INFORMATIK-ARBEITSTAGUNG
AB Programmiersprachen und Übersetzer
Institut für Computersprachen

Invitation

to the *Joint CompLang/RISC Workshop on*

Timing Analysis and Symbolic Computation (TASCo 2009)

4 - 5 February 2009

Library E185.1, Argentinierstr. 8, 4th Floor (Centre)

Workshop Programme

Wednesday, 4 Feb. 2009

10:00 Welcome and Opening

10:15 **Development of Infrastructures for Automatic Program Analysis**
Markus Schordan, University of Applied Sciences Technikum Wien, Vienna, Austria.

11:15 **A Constraint-based Loop Analysis for TuBound**
Adrian Prantl, Vienna University of Technology, Vienna, Austria.

Lunch

14:00 **Functional Program Verification in *Theorema*. Recent Achievements and Perspectives**
Nikolaj Popov and Tudor Jebelean, RISC-Linz, Hagenberg, Austria.

15:15 **Combining Automated Reasoning and Algebraic Methods in *Theorema***
Tudor Jebelean, RISC-Linz, Hagenberg, Austria.

16:30 **Forward Symbolic Execution for Program Verification in *Theorema* System**
Mădălina Eraşcu and Tudor Jebelean, RISC-Linz, Hagenberg, Austria.

17:45 Plenary Session

Workshop Dinner

Thursday, 5 Feb. 2009

09:00 **Towards Automatic Verification of Structural Code-Coverage Preservation**
Raimund Kirner, Vienna University of Technology, Vienna, Austria.

10:15 **Specification, Verification and Synthesis of Tail Recursive Programs in *Theorema***
Nikolaj Popov and Tudor Jebelean, RISC-Linz, Hagenberg, Austria.

11:30 Wrap-up Session

Lunch

14:00 Open Discussion Session and Farewell

Zu dieser Arbeitstagung lädt der *Arbeitsbereich für Programmiersprachen und Übersetzer am Institut für Computersprachen* herzlich ein.

Abstracts of Presentations

Development of Infrastructures for Automatic Program Analysis

Markus Schordan
University of Applied Sciences Technikum Wien
schordan@technikum-wien.at

As the volume of existing software in the industry grows at a rapid pace, the problems of understanding, maintaining, and developing software assume great significance. A strong support for analysis of programs is essential for a practical and meaningful solution to such problems. To be able to analyze such software systems, powerful tools are required that can handle the complexity of popular languages such as C++, Java, and C#. We present an approach for combining analysis and transformation tools that enables their application to popular programming languages without extending existing compilers.

The presented Static Analysis Tool Integration Engine (SATIrE) aims at integrating a broad range of analysis tools by providing additional gap-filling components, such that the selection of an arbitrary tool chain most suitable for a certain program analysis or manipulation task becomes feasable. The integrated tools are the LLNL-ROSE source-to-source infrastructure, the Program Analyzer Generator from AbsInt for abstract interpretation, and the language Prolog for manipulating terms representing C/C++ programs. Analysis results are made available as annotations of a common high-level intermediate representation and as generated source code annotations. We also support an external file format of the intermediate representation, allowing a tight integration with external tools.

A Constraint-based Loop Analysis for TuBound

Adrian Prantl
Vienna University of Technology
adrian@complang.tuwiena.ac.at

The safety of our day-to-day life depends crucially on the correct functioning of embedded software systems which control the functioning of more and more technical devices. Many of these software systems are time-critical. Hence, computations performed need not only to be correct, but must also be issued in a timely fashion. Worst case execution time (WCET) analysis is concerned with computing tight upper bounds for the execution time of a system in order to provide formal guarantees for the proper timing behaviour of a system. State-of-the-art WCET analysis tools rely on supporting analyses and manual annotations to provide them with information on the execution behaviour of the program such as loop bounds or maximum recursion depths. Typically, both steps are performed on the binary code of the program. The manual annotation of a binary program, however, imposes high demands on the programmer [1].

