VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kansliets noteringar Kod	Dnr
2013-45614-105023-81	2013-4977

2013
Project Grant Junior Researchers

Area of science
Natural and Engineering Sciences
Announced grants
Research grants NT April 11, 2013
Total amount for which applied (kSEK)

2014	2015	2016	2017	2018
1062	1076	1123	1169	1229

APPLICANT

Name(Last name, First name)
Kovacs, Laura

Date of birth
800426-2906

Gender
Female

Email address
laura.kovacs@chalmers.se

Academic title
Associate professor

Position
Associate professor

Phone
+46317721696

Doctoral degree awarded (yyyy-mm-dd)
2007-10-24

WORKING ADDRESS

University/corresponding, Department, Section/Unit, Address, etc.
Chalmers tekniska högskola
Institutionen för data-och informationsteknik
Programvaruteknik
Chalmers University of Technology, CSE Department, Rännvägen 6B
41296 Göteborg, Sweden

ADMINISTRATING ORGANISATION

Administrating Organisation
Chalmers tekniska högskola

DESCRIPTIVE DATA

Project title, Swedish (max 200 char)
GenPro: Generera och bevisa programegenskaper via symboleliminering

Project title, English (max 200 char)
GenPro: Generating and Proving Program Properties via Symbol Elimination

Abstract (max 1500 char)

Complex systems used in our daily life rely on software used in them. Such software is becoming increasingly more sophisticated, resulting in system malfunctioning and insecure behavior. Formal verification of such systems requires non-trivial automation, and automatic generation and reasoning about program properties is a key step to such an automation. Reasoning about program properties is an especially challenging task for programs with complex flow, in particular, with loops.

Our project addresses this challenge by designing new methods for reasoning about program loops. We will advance the power of the symbol elimination method, introduced recently by ourselves, for generating and proving program properties. We will carry out novel research in computer theorem proving to support symbol elimination and generate non-trivial first-order program properties, such as loop invariants and interpolants. Since reasoning about program properties requires reasoning in first-order theories, we will design new algorithms for automated reasoning with both theories and quantifiers.

The project combines computer science, mathematics, and logic. It will enhance capabilities of world-leading reasoning tools. Our proposal will advance the applicability of first-order theorem proving, in particular symbol elimination, in formal verification, pushing the boundaries of the technology further to the ultimate goal of automatic verification of very large computer programs.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2013-45614-105023-81

Name of Applicant
Kovacs, Laura

Date of birth
800426-2906

Abstract language
English

Keywords
program verification, automated reasoning, first-order theorem proving, program analysis, formal methods

Research areas
*Nat-Tek generellt

Review panel
NT-2

Classification codes (SCB) in order of priority
10201, 10205,

Aspects

Application is also submitted to

similar to: | **identical to:**

ANIMAL STUDIES

Animal studies
No animal experiments

OTHER CO-WORKER

Name(Last name, First name) | **University/corresponding, Department, Section/Unit, Addressetc.**
,

Date of birth | **Gender**

Academic title | **Doctoral degree awarded (yyyy-mm-dd)**

Name(Last name, First name) | **University/corresponding, Department, Section/Unit, Addressetc.**
,

Date of birth | **Gender**

Academic title | **Doctoral degree awarded (yyyy-mm-dd)**

Name(Last name, First name) | **University/corresponding, Department, Section/Unit, Addressetc.**
,

Date of birth | **Gender**

Academic title | **Doctoral degree awarded (yyyy-mm-dd)**

Name(Last name, First name) | **University/corresponding, Department, Section/Unit, Addressetc.**
,

Date of birth | **Gender**

Academic title | **Doctoral degree awarded (yyyy-mm-dd)**

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2013-45614-105023-81

Name of Applicant
Kovacs, Laura

Date of birth
800426-2906

ENCLOSED APPENDICES

A, B, C, N, S

APPLIED FUNDING: THIS APPLICATION

Funding period (planned start and end date)
2014-01-01 -- 2018-12-31

Staff/ salaries (kSEK)

Main applicant	**% of full time in the project**	**2014**	**2015**	**2016**	**2017**	**2018**
Laura Kovacs	20	278	288	299	309	320
Other staff						
PhD Student	80	626	648	671	695	720
Total, salaries (kSEK):		904	936	970	1004	1040

Other projectrelated costs (kSek)	**2014**	**2015**	**2016**	**2017**	**2018**
Travel cost	70	70	70	70	70
Laptop	20			20	
PhD exam/study costs			10		40
Premesis	58	60	62	64	67
Direct IT costs	10	10	11	11	12
Total, other costs (kSEK):	158	140	153	165	189

