ERC Starting Grant 2014
Research proposal [Part B1]

Symbolic Computation and Automated Reasoning for Program Analysis

SYMCAR

- **Principal Investigator (PI):** Laura Kovács
- **PI's Host Institution for the Project:** Chalmers University of Technology, Gothenburg, Sweden
- **Proposal Duration in Months**: 60

Individuals, industries, and nations are depending on software and systems using software. Automated approaches are needed to eliminate tedious aspects of software development, helping software developers in dealing with the increasing software complexity. Automatic program analysis aims to discover program properties preventing programmers from introducing errors while making changes and can drastically cut the time needed for program development.

This project addresses the challenge of automating program analysis, by developing rigorous mathematical techniques analyzing the logically complex parts of software. We will carry out novel research in computer theorem proving and symbolic computation, and integrate program analysis techniques with new approaches to program assertion synthesis and reasoning with both theories and quantifiers. The common theme of the project is a creative development of automated reasoning techniques based on our recently introduced symbol elimination method. Symbol elimination makes the project challenging, risky and interdisciplinary, bridging together computer science, mathematics, and logic.

Symbol elimination will enhance program analysis, in particular by generating polynomial and quantified first-order program properties that cannot be derived by other methods. As many program properties can best be expressed using quantified formulas with arithmetic, our project will make a significant step in analyzing large systems. Since program analysis requires reasoning in the combination of first-order logic and theories, we will design efficient algorithms for automated reasoning with both theories and quantifiers. Our results will be supported by the development of world-leading tools supporting symbol elimination in program analysis.

Our project brings breakthrough approaches to program analysis, which, together with other advances in the area, will reduce the cost of developing safe and reliable computer systems used in our daily life.

Section a: Extended Synopsis of the Scientific Proposal

(a.1) Project Summary

Individuals, organizations, industries, and nations are increasingly depending on software and systems using software. This software is large and complex and integrated in a continuously changing complex environment. New languages, libraries and tools increase productivity of programmers, creating even more software, but the reliability, safety and security of the software that they produce is still low. We are getting used to the fact that computer systems are error-prone and insecure. Software errors cost world economies billions of euros. They may even result in loss of human lives, for example by causing airplane or car crashes, or malfunctioning medical equipment. To improve software and methods of software development one can use a variety of approaches, including automated software verification and static analysis of programs.

The successful development and application of powerful verification methods and tools such as model checkers, static program analyzers, symbolic computation algorithms, decision procedures for common data structures, as well as theorem provers for first- and higher-order logic opened new perspectives for the automated verification of software systems. In particular, increasingly common use of concurrency in the new generation of computer systems has motivated the integration of established reasoning-based methods, such as satisfiability modulo theory (SMT) solvers and first-order (FO) theorem provers, with complimentary techniques such as software testing. This kind of integration has however imposed new requirements on verification tools, such as inductive reasoning [19, 16], interpolation [13], proof generation [7], and non-linear arithmetic symbolic computations [6]. Verification methods combining symbolic computation and automated reasoning are therefore of critical importance for improving software reliability. This project addresses this challenge by automatic program analysis. Program analysis aims to discover program properties preventing programmers from introducing errors while making software changes and can drastically cut the time needed for program development, making thus a crucial step to automated verification.

Purpose and Objectives. *The research proposed in this project develops new, unconventional methods of reasoning and reasoning-based program analysis, by exploiting new synergies between symbolic computation and FO theorem proving in the automated analysis of software systems. We will implement world-leading tools based on these methods, and apply them in concrete problems of high industrial relevance.* In particular, the project targets the combination of algorithmic combinatorics, computer algebra, FO theorem proving and static analysis of programs. Our proposal includes the following two project parts (PPs):

(PP1) Automatic generation of program properties by symbol elimination;

(PP2) Efficient reasoning with both theories and quantifiers for proving program properties.

Part (PP1) is intended to design algorithms for an *automatic generation of program properties*. Such properties express conditions to hold at intermediate program locations and are used to prove the absence of program errors, hence they are very important for improving automation of program analysis. (PP1) will rely on our recent symbol elimination method [16], but use symbol elimination in the combination of FO theorem provers and symbolic computation methods. Such a combination is highly non-trivial and requires the development of new reasoning methods based on superposition FO theorem proving, Gröbner basis computation, and quantifier elimination. (PP1) will provide fundamentally new ways of generating program properties, such as loop invariants and Craig interpolants, resulting in crucial improvements in the theory and practice of program analysis.

In (PP2) we will design efficient techniques for automatically proving properties of complex software systems. Such properties usually express arithmetic and logical operations over the computer memory, which can be represented by unbounded data types such as an array, list, pointer, or heap. Proving such properties automatically requires efficient *reasoning with both quantifiers and theories*. We expect (PP2) to have a deep and long-lasting impact in reasoning-based program analysis, bringing us closer to the goal of automatically analyzing programs containing millions of lines of code.

The algorithms developed in our project will be supported by the *development of the world-leading theorem prover Vampire [18], and its extension to support program analysis*. The new features of Vampire will be evaluated on academic and industrial programs, for the latter we have collaboration agreements with verification teams at Microsoft Research, Intel, Saab and Ericsson. We believe that the world-leading position of Vampire in FO theorem proving will be a basis for a major breakthrough in the use of FO provers in program analysis. Thanks to the full automation and tool support of our project, researchers and software engineers/developers will be able to use our results in their work, without the need to become experts in FO theorem proving and symbolic computation.

Timeliness and Importance. The high-gain aspect of our project comes from its creative use of automated reasoning methods, including new developments related to symbol elimination, program analysis, and combining quantifiers with theories. Our project will turn symbol elimination into a powerful tool for program analysis.

