TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

computer languages

Institut für Computersprachen
AB Programmiersprachen und Übersetzer

15. Kolloquium

Programmiersprachen und Grundlagen der Programmierung

— Programm und Vortragskurzzusammenfassungen —

Maria Taferl

12.-14. Oktober 2009

Jens Knoop, Adrian Prantl (Hrsg.)

Jens Knoop
Adrian Prantl
Institut für Computersprachen

Technische Universität Wien
http://www.complang.tuwien.ac.at

Bericht 2009-X-3
Schriftenreihe des Instituts für Computersprachen
Technische Universität Wien

Vorwort

Das Kolloquium *Programmiersprachen und Grundlagen der Programmierung* (*KPS*) findet dieses Jahr zum 15. Mal statt. Es setzt eine Reihe von Arbeitstagungen fort, die ursprünglich von den Professoren Friedrich L. Bauer (TU München), Klaus Indermark (RWTH Aachen) und Hans Langmaack (CAU Kiel) ins Leben gerufen wurde.

Aus den ursprünglich drei Arbeitsgruppen sind in der Zwischenzeit weitere Forschungsgruppen in ganz Deutschland und darüberhinaus hervorgegangen. Seit einigen Jahren präsentiert sich die Veranstaltung als ein offenes Forum für interessierte deutschsprachige Wissenschafter. Die folgende Liste gibt einen Überblick über die bisherigen Tagungsorte und Veranstalter und zeigt die lange Tradition der KPS-Treffen:

1980	Tannenfelde im Aukrug	Universität Kiel
1982	Altenahr	RWTH Aachen
1985	Passau	Universität Passau
1987	Midlum auf Föhr	Universität Kiel
1989	Hirschegg	Universität Augsburg
1991	Rothenberge bei Steinfurth	Universität Münster
1993	Garmisch-Partenkirchen	Universität der Bundeswehr München
1995	Alt-Reichenau	Universität Passau
1997	Avendorf auf Fehmarn	Universität Kiel
1999	Kirchhundem-Heinsberg	FernUniversität in Hagen
2001	Rurberg in der Eifel	RWTH Aachen
2004	Freiburg-Munzingen	Universität Freiburg
2005	Fischbachau	LMU München
2007	Timmendorfer Strand	Universität Lübeck

Das 15. Kolloquium Programmiersprachen und Grundlagen der Programmierung (KPS 2009) wird vom Institut für Computersprachen der Technischen Universität Wien organisiert. Wir freuen uns, zu diesem Jubiläumstreffen mehr als 50 Teilnehmer von 28 Universitäten aus Österreich, der Schweiz, Deutschland, Holland, England und Kanada vom 12. bis 14. Oktober 2009 im Wallfahrtsort Maria Taferl am Österreichischen Jakobsweg begrüßen zu können. Besonders freuen wir uns, unter den Teilnehmern Prof. Dr. Dr.h.c. Hans Langmaack als einen der Gründerväter dieser Tagungsreihe begrüßen zu können, sowie Prof. Dr. Rudolf Berghammer, Prof. Dr. Peter Thiemann, Prof. Dr. Clemens Grelck und Dr. Annette Stümpel als Organisatoren früherer KPS-Treffen. Ganz besonders freuen wir uns, dass nahezu alle Teilnehmer am diesjährigen KPS-Treffen in einem Vortrag über ihre Forschungsarbeit berichten werden. Um dies im Rahmen der zur Verfügung stehenden Zeit zu ermöglichen, verzichten wir in diesem Jahr auf einen eingeladenen Hauptvortrag.

Der Tagungsband ist als Bericht 2009-X-1 des Instituts für Computersprachen der TU Wien erschienen. Er enthält 46 Beiträge, zum Teil in Form von Kurzzusammenfassungen, von 62 Autoren. Weitere spät eingegangene Beiträge sind in einem Ergänzungsband gesammelt, der als Bericht 2009-X-2 des Instituts für Computersprachen der TU Wien erscheint. Alle Beiträge aus Haupt- und Ergänzungsband zeigen die Breite der wissenschaftlichen Forschung im Bereich Programmiersprachen und Grundlagen der Programmierung im deutschsprachigen Raum. Eine CD-ROM fasst zusätzlich alle Beiträge der beiden Tagungsbände zusammen. Unser Dank gilt allen Autoren für ihren Einsatz und die gute Zusammenarbeit, die diese beiden Bände ermöglicht haben. Unser besonderer Dank gilt darüberhinaus Frau Ewa Vesely und Herrn Leonid Narinsky, die bei der Vorbereitung und Organisation dieses Treffens von der Erstellung der Webseite über die Planung der Exkursion bis zum schnellen Auftun zusätzlicher Unterbringungsmöglichkeiten nach dem Hinauslaufen aus der Kapazität des Tagungshauses Hervorragendes geleistet haben. Unseren besonderen Dank drücken

wir dabei auch allen Mitarbeitern des Hauses Hotel Schachner für die gute Zusammenarbeit bei der Planung dieses Treffens aus.

Wir wünschen allen Teilnehmern interessante und spannende Vorträge, fruchtbare Diskussionen und Anregungen für die Forschungsarbeit, das Kennenlernen neuer Kollegen und das Wiedersehen guter Bekannter, das Anbahnen und Knüpfen neuer Kooperationen und die Verbreiterung und Vertiefung bestehender, eine erlebnis- und abwechslungsreiche Exkursion zu dem zum UNESCO-Weltkulturerbe gehörenden Benediktinerklosters Stift Melk und einen angenehmen und anregenden Aufenthalt in Maria Taferl am Eingang zur Wachau.

Willkommen zur KPS 2009!

Wien, im Oktober 2009

Jens Knoop
Adrian Prantl

Vorwort zum Programm- und Kurzzusammenfassungsheft

Das vorliegende Heft enthält das Programm und die Kurzzusammenfassungen der Vorträge auf dem 15. Kolloquium Programmiersprachen und Grundlagen der Programmierung (KPS 2009). Es ist als Bericht 2009-X-3 des Instituts für Computersprachen der TU Wien erschienen und soll besonders den Teilnehmern an diesem Kolloquium während der Tagung ein nützliches und hilfreiches *vade mecum* zu Überblick und Orientierung sein.

Wien, im Oktober 2009

Jens Knoop
Adrian Prantl

Teilnehmer

Sebastian Altmeyer	Universität des Saarlandes, Saarbrücken
Wolfram Amme	Friedrich-Schiller-Universität Jena
Enes Bajrović	Universität Wien
Gergö Barany	Technische Universität Wien
Rudolf Berghammer	Christian-Albrechts-Universität zu Kiel
Dirk Beyer	Simon Fraser University, Surrey, B.C.
Walter Binder	Università della Svizzera italiana, Lugano
Florian Brandner	Technische Universität Wien
Stefan Brunthaler	Technische Universität Wien
Jens Dörre	Universität Passau
M. Anton Ertl	Technische Universität Wien
Christoph Feller	Technische Universität Kaiserslautern
Sebastian Fischer	Christian-Albrechts-Universität zu Kiel
Gerhard Goos	Universität Karlsruhe (TH)
Clemens Grelck	Universiteit van Amsterdam und University of Hertfordshire
Jürg Gutknecht	ETH Zürich
Sebastian Hack	Universität des Saarlandes, Saarbrücken
Michael Hanus	Christian-Albrechts-Universität zu Kiel
Thomas Heinze	Friedrich-Schiller-Universität Jena
Christoph Höger	Technische Universität Berlin
Christina Jansen	RWTH Aachen
Tudor Jebelean	RISC, Johannes Kepler Universität Linz
Raimund Kirner	Technische Universität Wien
Jens Knoop	Technische Universität Wien
Peter Lammich	Westfälische Wilhelms-Universität Münster
Oliver Lampl	Alpen-Adria-Universität Klagenfurt
Hans Langmaack	Christian-Albrechts-Universität zu Kiel
Florian Lorenzen	Technische Universität Berlin
Tim A. Majchrzak	Westfälische Wilhelms-Universität Münster
Volker Mattick	Technische Universität Dortmund
Eduard Mehofer	Universität Wien
Thomas Meyer	Universität Basel
Patrick Michel	Technische Universität Kaiserslautern
Leonid Narinsky	Technische Universität Wien
Ulrich Neumerkel	Technische Universität Wien
Nikolaj Popov	RISC, Johannes Kepler Universität Linz
Adrian Prantl	Technische Universität Wien
Franz Puntigam	Technische Universität Wien
Fabian Reck	Christian-Albrechts-Universität zu Kiel
Dirk Richter	Martin-Luther-Universität Halle-Wittenberg