Total amount for which applied (kSEK)

2014	2015	2016	2017	2018
1062	1076	1123	1169	1229

ALL FUNDING

Other VR-projects (granted and applied) by the applicant and co-workers, if applic. (kSEK)

Funds received by the applicant from other funding sources, incl ALF-grant (kSEK)

Funding source	**Total**	**Proj.period**	**Applied 2014**
Austrian Science Fund (FWF)	1491	2011-2015	
Project title	**Applicant**		
Interpolation and Symbol Elimination	Laura Kovacs		

POPULAR SCIENCE DESCRIPTION

Popularscience heading and description (max 4500 char)

Mycket komplexa datorsystem har blivit en viktig del av våra liv. Banker, sjukhus, företag, organisationer och individer använder sådana system och är beroende av dem. Dessa system, bl.a. Internet, mobilnätet, betalsystem, autonoma enheter, trafikövervakning, mm, är helt beroende av programvaran de innehåller. Om den här programvaran inte är pålitlig kan kostnaderna för företag och samhälle bli enorma.

Utmaningen att skapa pålitlig programvara har varit i fokus för datavetenskapen under flera decennier. Det är ett svårt problem

eftersom programvaran och därmed den underliggande matematiska modellen hela tiden blir mer komplex.

Många företag och organisationer hanterar nu för tiden rutinmässigt programvara som består av flera miljoner rader kod, skrivna av många olika personer och uttryckta i flera olika programspråk, med olika verktyg och i olika stilar. Sådan programvara är svår att förstå i sig och den är dessutom integrerad i en föränderlig kontext via datorer, nätverk, olika styrenheter, säkerhetsprotokoll och andra komponenter. Programvaruutveckling blir därför dyr och felkällorna många vilket gör att en stor andel av de industriella programutvecklingsresurserna läggs på att hitta och korrigera fel.

Målet med formell programverifiering är att erbjuda automatiserade metoder att hitta och korrigera fel och på så sätt producera mer pålitliga och robusta system. Det finns ett växande intresse för att tillämpa formella metoder för att garantera pålitligheten hos högkvalitativa IT-system som vi stöter på i vårt dagliga liv. Att hantera sådana komplexa system och öka deras pålitlighet är ett utmanande forskningsämne både i akademin och industrin, och det är av avgörande betydelse för IT-industrin. I det här projektet vill vi tackla denna utmaning.

Eftersom bevis av mycket stora program skulle kräva mycket stora mänskliga resurser är huvudidén i formell verifiering att designa automatiska programverktyg (också program!) som tar hand om bevisen. Användningen av automatiska programverktyg är nyckeln till formell verifierings framgång.

Detta projekt har som mål att utveckla nya verifieringsmetoder genom att använda datorstödd bevisföring på nya sätt. Vår forskning kommer att utgå från den nyligen introducerade symboleliminingstekniken inom datorstödd bevisföring och den kommer att rikta in sig på följande spår.

* Vi kommer att automatiskt härleda egenskaper och tips som hindrar programmerare från att göra fel när de ändrar i sina program. Att ge sådana tips manuellt kräver en hel del arbete av välutbildad personal vilket ofta gör det för dyrt i praktiken. Därför är automatiskt genererade tips utan anvisning från användarna ovärderligt när det handlar om att göra programverifiering ekonomiskt gångbart.

* Vi kommer också designa nya algoritmer som stödjer bevisföring om programegenskaper och implementera dessa algoritmer i den prisbelönta teorembevisaren Vampire som huvudsökanden är aktivt involverad i utvecklingen av. Vampire används av flera tusen akademiker och industriexperter över hela världen.

Vi förväntar oss att vårt projekt kommer att få kraftig och långvarig effekt på bevisföringsbaserad formell verifiering, och kommer att ge nya metoder och vetenskapliga genombrott inom automatisk bevisföring och formell verifiering. Vårt projekt angriper problem som andra angreppssätt ännu inte kunnat lösa och vi ämnar flytta forskningsfronten framåt med det långsiktiga målet att nästan helt automatisera verifiering av stora datorprogram.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2013-45614-105023-81

Name of applicant
Kovacs, Laura

Date of birth
800426-2906

Title of research programme
GenPro: Generating and Proving Program Properties via Symbol Elimination

Appendix A

Research programme

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Appendix A – Research Programme