With TuBound, we are providing an improved work-flow by lifting manual annotations and supporting analyses to the source code level of a program. The information computed on this level and annotated in the code is then conjointly transformed throughout the compilation and optimization of the program to the binary code level to make it accessible to the WCET analysis component of our TuBound tool [2].

In this talk, we highlight the static program analysis component of TuBound at whose heart is an interprocedural interval analysis and an approach to loop analysis that is based on constraint logic programming [3].

References

[1] Raimund Kirner, Jens Knoop, Adrian Prantl, Markus Schordan, and Ingomar Wenzel. WCET Analysis: The Annotation Language Challenge. In *Proceedings 7th Int. Workshop on Worst-Case Execution Time Analysis* (*WCET 2007*), 83 - 99, Pisa, Italy, 2007.

[2] Adrian Prantl, Markus Schordan, and Jens Knoop. TuBound A Conceptually New Tool for Worst-Case Execution Time Analysis. In *Proceedings 8th Int. Workshop on Worst-Case Execution Time Analysis* (*WCET 2008*), 141-148, Prague, Czech Republic, 2008. ISBN: 978-3-85403-237-3.

[3] Adrian Prantl, Jens Knoop, Markus Schordan, and Markus Triska. Constraint solving for high-level WCET analysis. In *Proceedings of the 18th Int. Workshop on Logic-based methods in Programming Environments* (*WLPE 2008*), Udine, Italy, 2008.

Functional Program Verification in *Theorema*. Recent Achievements and Perspectives

Nikolaj Popov **and Tudor Jebelean**
RISC–Linz
{popov,Tudor.Jebelean}@risc.uni-linz.ac.at

We present an environment designed for proving total correctness of recursive functional programs.

As usual, correctness is transformed into a set of first-order predicate logic formulae – verification conditions. As a distinctive feature of our method, these formulae are not only sufficient, but also necessary for the correctness [2].

We demonstrate our method on several examples and show how correctness of those may be proven fully automatically.

In fact, even if a small part of the specification is missing – in the literature this is often a case – the correctness cannot be proven. Furthermore, a relevant counterexample may be constructed automatically [3].

A specialized strategy for proving termination of recursive functional programs is developed [5]. The detailed termination proofs may in many cases be skipped, because the termination conditions are reusable and thus collected in specialized libraries. Enlargement of the libraries is possible by proving termination of each candidate, but also by taking new elements directly from existing libraries.

During the talk, we emphasize on the most recent achievements we have made, and in particular verification of functions defined by mutual recursion and functions containing nested recursion [4].

Our work is performed in the frame of the *Theorema* system [1], which is a mathematical computer assistant aiming at supporting all the phases of mathematical activity: construction and exploration of mathematical theories, definition of algorithms for problem solving, as well as experimentation and rigorous verification of them. Moreover, the logical verification conditions can be passed to the automatic provers of the system. *Theorema* includes a collection of general as well as specific provers for various interesting domains (e.g., integers, sets, reals, tuples, etc.).

References

[1] B. Buchberger, A. Craciun, T. Jebelean, L. Kovacs, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, and W. Windsteiger. Theorema: Towards Computer-Aided Mathematical Theory Exploration. *Journal of Applied Logic*, 2005.

[2] T. Jebelean, L. Kovacs, and N. Popov. Experimental Program Verification in the Theorema System. *Int. Journal on Software Tools for Technology Transfer* (*STTT*), 2008. To appear.

[3] N. Popov and T. Jebelean. A Prototype Environment for Verification of Recursive Programs. In Z. Istenes, editor, *Proceedings of FORMED'08*, pages 121–130, March 2008. To appear as ENTCS volume, Elsevier.

[4] N. Popov and T. Jebelean. Verification of Functional Programs Containing Nested Recursion. In B. Buchberger, T. Ida and T. Kutsia, editors, *Proceedings of SCSS'08*, pages 163–175, Hagenberg, Austria, July 2008.

[5] N. Popov and T. Jebelean. Proving Termination of Recursive Programs by Matching Against Simplified Program Versions and Construction of Specialized Libraries in Theorema. In D. Hofbauer and A. Serebrenik, editors, *Proceedings of 9th International Workshop on Termination* (*WST'07*), pages 48–52, Paris, France, June 2007.