Analyzing and verifying large programs requires non-trivial automation, and automatic generation and proving of program properties is a key step to such automation. Our project brings new non-standard approaches to program property generation, yielding properties that cannot be generated by others. By doing so, we will considerably reduce the required manual work. We believe reasoning with both theories and quantifiers will be the main topic in automatic reasoning for the next two decades: advances in program analysis and verification crucially depend on it, it is very complex, and requires developing new mathematics and non-trivial algorithms.

The timeliness of our research is witnessed by the growing academic and industrial interest in program analysis. Microsoft is now routinely using program analysis techniques, for example on the Windows operating system having over 50 millions lines of code, which allowed it to reduce the main source of the Windows system crashes down to almost zero. Just recently, in Summer 2013 Facebook bought the academic startup verification company Monoidics and now uses program analysis and verification for its software. According to the Open Web Application Security Project - OWASP, the highest ranked security vulnerability comes from the "execution of unintended commands or accessing data without proper authorization", an error that can be discovered by program analysis. Time is thus now ripe for reasoning-based program analysis.

Our project will enable the analysis, and hence partial verification of programs that are beyond the power of existing methods, since advanced symbolic computation techniques and their combination with FO theorem proving are not yet used in state-of-the-art program analysis tools. Our results will bring breakthrough approaches to software analysis, which, together with other advances in the area, will reduce the cost of developing safe and reliable computer programs used in our daily life.

(a.2) Methodology

Our project is divided in two connected Project Parts (PPs), as illustrated in Figure 1. The theory, algorithms, and tools developed in (PP2) will provide the necessary reasoning basis for (PP1). On the other hand, (PP1) will supply (PP2) with demanding testcases, resulting in improvements in (PP2). These improvements will be evaluated in (PP2), yielding a better understanding of what is crucial for essential progress in theory and methods. Therefore, work in both (PP1) and (PP2) will be carried out during the entire project.

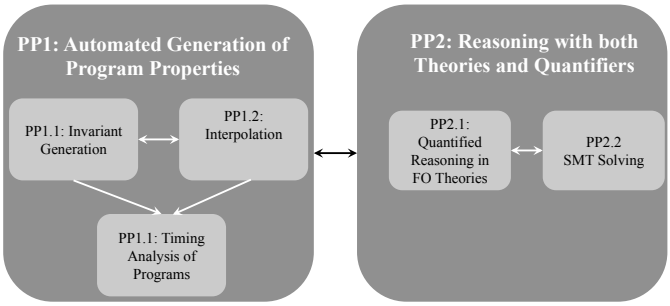

Figure 1: Project Parts and Their Relations.

(PP1) Automated Generation of Program Properties. We are going to find new methods for generating program properties by extending our recently introduced symbol elimination method [16], which works as follows. Given a program, we first extend the language of the program with additional symbols, such as loop counters. Next, we automatically discover program properties that can be expressed by FO formulas in the extended language. To generate properties in the original language of the program, we run a FO theorem prover to eliminate the auxiliary symbols. Unlike other approaches, symbol elimination requires no user guidance, and was the first ever method to automatically discover complex properties with quantifier alternations in the FO theories of various data structures, such as integers or arrays.

The work in (PP1) aims at developing new symbol elimination algorithms for the automated generation of auxiliary program properties that can be used to prove safety and liveness properties of software. Analyzing such properties becomes an especially challenging task for programs with complex flow and, in particular, with loops or recursion. For such programs one needs additional information, in the form of loop invariants, conditions on ranking functions, or interpolants, that express properties to hold at intermediate program points.

Consider for example the program given in Figure 2, written in a C-like syntax. The program fills an integer-valued array B by the positive values of a source array A added to the values of a function call f, and an integer-valued array C with the non-positive values of A. In addition, it computes the sum s of squares of the visited positions in A. A safety assertion, in FO logic (FOL), is specified at the end of the loop, using the `assert` construct. The program of Figure 2 is clearly safe as the assertion is satisfied when the loop is exited. However, to prove program safety we need additional loop properties, i.e. invariants, that hold at any loop iteration. It is not hard to derive that after any iteration k of the loop (assuming $0 \leq k \leq n$), the linear invariant relation $a = b + c$ holds. It is also not hard to argue that, upon exiting the loop, the value of a is n. However, such properties do not give us much information about the arrays A, B, C and the integer s. For proving program safety, we need to derive that each $B[0], \ldots, B[b-1]$ is the sum of a strictly positive element in A and the value of f at the corresponding position of B. We also need to infer that s stores the sum of square of the first n non-negative integers, corresponding to the visited positions in A. Formulating these properties in

FOL yields the loop invariant:

$$(\forall p)(0 \leq p < b \implies (\exists q)(0 \leq q < a \wedge A[q] > 0 \wedge B[p] = A[q]+h(p))) \wedge 6*s = a*(a+1)*(2*a+1) \quad (1)$$

The above property requires quantifier alternations and polynomial arithmetic and can be used to prove the safety assertion of the program. Generating such loop invariants requires reasoning in full FOL with theories, in our example in the FO theory of arrays, polynomial arithmetic and uninterpreted functions. Adding such properties manually involves a considerable amount of work by highly qualified personnel and makes program analysis prohibitively expensive. Therefore, automation of the generation of program properties is invaluable in making industrial program analysis economically feasible. (PP1) will advance symbol elimination in the combination of superposition FO theorem proving and symbolic computation methods, as follows.

(PP1.1) We will automatically derive polynomial equality and inequality invariants for programs with arbitrary polynomial assignments. Using compositional analysis, loops will be modeled by a system of algebraic recurrence equations with polynomial coefficients in the loop variables and iteration counters. For example, one recurrence equation of Figure 2 is $s^{(k+1)} = s^{(k)} + a^{(k)} * a^{(k)}$, where $s^{(k)}$ denotes the value of s at the loop iteration k. Polynomial invariants from the closed form solutions of these recurrences will be derived by symbol elimination in Gröbner basis computation and cylindrical algebraic decomposition. The polynomial invariants will further be used by symbol elimination in FO theorem proving and quantified properties of programs with unbounded data types will be inferred. We will use consequence finding in superposition theorem proving and aim at deriving a minimized set of quantified loop invariants, possibly with quantifier alternations.