GenPro: Generating and Proving Program Properties via Symbol Elimination

1. Purpose and Aims

Software systems used in our daily life, such as networking, security, autonomous devices, traffic control, etc., heavily rely on software used in them. Such software is becoming increasingly more sophisticated, resulting in system malfunctioning and error-prone and insecure system behavior. Software errors are very costly. There are many studies attempting to quantify the cost of software failures, including survey reports by the European Services Strategy Unit, the British Computer Society, and the US National Institute of Standards and Technology. For example, according to US National Institute of Standards and Technology software errors cost the US economy nearly 60 billion annually, and 80% of development costs involve identifying and correcting errors. For this reason, there is a growing interest, both industrial and academic, in applying formal methods for ensuring reliability of long-lived, high-quality software systems.

Formal verification aims at providing a methodology that produces more reliable and robust systems. Companies developing and running safety-critical systems, such as Microsoft, IBM, and Intel, have therefore started to use precise formal methods and scalable tools developed in this area. Precision of designed methods is necessary in order to ensure that the reported potential system errors are indeed errors, and that no bugs are omitted during the verification process. Scalability is required so that the tools work efficiently for large systems. Among the main techniques of formal verification are model checking, abstract interpretation, satisfiability modulo theory (SMT) reasoning, and first-order theorem proving. These methods are interrelated and many modern formal verification tools use a combination of them.

The objective of our proposal is to develop new methods advancing the applicability of first-order theorem proving in formal software verification. The project will explore and advance the power of the symbol elimination method, introduced recently by ourselves, for generating and proving program properties. Symbol elimination is based on first-order theorem proving and is fully automatic. It was the first ever method able to automatically discover complex program properties with quantifier alternations. The method is based on the following steps. Given a program loop, we first extend the language of the program with additional symbols, such as a loop counter. Next, we extract various information about the program that can be expressed by first-order formulas. These formulas can however use the introduced auxiliary symbols. Therefore, we run a superposition-based first-order theorem prover to eliminate the auxiliary symbols and obtain program properties expressed in the original language of the program.

In our project we will pay special attention to developing new theory and tools based on symbol elimination, solve automated reasoning problems arising in program verification, and thus prove automatically the validity of safety properties of software.

Safety properties ensure that something ”bad” never happens in the program. Verifying such properties becomes an especially challenging task when programs contain (nested) loops or recursion. The verification of such programs needs additional information, so-called program assertions, that express conditions to hold at certain intermediate points of the program. Typical auxiliary assertions are loop invariants, which describe program states that can be reached during program computations and thus are essential for safety property verification. The effectiveness of using symbol elimination in formal verification thus crucially depends on whether such assertions, even trivial ones, can be deduced automatically.

The **overall purpose** *of our project is to design new, unconventional methods for reasoning about program properties, by using symbol elimination in first-order theorem proving. We will implement world-leading tools that are likely to be used by others in the area, and apply our methods and tools on problems of industrial relevance.* The results of our project will provide fundamentally new ways of generating and proving program properties by symbol elimination,

and have a substantial impact both on the theory and practice of program verification. To achieve the aim of the project, the **specific goals of our project** includes the following three directions:

1. Developing new methods for *generating complex loop invariants* using symbol elimination. This research will include extensions of symbol elimination to reason about loops with unbounded data types such as arrays, lists, pointers, and heaps, and generate quantified invariants, possibly with quantifier alternations, fully automatically. We believe that the results will be highly original and ground-breaking, since the research we propose is the only method for property generation based on consequence finding by symbol elimination in first-order theorem provers.
2. Designing new techniques for *constructing small Craig interpolants* from proofs in full first-order logic with theories, and derive interpolants over complex data types. Craig interpolants will be further used in the process of invariant generation and model checking. We will develop efficient interpolation algorithms using symbol elimination in first-order theorem proving and apply our interpolation algorithms both for first-order and SMT reasoning. Our results will thus not be limited to reasoning in ground (that is, quantifier-free) theories, but provide automated methods to derive interpolants in quantified first-order theories, such as the theory of arrays, lists, pointers, and heaps.
3. Developing and implementing *efficient reasoning methods with theories and quantifiers*. Further progress in program verification needs efficient methods and tools for analyzing complex data structures, pointers and memory allocation. Reasoning with quantifiers and theories is the main ingredient required for a major breakthrough making such analysis feasible. We therefore expect that this research direction will significantly advance the use of theorem provers in verification. In particular, we will focus on theory reasoning in first-order superposition calculus and instantiation-based theorem proving, and design new ways of cooperation between first-order provers and SMT solvers.