Combining Automated Reasoning and Algebraic Methods in *Theorema*

Tudor Jebelean
RISC–Linz
Tudor.Jebelean@risc.uni-linz.ac.at

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 a 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] B. Buchberger, A. Craciun, T. Jebelean, L. Kovacs, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, and W. Windsteiger. Theorema: Towards Computer-Aided Mathematical Theory Exploration. *Journal of Applied Logic*, 2005.

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

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

[4] L. Kovacs and T. 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] R. Vajda, T. Jebelean, and B. Buchberger. Combining Logical and Algebraic Techniques for Natural Style Proving in Elementary Analysis. *Mathematics and Computers in Simulation*, 2008 (in print).

Forward Symbolic Execution for Program Verification in *Theorema* System

***Mădălina Eraşcu* and Tudor Jebelean**
RISC–Linz
{merascu,Tudor.Jebelean}@risc.uni-linz.ac.at

We present a static analysis method for imperative program verification based on forward symbolic execution; given the Hoare triple (Input Specification, Program Body, Output Specification), we want to check whether the program fulfils its specification. The problem of generating the verification conditions is approached using an axiomatic calculus characterizing inference rules for each statement encountered in the program: assignments (including recursive calls), conditionals and abrupt statements (`Return`). `While` loops can be simulated using conditionals and recursion. Detailed theoretical aspects of this method are stated in [1]. The method is implemented in a prototype framework on top of the computer algebra system Mathematica and uses the existing Theorema imperative language. Our goal is to automatically prove/disprove the verification conditions generated using logical, algebraic and combinatorial techniques. At this aim, we combined logical (natural deduction) and simple algebraic inferences for preprocessing the verification conditions. For further reasoning about the resulting formulae, we will use polynomial algebra algorithms which might (e.g. Cylindrical Algebraic CAD Decomposition works on the theory of real closed fields) or might not (e.g. Groebner basis algorithms works on a commutative ring with 1) need to set an underlying theory. Although CAD method is powerful enough for handling our formulae, it has a high complexity and therefore we avoid to use it until the latest. We will de ne classes of verification conditions which can be handled by other means and, if possible, hold also in weaker theories than reals.

References

[1] M. Eraşcu and T. Jebelean. Practical Program Verification by Forward Symbolic Execution: Correctness and Examples. In B. Buchberger, T. Ida, T. Kutsia, editors, *Proceedings of Austrian-Japan Workshop on Symbolic Computation in Software Science*, pages 47-56, 2008.

Towards Automatic Verification of Structural Code-Coverage Preservation*

Raimund Kirner
Vienna University of Technology
raimund@vmars.tuwien.ac.at

Embedded Systems are ofen used in safty-critical environments. Thus, thorough testing of them is mandatory. To achieve a required structural code-coverage criterion it is beneficial to derive the test data at a higher program-representation level than machine code. Higher program-representation levels include, besides the source-code level, languages of domain-specific modeling environments with automatic code generation. This enables for a testing framework with automatic test-data generation to achieve high retargetability.

Within the project "Sustaining Entire Code-Coverage on Code Optimization" (SECCO) we address the challenge of ensuring that the structural code coverage achieved at a higher program representation level is preserved during the code generations and code transformations down to machine code [1,2]. We define the formal properties that have to be fulfilled by a code transformation to guarantee preservation of structural code coverage. Based on these properties we will formalize code transformations to automatically prove whether a given code coverage preserves the code coverage of interest.

* The research leading to these results has received funding from the Austrian Science Fund (Fonds zur Förderung der wissenschaftlichen Forschung) within the research project "Sustaining Entire Code-Coverage on Code Optimization" (SECCO) under contract P20944-N13.

References

[1] Raimund Kirner. Towards preserving model coverage and structural code coverage. *Submitted to the EURASIP Journal on Embedded Systems*, 2008. Research report 49/2008.

[2] Raimund Kirner and Susanne Kandl. Test coverage analysis and preservation for requirements-based testing of safety-critical systems. *ERCIM News*, (75):40-41, Oct. 2008.