Wolfgang Scholz	Universität Passau
Sven-Bodo Scholz	University of Hertfordshire
Markus Schordan	Fachhochschule Technikum Wien
Dietmar Schreiner	Technische Universität Wien
Martin Schwarz	Technische Universität München
Lukas Stadler	Johannes Kepler Universität Linz
Annette Stümpel	Universität zu Lübeck
Michael Tautschnig	Technische Universität Darmstadt
Peter Thiemann	Universität Freiburg
Baltasar Trancón y Widemann	Universität Bayreuth
Christian Tschudin	Universität Basel
Helmut Veith	Technische Universität Darmstadt
Alexander Wenner	Westfälische Wilhelms-Universität Münster
Thomas Würthinger	Johannes Kepler Universität Linz
Wolf Zimmermann	Martin-Luther-Universität Halle-Wittenberg
Florian Zuleger	Technische Universität Darmstadt

Inhaltsverzeichnis

KPS 2009 – Programm

Sonntag, 11.10.2009

17:00 - 19:00 Registrierung

19:00 - ... Gemeinsames Abendessen

Montag, 12.10.2009

08:00 - 08:30 Registrierung

08:30 - 08:45 Begrüßung

08:45 - 10:15 Sitzung 1

Leitung: *Jens Knoop*

- *Computing and Visualizing Closure Objects Using Relation Algebra and* RelView
  Rudolf Berghammer (Christian-Albrechts-Universität zu Kiel)
- *Type-Safe Bytecode Generation in Scala*
  Peter Thiemann (Universität Freiburg)
- *Concurrency Engineering with S-Net*
  Clemens Grelck (Universiteit van Amsterdam und University of Hertfordshire)
- *Communication-based System Development using Standard Programming Languages*
  Annette Stümpel (Universität zu Lübeck)
- *On Undecidability Results of Real Programming Languages*
  Wolf Zimmermann (Martin-Luther Universität Halle-Wittenberg)
- *Set Functions for Functional Logic Programming*
  Michael Hanus (Christian-Albrechts-Universität zu Kiel)

10:15 - 10:50 Kaffeepause

10:50 - 12:30 Sitzung 2

Leitung: *Peter Thiemann*

- *Rapid Development of Dynamic Analysis Tools for the Java Virtual Machine*
  Walter Binder (Università della Svizzera italiana, Lugano)
- *Lazy Continuations for Java Virtual Machines*
  Lukas Stadler (Johannes Kepler Universität Linz)
- *Towards Dynamic Code Evolution for the HotSpot VM*
  Thomas Würthinger (Johannes Kepler Universität Linz)
- *Inline Caching meets Quickening*
  Stefan Brunthaler (Technische Universität Wien)
- *Robots, Software, Mayhem? Towards a Design Methodology for Robotic Software Systems*
  Dietmar Schreiner (Technische Universität Wien)

12:30 - 14:20 Mittagessen

14:20 - 16:00 Sitzung 3

Leitung: *Helmut Veith*

- *An Algebraic Framework for Modeling and Processing Hyperdocuments*
  Volker Mattick (Technische Universität Dortmund)
- *Maintaining XML Data Integrity in Programs – An Abstract Datatype Approach*
  Patrick Michel (Technische Universität Kaiserslautern)
- *Tree Automata for Analyzing Dynamic Pushdown Networks*
  Peter Lammich (Westfälische Wilhelms-Universität Münster)
- *Adding Weights to Dynamic Pushdown Networks*
  Alexander Wenner (Westfälische Wilhelms-Universität Münster)
- *The Reachability-Bound Problem*
  Florian Zuleger (Technische Universität Darmstadt)

16:00 - 16:45 Kaffeepause

16:45 - 18:25 Sitzung 4

Leitung: *Michael Hanus*

- *Typsicheres und generisches MapReduce*
  Jens Dörre (Universität Passau)
- *Reinventing Haskell Backtracking*
  Sebastian Fischer (Christian-Albrechts-Universität zu Kiel)
- *Towards a Parallel Search for Solutions of Non-deterministic Computations*
  Fabian Reck (Christian-Albrechts-Universität zu Kiel)
- *Lambdas und Schleifen in monotonen Logikprogrammen*
  Ulrich Neumerkel (Technische Universität Wien)
- *Erfahrungen bei der Auswahl von maschinenunabhängigen Optimierungen im Bereich des mobilen Codes*
  Wolfram Amme (Friedrich-Schiller-Universität Jena)

19:00 - ... Abendessen

Dienstag, 13.10.2009

08:30 - 10:10 Sitzung 5

Leitung: *Annette Stümpel*

- *Static Timing Analysis for Hard Real-Time Systems*
  Sebastian Altmeyer (Universität des Saarlandes, Saarbrücken)
- *Automatic Calculation of Coverage Profiles for Coverage-based Testing*
  Raimund Kirner (Technische Universität Wien)
- *Rekursionspräzise Intervallanalysen*
  Dirk Richter (Martin-Luther-Universität Halle-Wittenberg)
- *Generation of Incremental Parsers for Modern IDEs*
  Christoph Höger (Technische Universität Berlin)
- *Programmierung als Leitbild in der Theorie der Ökosysteme*
  Baltasar Trancón y Widemann (Universität Bayreuth)

10:10 - 10:45 Kaffeepause

10:45 - 12:15 Sitzung 6

Leitung: *Jürg Gutknecht (angefragt)*

- *Fraglets: Chemical Programming with a Packet Prefix Language*
  Christian Tschudin (Universität Basel)
- *Fraglets: Stochastic Programming for Provable Program Dynamics and Self-Healing Programs*
  Thomas Meyer (Universität Basel)
- *MultiMediaC# – QoS-Aware Programming with C#*
  Oliver Lampl (Alpen-Adria Universität Klagenfurt)
- *FQL: A Query Language for Program Testing*
  Michael Tautschnig (Technische Universität Darmstadt)
- *Technische Aspekte des erfolgreichen Testens von Software in Unternehmen*
  Tim A. Majchrzak (Westfälische Wilhelms-Universität Münster)
- *Verbesserung der Modellbildung und Analyse verteilter Geschäftsprozesse durch Prozessumstrukturierung*
  Thomas Heinze (Friedrich-Schiller-Universität Jena)

12:15 - 13:45 Mittagessen

13:45 - 14:45 Sitzung 7

Leitung: *Rudolf Berghammer*

- *SATIrE within ALL-TIMES: Improving Timing Technology with Source Code Analysis*
  Gergö Barany (Technische Universität Wien)
- *ARAL: a Language for Information Exchange between Program Analysis Tools*
  Markus Schordan (Fachhochschule Technikum Wien)
- *Towards a Static Profiler*
  Adrian Prantl (Technische Universität Wien)
- *Automatic Tool Generation from Structural Processor Descriptions*
  Florian Brandner (Technische Universität Wien)

15:00 - 19:00 Ausflug nach Melk

19:30 - ... Abendessen mit Weinverkostung

Mittwoch, 14.10.2009

08:30 - 10:00 Sitzung 8

Leitung: *Clemens Grelck*

- *Automatische Paketisierung*
  Sebastian Hack (Universität des Saarlandes, Saarbrücken)
- *Utilizing Multiple Hardware Threads with Pipeline Parallelism*
  M. Anton Ertl (Technische Universität Wien)
- *A New Look on Data Parallelism: Space vs. Time*
  Sven-Bodo Scholz (University of Hertfordshire)
- *A Formalisation of the OSEK Concurrency Model*
  Martin Schwarz (Technische Universität München)
- *How to Specify the Flow of Data Accessibility: An OO Way of Concurrent Programming*
  Franz Puntigam (Technische Universität Wien)
- *Programming Support for Cell/BE Multiprocessor*
  Eduard Mehofer (Universität Wien)

10:00 - 10:35 Kaffeepause

10:35 - 12:15 Sitzung 9

Leitung: *Hans Langmaack*

- *CPAchecker: A Tool for Configurable Software Verification*
  Dirk Beyer (Simon Fraser University, Surrey, B.C.)
- *A Sound, Complete and Usable Hoare-Style Logic for a Sequential Java Subset*
  Christoph Feller (Technische Universität Kaiserslautern)
- *Generierung von Hyperkantenersetzungsgrammatiken zur Heapabstraktion*
  Christina Jansen (RWTH Aachen)
- *Statische Erkennung von semantischen Feature-Interaktionen*
  Wolfgang Scholz (Universität Passau)
- *Functional Program Verification in Theorema. Soundness and Completeness*
  Nikolaj Popov (RISC, Johannes Kepler Universität Linz)

12:15 - 12:30 Kaffeepause

12:30 - 13:00 Sitzung 10

Leitung: *Christian Tschudin*

- *Combining Automated Reasoning and Algebraic Methods in Theorema*
  Tudor Jebelean (RISC, Johannes Kepler Universität Linz)
- *From Trusted Annotations to Verified Knowledge*
  Jens Knoop (Technische Universität Wien)

13:00 - 13:10 Verabschiedung

13:10 - 14:30 Mittagessen

Static Timing Analysis for Hard Real-Time Systems

Sebastian Altmeyer
Compiler Design Lab, Saarland University, Saarbrücken

Hard real-time systems impose strict timing constraints. To prove that these constraints are met, timing analyses aim to derive safe upper bounds on a task's execution time. Especially modern processor features (caches, out-of-order pipelines, etc.) have a strong impact on the variation of a task's execution time and thus, on the precision and the complexity of the timing analysis. Naive or measurement-based approaches usually can not guarantee safe and reliable timing bounds.