The proposed research directions will be accompanied by the development of the award-winning first-order theorem prover Vampire, to which the PI of the project is actively contributing. In particular, we will implement new extensions of Vampire to support generating and proving program properties. The new features of Vampire will be *applied and evaluated on academic and industrial programs*, for the latter we have informal agreements about collaboration with verification teams at Microsoft Research and Intel.

We expect our project to have a deep and long-lasting impact in reasoning-based formal verification, yielding novel methods and scientific breakthroughs in using symbol elimination for software verification. The timeliness of the proposed research is witnessed by the growing academic and industrial interest in formal verification, which is due to availability of very powerful tools and methods based on previous research in the area.

2. Survey of the Field

We overview the state-of-the-art in invariant generation, interpolation, and reasoning in first-order theories. We give key references to existing methods and discuss the novel features of our project in relation to these methods. Due to the page limit, we do not discuss the wide range of tools in this area.

2.1 Quantified Invariant Generation. Invariants with quantifiers are important for the verification of programs over non-numeric data structures, such as arrays, due to the unbounded nature of array structures. Such invariants can express relationships among array elements and properties involving arrays and scalar variables of the loop, and thus significantly ease the verification task. Automated discovery of array invariants is therefore a challenging topic.

In [36, 10] loop invariants are inferred by predicate abstraction over a set of a priori defined predicates, while [13] employs constraint solving over invariant templates. A different approach

is given in [33], where input predicates in conjunction with interpolation are deployed to compute invariants. Unlike these techniques, in [30] we described a framework for automatically inferring array invariants without any user guidance, by introducing the symbol elimination method in first-order theorem proving. User guidance is also not required in [14, 4], but invariants are derived by abstract interpretation over array indexes [14] and array segments [4]. However, these approaches can only infer universally quantified invariants. Our approach in [30] is fundamentally different from the afore-cited techniques. The use of symbol elimination in quantified first-order theories has not yet been addressed by other methods. Based on our previous work [30, 20], we will provide an automatic method for generating quantified invariants, including those with quantifier alternations, which cannot be handled by other techniques.

2.2 Interpolation in Formal Verification. Interpolation [5] is an important technique in verification and static analysis of programs, in particular in invariant generation and bounded model checking, see e.g. [23, 33]. The key ingredient of theorem proving based interpolation is the notion of a so-called local proof, since local proofs admit efficient interpolation algorithms – see [32] and our work [31].

There are several interpolant generation algorithms for various theories. For example, [23, 3] derive interpolants from resolution proofs in the ground theory of linear arithmetic and uninterpreted functions. The approach described in [35] 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 [33] computes quantified interpolants from first-order resolution proofs over scalars, arrays and uninterpreted functions. Craig interpolation has recently been generalized to so-called tree interpolants for their use in concurrent [11] and recursive [15] programs. Results of [12] can however be applied only for interpolation in the quantifier-free theory of uninterpreted functions and linear arithmetic. In contrast to these techniques, in [31] we gave an algorithm for interpolant extraction from local proofs in any sound calculus, including calculi for various theories, such as arithmetic or a theory of arrays. Thus, our interpolation method in [31] is not limited to decidable theories for which interpolation algorithms are known. However, a consequence of this generality is that, unlike [23, 3, 35, 11, 15], in [31] we do not guarantee finding interpolants even for decidable theories or for theories where interpolants are known to exist. In [22] we addressed qualitative aspects of interpolation, and gave a general framework to minimize interpolants wrt their size and number of quantifiers. Our work in [22] was the first method providing a flexibility in computing interpolants of different strength and structure.

2.3 Reasoning with Theories and Quantifiers. For reasoning about properties involving both quantifiers and theories, SMT solvers and first-order theorem provers are complementary. SMT solvers, see e.g. Z3 [7] and Yices [8], are powerful when it comes to proving formulas without quantifiers in combinations of common first-order theories such as linear arithmetic, arrays and uninterpreted functions, but are inefficient for reasoning in full first-order logic. On the contrary, first-order theorem provers, see e.g. Vampire [19, 21] and iProver [26] are very efficient in working with quantifiers, but have essentially no support for theory reasoning.

Combining first-order proving and theory reasoning is very hard. For example, some simple fragments of combining first-order reasoning with linear arithmetic are already Π^1_1-complete [27]. 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, 8] and (incomplete) axiomatization of theories in first-order provers [30, 20]. A tighter integration is described in [6, 1]. Motivated by our previous results using Vampire [30, 20], we will investigate how to integrate theory reasoning with superposition-based first-order theorem proving and implement our results in Vampire. We believe that our project will enable solving problems beyond the reach of other methods.

