J Autom Reasoning (2017) 59:1–2
DOI 10.1007/s10817-016-9399-8

Preface: Special Issue on Automatic Resource Bound Analysis

Jürgen Giesl[1] · **Jan Hoffmann**[2]

Received: 28 November 2016 / Accepted: 29 November 2016 / Published online: 5 December 2016

Resource usage—such as time, memory, and energy usage—is one of the most important quality measures of software and programs that exceed the available resources when executed can barely be considered correct. The ever increasing complexity of software systems and library code makes it more difficult for developers to predict the resource usage of their code. As a result, performance bugs are common and among the bugs that are most difficult to detect. Moreover, many security vulnerabilities exploit the space and time usage of software.

This special issue covers *automatic resource bound analysis*, a research area that studies tools and techniques which help programmers to understand the performance characteristics of their code. The goal of many automatic analyses is to summarize the resource usage of a program with an easily-understood symbolic expression that is a function of program variables or the inputs of the program. Since the derivation of such resource bounds is an undecidable problem it is not possible to automatically analyze all programs. However, the driving force in the field is to improve the analysis techniques to automatically derive more precise bounds for more and more programs.

Resource bound analysis brings together techniques from diverse fields such as abstract interpretation, recurrence solving, term rewriting, linear optimization, analysis of algorithms, compiler construction, and type theory. The research focus ranges from the support of challenging language features (e.g., lazy evaluation and concurrency), to deriving bounds for programs that are difficult to analyze (e.g., because they contain challenging loop patterns), to providing stronger guaranties on the precision of the bounds (e.g., by deriving lower bounds on the worst-case behavior).

✉ Jürgen Giesl
giesl@informatik.rwth-aachen.de

Jan Hoffmann
jhoffmann@cmu.edu

1 RWTH Aachen University, Aachen, Germany

2 Carnegie Mellon University, Pittsburgh, PA, USA

For this special issue we were trying to select papers that contain cutting-edge research results and represent a compilation of current research topics and state-of-the-art techniques in resource bound analysis.

The first paper *Complexity and Resource Bound Analysis of Imperative Programs using Difference Constraints* by Moritz Sinn, Florian Zuleger, and Helmut Veith adapts an approach from termination analysis in order to derive invariants and upper complexity bounds. In this way, upper bound analysis can be extended to challenging forms of loops (e.g., with increments and resets) that occur in many typical imperative programs.

In the paper *Rely-Guarantee Termination and Cost Analyses of Loops with Concurrent Interleavings* by Elvira Albert, Antonio Flores-Montoya, Samir Genaim, and Enrique Martin-Martin, the authors develop the first approach to infer upper bounds for loops with concurrent interleavings in an imperative actor-based language. They use a form of rely-guarantee reasoning in order to handle loops separately. Then, upper bounds can be obtained by analyzing how often assertions on the handling of shared data are violated by the program.

The article *Type-Based Cost Analysis for Lazy Functional Languages* by Steffen Jost, Pedro Vasconcelos, Mário Florido, and Kevin Hammond combines two recent breakthroughs that allow us for the first time to automatically analyze lazy functional programs that use recursive and co-recursive data structures. It uses type-directed analysis that is based on amortized analysis.

While the previous papers were concerned with upper complexity bounds, the article *Lower Bounds for Runtime Complexity of Term Rewriting* by Florian Frohn, Jürgen Giesl, Jera Hensel, Cornelius Aschermann, and Thomas Ströder presents the first techniques to infer lower bounds on the worst-case runtime of term rewrite systems. In this way, one can detect vulnerabilities and (in combination with approaches to find upper bounds) prove tight complexity results.

Acknowledgements We thank the Editor-in-Chief Tobias Nipkow for approaching us with the idea for this special issue, for giving us the opportunity to edit it, and for his patience throughout the submission and review process. We also thank the anonymous reviewers for providing detailed feedback and great suggestions for improvement. Finally, we thank the authors of the selected articles for their high-quality submissions and the quick turnaround when incorporating the comments of the reviewers.