This talk provides an overview of the current timing analysis technique and shortly presents ongoing research at the Compiler Design Lab at Saarland University. A detailed presentation can be found in [1].

(This is joint work with Mohamed Abdel Maksoud, Claire Burguiere, Daniel Grund, Jörg Herter, Philipp Lucas, Oleg Parshin, Markus Pister, Jan Reineke, Marc Schlickling, Björn Wachter, and Reinhard Wilhelm.)

Reference

[1] Reinhard Wilhelm. Determining bounds on execution time. In R. Zurawski, editor, *Handbook on Embedded Systems*, pages 14-1, 14-23. CRC Press, 2005.

Erfahrungen bei der Auswahl von maschinenunabhängigen Optimierungen im Bereich des mobilen Codes

Wolfram Amme
Friedrich-Schiller-Universität Jena

Mobiler Code wird heutzutage auf der Basis einer Just-in-Time (JIT)-Übersetzung ausgeführt. Da die für eine JIT-Übersetzung notwendige Zeit direkt in die Ausführungszeit der Programme einfließt, ist man darauf bedacht, die für eine solche Übersetzung notwendigen Zeiten möglichst gering zu halten. Eine Möglichkeit dieses Vorhaben zu unterstützen, ist schon während der Erzeugung von mobilen Programmen auf der Produzentenseite eines Systems zum Transport mobiler Programme maschinenunabhängige Optimierungen auszuführen.

Im Vortrag wird die Wirkungsweise vorgestellt, die eine Verwendung von maschinenunabhängigen Optimierungen auf die Ausführung von mobilen Programmen hat. Alle dabei betrachteten Experimente wurden unter Verwendung des SafeTSA-Formats durchgeführt, eines auf der SSA-Form basierenden Zwischencodeformats für mobilen Code. Die auf der PowerPC- and IA32-Architektur durchgeführten Messungen zeigen, dass üblicherweise als maschinenunabhängig eingestufte Optimierungsformen tatsächlich oftmals sehr maschinenabhängiger Natur sind. Nichtsdestotrotz belegen die Ergebnisse jedoch auch, dass die Verwendung von maschinenunabhängigen Optimierungen auch im Bereich des mobilen Codes durchaus einen Nutzen haben kann.

SATIrE within ALL-TIMES: Improving Timing Technology with Source Code Analysis

Gergö Barany
Technische Universität Wien

We present the SATIrE source-to-source analysis framework within the context of ALL-TIMES, a European research and development project aimed at improving and integrating existing tools in the area of timing analysis. Within the project, SATIrE contributes by performing source-level static analysis on C programs and exporting its results for other tools to use.

This work gives an overview of SATIrE and how its analyses may improve timing analysis results obtained by other tools. We discuss SATIrE's efficient and powerful context-sensitive unification-based points-to analysis and its value interval analysis. We explain how SATIrE's integration with other analysis tools handles issues such as combination of source-level with binary-level analysis and communication of views of function call contexts between tools.

Computing and Visualizing Closure Objects Using Relation Algebra and RelView

Rudolf Berghammer
Christian-Albrechts-Universität zu Kiel

Closure objects play an important role in mathematics and computer science. We develop relation-algebraic specifications to recognize several classes of them, compute the complete lattices they constitute and transform any of these closure objects into another. All specifications are algorithmic and can directly be translated into the language of RelView. We show that the system is well suited for computing and visualizing closure objects and their lattices.

(This is joint work with Bernd Braßel.)

CPAchecker: A Tool for Configurable Software Verification

Dirk Beyer
Simon Fraser University, Surrey, B.C.

Configurable software verification is a recent concept for expressing different program analysis and model checking approaches in one single formalism. This paper presents CPAchecker, a tool and framework that aims at easy integration of new verification components. Every abstract domain, together with the corresponding operations, is required to implement the interface of configurable program analysis (CPA). The main algorithm is configurable to perform a reachability analysis on arbitrary combinations of existing CPAs. The major design goal during the development was to provide a framework for developers that is flexible and easy to extend. We hope that researchers find it convenient and productive to implement new verification ideas and algorithms using this platform and that it advances the field by making it easier to perform practical experiments. The tool is implemented in Java and runs as command-line tool or as Eclipse plug-in. We evaluate the efficiency of our tool on benchmarks from the software model checker Blast. The first released version of CPAchecker implements CPAs for predicate abstraction, octagon, and explicit-value domains. Binaries and the source code of CPAchecker are publicly available as free software.

(This is joint work with M. Erkan Keremoglu.)

Rapid Development of Dynamic Analysis Tools for the Java Virtual Machine

Walter Binder
Università della Svizzera italiana, Lugano

Many software engineering tools for the Java Virtual Machine that perform some form of dynamic program analysis, such as profilers or debuggers, are implemented with low-level bytecode instrumentation techniques. While program manipulation at the bytecode level is very flexible, because the possible bytecode transformations are not restricted, tool development is tedious and error-prone. Specifying bytecode instrumentations at a higher level using aspect-oriented programming (AOP) is a promising alternative in order to reduce tool development time and cost. However, prevailing AOP frameworks lack some features that are essential for certain dynamic analyses. In this paper, we focus on two common shortcomings in AOP frameworks with respect to the development of aspect-based tools - (1) the lack of mechanisms for passing data between woven advices in local variables, and (2) the support for user-defined static analyses at weaving time. We introduce J, an annotation-based AOP language and weaver that integrates support for these two features. We illustrate the benefits of the proposed features with an example.

(This is joint work with Alex Villazón, Danilo Ansaloni and Philippe Moret.)

Automatic Tool Generation from Structural Processor Descriptions

Florian Brandner
Technische Universität Wien

The success of embedded systems in mobile communication and entertainment devices, in commodity appliances, the domestic environment, as well as in the (safety) critical control systems of cars and airplanes made these small computer systems an indispensable part of every bodies daily lives. The demands on these systems in terms of reliability, efficiency, and computational power are steadily rising, while at the same time the physical dimensions and costs per unit are expected to shrink for every new product generation.

Application specific instruction processors (ASIPs) have become a valuable tool to deliver high computing power under rigid power and area constraints. The development of such a processor is delicate task that requires intimate knowledge of processor design, software development, compilers, and of course the particular problem domain.

Processor description languages – often referred to as architecture description languages (ADLs) – are a promising approach to capture the behavior, structure, and instruction set of an ASIP using a compact and concise description. These languages typically provide various views of the processor at different abstraction levels: (1) the behavioral level is primarily concerned with the abstract behavior of individual instructions, while (2) the structural view defines the hardware organization and resources. Processor description languages that focus on the former are often called behavioral languages, those following the latter approach structural. If a language combines specifications of both layers, it is called mixed. From a processor model, specified in one of these languages, software development tools, such as a compiler, linker, or assembler, can be (semi-)automatically customized for that particular processor. In addition, simulation tools, test cases, and even hardware models can be derived. Mixed languages typically provide the most flexibility in terms of these applications, because both a rather high-level, but also a rather low-level view of the processor is provided.

In this work a structural processor description language is presented that allows to derive a behavioral model from its structural specifications automatically. The language captures the behavioral and structural

details of a processor and thus provides great flexibility, but avoids redundancies known from mixed languages. We demonstrate the feasibility of our approach using two generators: (1) a compiler generator that automatically derives a highly optimizing code generator and (2) a simulator generator that derives a high-speed simulator based on dynamic binary translation. Experiments show that the derived tools can compete with hand crafted tools. The code produced by the generated compilers achieves speedups of up to 20%, on average moderate slowdowns of 5-15% have been observed. The simulation framework is similarly competitive and achieves a simulation speed that often matches the speed of the processor implementation in hardware.