3. Project Description

3.1 Methodology. To formally verify computer programs means to apply mathematical arguments to prove the correctness of systems. Since systems have bugs, formal verifications aims at finding and correcting such bugs. There are theoretical results showing undecidability of almost every problem related to program correctness, in particular to reason and prove properties about the logically complex part of the code characterized by (nested) loops and recursion.

The practice of designing new tools supporting the verification of program with loops meets very hard everyday challenges, and faces the problem of generating and reasoning about loop properties, in particular loop invariants. Providing loop properties manually requires a considerable amount of work by highly qualified personnel and thus often makes verification prohibitively expensive. Therefore, generation of program properties without user-provided annotations is invaluable in making program analysis and verification economically feasible.

In our project we will focus on developing methods for the automated synthesis of loop properties of complex programs with unbounded data structures, such as arrays, lists, pointers, and heaps. First, we will focus on the generation of loop invariants. Next, as Craig interpolation provides an alternative way to reason about loop invariants, we will compute interpolants and use them further in invariant synthesis. Finally, as loop properties over unbounded data types involve quantifiers and theory symbols, we will address reasoning in full first-order (FO) logic with theories. The common method of our project is the *symbol elimination method*, introduced by ourselves in [30].

In a nutshell, symbol elimination is based on the following ideas. Suppose we have a program P with a set of variables V. The set V defines the language of P. We extend the language of P to a richer language V^+ by adding functions and predicates, such as loop counters. After that, we automatically generate a set Π of FO properties of the program in the extended language V^+, by using techniques from symbolic computation and theorem proving. These properties are valid properties of the program, however they use the extended language V^+. Then we derive from Π program properties in the original language P, thus "eliminating" the symbols in $V^+ \setminus V$.

The distinctive feature of the symbol elimination method is its power to automatically generate, with no user guidance, statements expressing complex computer program properties, including those with quantifier alternations. The method uses, in a new way, the power of a superposition-based FO theorem prover and symbolic computation to derive program properties that hold at intermediate points of the program. Such properties are crucial to ensure the safety of computer systems.

To illustrate the need of quantified loop properties, consider the program given in Figure 1, written in a C-like syntax. The program fills an array B with the non-negative values of a source array A, and an array C with the negative values of A. It is not hard to derive that at after any iteration n of this loop (assuming $0 \leq n \leq k$), the linear invariant relation $a = b + c$ holds. For example, this property can be derived by the methods of [34, 29]. However, such a property does not give us much information about the arrays A, B, C and their relationships. For example, one may want to derive the following properties of the loop (n denotes the loop counter):

```
a := 0; b := 0; c := 0;
while (a ≤ k) do
  if A[a] ≥ 0
    then B[b] := A[a];b := b + 1;
    else C[c] := A[a];c := c + 1;
  a := a + 1;
end do
```

Figure 1: Array partition.

1. Each of $B[0], \ldots, B[b-1]$ is non-negative and equal to one of $A[0], \ldots, A[a-1]$. Formulating this property in FO logic, yields the loop invariant:

$$(\forall p)(0 \leq p < n \implies B[p] \geq 0 \ \wedge \ (\exists k)(0 \leq k < a \wedge A[k] = B[p])).$$

2. For every $p \geq b$, the value of $B[p]$ is equal to its initial value. Writing it in FO logic, this loop invariant is:
$$(\forall p)(p \geq b \implies B[p] = B_0[p]),$$
where B_0 denotes the value of B before the first iteration of the loop.

These loop properties in fact describe much of the intended function of the loop and can be used to verify properties of programs manipulating arrays in which this loop is embedded. Note however that the first invariant, when formulated in first-order logic, requires quantifier alternations, whereas the second property involves only universal quantification. Generating such properties requires reasoning in full FO logic with theories, in our example in the FO theory of arrays and integers.

This proposal addresses the quest of generating such quantified program properties. We aim at improving our symbol elimination method for generating quantified loop invariants and interpolants for programs with complex flow and unbounded data structures, such as arrays, lists, pointers, and heaps. As reasoning about such programs requires reasoning with both theories and quantifiers, our project will also address proving in FO logic with quantified theories. Our project will thus be structured in three working packages (WP):

(WP 1). Inferring quantified invariants in theories with unbounded data types;
(WP 2). Generating quantified interpolants from proofs in such theories;
(WP 3). Efficient reasoning with theories and quantifiers.