Specification, Verification and Synthesis of Tail Recursive Programs in *Theorema*

Nikolaj Popov **and Tudor Jebelean**
RISC–Linz
{popov,Tudor.Jebelean}@risc.uni-linz.ac.at

We describe an innovative method for proving total correctness of tail recursive programs having a specific structure, namely programs in which an auxiliary tail recursive function is driven by a main nonrecursive function, and only the specification of the main function is provided.

The specification of the auxiliary function is obtained almost fully automatically by solving coupled linear recursive sequences with constant coefficients [3].

The process is carried out by means of CA (Computer Algebra) and AC (Algorithmic Combinatorics). We demonstrate this method on an example involving polynomial expressions.

Furthermore, we develop a method for synthesis of recursive programs for computing polynomial expressions of a fixed degree by means of "cheap" operations e.g., additions, subtractions and multiplications. For a given polynomial expression, we define its recursive program in a schemewise manner [5].

The correctness of the synthesized programs follows from the general correctness of the synthesis method, which is proven once for all, using the verification method developed in [4].

References

[1] B. Buchberger, A. Craciun, T. Jebelean, L. Kovacs, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, and W. Windsteiger. Theorema: Towards Computer-Aided Mathematical Theory Exploration. *Journal of Applied Logic*, 2005.

[2] N. Popov and T. Jebelean. Proving Termination of Recursive Programs by Matching Against Simplified Program Versions and Construction of Specialized Libraries in Theorema. In D. Hofbauer and A. Serebrenik, editors, *Proceedings of 9th International Workshop on Termination* (*WST'07*), pages 48–52, Paris, France, June 2007.

[3] L. Kovacs, N. Popov, and T. Jebelean. Verification Environment in Theorema. *Annals of Mathematics, Computing and Teleinformatics* (*AMCT*), 1(2):27–34, 2005.

[4] N. Popov. *Functional Program Verification in Theorema*. PhD thesis, RISC, Johannes Kepler University Linz, Austria, July 2008.

[5] N. Popov and T. Jebelean. Using Computer Algebra Techniques for the Specification, Verification and Synthesis of Recursive Programs. *Mathematics and Computers in Simulation*, 2008. To appear.

About the Speakers

- **Mădălina Eraşcu, M.Sc.**
  Research Institute for Symbolic Computation (RISC)
  Johannes Kepler University of Linz
  A-4040 Linz, Austria

  merascu@risc.uni-linz.ac.at
  http://www.risc.uni-linz.ac.at/home/merascu

  Mădălina Eraşcu is a 1st year PhD student in the Theorema research group (leader: Bruno Buchberger) at the Research Institute for Symbolic Computation (RISC), Johannes Kepler University of Linz, Austria. She is interested in program analysis using formal methods, computer algebra and automated theorem proving. She holds a MSc from Johannes Kepler University (International School for Informatics), Linz (www.risc.jku.at).

- **Prof. Dr. Tudor Jebelan**
  Research Institute for Symbolic Computation (RISC)
  Johannes Kepler University of Linz
  A-4040 Linz, Austria

  Tudor.Jebelean@risc.uni-linz.ac.at
  http://www.risc.uni-linz.ac.at/home/tjebelea/

  Tudor Jebelean is an Associate Professor in the Theorema group (leader: Bruno Buchberger and Tudor Jebelean) at the Research Institute for Symbolic Computation (RISC), Johannes Kepler University of Linz, Austria. His main research interests are automated reasoning and program verification using logic and algebraic methods. He holds a PhD in Computer Science and Habilitation title from Johannes Kepler University of Linz.