$a := 0; b := 0; c := 0; s := 0;$
while $(a < n)$ **do**
  if $A[a] > 0$
    then $B[b] := A[a] + h(b); b := b + 1;$
    else $C[c] := A[a]; c := c + 1;$
  $a := a + 1; s := s + a * a;$
end do
assert$((\forall p)(0 \leq p < b \implies B[p] - h(p) > 0) \wedge$
$6 * s = n * (n + 1) * (2 * n + 1))$

Figure 2: Motivating example.

(PP1.2) Symbol elimination will also be used to develop FO interpolation algorithms, such that the resulting interpolants will give better scalability for inferring system requirements. For example, an interpolant after one loop unrolling of Figure 2 is $B[0] - h(0) > 0 \wedge s = 1$. We will use special orderings and literal selection functions to generate proofs from which interpolants of different logical strength can efficiently be derived.

(PP1.3) Symbol elimination will also be used to derive bounds on program resources, such as time and memory. For example, in Figure 2 the number of loop iterations is bounded by n. The invariants and interpolants from (PP1.1) and (PP1.2) will be used in conjunction with loop tests and predicate abstraction, and tight bounds on the number of loop iterations will be inferred using quantifier elimination. These loop bounds will be used in the execution time analysis of programs.

Unlike current techniques, results of (PP1) will yield FO program properties in a fully automated way, including properties with quantifier alternations and non-linear arithmetic.

(PP2) Reasoning with both Theories and Quantifiers. Program components involve both bounded and unbounded data structures. Therefore, analyzing the intended behavior of a computer program faces the challenging task of reasoning in the combination of FOL and various theories, such as arithmetic, arrays, and lists. For example, for proving safety of Figure 2 using the invariant (1), one needs to reason in the FO theory of arrays, polynomial arithmetic and uninterpreted functions. Without efficient decision procedures and theorem provers, it is impossible to build effective tools for program analysis, which is the main topic of our project.

Decision procedures over quantifier-free theories, implemented in SMT solvers, can process huge formulas, but SMT solvers are yet inefficient in handling quantifiers. On the other hand, FO theorem provers are very efficient in working with quantifiers but weak in theory reasoning. Note that a large number of problems in program analysis (for example, reasoning about dynamic memory allocation) can best be expressed using both theories and quantifiers. Reasoning with quantifiers and theories is therefore regarded as probably the main obstacle in applications of reasoning to software analysis and verification. *(PP2) targets the development of automated theorem proving methods and tools supporting reasoning with both quantifiers and theories.*

(PP2.1) We will start using (incomplete but sound) theory axiomatizations, moving then to more efficient procedures for combining FOL with various theories. We will design special term orderings and literal selection functions for superposition theorem proving in FO theories, without radical changes in the underlining inference mechanism and implementations of superposition. We will target incremental reasoning in FO theories, by using instantiation-based FO theorem proving and tightly integrating the FO superposition calculus and the SMT reasoning part of (PP2.2).

(PP2.2) Further, SMT decision procedures for various fragments of FOL will be used in conjunction with FO theorem proving, and we will use FO theorem proving to generate quantifier-free instances that can be processed using SMT solving. Information extracted from SMT proofs will be then passed to the (PP2.1) in order to refine instance generation.

(PP2) will solve problems beyond the reach of current methods, since we will extend FO theorem proving with theory-specific reasoning rather than using heuristic-based quantifier instantiations in SMT solvers. Our preliminary experiments on proving properties about data collections, such as sets or arrays, show that new superposition rules can efficiently be added to FO provers, solving problems no theorem prover can solve.

Tool Development. Our project will provide automatic tool support for (PP1) and (PP2) by extending the FO theorem prover Vampire [18], to which the PI is actively contributing. Our project will make Vampire a powerful automated reasoning tool able to perform program analysis, polynomial and quantified invariant generation, interpolation, resource bound analysis, and prove properties involving both quantifiers and theories.

(a.3) State-of-the-Art and Innovative Aspects of the Project

Our project addresses problems that other approaches could not solve so far. Due to the page limit, we compare innovative aspects of our proposal only with some key references in the area.

(1) Developing polynomial invariant generation algorithms for loops with arbitrary polynomial arithmetic. In [20, 23] polynomial equalities of a bounded degree are computed as invariants by using linear and polynomial algebra. Without an a priori fixed polynomial degree, in [21] the polynomial invariant ideal is approximated by a fixed point procedure based on polynomial algebra and abstract interpretation. In [15] we described an automatic approach computing all polynomial invariants of a certain class of loops which generalizes the programming model of [21]. Our project extends [15] in new ways, uses recent advances in computer algebra and FO theorem proving, and requires no a priori fixed set of predicates or polynomial degree bounds. Our results will hence simplify the analysis of programs with complex arithmetic.

(2) Developing quantified invariant generation methods for loops over unbounded data structures, such as arrays, lists, pointers, and heaps. Developing such methods is very important for complex programs, since such programs extensively use both complex data types and memory management. In [24] loop invariants are inferred by predicate abstraction over a set of a priori defined predicates, while [9] employs constraint solving over invariant templates. In [19] invariants are computed using input predicates and interpolation. Unlike these approaches, our project will not use a priori defined predicates or templates. User guidance is also not required in [11, 4], but invariants are derived using abstract interpretation over array indexes [11] and array segments [4]. However, these approaches can only infer universally quantified invariants. Our project, based on our previous works [16], will provide a fully automatic method for generating quantified invariants, including those with quantifier alternations, which cannot be handled by other techniques.