Inline Caching meets Quickening

Stefan Brunthaler
Technische Universität Wien

Inline caches effectively eliminate the overhead implied by dynamic typing. Unfortunately, inline caching is mostly used in code generated by just-in-time compilers. We present efficient implementation techniques for using inline caches without dynamic translation, thus enabling future interpreter implementers to use this important optimization technique—we report speedups of up to 1.4—without the additional implementation and maintenance costs incurred by using a just-in-time compiler.

Typsicheres und generisches MapReduce

Jens Dörre
Universität Passau

MapReduce ist ein bei Google entwickeltes Programmiermodell, das dort die Entwicklung datenparalleler Programme für die massiv verteilte Ausführung ermöglicht. Dieses Modell basiert theoretisch auf lange bekannten Konzepten aus der Funktionalprogrammierung.

Im praktischen Einsatz fehlt allerdings noch ein formales und technisches Fundament, auf dessen Basis Eigenschaften wie Sicherheit und Korrektheit von MapReduce-Programmen garantiert werden können. Auch ist noch unklar, wie dieses Modell generisch in verschiedenen Szenarien angewendet werden kann.

Utilizing Multiple Hardware Threads with Pipeline Parallelism

M. Anton Ertl
Technische Universität Wien

In recent years general-purpose CPUs have aquired multiple hardware threads by providing multiple cores per CPU (multi-core) and multiple hardware threads per core (simultaneous multi-threading). Making good use of such resources has been a challenge for several decades that has been successfully attacked for scientific applications, but not to a significant extent for general-purpose applications. The main current paradigm for this programming problem is to have explicit threads that share memory and are synchronized by a variety of synchronzation constructs. Unfortunately, it seems to be too hard to program profitably in this paradigm for general-purpose applications. Pipeline parallelism is a programming paradigm that has proved so easy to understand that shell programmers use it even on machines that have only one thread. In this work we present the case for better support for pipeline parallelism in programming languages, and present ideas on how to make the implementation of pipeline parallelism scalable.

A Sound, Complete and Usable Hoare-Style Logic for a Sequential Java Subset

Christoph Feller
Technische Universität Kaiserslautern

Proving a Hoare-style logic sound and complete is not a new idea. In [1] we find a formalization of a rather large subset of sequential Java together with a Hoare-style logic and a proof of its soundness and completeness. Why do we do it again?

Our long-term goal is to find a way to modularize reasoning about programs. So we have to find a proper notion of modules but also a way to reason about these. An axiomatic semantics seems to be the best way to do the latter. So this work can be seen as a preparation for our long-term goals: To acquire more insight into program logics and a better understanding of the used theorem prover Isabelle/HOL.

Another aim of this work is to provide a more usable logic than [1]. That essentially means to keep the important definition and especially the rules of the logic as legible and concise as possible. As a technical detail we will show how the locale mechanism of Isabelle/HOL contributed towards this aim.

Reference

[1] David von Oheimb. Analyzing Java in Isabelle/HOL: Formalization, Type Safety and Hoare Logic. Ph.D. Thesis, Technische Universität München, Germany, 2001.

Reinventing Haskell Backtracking

Sebastian Fischer
Christian-Albrechts-Universität zu Kiel

Almost ten years ago, Ralf Hinze has written a functional pearl on how to derive backtracking functionality for the purely functional programming language Haskell. In these notes, we show how to arrive at the efficient, two-continuation based backtracking monad derived by Hinze starting from an intuitive inefficient implementation that we subsequently refine using well known program transformations.

It turns out that the technique can be used to build monads for non-determinism from modular, independent parts which gives rise to various new implementations. Specifically, we show how the presented approach can be applied to obtain new implementations of breadth-first search and iterative deepening depth-first search.

Concurrency Engineering with S-Net

Clemens Grelck
Universiteit van Amsterdam und University of Hertfordshire

The current hardware trend towards multicore processors and systems-on-a-chip will increasingly expose ordinary programmers to the notorious pitfalls of parallel programming. Since automatic parallelisation of sequential programs has long been identified as too ambitious even in a functional setting, the quest is for abstractions that hide machine-level synchronisation and communication issues on the application programming level in an intuitive way.

We present the declarative coordination language and component model S-Net. With S-Net we aim at complete separation of concerns between (sequential) application engineering and concurrency engineering.

Sequential code (existing or new) is encapsulated in S-Net boxes, which form atomic stream processing components. Their interaction in a semi-static streaming network is defined by algebraic formulae built out of four network combinators. Network integrity is statically guaranteed through a type system. Types also control the flow of data through the streaming network. Structural subtyping on the level of boxes and networks and a tailor-made inheritance mechanism facilitate the adaptation of existing boxes and networks to different instantiation contexts.

(This is joint work with Sven-Bodo Scholz and Alex Shafarenko.)

Automatische Paketisierung

Sebastian Hack
Universität des Saarlandes, Saarbrücken

„Packetization“(also: Data Parallelization or SIMDfication) is the process of transforming scalar code, given by a CFG G, into a CFG G' that works on N scalar input values at once (where N is the target-architecture's SIMD width). One execution of G' is equivalent to N parallel executing instances of G. This technique is important for data-parallel algorithms, in particular applications in computer graphics such as ray tracing.

The benefit of packetization lies in the usage of SIMD instructions. To this end, a value in the scalar instance is mapped to a SIMD "packet" in the packetized version. However, each of the scalar instances may take different control-flow paths. To avoid splitting SIMD packets apart, control flow is almost completely replaced by predicated execution (only loop-back branches have to be kept).

We present a packetization algorithm and discuss its limitations. Furthermore, we present first results of using our algorithm in the shading system of a ray tracer: the programmer writes scalar code in C/C++ that is then automatically packetized. Although still lacking some optimizations, the packetized CFGs already outperform their scalar counterparts by an average factor of 3.6 using a vector width of 4.

Set Functions for Functional Logic Programming

Michael Hanus
Christian-Albrechts-Universität zu Kiel

We propose a novel approach to encapsulate non-deterministic computations in functional logic programs. Our approach is based on set functions that return the set of all the results of a corresponding ordinary operation. A characteristic feature of our approach is the complete separation between a usually-non-deterministic operation and its possibly-non-deterministic arguments. This separation leads to the first provably order-independent approach to computing the set of values of non-deterministic expressions. Our approach solves easily and naturally problems mishandled by current implementations of functional logic languages.

Verbesserung der Modellbildung und Analyse verteilter Geschäftsprozesse durch Prozessumstrukturierung

Thomas Heinze
Friedrich-Schiller-Universität Jena

Die Web Services Business Process Execution Language (WS-BPEL) ist eine vielversprechende Modellierungssprache für verteilte Geschäftsprozesse auf Grundlage der Komposition von Web Services. Eine

wichtige Voraussetzung für einen fehlerfreien WS-BPEL-Prozess bildet die Kompatibilität der komponierten Web Services. Die Frage nach der Kompatibilität umfasst dabei nicht nur die Übereinstimmung von Schnittstellen. Insbesondere in Situationen, in denen Web Services selbst Geschäftsprozesse realisieren, muss auch sichergestellt werden, dass es während der Prozessinteraktion zu keinem unerwünschten Verhalten, beispielsweise zu Verklemmungen, kommt. In diesem Zusammenhang wird von der Verhaltenskompatibilität der interagierenden Prozesse gesprochen.

Eine geeignete Methode zur Analyse der Verhaltenskompatibilität von WS-BPEL-Prozessen beruht auf der Modellbildung durch Petrinetze. Herkömmliche Abbildungsvorschriften ignorieren jedoch aus Gründen der Analysierbarkeit des Petrinetzmodells die Datenaspekte und -abhängigkeiten der Prozesse. Als Folge dieser Abstraktion wird der bedingte Kontrollfluss durch nichtdeterministische Strukturen repräsentiert und fehlerhafte Analyseergebnisse, auch für einfache Prozesse, in Kauf genommen.

Im Vortrag wird eine Umstrukturierungstechnik für verteilte Geschäftsprozesse der Sprache WS-BPEL vorgestellt. Unter Verwendung dieser Umstrukturierungstechnik lassen sich Schleifen und Verzweigungen dann immer so transformieren, dass Daten- in Kontrollabhängigkeiten umgewandelt werden, falls deren Schleifen- oder Verzweigungsbedingungen zur Laufzeit nur auf Konstanten zugreifen. Als Resultat der Transformation ergeben sich semantisch äquivalente Prozesse, in denen die Bedingungen der Schleifen oder Verzweigungen statisch ausgewertet und entfernt werden können. Derart lässt sich die Anzahl von nichtdeterministischen Strukturen innerhalb des Petrinetzmodells reduzieren und als mögliche Ursache von Analysefehlern einschränken.