- **Dr. Raimund Kirner**
  Institute of Computer Engineering
  Vienna University of Technology
  A-1040 Vienna, Austria

  raimund@vmars.tuwien.ac.at
  https://ti.tuwien.ac.at/rts

  Univ. Assistent Dr. Raimund Kirner received his Master's degree and his PhD degree at the Vienna University of Technology (TU Vienna) in 2000 respectively 2003. During this time he has been working as a research and teaching assistant at the Institut für Technische Informatik at TU Vienna. The main focus of Kirner's research is worst-case execution time analysis of real-time programs, including compiler support and design methodologies to make systems predictable. He has published several papers on WCET analysis and was involved in two projects funded by the European Commission (SETTA, NEXT TTA). From 2003-2005 Raimund Kirner has worked on the FIT-IT project MoDECS, and from 2005-2007 he worked on the FIT-IT project TeDES, both funded by the Federal Ministry of Transport, Innovation, and Technology (BMVIT). Currently, Raimund Kirner is principal investigator of the following projects: "Compiler-Support for Timing Analysis" (COSTA), "Formal Timing Analysis Suite" (FORTAS), and "Sustaining Entire Code-Coverage on Code Optimization" (SECCO). He is a member of the IEEE Computer Society, the ACM, and the Austrian Computer Society (OCG).

- **Dr. Nikolaj Popov**
  Research Institute for Symbolic Computation (RISC)
  Johannes Kepler University of Linz
  A-4040 Linz, Austria

  popov@risc.uni-linz.ac.at
  http://www.risc.uni-linz.ac.at/home/npopov

  Nikolaj Popov is a postdoctoral researcher in the Theorema group of Prof. Bruno Buchberger and Prof. Tudor Jebelean, at the Research Institute for Symbolic Computation, University of Linz, Austria.

  His research deals with the development of a relevant theory for proving correctness of recursive programs in an automatic manner. His particular focus is on the automatic generation of a necessary and sufficient set of verification conditions in order for the program to be correct.

  He holds an MSc from Sofia University, Bulgaria, and a PhD degree from the Research Institute for Symbolic Computation of the Johannes Kepler University, Linz, Austria.

- **Dipl.-Ing. Adrian Prantl**
  Institute of Computer Languages
  Vienna University of Technology
  A-1040 Vienna, Austria

  adrian@complang.tuwien.ac.at
  http://www.complang.tuwien.ac.at/adrian

  Adrian Prantl studied computer science at the Vienna University of Technology. During the final year he worked with OnDemand Microelectronics designing and implementing the compiler tool chain for the Chili family of VLIW processors. He is currently working towards a PhD and engaged in the FWF-funded project "Compiler Support for Timing Analysis" (CoSTA) aiming at bringing the power of source code transformations to the field of worst-case execution time analysis.

- **Dr. Markus Schordan**
  University of Applied Sciences Technikum Wien
  A-1200 Vienna, Austria

  schordan@technikum-wien.at
  http://www.technikum-wien.at/

  In 1997-2001 Markus Schordan was a research and teaching assistant at the University Klagenfurt (Department of Information Technology) in Austria. His research focused on alias analysis and data-flow analysis of object-oriented languages, in particular Java. He lectured on the subjects of formal languages and compiler construction, and taught courses in object-oriented programming, functional and logic programming. He earned his Dr.sc.techn. with distinction (mit ausgezeichnetem Erfolg) in Computer Science from the University Klagenfurt, Austria, in June 2001.

  In 2001-2003 he gained international experience as post doctoral researcher at the Lawrence Livermore National Laboratory (Center for Applied Scientific Computing (CASC)), CA, USA. Working on the source-to-source infrastructure project ROSE his research focused on design and implementation of intermediate representations of object-oriented languages, domain specific high-level transformations, and parallelization.

In January 2004 he became university assistant at the Vienna University of Technology, Austria. He lectured on compiler construction and software frameworks. His research focused on tool integration, static analysis of object-oriented languages, source-to-source transformation, high-level optimization, and parallelization. In December 2007 he also became project leader of the ALL-TIMES project at TU Vienna. ALL-TIMES is a medium-scale focused-research project within the European Commission's 7th Framework Programme on Research, Technological Development and Demonstration.

In September 2008 he moved to a permanent position at University of Applied Sciences Technikum Wien and became Deputy Program Director of Game Engineering and Simulation. He continues to lecture on topics in the field of programming languages and also lectures on game engineering. His research focuses on analysis of object-oriented systems, including state-of-the-art game engines.