(3) Developing efficient techniques for an automatic generation and optimization of interpolants, including those with quantifiers. Craig interpolation turned out to provide a powerful technique for verifying safety properties of software. Papers [13, 3] derive interpolants from resolution proofs in the ground (that is, quantifier-free) theory of linear arithmetic and uninterpreted functions. The approach described in [22] generates ground interpolants in the combined theory of arithmetic and uninterpreted functions using constraint solving techniques over an a priori defined interpolant template. The method presented in [19] computes quantified interpolants from restricted FO proofs over scalars, arrays and uninterpreted functions. Unlike these approaches, our project is not limited to reasoning in ground decidable theories, but use symbol elimination in FO theorem proving in conjunction with quantified theories. Our project builds upon our previous results [17, 12], and computes interpolants of different strength from proofs in FO logic with theories.

(4) Designing an efficient method for the timing analysis of programs. Several existing works make use of abstract interpretation to estimate the number of loop iterations, i.e. loop bounds; however, often loop bounds are assumed to be a priori given, in part, by the user – see e.g. [10]. In [8], an abstract interpretation based linear invariant generation method is used to derive linear bounds over each loop counter variable. Abstract interpretation is also used in [1] in conjunction with analyzing the non-deterministic behavior of multi-path loops. Contrary to these techniques, our project computes complex algebraic upper bounds on the number of loop iterations, by using symbol elimination in symbolic computation. Our project significantly extends our previous results [14] and advances the timing analysis of programs.

(4) Developing new algorithms for reasoning in FO logic and theories. Combining FO proving and theory reasoning is very hard. Most of the modern systems are based on two approaches to such reasoning: trigger-based (heuristic) ground instantiation of quantified axioms in SMT solvers [7] and (incomplete) axiomatization of theories in FO provers [18]. A tighter integration is described in [5, 2]. Based on [18], we will integrate theory reasoning with superposition FO theorem proving, solving problems beyond the reach of other methods.

(a.4) Further Considerations

Interdisciplinarity. The project will use and design rigorous mathematical techniques in polynomial algebra, quantifier elimination, algorithmic combinatorics, and mathematical logic. The project will thus encourage interactions with mathematicians and logicians, bridging together computer science, mathematics and logic. Our project will also help to derive program properties that can be used by computer security tools.

Collaborations. We will collaborate with the following researchers: Dr. Wolfgang Ahrendt, Chalmers; Dr. Nikolaj Bjørner, Microsoft Research; Prof. Armin Biere, JKU Linz; Prof. Koen Claessen, Chalmers; Prof. Thomas A. Henzinger, IST Austria; Dr. Zurab Khasidashvili, Intel; Prof. Jens Knoop, TU Vienna; Prof. Andrei Sabelfelf, Chalmers; Prof. Helmut Veith, TU Vienna; Prof. Andrei Voronkov, U. of Manchester.

Career Development of the PI. The PI was appointed to Chalmers as an associate professor in April 2013, with a purpose of building an internationally competitive research group in formal methods. The ERC Starting Grant would support the PI in building up her own research group at Chalmers by providing funding for researchers working under her supervision.

Requested Funding. The PI will dedicate at least 70% of her time to the project. We request funding for the PI, one postdoc and one PhD student; partial funding for a project administrator; and funding for travels, computers, and publications.

Literature

[1] E. Albert, P. Arenas, S. Genaim, and G. Puebla. Closed-Form Upper Bounds in Static Cost Analysis. *J. Automated Reasoning*, 46(2):161–203, 2011.

[2] L. Bachmair, H. Ganzinger, and U. Waldmann. Refutational Theorem Proving for Hierarchic First-Order Theories. *Appl. Algebra Eng. Commun. Comput.*, 5:193–212, 1994.

[3] A. Cimatti, A. Griggio, and R. Sebastiani. Efficient Interpolant Generation in Satisfiability Modulo Theories. In *TACAS*, pages 397–412, 2008.

[4] P. Cousot, R. Cousot, and F. Logozzo. A Parametric Segmentation Functor for Fully Automatic and Scalable Array Content Analysis. In *POPL*, pages 105–118, 2011.

[5] L. de Moura and N. Bjørner. Engineering DPLL(T) + Saturation. In *IJCAR*, pages 475–490, 2008.

[6] L. de Moura and G. Passmore. Computation in Real Closed Infinitesimal and Transcendental Extensions of the Rationals. In *CADE*, pages 178–192, 2013.

[7] L. M. de Moura and N. Bjorner. Proofs and refutations, and z3. In *CEUR Workshop Proceedings*, 2008.

[8] S. Gulwani and F. Zuleger. The Reachability-Bound Problem. In *PLDI*, pages 292–304, 2010.

[9] A. Gupta and A. Rybalchenko. InvGen: An Efficient Invariant Generator. In *CAV*, pages 634–640, 2009.

[10] J. Gustafsson, A. Ermedahl, C. Sandberg, and B. Lisper. Automatic Derivation of Loop Bounds and Infeasible Paths for WCET Analysis Using Abstract Execution. In *RTSS*, pages 57–66, 2006.

[11] N. Halbwachs and M. Peron. Discovering Properties about Arrays in Simple Programs. In *PLDI*, pages 339–348, 2008.

[12] K. Hoder, L. Kovács, and A. Voronkov. Playing in the Grey Area of Proofs. In *POPL*, pages 259–272, 2012.

[13] R. Jhala and K. L. McMillan. A Practical and Complete Approach to Predicate Refinement. In *TACAS*, pages 459–473, 2006.

[14] J. Knoop, L. Kovács, and J. Zwirchmayr. WCET Squeezing: On-demand Feasibility Refinement for Proven Precise WCET-bounds. In *RTNS*, pages 161–170, 2013.

[15] L. Kovács. Reasoning Algebraically About P-Solvable Loops. In *TACAS*, pages 249–264, 2008.