These work packages are strongly related and depend on each others' results. For example, properties generated in (WP1) and (WP2) will be used for testing (WP3), and any development in (WP3) will improve the results of (WP1)-(WP2). In what follows, we describe the proposed research for each work package, along with their deliverables and milestones.

WP1. Quantified Loop Invariant Generation. In (WP1) we plan to address the following five aspects of symbol elimination:
(WP1.1) We will first design various *FO theories of data types*. For doing so, auxiliary predicates expressing properties over the content of arrays, lists, pointers, and heaps will be added. Reasoning in FO theories of such unbounded data types will be necessary for extending symbol elimination to derive quantified invariants. (WP1.1) will use results of (WP3) and help (WP3) by providing hard benchmarks for reasoning with quantifiers and theories under study.
(WP1.2) We will extend symbol elimination with *generation of various classes of clause sets with eliminated symbols*. One interesting extension comes with inferring a minimized set of invariants to avoid redundant clauses that imply each other.
(WP1.3) Further development of symbol elimination will also require *analyzing programs with nested loops and complex arithmetic* followed by or interleaved with theorem proving and symbolic computation. For such analysis, we will apply advanced recurrence solving techniques from [29] together with the FO interpolation results of (WP2), as well as with the best existing static analysis methods and tools, for example [36, 14, 4].
(WP1.4) An open challenge of (WP1) is to show when symbol elimination is (or can be made) a *complete approach for invariant generation*. By completeness we mean that if a program has a quantified invariant of a certain form, our approach will find invariants implying them. We will also generalize (WP2) to *generate interpolants from which inductive invariants are inferred*.
(WP1.5) We will *extend Vampire* to analyze complex programs over various data types, and generate invariants using symbol elimination. Our methods will be *tested on benchmarks*, including large industrial examples coming from Intel and Microsoft (see our collaboration list).
Milestones and Deliverables:

- first-order theories of data types;

- new and complete invariant generation methods in such theories using symbol elimination;
- a first-order theorem prover with invariant generation;
- case studies of invariant generation for large programs.

WP2. Generating Quantified Interpolants. (WP2) will be structured on the following five applications of symbol elimination:
(WP2.1) We will use symbol elimination in FO theorem proving and design new methods to *extract interpolants from arbitrary FO superposition proofs*. In particular, we will address interpolation in FO non-local proofs.
(WP2.2) Using the theorem proving engine of (WP3), we will design efficient superposition algorithms for *generating interpolants in the combination of various FO theories*, such as arithmetic, unbounded data structures and uninterpreted function symbols. Various methods of *minimizing interpolants* will also be studied, possibly based on proof transformations.
(WP2.3) We will use generalized Craig interpolants, called tree interpolants [11, 15], and *compute FO tree interpolants for concurrent programs*.
(WP2.4) We will *extend Vampire* with efficient interpolation algorithms for complex programs, and make Vampire work together with model checkers, such as CPAChecker [2]. Our results will be evaluated on various benchmarks, including examples coming Intel and Microsoft (see our collaboration list).
(WP2.5) A the quality of interpolation algorithms can only be evaluated in the context of practical software verification, our results will be *integrated in concrete verification tools*. We will experiment how the quality of minimized interpolants effects the efficiency of interpolation-based verification methods, in particular invariant generation, as discussed in (WP1).
Milestones and Deliverables:

- new methods of interpolation in FO theories using symbol elimination;
- an interpolating theorem prover for first-order formulas with data types;
- integration into model checking based verification tools;
- case studies of interpolation on concrete programs of industrial relevance.

WP3. Efficient Reasoning with Theories and Quantifiers. (WP3) will consist of the following four research directions:
(WP3.1) We will first add *existing SMT algorithms* for the theory of integers, reals and arrays to Vampire, study theory behavior and understand the best ways of using them within a FO theorem prover. After that we are interested in *new, less understood, theories*, such as those of pointers, queues and heaps.
(WP3.2) We will next extend FO theorem provers with *sound but incomplete theory axiomatizations*. In particular, using testcases from (WP1) and (WP2), we will study necessary and sufficient sets of theory axioms from which interesting program properties can be derived in extensions of the superposition calculus. We will need to understand what are the best simplification rules, literal selections, variable orderings and maybe also add some theory rules.
(WP3.3) We will address the *instantiation based theorem proving* framework of [9, 26], and use (WP3.1) for proving ground instanced of a FO problem. Proofs produced by (WP3.1) will be analyzed with the purpose of generating proof certificates. The proof certificates will be propagated back to the FO theorem prover, and new ground instances will be generated. However, understanding what parts of the proof are relevant is a non-trivial task, and requires a deeper understanding of how the SMT engine and the FO prover can work together.
(WP3.4) Our work will be based on *empirically driven development* of methods, resulting in improvements in algorithms and their use and combination in Vampire. These improvements will be evaluated on examples coming from (WP1) and (WP2), as well on industrial benchmarks coming from Microsoft and Intel (see our collaboration list).
Milestones and Deliverables:

- theory reasoning in superposition calculus;
- theory reasoning in instantiation-based theorem proving;
- new reasoning methods combining first-order provers and SMT solvers;
- case studies on academic and industrial examples.

3.2. Timetable and Organization. The project will last 5 years and will be led by Laura Kovács (PI). The work will be carried out by the PI and a PhD student under her supervision. There will be weekly project meetings to exchange ideas and discuss the progress.

The work plan of the project is illustrated in Table 2 and is is designed according to the dependencies among WPs. (WP1) and (WP2) crucially depend on (WP3), and therefore work in (WP3) is planned during the entire phase of the project. The work on (WP1) will start already in year 1, yielding results that can improve the research of (WP2) in year 2. Results of all WPs will be joined and tested in year 3 of the project, identifying this way the essential research directions for years 4 and 5. The working plan in years 4 and 5 is similar as in years 1 and 2.

Work Package	Year1	Year2	Year3	Year4	Year5
WP1	✓		✓	✓	
WP2		✓	✓		✓
WP3	✓	✓	✓	✓	✓

Figure 2: The GenPro Work Plan.

Our results will be disseminated as follows. We will publish articles in leading journals and conferences. We will develop the world-leading theorem prover Vampire. We will participate in tool competitions in the area. We will al organize workshops and tutorials at top international conferences, and scientific seminars at Chalmers.

4. Significance

Analyzing and verifying large computer programs requires non-trivial automation. Automatic generation and proving program properties is a key step to such an automation. Reasoning about program properties becomes an especially challenging task for programs with complex flow and, in particular, with loops. Further progress in reasoning about software therefore crucially depends on the existence of efficient methods and tools analyzing programs with nested loops and complex data structures. Our project addresses this challenge, and brings new non-standard approaches to generate and prove program properties via symbol elimination in first-order theorem proving. Our project will give unique, novel and significant contributions for the following reasons:

- Symbol elimination offers a fully automatic framework to generate invariants with arbitrary quantifier alternations, without using user/provided templates and annotations. By extending the application of symbol elimination to more complex programs and unbounded data types, our project will automatically infer properties that cannot be generated by other approaches.
- Generating quantified loop invariants using theorem provers is a new research area, initiated by our work in [30] over programs with integers and arrays. No technique is known to infer invariants over other data structures or theories by using theorem proving.
- Interpolation was mainly studied and implemented in the context of quantifier-free theories, see e.g. [23, 3, 11]. Our results will however target interpolation in fist-order theories, and will not limited to decidable theories. That is, results of our project can even be applied to theories for which no interpolation algorithms are known.
- We will design and implement efficient methods combining superposition theorem proving and SMT reasoning. We expect that our methods will be as efficient as first-order provers in reasoning with quantifiers and as efficient as SMT solvers in ground theory reasoning, solving thus some problems that are beyond the scope of existing technologies.

The proposed research for a PhD student is in the rapidly developing area of formal verification. This area attracts a lot of interest in industry, which means that successful PhD students can be employed by industry and/or apply their results in an industrial setting. The project will use and design rigorous techniques in computer science (formal methods), mathematics (combinatorics and computer algebra), and logic (reasoning and decision procedures), to support systems engineering methods and tools. Our project will thus also encourage interactions with mathematicians, logicians, and computer scientists.

5. Preliminary Results

Our work [30] on symbol elimination in theorem proving resulted in the first ever method to generate complex loop invariants with quantifier alternations. Symbol elimination, unlike other known methods, is completely automatic and requires no annotation or patterns. When applying symbol elimination on industrial safety-critical applications, in [20] we have shown that the loop properties obtained by symbol elimination could replace human-produced annotations in over 80% of test cases. The impact of symbol elimination method in formal verification is witnessed by the number of citations of our 2009 paper [30]: 63 international citations since 2009.