(Dies ist eine gemeinsame Arbeit mit Wolfram Amme und Simon Moser.)

Generation of Incremental Parsers for Modern IDEs

Christoph Höger
Technische Universität Berlin

Tools in modern Integrated Developments Environments (IDEs) usually work on models derived from the abstract syntax tree. Because those models should be validated and reconciled in real time, parsing the source code is subject to strict limitations in runtime and memory. In this paper we provide a Java based approach to extend a special LR Parser with methods for incremental re-parsing a given text. We also show that while incremental parsers cannot be faster than batchparsers in general (that is: for any given grammar), our implementation reaches O(log(n)) in the median. We will conclude by showing some optimizations on the approach and measuring their influence on the runtime of the parser.

Generierung von Hyperkantenersetzungsgrammatiken zur Heapabstraktion

Christina Jansen
RWTH Aachen

Bei der Verifizierung von heapbasierten, dynamischen Datenstrukturen stellen unendliche Zustandsräume ein Problem dar. Die potentiell unendliche Menge von Heapzuständen lässt sich durch Abstraktion endlich repräsentieren und mithilfe dieser Repräsentation durch Techniken wie Model Checking verifizieren. Grundlage dieser Abstraktion sind Hyperkantenersetzungsgrammatiken, wobei die Form einer geeigneten Grammatik stark von der zugrundeliegenden Datenstruktur und der Anwendung abhängt. Erfüllt sie dabei alle notwendigen Eigenschaften, bezeichnet man sie als Heapabstraktionsgrammatik. Wir möchten vorstellen, wie die Konstruktion einer solchen Heapabstraktionsgrammatik abläuft und wie das Lernen von Produktionsregeln während der Programmausführung aussehen kann.

(Dies ist eine gemeinsame Arbeit mit Jonathan Heinen.)

Combining Automated Reasoning and Algebraic Methods in Theorema

Tudor Jebelean
RISC, Johannes Kepler Universität Linz

We present some applications of the *Theorema* system to the generation of invariants for imperative loops and to automated proving in elementary analysis, which are based on the interaction of logic techniques with methods from computer algebra and from algebraic combinatorics.

The *Theorema* project (`www.theorema.org`), provides an uniform logic frame for the exploration of mathematical theories – [1], based on automatic reasoning.

The use of combinatorial and algebraic methods in conjunction with automated reasoning leads to powerful analysis tools, because they allow the automatic generation of inductive assertions for programs [4] – joint work with Laura Kovacs. The method generates all the invariants which can be represented as polynomial equations (in fact, a basis for the ideal generated by the corresponding polynomials) in two stages: first the recursive equations corresponding to the evolution of loop variables are transformed into closed formulae (depending on the loop counter) using combinatorial techniques; second these closed forms are used in successive applications of the Buchberger algorithm in order to find out the invariant ideal.

We also show how to significantly enhance the power of automatic provers [5,3] – joint work with Bruno Buchberger and Robert Vajda – in particular for reasoning in numeric domains (reals, integers) by using the CAD method (Cylindrical Algebraic Decomposition) in order to generate natural proofs in elementary analysis (the so called epsilon–delta proofs). Namely, by applying the S-Decomposition [2] logical technique we decompose the original proof problem into several numerical conjectures which involve existential quantifiers, whose witnesses are then found by CAD. This combination of techniques builds a prover with the distinctive feature that it does not need all the axioms of the underlying domain (e.g. the reals), but it automatically finds the appropriate lemmata which are necessary for completing the proof.

References

[1] Bruno Buchberger, Adrian Craciun, Tudor Jebelean, Laura Kovacs, Temur Kutsia, Koji Nakagawa, Florina Piroi, Nikolaj Popov, Judit Robu, Markus Rosenkranz, and W. Windsteiger. Theorema: Towards Computer-Aided Mathematical Theory Exploration. *Journal of Applied Logic*, 2005.

[2] Tudor Jebelean. Natural Proofs in Elementary Analysis by S-Decomposition. RISC Report 01-33, November 2001.

[3] Tudor Jebelean. Using Computer Algebra for Automated Reasoning in the Theorema System, 2005. Invited talk at Seventh Asian Symposium on Computer Mathematics (ASCM 2005).

[4] Laura Kovacs and Tudor Jebelean. Finding Polynomial Invariants for Imperative Loops in the Theorema System. In S. Autexier and H. Mantel, editors, *Proceedings of Verify'06 Workshop, IJCAR'06, The 2006 Federated Logic Conference*, pages 52–67, 2006.

[5] Robert Vajda, Tudor Jebelean, and Bruno Buchberger. Combining Logical and Algebraic Techniques for Natural Style Proving in Elementary Analysis. Mathematics and Computers in Simulation, 79 (8), pp: 2310–2316, Elsevier, 2009.

Automatic Calculation of Coverage Profiles for Coverage-based Testing

Raimund Kirner
Technische Universität Wien

Code-coverage-based testing is a widely-used testing strategy with the aim of providing a meaningful decision criterion for the adequacy of a test suite. Code-coverage-based testing is also used for the development of safety-critical applications, as the modified condition/decision coverage (MCDC) is proposed by the DO178b document.

One critical issue of code-coverage testing is that they are typically applied to source code while the generated machine code may result in a different code structure due to code optimizations performed by an compiler. In this talk we describe the automatic calculation of coverage profiles describing which structural code-coverage criteria are preserved by which code optimization. These coverage profiles allow to easily extend compilers with the feature of preserving any given code-coverage criteria by enabling only those code optimizations that preserve it.

(This is joint work with Walter Haas.)

From Trusted Annotations to Verified Knowledge

Jens Knoop
Technische Universität Wien

WCET analyzers commonly rely on user-provided annotations such as loop bounds, recursion depths, region- and program constants. This reliance on user-provided annotations has an important drawback. It introduces a Trusted Annotation Basis into WCET analysis without any guarantee that the user-provided annotations are safe, let alone sharp. Hence, safety and accuracy of a WCET analysis cannot be formally established. In this paper we propose a uniform approach, which reduces the trusted annotation base to a minimum, while simultaneously yielding sharper (tighter) time bounds. Fundamental to our approach is to apply model checking in concert with other more inexpensive program analysis techniques, and the coordinated application of two algorithms for Binary Tightening and Binary Widening, which control the application of the model checker and hence the computational costs of the approach. Though in this talk we focus on the control of model checking by Binary Tightening and Widening, this is embedded into a more general approach in which we apply an array of analysis methods of increasing power and computational complexity for proving or disproving relevant time bounds of a program. First practical experiences using the sample programs of the Mälardalen benchmark suite demonstrate the usefulness of the overall approach. In fact, for most of these benchmarks we were able to empty the trusted annotation base completely, and to tighten the computed WCET considerably.

(This is joint work with Albrecht Kadlec, Raimund Kirner, Adrian Prantl, and Markus Schordan.)

Tree Automata for Analyzing Dynamic Pushdown Networks

Peter Lammich
Westfälische Wilhelms-Universität Münster

Dynamic Pushdown Networks (DPNs) are an abstract model for concurrent programs with recursive procedures and dynamic process creation. Usually, DPNs are described with an interleaving semantics, where an execution is a sequence of steps. Recently, we introduced a true-concurrency semantics for DPNs, where executions are trees.

The standard analysis methods for DPNs are based on a saturation algorithm, that, given a set of configurations, computes the set of all predecessor configurations.

In this talk we present an alternative analysis algorithm that is based on tree automata. DPN executions as well as the properties to be analyzed are represented as tree automata, and the analysis is done by standard tree-automata algorithms for intersection and emptiness check.

MultiMediaC# – QoS-Aware Programming with C#

Oliver Lampl
Alpen-Adria Universität Klagenfurt

Providing QoS-awareness and adaptive behavior in multimedia applications is cumbersome and errorprone. Many different frameworks and even additional languages exist which try to support QoS-aware application development. In many cases this is done by shifting away QoS processing facilities from the prgrammer and hiding them inside resource layers or other middle-ware. Instead, we integrate QoS directly into the common programming language C#.