[16] L. Kovács and A. Voronkov. Finding Loop Invariants for Programs over Arrays Using a Theorem Prover. In *FASE*, pages 470–485, 2009.

[17] L. Kovács and A. Voronkov. Interpolation and Symbol Elimination. In *CADE*, pages 199–213, 2009.

[18] L. Kovács and A. Voronkov. First-Order Theorem Proving and Vampire. In *CAV*, pages 1–35, 2013.

[19] K. L. McMillan. Quantified Invariant Generation Using an Interpolating Saturation Prover. In *TACAS*, pages 413–427, 2008.

[20] M. Mueller-Olm and H. Seidl. Precise Interprocedural Analysis through Linear Algebra. In *POPL*, pages 330–341, 2004.

[21] E. Rodriguez-Carbonell and D. Kapur. Generating All Polynomial Invariants in Simple Loops. *J. Symbolic Computation*, 42(4):443–476, 2007.

[22] A. Rybalchenko and V. Sofronie-Stokkermans. Constraint Solving for Interpolation. In *VMCAI*, pages 346–362, 2007.

[23] S. Sankaranaryanan, H. B. Sipma, and Z. Manna. Non-Linear Loop Invariant Generation using Groebner Bases. In *POPL*, pages 318–329, 2004.

[24] S. Srivastava and S. Gulwani. Program Verification using Templates over Predicate Abstraction. In *PLDI*, pages 223–234, 2009.

Section b: Curriculum Vitae

PERSONAL INFORMATION:
- Family name, First name: KOVÁCS, Laura Ildikó
- Date of birth: April 26, 1980
- Web site: `www.cse.chalmers.se/~laurako`

EDUCATION:
- 2012 November: Habilitation in Computer Science, Vienna University of Technology (TU Vienna), Austria
- 2007 October: PhD in Computer Science with highest distinction, Research Institute for Symbolic Computation (RISC-Linz), Johannes Kepler University Linz, Austria
- 2004 February: Master in Computer Science, Department of Computer Science, West University of Timişoara, Romania.

CURRENT POSITIONS:
- April 2013 - present: Associate Professor, Department of Computer Science and Engineering, Chalmers University of Technology, Sweden
- April 2013 - present: Adjunct Associate Professor, Faculty of Informatics, TU Vienna, Austria

PREVIOUS POSITIONS:
- 2010 - 2013: Hertha-Firnberg research fellow of the Austrian Science Fund (FWF); Assistant Professor, TU Vienna, Austria
- 2009 - 2010: Postdoctoral scientist at ETH Zürich, Switzerland
- 2007 - 2009: Postdoctoral scientist at EPFL Lausanne, Switzerland
- 2003 - 2007: PhD researcher at RISC-Linz, Austria

Invited positions:
- 2012 June: Visiting professor, University Joseph Fourier, Grenoble, France

FELLOWSHIPS AND AWARDS:
- 2013 November: Swedish Research Council - VR grant for Junior Researchers, Sweden (a highly competitive fellowship: 769 applications, 64 approved)
- 2012 June: Visiting professorship at the University Joseph Fourier, Grenoble, France
- 2011 November: FESTO Austria Prize for Young Researchers and Scientists, FESTO IT company in process automation, Austria
- 2009 June: FWF Hertha-Firnberg Fellowship, Austria (a very competitive fellowship: 53 applications, 13 approved)
- 2008: Best paper award in application track, 3rd Int. Computer Science Symposium in Russia (CSR), Russia
- 2007 September: Japan Society for the Promotion of Science fellowship, Tsukuba University
- 2005: Best student presentation award, 2nd South-East European Workshop on Formal Methods (SEEFM), FYR Macedonia

SUPERVISION OF GRADUATE STUDENTS AND POSTDOCTORAL FELLOWS:
Currently, I supervise and collaborate with two doctoral students and one master student. In addition, I also supervise five bachelor projects. To date, one doctoral student, two master students and one bachelor student have graduated under my supervision. One undergraduate student completed a summer internship project.
- 2013 - 2018: Main PhD supervisor of 1 PhD student at Chalmers:
  - MSc. Evgenii Kotelnikov, with expected date of PhD examination: September 2018
- 2010 - 2015: Main PhD supervisor of 2 PhD students at the TU Vienna:
  - MSc. Ioan Dragan, with expected date of PhD examination: February 2015
  - Dipl.Ing. Jakob Zwirchmayr, with PhD examination date: October 2013
- 2011 - 2014: Main Master thesis supervisor of 2 Master students at the TU Vienna:
  - MSc. Bernhard Kragl, with expected date of Master examination: April 2014
  - MSc. Ioana Jucu, with Master examination date: November 2013
- 2008 - 2009: Master thesis co-supervisor of 1 Master students at EPFL Lausanne:
  - MSc. Thibaud Hottelier, with Master examination date: June 2009

TEACHING ACTIVITIES (only lecture courses):
My teaching experience includes sole-lecturing at Chalmers, TU Vienna, ETH Zürich, University of Zürich and EPFL Lausanne; all together 5 master courses, 1 master seminar, and 4 bachelor courses, as follows:

- 2013 - 2014: Sole-lecturer of the "Automated Reasoning for Program Verification" master course, Chalmers, Fall 2013
- 2012 - 2014: Sole-lecturer of the "Automated Reasoning and Program Verification" master course, TU Vienna, Spring 2014, 2013, and 2012
- 2012 - 2013: Sole-lecturer of the "Advanced Topics in Theoretical Computer Science" master course, TU Vienna, Fall 2012
- 2010 - 2012: Sole-lecturer of the "Advanced Theoretical Computer Science" bachelor course , TU Vienna, Fall 2011 and 2010
- 2009 - 2010: Sole-lecturer of the "Foundations of Computer Science 1 bachelor course, University of Zürich, Fall 2009
- 2009 - 2010: Lecturer of the "Software Engineering Seminar" master seminar, ETH Zürich, Fall 2009
- 2008 - 2009: Sole-lecturer of the "Advanced Theoretical Computer Science" bachelor course, EPFL Lausanne, Spring 2009

