokolls ist ferner ein einfacher Verklemmungsfreiheitsbeweis in STeP geführt worden. Implementierungssprachen waren C und Haskell.

Bordeaux — Formale Entwicklung einer Rollstuhlsteuerung.

Betreut in Zusammenarbeit mit Lutz Schröder und Till Mossakowski, Wintersemester 2004/2005 bis Sommersemester 2006.

Dieses studentische Projekt befasst sich mit dem Bremer autonomen Rollstuhl als Anwendungsbeispiel für die formale Spezifikation eingebetteter Systeme. Hierbei geht es sowohl um Sicherheitseigenschaften (z.B. dass der Rollstuhl nie mit einem Hindernis kollidiert) als auch um Verklemmungsfreiheitseigenschaften bzw. Leistungsgarantien (z.B. dass der Rollstuhl auch wirklich zu einem gegebenen Ziel fährt, wenn er dazu aufgefordert wird).

Kernthema des Projektes ist die Implementierung einer Rollstuhlsteuerung, die schrittweise von technischen Details abstrahiert und somit leichter zu verifizieren ist. Dazu wird die Schnittstelle zur unteren Ebene der Rollstuhlsteuerung verkapselt und in eine unter Haskell zu entwickelnde anwendungsspezifische Sprache (domain-specific language) eingebettet, die einer formalen Verifikation zugänglich ist. Nach der Spezifikation des Verhaltens des Rollstuhls folgt die Implementierung und der halbautomatische Beweis der Korrektheit. Für Systemtests wird es neben dem Rollstuhl selbst auch einen Simulator geben, dessen Anbindung an das System Teil des Projekts ist.

Diplomarbeiten

- *EMMOR — Entwicklung des mathematischen Modells zu Rolland*

Gunnar Spiess, September 2002.

Diese Diplomarbeit entwickelt einen Formalismus, mit dem eine Kollisionsvermeidung für den autonomen Bremer Rollstuhl Rolland implementiert werden kann. Ziel dieses Formalismus ist es, die Navigation des Rollstuhls in einer Welt von Hindernissen zu modellieren, und dem Rollstuhl gegebene Fahrtkommandos auf Sicherheit zu überprüfen, d.h. sicherzustellen, dass ein gegebenes Fahrtkommando nicht zu einer Kollision mit einem Hindernis führt.

- *Entwicklung und Implementation eines generischen Window-Inferencing Kalküls*

Pascal Schmidt, August 2004.

Window Inferencing ist eine auf Staples und Grundy zurückgehende Formalisierung des hierarchischen, transformationellen Schließens, wie es zum Beispiel in der Schulmathematik (Äquivalenzbeweise in der Algebra) verwendet wird. In der Diplomarbeit wird ein generischer Kalkül für Window Inferencing für den Theorembeweiser Isabelle, welches auf der Arbeit [16] basiert, entwickelt und implementiert.

Themenvorschläge

Hiermit schlage ich folgende Themen für den zu meiner mündlichen Habilitationsleistung gehörigen wissenschaftlichen Vortrag vor:

1. QML — a functional quantum programming language.
2. Premonoidal categories and notions of computation.
3. Specification and development of interactive systems — FOCUS on streams, interfaces and refinements.

Erklärung über frühere Anträge auf Zulassung zur Habilitation

Hiermit erkläre ich, dass ich zu keinem früheren Zeitpunkt die Zulassung zur Habilitation beantragt habe.

Christoph Lüth. Bremen, 01.02.2005