In this paper we introduce the specification of MMC# as an extension to the C# programming lanugage focusing on adaptive, QoS-aware programming. The new language features provide a solution to the three common problems in the field of QoS processing: (1) constraint declaration, (2) constraint monitoring and (3) providing adaptive behavior in multimedia applications. Furthermore, declarative definition of QoS requirements directly within the programming language allows to apply semantic constraint analysis - partly at compile-time, partly run-time - to check the correctness of specified requirements and further provide optimizations by automatically removing not required constraints. Last but not least, this paper describes the structure of the Mono compiler which has been the basis for development. We illustrate how the compiler has been modified to support these novel language extensions and the semantic analysis of QoS constraints.

(This is joint work with Laszlo Böszörmenyi.)

Technische Aspekte des erfolgreichen Testens von Software in Unternehmen

Tim A. Majchrzak
Westfälische Wilhelms-Universität Münster

Um Softwareprodukte von hoher Qualität zu erstellen, ist das Testen unerlässlich. Da es sich bei Softwaretests um eine teure Aufgabe handelt, die nur schwierig zu beherrschen ist, muss ihre organisatorische Einbettung wohlüberlegt sein. Wir haben mit regionalen Unternehmen zusammengearbeitet, um ihre individuellen Stärken und Schwächen hinsichtlich der Entwicklung und insbesondere des Testens von Software kennenzulernen. In der Folge war es uns möglich, erfolgreiche Vorgehensweisen ("best practices") abzuleiten und Empfehlungen zu formulieren. In diesem Vortrag wird das Projekt und die gewählte Forschungsmethodik vorgestellt. Danach werden fünf Empfehlungen vorgestellt, deren Fokus auf technischen bzw. technologischen Aspekten des Testens liegt. Es wird insbesondere auch berücksichtigt, welchen Einfluss Programmierpraktiken sowie -paradigmen bzw. die Wahl der Programmiersprache haben. Für jede Empfehlung wird erörtert, unter welchen Gegebenheiten sich ihre Umsetzung anbietet.

An Algebraic Framework for Modeling and Processing Hyperdocuments

Volker Mattick
Technische Universität Dortmund

The World Wide Web is dominated by hyperdocuments. Markup languages, accompanied by schema and stylsheet languages, are the leading technology in implementing hyperdocuments and they can be seen as a special kind of programming languages. The talk presents a short overview how common terms of XML-based markup languages, schemas and stylesheets can be defined precisely by well-founded algebraic methods, in particular by signatures, the resulting abstract syntax and their algebraic interpretations. It is shown how concepts developed for compilers can be adequatly adapted to process and edit hyperdocuments. Finally there is a short outlook how this techniques can be modified to support a specific kind of adaptable hyperdocuments.

Programming Support for Cell/BE Multiprocessor

Eduard Mehofer
Universität Wien

An emerging class of architectures are accelerator-based heterogeneous multiprocessors with software-managed memory hierarchies like Cell/BE. A major difficulty in programming such kind of machines are the explicit data transfers between the different memories raising new programming challenges. In this paper we discuss a programming approach which supports application programmers in writing efficient code for non-cache-based architectures. A crucial role plays the interplay between the three parties programmer, parallelization framework, and native compiler. Based on our experiences with past programming approaches, we propose language extensions to orchestrate parallel execution of threads and to control data transfers. Experiments are performed to analyze the roles of programmer and compiler in more detail.

(This is joint work with Enes Bajrović.)

Fraglets: Stochastic Programming for Provable Program Dynamics and Self-Healing Programs

Thomas Meyer
Universität Basel

In Fraglets - a chemically inspired programming language - we are able to predict and even prove the dynamic behavior of programs, despite the stochastic execution of its code. In addition to traditional symbolic computation, we exploit this execution dynamics to perform implicit computations on a macroscopic level, where information is represented by the multiplicity of entities and the rate of their production. This allows us to maintain a stable population of redundant code parts that replicate themselves, yielding intrinsic self-healing code, which is robust to the deletion of its individual parts.

Maintaining XML Data Integrity in Programs – An Abstract Datatype Approach

Patrick Michel
Technische Universität Kaiserslautern

XML technology can be supported in high level programming languages in many different ways. APIs like DOM and SAX offer support for data in the generic XML format. Data binding approaches like JAXB and language extensions like XJ offer support for the combination of XML with XML schemata. Utilizing the type system of languages to guarantee validity of XML data is one way to improve language support.

There is almost no support, however, for the combination of XML with more powerful schema languages like Schematron. Such languages allow the definition of arbitrary boolean expressions on the shape and more important the content of XML data. Tools supporting Schematron are able to check all these integrity constraints for any given document, but offer no support for the correct manipulation of such documents. Schematron also does not define any method of manipulation, but allows to define valid sets of documents only.

We propose to view XML documents with such complex constraints as abstract datatype, with an interface of schema-specific procedures. Such an interface integrates well with object-oriented languages like Java and encapsulates the alien aspects of tree-shaped data. By automatically guarding these procedures with minimal preconditions, the lasting correctness and validity of an XML document can be guaranteed.

The procedures are implemented in a language accessible to Java programmers and domain experts, incorporating abstractions of the XML domain. It is powerful enough to define basic atomic manipulations that can leave valid documents in a valid state. An automated analysis is able to derive minimal preconditions that check if a procedure is applicable to a valid document. Failure of these preconditions can be dealt with by the Java programmer, for example, by changing parameters or calling other procedures.

Together with typical data binding approaches, which allow the type correct and convenient navigation and reading on documents, our approach offers comprehensive support for XML data with complex integrity constraints within languages like Java.

Lambdas und Schleifen in monotonen Logikprogrammen

Ulrich Neumerkel
Technische Universität Wien

Lambda-Abstraktionen und Schleifenkonstrukte sind in der Logikprogrammierung bisher kaum aufgenommen worden. Diese Zurückhaltung hängt unmittelbar mit den algebraischen Eigenschaften der verwendeten Konstrukte zusammen. Die bisher vorgeschlagenen Ansätze verletzen grundlegende Eigenschaften wie die der Monotonie, wodurch nicht nur eine deklarative Betrachtungsweise verunmöglicht wird und diagnostische Verfahren stark eingeschränkt werden, sondern auch effiziente Implementierungen behindert werden.

Mit der vorgestellten Umsetzung von Lambda-Ausdrücken werden die meisten Mängel unmittelbar vermieden. Es eröffnen sich nun neue Möglichkeiten monotone Schleifenkonstrukte einzubinden.

Functional Program Verification in Theorema. Soundness and Completeness

Nikolaj Popov
RISC, Johannes Kepler Universität Linz

We present a method for verifying recursive functional programs. We define a Verification Condition Generator (VCG) which covers the most frequent types of recursive programs (including nested recursive and mutual recursive ones). These programs may operate on arbitrary domains. Soundness and Completeness of the VCG are proven on the meta level, and this provides a warranty that any system based on our results will be sound.

The research is performed in the frame of the Theorema system, a mathematical computer assistant which aims at supporting the entire process of mathematical theory exploration.

(This is joint work with Tudor Jebelean.)

Towards a Static Profiler

Adrian Prantl
Technische Universität Wien

The profiler is an important tool to identify performance bottlenecks in programs. A traditional profiler works by compiling a specially instrumented version of the target program that collects performance data over a series of program executions with typical input data. The collected data is then interpreted and presented to the user as tables and execution graphs.

Since it is up to the user to supply the program with a representative set of input data during the profiling run, it is often left to chance that performance critical paths are triggered during profiling. While this is acceptable for a user interested in the average performance of the program, it is not purposeful for investigating the program's worst-case behaviour; a property especially important to developers of real-time and embedded systems.

In this talk we present the ingredients for a *static profiler* which visualizes an unfolded interprocedural control flow graph by using the results from multiple static analyzers that identify feasible paths and provide upper bounds for loop constructs. The resulting graph is guaranteed to always include the worst-case behaviour of the program.

Our prototype implementation was built using SATIrE program analysis framework as frontend and can analyze C programs. The graph calculation and flow analysis was implemented in Prolog using the Termite library.

References

– SATIrE: `http://www.complang.tuwien.ac.at/satire/`
– Termite: `http://www.complang.tuwien.ac.at/adrian/termite/`

How to Specify the Flow of Data Accessibility: An OO Way of Concurrent Programming

Franz Puntigam
Technische Universität Wien

Program structures appropriate for concurrency are often in conflict with object-oriented principles. Especially average programmers need high-level language constructs for concurrency with good integration into the object-oriented paradigm. A key concept in this respect is better static knowledge of the data flow. We propose to explicitly specify the flow of data accessibility in a program. This information is sufficient for a compiler to automatically spawn threads and add synchronization as necessary. Programmers regard the specifications just as assertions.