ORGANIZATION OF SCIENTIFIC MEETINGS:
- 2012: Organizer of the Dagstuhl Seminar 12461 on "Games and Decisions for Rigorous Systems Engineering, Germany, 50 participants
- 2012: Local chair of the ICNPAA 2012 World Congress: 9th Int. Conference on Mathematical Problems in Engineering, Aerospace and Sciences, TU Vienna, Austria, 300 participants
- 2011: Local chair of the Int. Workshop on Logic and Computer Science, University of Vienna, Austria, 30 participants
- 2010: Local chair of the Int. Workshop on Automated Specification and Verification of Web Systems (WWV), TU Vienna, Austria, 20 participants
- 2007: Local organization chair of the Int. Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (CALCULEMUS) and Mathematical Knowledge Management (MKM), RISC-Linz, Austria, 150 participants

INSTITUTIONAL RESPONSIBILITIES:
- 2013-2015: Academic senate member, Chalmers
- 2011-2013: Computer Science curriculum committee member, TU Vienna

COMMISSIONS OF TRUST:
A complete list of activities is available at `www.cse.chalmers.se/~laurako/CV_Kovacs.pdf`. Here we report on the most important activities.
- Program committee (PC) chair: PC chair of 4 conferences and 6 workshops in the area of formal methods, including the Symbolic Computation in Software Science (SCSS) conference, 2013; Workshop on Invariant Generation, 2010 and 2009
- Program committee (PC) member: PC member of 30 international conferences, including the top tier conferences Logic in Computer Science (LICS), 2014; Symposium on Static Analysis (SAS), 2014; Computer Aided Verification (CAV), 2014; Logic for Programming, Artificial Intelligence and Reasoning (LPAR), 2009-2013; Computer Science Logic (CSL), 2012; Symbolic and Algebraic Computation (ISSAC), 2010
- Guest editor of 4 special issues of international journals in formal methods, including the J. of Symbolic Computation 2012 and 2010; J. of Applied Logic, 2012; J. of Logic and Algebraic Programming, 2013
- Journal and conference referee: reviewer for 1 book chapter, 15 international journals, and 12 international conferences and workshops, including the Handbook of Model Checking, 2012; ACM Transactions on Programming Languages and Systems, 2011 and 2012; J. of Symbolic Computation, 2010 and 2011; Int. Conference on Programming Language Design and Implementation (PLDI), 2010 and 2009

MEMBERSHIPS OF SCIENTIFIC SOCIETIES:
- 2010 - : Founding member of the Austrian Society for Rigorous Systems Engineering (ARiSE)
- 2011 - : Advisory board member of the Vienna Center for Logic and Applications (VCLA)
- 2008 - : Member of the Association for Automated Reasoning and the European Association for Programming Languages and Systems

MAJOR COLLABORATIONS:
- 2009 - present: Joint work with Prof. Andrei Voronkov (U. of Manchester) on automated reasoning and developing the world-leading theorem prover Vampire
- 2009 - 2010: Industrial consulting on theorem proving and program analysis at Dassault Aviation, France
- 2010: Industrial consulting on program analysis at Intel, Haifa

Appendix: All Ongoing and Submitted Grants and Funding of the PI (Funding ID)

1. Ongoing Grants

The current research projects of the PI are summarized below.

Project Title	Funding Source	Amount (Euros)	Period	Role of the PI	Relation to current ERC proposal
Interpolation and Symbol Elimination	Austrian Science Fund (FWF)	178,668	03/2011 - 02/2015	PI	see below
GenPro: Generating and Proving Program Properties via Symbol Elimination	Swedish Research Council (VR)	379,220	09/2014 - 08/2018	PI	see below
Assurance as a Service: Upfront Quality and Safety in Continuous Software Engineering	Chalmers Software Center	15,800	01/2014 - 06/2014	co-PI	see below

Relation of ongoing grants to the current ERC proposal.

• Some parts of (PP1.2) of the current ERC proposal build upon results obtained during the PI's project "Interpolation and Symbol Elimination" funded by the Austrian Science Fund – FWF (2011-2015). Nevertheless, the current project is not merely a continuation of this FWF project. It proposes symbol elimination as a unifying framework for invariant generation, interpolation, timing analysis of programs, and reasoning with both quantifiers and theories. All parts of the current ERC proposal, except (PP1.2), are thus completely new research directions, whereas (PP1.2) strengthens the objectives of the PI's FWF grant by generating interpolants of different strength in various first-order theories.

• The current ERC proposal is also related to the PI's recently granted project "GenPro: Generating and Proving Program Properties via Symbol Elimination", funded by the Swedish Research Council - VR. The VR grant provides financial support for one PhD position. The use of computer algebra methods in combination with theorem proving is however not addressed in the VR project, and hence polynomial invariant generation, resource bound analysis and quantified reasoning with both arithmetic and quantifiers are completely new as compared to the VR project. Our proposal is thus complementary to the VR grant, requesting funding for basic and fundamental research in all parts of the project.

• Recently, with the "Assurance as a Service: Upfront Quality and Safety in Continuous Software Engineering" project, the PI has secured seed funding from the industrial board of the Chalmers Software Center for exploring how formal methods can be used by the local Swedish industry, in particular by Ericsson and Saab. Results of this grant will give a solid practical basis for the work proposed within the current ERC proposal.

2. Applications

The submitted ongoing research applications of the PI are summarized below. Funding decisions on these applications are expected in December 2014.

Project Title	Funding Source	Amount (Euros)	Period	Role of the PI	Relation to current ERC proposal
Algebraic and Theorem Proving Methods for Verified Software	Swedish Foundation for Strategic Research (SSF): Career Development Programme Promoting Gender Equality	670,000	1/2015 - 12/2019	PI	see below
TheProSE: Theorem Proving and Symbol Elimination for Software Analysis and Verification	Wallenberg Academy Fellowship 2014 (KAW)	560,000	1/2015 - 12/2019	PI	see below

Relation of ongoing applications to the current ERC proposal.

This ERC proposal is related to two recently submitted grant applications of the PI. Both of these grant applications are individual fellowships promoting the career development of young researchers; in addition, one of them (the SSF grant application) makes a special emphasis on promoting gender equality in science and technology.

• The overall goal of the SSF grant application is the design of a reasoning-based verification framework using algebraic and theorem proving methods for invariant generation. Compared to the SSF grant application, this ERC proposal brings completely new research directions on interpolant generation and reasoning with both theories and quantifiers for automated program analysis. Our proposal is thus complementary to the SSF grant application, requesting funding for basic and fundamental research in all parts of the project.

• The PI has recently been nominated by the Chalmers University of Technology to the prestigious KAW 2014 fellowship program of Sweden, which is an individual fellowship program promoting young researchers and providing fellowships partially covering salaries of KAW fellows. The research objectives of this ERC proposal are strongly connected to the KAW fellowship application. However, compared to the KAW fellowship application, this ERC proposal brings new research directions to program analysis by using symbolic computation methods in conjunction with first-order theorem proving. If granted, this ERC proposal will provide the PI with the necessary funding for building up her own research group, by funding graduate students and a postdoctoral researcher working under her supervision.

Section c: Early Achievements Track-Record

Scientific novelty. In 2009 I have introduced the symbol elimination method for program analysis. Symbol elimination generalizes other approaches to program analysis, such as Gröbner basis computation, quantifier elimination and Craig interpolation, and was the first ever method able to generate high quality loop invariants with quantifier alternations. When testing symbol elimination on industrial safety-critical applications, my results have shown that symbol elimination actually works well in practice: the loop properties obtained by symbol elimination could replace annotations produced by human experts in over 80% of test cases. The impact of symbol elimination in program analysis is witnessed by the 83 citations my FASE 2009 paper.
Industrial relevance. During 2009–2010, I was consulting the Dassault Aviation company, France, in program analysis. In December 2010, I was also invited to consult for Intel in Haifa, with the purpose of applying theorem proving and program analysis in hardware verification. Starting with January 2014, I secured seed funding from the Chalmers Software Center for exploring how formal methods can be used by the Swedish industry.
Scientific tools. I implemented the symbolic computation engine Aligator to derive polynomial invariants and contributed to the timing analysis of programs with the r-TuBound toolchain. Starting with 2009, I become one of the developer of the Vampire theorem prover, significantly increasing the number of Vampire users. For example, in 2010-2011 Vampire was downloaded more than 1,000 times, that is, over 10 times a week. My work on Vampire was invited for publication at CAV 2013, the leading conference in verification.
Research Summary. My preliminary results on symbol elimination in program analysis provide a solid background for further progress in the area, addressing the challenge of reasoning about software with complex flow and various data types. My academic contribution and my experience in developing reasoning-based verification methods and tools enable me to make significant developments in both theory and implementation and ensure that the research directions proposed in the current project successfully meet their scientific goals.

Scientific Publications, Talks, Prizes

Publications. Computer science is a conference-oriented field. All my conference proceeding papers have been subject to high standards of peer reviews from top conferences (acceptance rate below 20%), including publications at flagship venues in automated reasoning (2 CADE papers, 2 IJCAR, 5 LPAR), programming languages (1 POPL), and verification (1 ATVA, 1 CAV, 1 FASE, 2 TACAS). *These publications attracted 605 citations and yield my h-index of 13 (source: Google Scholar).* All together, I published 1 textbook, 3 invited papers, 7 journal papers, and 50 peer-reviewed proceedings paper. My complete list of publications is available at: `www.cse.chalmers.se/~laurako/publication_Kovacs.pdf`. Here we give a selected list of the most important publications since defending my PhD thesis in 2007 – all these papers are **authored without the presence of my PhD supervisor** and have their authors listed in alphabetical order. Citations are given according to Google Scholar (GS). The five most representative papers are highlighted (papers 1, 8, 20, 21, 25).

1. **Laura Kovács and Andrei Voronkov. *First-Order Theorem Proving and Vampire*. Computer Aided Verification (CAV), pg. 1-35, 2013, (GS): 5. Invited Paper.**
2. Régis Blanc, Ashutosh Gupta, Laura Kovács, and Bernhard Kragl. *Tree Interpolation in Vampire*. Logic for Programming Artificial Intelligence and Reasoning (LPAR-19), pg. 173-181, 2013, (GS): 0.
3. Laura Kovács, Natasha Sharygina, and Simone Fulvio Rollini. *A Parametric Interpolation Framework for First-Order Theories*. Mexican Conf. on Artificial Intelligence (MICAI), 2013, (GS): 0.
4. Armin Biere, Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. *SmacC: A Retargetable Symbolic Execution Engine*. Automated Technology for Verification and Analysis (ATVA), pg. 482-486, 2013, (GS): 2.
5. Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. *WCET Squeezing: On-demand Feasibility Refinement for Proven Precise WCET-bounds*. Real-Time and Network Systems (RTNS), pg. 161-170, 2013. (GS): 1.
6. Ioan Dragan, Konstantin Korovin, Laura Kovács, and Andrei Voronkov. *Bound Propagation for Arithmetic Reasoning in Vampire*. Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2013, To appear, (GS): 0.
7. Armin Biere, Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. *The Auspicious Couple: Symbolic Execution and WCET Analysis*. Worst-Case Execution Time Analysis (WCET), pg. 53-63, 2013, (GS): 1.
8. **Krystof Hoder, Laura Kovács, and Andrei Voronkov. *Playing in the Grey Area of Proofs*. Principles of Programming Languages (POPL), pg. 259-272, 2012. (GS): 12.**
9. Krystof Hoder, Andreas Holzer, Laura Kovács, and Andrei Voronkov. *Vinter: A Vampire-Based Tool for Interpolation*. Asian Symp. on Programming Languages and Systems (APLAS), pg. 148-146, 2012, (GS): 0.
10. Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. *r-TuBound: Loop Bounds for WCET Analysis*. Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-18), pg. 435-444, 2012, (GS): 9.
11. Krystof Hoder, Laura Kovács, and Andrei Voronkov. *Case Studies on Invariant Generation Using a Saturation Theorem Prover*. Mexican Conf. on Artificial Intelligence (MICAI), pg. 1-15, 2011, (GS): 6.