Towards a Parallel Search for Solutions of Non-deterministic Computations

Fabian Reck
Christian-Albrechts-Universität zu Kiel

Non-determinism is one of the distinctive built-in features of logic and functional programming languages such as Prolog and Curry. In other programming languages non-determinism can be modeled using appropriate abstractions. In Haskell this is usually done by means of non-determinism monads. Since the different results of a non-deterministic computation do not depend on each other it should be possible to compute them in parallel. In this talk we explore different possibilities to execute non-deterministic Haskell programs in parallel. Together with our latest attempts to compile Curry programs to monadic Haskell we already achieve preliminary but encouraging results.

We present three different implementations of a parallel execution of non-deterministic monadic Haskell programs:

1. dividing the computation into a fixed number of sub-computations and assign each of them to a different thread,
2. using a Bag-of-Tasks approach to distribute small tasks to a fixed number of threads, and
3. using the `par`-combinator that is implemented in the GHC to give *hints* to the Haskell runtime system about which parts of the computation can be done in parallel.

A detailed description of these approaches together with a discussion of their properties and selected benchmarks is published in the proceedings of the 39th annual conference of German Gesellschaft für Informatik [1].

(This is joint work with Sebastian Fischer.)

Reference

[1] Fabian Reck and Sebastian Fischer. Towards a Parallel Search for Solutions of Non-deterministic Computations. In: INFORMATIK 2009, Im Focus das Leben, 39. Jahrestagung der Gesellschaft für Informatik. LNI, GI (2009).

Rekursionspräzise Intervallanalysen

Dirk Richter
Martin-Luther-Universität Halle-Wittenberg

Intervallanalysen bestimmen zu einem gegebenem Programm konservative oder nicht-konservative Wertebereiche von Variablen. Desto genauer diese Wertebereiche bestimmt werden können, desto präziser sind die darauf basierenden Analysen wie z.B. die Laufzeitschätzung von Programmen bzw. führen zu weniger Fehlalarmen wie z.B. bei der Prüfung auf Feldzugriffe auѣerhalb zulässiger Indizees (ArrayOutOfBounds). Bei unbeschränkter Rekursion und unbeschränktem Speicher (Heap/Halde) ist das Problem der Bestimmung von Wertebereichen maximaler Genauigkeit (auch als exakte Wertebereiche bezeichnet) unentscheidbar. Im Folgenden wird gezeigt, wie bei unbeschränkter Rekursion und beschränktem Speicher stets derartige exakte Wertebereiche in komplexitätstheoretisch optimaler Zeit bestimmt werden können.

So konnten mit unserer Methode in Experimenten durch Speicherbegrenzung in Java-Programmen mittels des Tools JMoped sehr genaue Wertebereiche der Variablen bestimmt werden. Die Java-Programme werden dazu zunächst in ein Rekursionsmodell (Kellersystem) überführt, welches präzise das rekursive Verhalten des Programms beschreibt. Für diese Rekursionsmodelle können exakte Wertebereiche berechnet werden.

Je nach Art der Modellgenerierung lassen diese Wertebereiche im Modell Rückschlüsse auf die Wertebereiche des ursprünglichen Java-Programms zu. Da die Bestimmung exakter Wertebereiche (nicht nur komplexitätstheoretisch) aufwändig ist, sind möglichst kleine Modelle für die Durchführbarkeit entscheidend. Derartige Verkleinerung von Modellen ist Gegenstand früherer Veröffentlichungen. Die Modellgenerierung mittels JMoped ermöglicht die Modellierung ganzzahliger Datentypen mit nur wenigen Bits und verkleinert somit auch den adressierbaren Speicher (in welchem sich die zur Laufzeit erzeugten Objekte befinden), was bereits in 80% (191 von 240) der untersuchten Fälle genügte. Eine Modell-Verkleinerung bzw. -Verbesserung führt in diesen Fällen dann nur noch zu einer Beschleunigung der rekursionspräzisen Intervallanalyse.

A New Look on Data Parallelism: Space vs. Time

Sven-Bodo Scholz
University of Hertfordshire

Data Parallelism often is used synonymously with “forall”-loops or “map”-functions. This leads to a perception of Data Parallelism as being a niche approach, mainly suitable for autoparallelisation attempts in the context of scientific codes written in Fortran or similar languages.

In this talk we try to present a new look at Data Paralellism which suggests that Data Parallelism as programming model has far wider applicability. Furthermore, it seems that it is inherently better suited than many other approaches when it comes to generating code for various different modern multicore architectures.

Statische Erkennung von semantischen Feature-Interaktionen

Wolfgang Scholz
Universität Passau

Interaktionen zwischen verschiedenen Programmteilen erschweren die Entwicklung komplexer Softwaresysteme. Feature-orientierte Softwareentwicklung hat das Ziel, Programmfeatures in verschiedenen Kombinationen zu einem korrekten Programm zusammenzufügen, und bietet damit aussichtsreiche Möglichkeiten für die modulare Programmentwicklung. Die Erkennung und Behebung von ungewollten Feature-Interaktionen wird bisher manuell vorgenommen und ist ein aufwändiger Prozess, da er detailliertes Wissen über die Implementierung aller involvierten Features erfordert. Um die Effektivität von Feature-orientierter Programmierung zu erhöhen ist es daher notwendig, Werkzeuge zur Verfügung zu stellen, die in der Lage sind, semantische Feature-Interaktionen aufzudecken.

Ziel des Projekts ist es, auf der Basis des Eclipse-Plugins FeatureIDE und der Verifikationsplattform Why ein Programmierwerkzeug zu entwickeln, mit dem es möglich ist, eine bestimmte Klasse von semantischen Feature-Interaktionen in Java-Software-Produktlinien statisch bei der Kompilation zu erkennen.

ARAL: a Language for Information Exchange between Program Analysis Tools

Markus Schordan
Fachhochschule Technikum Wien

The analysis results annotation language (ARAL) is designed to be a general format that allows the exchange of analysis information between program analysis tools either in a separate file or through program annotations. It is suitable for representing flow-sensitive and context-sensitive information and aims at supporting a

wide range of analyses with focus on data-flow analysis and abstract interpretation in general. For example, the language is general enough to represent any analysis information that is computed by AbsInt's Program Analyzer Generator (PAG). Beside exchange of analysis information, the purpose of the language also is the support of testing analyzers and allowing to manually add annotations, as is important for worst-case execution time analysis. The development of ARAL is motivated by the need of making analysis results persistent for enabling whole-program analysis of very-large software, and the goal of information exchange between program analysis tools in the ALL-TIMES project.

Robots, Software, Mayhem?

Towards a Design Methodology for Robotic Software Systems

Dietmar Schreiner
Technische Universität Wien

The development of autonomous robotic systems has experienced a remarkable boost within the last years. Away from stationary manufacturing units, current robotic systems have grown up into autonomous, mobile systems that not only interact with real world environments, but also fulfill mission critical tasks in collaboration with human individuals on a reliable basis. Therefore, todays robotic systems can be described as highly reactive, inherent parallel, distributed real-time systems. Our work aims at an improved software development methodology, that on the one hand allows high level development of certifiable robotic software, but on the other hand is capable of synthesizing optimized low-level code for robust concurrent real-time environments.

A Formalisation of the OSEK Concurrency Model

Martin Schwarz
Technische Universität München

The OSEK specification was created to increase portability of applications written for embedded systems with interrupt-driven concurrency. Despite of the complex control-flow due to scheduling and interrupts, programs to be executed on an OSEK-compliant system are meant to exhibit an essentially sequential behaviour. Our goal is to make this explicit and exploit it for transferring techniques for the analysis of sequential programs to OSEK programs. Building on that, we define a notion of races for interrupt-driven programs and provide a precise algorithm for detecting all races.

The OSEK specification defines a unified operating system (OSEK OS), which executes the tasks and interrupts of OSEK programs in a well-defined manner and provides a set of library functions for resource management and scheduling. The basic scheduling policy of OSEK OS is to work through the activated tasks in priority order. Once started, a task will run to termination unless a task of higher priority is activated and pre-empts it, i.e. no time-slicing is used. This property allows the translation of OSEK programs to a sequential, stack-based execution model.

For synchronisation the Priority Ceiling Protocol (PCP) is used. The key idea of the PCP is to raise the priority of a task which has acquired a resource to the highest priority of all tasks declared to use that resource. This policy ensures dead-lock freedom and minimises priority inversion, where a high-priority task waits for a lower-priority task to release a resource.