12. Krystof Hoder, Laura Kovács, and Andrei Voronkov. *Invariant Generation in Vampire*. Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pg. 60-64, 2011, (GS): 5.
13. Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. *Symbolic Loop Bound Computation for WCET Analysis*. Perspectives of System Informatics (PSI), pg. 224-239, 2011, (GS): 15.
14. Laura Kovács, Georg Moser, and Andrei Voronkov. *On Transfinite Knuth-Bendix Orders*. Conference on Automated Deduction (CADE), pg. 384-399, 2011, (GS): 3.
15. Régis Blanc, Thomas Henzinger, Thibaud Hottelier, and Laura Kovács. *ABC: Algebraic Bound Computation for Loops*. In Logic for Programming, Artificial Intelligence and Reasoning (LPAR-16), pg. 103-118, 2010, (GS): 14.
16. Thomas Henzinger, Thibaud. Hottelier, Laura Kovács, and Andrey Rybalchenko. *Aligators for Arrays*. Logic for Programming, Artificial Intelligence and Reasoning (LPAR-17), pg. 348-356, 2010, (GS): 4.
17. Thomas Henzinger, Thibaud Hottelier, Laura Kovács, and Andrei Voronkov. *Invariant and Type Inference for Matrices*. Verification, Model Checking, and Abstract Interpretation (VMCAI), pg. 163-179, 2010, (GS): 8.
18. Krystof Hoder and Laura Kovács and Andrei Voronkov. *Interpolation and Symbol Elimination in Vampire*. Joint Conference on Automated Reasoning (IJCAR), pg. 188-195, 2010, (GS): 23.
19. Laura Kovács. *A Complete Invariant Generation Approach for P-solvable Loops*. Perspectives of System Informatics (PSI), pg. 242-256, 2009, (GS): 6.
20. **Laura Kovács and Andrei Voronkov. *Finding Loop Invariants for Programs over Arrays Using a Theorem Prover*. Fundamental Approaches to Software Engineering (FASE), pg. 470-485, 2009, (GS): 83.**
21. **Laura Kovács and Andrei Voronkov. *Interpolation and Symbol Elimination*. Conference on Automated Deduction (CADE), pg. 199-213, 2009, (GS): 32.**
22. Thomas Henzinger, Thibaud Hottelier, and Laura Kovács. *Valigator: A Verification Tool with Bound and Invariant Generation*. on Logic for Programming, Artifical Intelligence and Reasoning (LPAR-15), pg. 333-342, 2008, (GS): 14.
23. Laura Kovács. *Aligator: A Mathematica Package for Invariant Generation*. Joint Conference on Automated Reasoning (IJCAR), pg. 275-282, 2008, (GS): 13.
24. Laura Kovács. *Invariant Generation for P-solvable Loops with Assignments Only*. Computer Science in Russia (CSR), pg. 349-359, 2008, (GS): 11. **Best Paper Award in the Application Track**.
25. **Laura Kovács. *Reasoning Algebraically About P-solvable Loops*. Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pg. 249-264, 2008, (GS): 29.**

Invited Talks at Conferences and International Advanced Schools.

1. *Generating Program Properties using Symbol Elimination*. Automated Verification of Critical Systems (AVOCS), The Netherlands, September 2014.
2. *Symbol Elimination in Program Analysis*. Program Verification, Automated Debugging and Symbolic Computation (PAS), China, October 2012.
3. *Symbol Elimination in Program Analysis*. Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Romania, September 2011.
4. *Quantified Invariant Generation using Symbolic Computation and Theorem Proving*. Symbolic Computation and Software Verification (SCSV), Japan, April 2010.
5. *Program Assertion Synthesis using Symbol Elimination*. VTSA Summer School on Verification Technology, Systems, and Applications, Luxemburg, October 2014.
6. *First-Order Theorem Proving and Vampire*. ReRiSE Advanced Winter School on Reasoning Engines for Rigorous Systems Engineering, Austria, February 2014.
7. *Automated Theorem Proving*. ARiSE/VCLA Winter School on Verification, Austria, February 2012.
8. *Invariant Generation by Algebraic Techniques for Software Verification*. Tbilisi Summer School in Logic and Language, Georgia, September 2008.

In addition, I was a **tutorial speaker at 5 international conferences**, including CADE 2011.

Prizes and Awards.

- 2013 November: Swedish Research Council - VR grant for Junior Researchers (769 applications, 64 approved).
- 2012 June: Visiting professorship at the University Joseph Fourier, Grenoble, France.
- 2011 November: FESTO Austria Prize for Young Researchers and Scientists, FESTO IT company, Austria.
- 2009 June: FWF Hertha-Firnberg Fellowship, Austria (53 applications, 13 approved).
- 2008: Best paper award in application track, 3rd Int. Computer Science Symposium in Russia (CSR), Russia.
- 2007 September: Japan Society for the Promotion of Science fellowship, Tsukuba University.
- 2005: Best student presentation award, 2nd South-East European Workshop on Formal Methods, Macedonia.