Lazy Continuations for Java Virtual Machines

Lukas Stadler
Johannes Kepler Universität Linz

Continuations, or 'the rest of the computation', are a concept that is most often used in the context of functional and dynamic programming languages. Implementations of such languages that work on top of the Java virtual machine (JVM) have traditionally been complicated by the lack of continuations because they must be simulated.

This talk will present our implementation of continuations in the Java virtual machine with a lazy or on-demand approach. Our system imposes zero run-time overhead as long as no activations need to be saved and restored and performs well when continuations are used. Although our implementation can be used from Java code directly, it is mainly intended to facilitate the creation of frameworks that allow other functional or dynamic languages to be executed on a Java virtual machine.

Along with some preliminary performance numbers this talk will also feature a discussion on how the concept of continuations fits into the Java world and which problems and needs it addresses.

Communication-based System Development using Standard Programming Languages

Annette Stümpel
Universität zu Lübeck

A specification of a system first describes the black box view which concentrates on the externally observable behaviour characterized by the communication of the system components. Software developers employ

communication-based graphical models such as message sequence charts or UML sequence diagrams for these communication-based specifications. Stream processing constitutes a formal model for communication-based specifications. Stream functions describe the input / output behaviour of components and can be composed to larger systems.

However, there is a gap between the communication-based specification and the implementation: the implementation of such systems is often not constructed from the communication-based specification. With some programming languages, even some standard ones, it is possible to mould the communication-based specification into a specification which clearly reflects the underlying communication-based specification. We show how to achieve that with a lazy functional language, Haskell, and an object oriented language with threads, Java.

We show under which conditions and how Haskell's lazy lists can be used to implement a communication-based specification in a straightforward way.

Furthermore we present our STREAMS tool which supports the simulation of communication-based systems in Java: the components as well as the communication channels run as concurrent processes.

FQL: A Query Language for Program Testing

Michael Tautschnig
Technische Universität Darmstadt

Coverage criteria for white box testing naturally fall into two groups: The first group are generic off-the-shelf criteria such as basic block coverage and condition coverage which are applied in the same uniform way to different source code. Although generic criteria are supported by industrial strength tools, simple experiments reveal that the tools disagree on standard notions such as condition coverage. The second group are code specific coverage criteria which are either defined on the fly during program development, or consciously designed to reflect requirements specific to the application and architecture. For the second group, there is a notable lack of tool support. In this paper, we aim to solve the problems of both groups – lack of precision and lack of adequate tool support – in a unified framework: We describe a specification language which designates coverage targets in the source code as building blocks for formal coverage criteria. On the one hand, our language FQL facilitates the precise specification of coverage criteria on the basis of a clean and intuitive semantics; on the other hand, we show how to apply the query-driven program testing paradigm presented at VMCAI 2009 for efficient test case generation with the help of a model checker. Experimental results demonstrate the practical feasibility of our framework.

(This is joint work with Andreas Holzer, Christian Schallhart und Helmut Veith.)

Type-Safe Bytecode Generation in Scala

Peter Thiemann
Universität Freiburg

Mnemonics is a novel bytecode generation library for the Java Virtual Machine (JVM), written in Scala. It avoids the generation of ill-formed bytecode sequences by adopting a typed representation of the stack and the local variables. Individual instructions are modeled as functions that transform these types according to the action of the instruction. The library exploits a number of advanced features of Scala's type system (type inference with bounded polymorphism, objects with type members, implicit parameters, and reflection) to guarantee that the compiler rejects illegal combinations of instructions at compile time.

Programmierung als Leitbild in der Theorie der Ökosysteme

Baltasar Trancón y Widemann
Universität Bayreuth

Die Umweltwissenschaften haben im Lauf ihrer Entstehung als Disziplin das Forschungsprogramm der klassischen Physik mit dem Paradigma der mechanistischen Systeme weitgehend unhinterfragt übernommen. Es zeigt sich, dass dieses Vorgehen umso problematischer ist, je mehr lebende Bestandteile das Verhalten des Systems dominieren. Andererseits gibt es zahlreiche alte, gesellschaftlich hochrelevante und erfolgreiche Nutzungstraditionen von Umweltsystemen, die theoriefrei und rein heuristisch funktionieren. Als Folge entsteht eine im Bereich der Naturwissenschaften einmalige Spannung zwischen Theorie und Praxis, und es zeigen sich Symptome einer wissenschaftlichen Krise, wie sie oft nur durch einen Paradigmenwechsel zu lösen ist. Dieser Beitrag soll dreierlei zeigen: Erstens, dass aktuelle Entwicklungen im Bereich der computergestützten Umweltmodellierung sich dem Informatiker als Krisensymptome darstellen, die dem „laienhaften Anwender“verschlossen bleiben. Zweitens, dass Begriffe aus der Softwaretechnik und der Programmanalyse geeignet sind, die Rolle von Theorie und Modellierung von Ökosystemen metaphorisch zu beschreiben. Drittens, dass sich Formalismen aus dem Bereich der Semantik von Programmiersprachen und -kalkülen ganz konkret als Basis eines alternativen Paradigmas der Theorie der Ökosysteme eignen.

Fraglets: Chemical Programming with a Packet Prefix Language

Christian Tschudin
Universität Basel

In a computer network, the data packet header “instructs” the receiving router or end node what it has to do with the packet. We generalize this concept and construct a prefix programming language where each packet becomes a computation fragment (fraglet) for distributed computations and protocols. In this talk we will introduce this programming model, show its links to string rewriting systems as well as the chemical programming metaphor, and discuss some key design decisions of the Fraglets language. The dynamic aspect of chemical programs will be covered in a second talk.

(This is joint work with Thomas Meyer.)

Adding Weights to Dynamic Pushdown Networks

Alexander Wenner
Westfälische Wilhelms-Universität Münster

We develop a generic framework for the analysis of programs with recursive procedures and dynamic process creation. To this end we combine the approach of weighted pushdown systems (WPDS) with the model of dynamic pushdown networks (DPN). The resulting model, weighted dynamic pushdown networks (WDPN), describes processes running in parallel, each of them being able to perform pushdown actions, that may spawn new processes as a side effect. As with WPDS, transitions are labelled by weights to carry additional information. Starting from techniques for WPDS and DPN, we derive a method to determine meet-over-all-paths values for the paths between regular sets of configurations of a WDPN. Using this method we are able to solve basic dataflow analysis problems in a parallel context.

Towards Dynamic Code Evolution for the HotSpot VM

Thomas Würthinger
Johannes Kepler Universität Linz

Dynamic code evolution is a technique to change the source code of a running program. The current hotswapping mechanism in the Java HotSpot VM only allows changing the bodies of methods at runtime. We are working on an approach that allows arbitrary changes. This talk is about our experiences implementing a prototype in HotSpot that supports adding and deleting of methods and fields as well as performing changes to the class hierarchy. What are the conditions for dynamic code evolution to have a clear semantics and what kind of changes should be forbidden? What are meaningful applications of dynamic code evolution for Java and their requirements? The talk will conclude with ideas for future work to make dynamic code evolution stable and performant.

On Undecidability Results of Real Programming Languages

Wolf Zimmermann
Martin-Luther Universität Halle-Wittenberg

Often, it is argued that some problems in data-ow analysis such as e.g. worst case execution time analysis are undecidable (because the halting problem is) and therefore only a conservative approximation of the desired information is possible. In this paper, we show that the semantics for some important real programming languages – in particular those used for programming embedded devices – can be modeled as finite state systems or pushdown machines. This implies that the halting problem becomes decidable and therefore invalidates popular arguments for using conservative analysis. The paper shows some exact program analyses directly base on the above semantics.

(This is joint work with Raimund Kirner and Dirk Richter.)

The Reachability-Bound Problem

Florian Zuleger
Technische Universität Darmstadt

We define the reachability-bound problem to be the problem of finding a symbolic worst-case bound on the number of times a given control location inside a procedure is visited in terms of the inputs to that procedure. This has applications in bounding resources consumed by a program such as time, memory, network- traffic, power, as well as estimating quantitative properties (as opposed to boolean properties) of data in programs, such as amount of information leakage or uncertainty propagation.

Our approach to solving the reachability-bound problem brings together two very different techniques for reasoning about loops in an effective manner. One of these techniques is an abstract- interpretation based iterative technique for computing precise disjunctive invariants (to summarize nested loops). The other technique is a non-iterative proof-rules based technique (for loop bound computation) that takes over the role of doing inductive reasoning, while deriving its power from use of SMT solvers to reason about abstract loop-free fragments.