Since recent results on Craig interpolation provide an alternative approach to invariant generation, we extended symbol elimination to interpolation and gave a new algorithm for building interpolants from first-order local proofs [31]. Further, we designed a new method for computing small interpolants [22], by considering proof transformations and encoding the interpolant minimization problem as a pseudo-boolean optimization problem. The evaluation of our method on bounded model checking examples shows that minimization considerably decreases the interpolant size [18].

We also investigated the use of symbol elimination in symbolic computation and gave a complete algorithm for generating all polynomial invariants, by using symbol elimination in Gröbner basis computation [29]. In addition, when extending symbol elimination to quantifier elimination methods, our results in [16] gave an automated approach to derive linear inequality invariants. Interestingly, the application of symbol elimination in symbolic computation turned out to be a useful method for the timing analysis of real-time system [24].

On the implementation side, we made the Vampire theorem prover into an interpolating first-order prover with invariant generation [19, 21]. The PI of this project started her work on Vampire in August 2008. As a result of the PIs involvement in the development of Vampire, starting from 2009 the number of Vampire users has significantly increased. For example, in 2010–2011 Vampire was downloaded more than 1,000 times (counting only real downloads, with verified email addresses), that is, over 10 times a week.

We also designed the symbolic computation engine Aligator to derive polynomial invariants [28, 17]. Our work in using symbol elimination for the timing analysis of software has materialized in the r-TuBound software package [25].

Summarizing, I believe that my preliminary results of symbol elimination in formal verification 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 enables me to make significant developments in both theory and implementation and ensures that the project successfully meets its scientific goals.

6. National and International Collaborations

We plan to collaborate with well-known researchers in the area, as listed below.

Chalmers. Starting with April 2013, I have joined Chalmers as an associate professor. My appointment to Chalmers was made with a purpose of building a competitive research group in formal methods and strengthen Chalmers' research expertise and international scientific competitiveness in formal verification. Topics of the current project are in the center of interest of several

researchers at Chalmers, including top scientists such as Prof. Wolfgang Ahrendt, Prof. Koen Claessen, Dr. Moa Johansson, Prof. Andrei Sabelfeld, and Prof. Geraldo Schneider. Our project will thus encourage cross-institutional collaboration at Chalmers and will benefit from the research expertise and the infrastructure of the department. To be more precise, our collaboration plans include the following actions: collaboration with Prof. Ahrendt and Prof. Schneider on invariant generation and verification tools (WP1)-(WP2), joint work with Prof. Claessen and Dr. Johansson on theorem proving (WP3), and collaboration with Prof. Sabelfeld on reasoning about properties over concurrent data structures such as queues and heaps (WP1).

Austrian RiSE. Together with 8 other researchers, I am a founding member of the Austrian Society for Rigorous Systems Engineering (ARiSE). The current project naturally targets collaboration with the other members of ARiSE, in particular with Prof. Thomas A. Henzinger (IST Austria) and Prof. Helmut Veith (TU Vienna) on (WP1) and (WP2); and with Prof. Armin Biere (JKU Linz) on (WP3).

The University of Manchester, UK. Our main results on symbol elimination in first-order theorem proving and Vampire were obtained in collaboration with Prof. Andrei Voronkov at the University of Manchester. We are keen to continue this collaboration on all parts of the project – (WP1), (WP2), and (WP3). Joint work is also scheduled during the entire development of Vampire. The collaboration between the PI and Prof. Voronkov has started in 2008, yielding several joint publications that are of direct relevance to the proposed project.

Industrial Collaborators. We propose joint work with Dr. Nikolaj Bjørner from Microsoft Research, US on (WP1) and (WP3), as well as with Dr. Zurab Khasidashvili from Intel Haifa on (WP1). Industrial examples coming from Microsoft and Intel will be used to evaluate the results of our project. The PI has recently visited Microsoft Research Redmond in February 2013 with the aim of starting deeper collaboration and joint work. The PI already consulted Intel in December 2010, and has recently restarted her consulting activities with Intel.

7. Other Grants

I have joined Chalmers as an associate professor in April 2013. The current application is my first VR proposal, and I do not apply for other VR grants. The VR grant would make it possible for me to start my own independent research group in formal methods at Chalmers. The VR grant would also provide me the opportunity to integrate myself in the Swedish scientific community.

Literature

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

[2] D. Beyer and M. E. Keremoglu. CPAchecker: A Tool for Configurable Software Verification. In *Proc. of CAV*, pages 184–190, 2011.

[3] A. Cimatti, A. Griggio, and R. Sebastiani. Efficient Interpolant Generation in Satisfiability Modulo Theories. In *Proc. of 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 *Proc. of POPL*, pages 105–118, 2011.

[5] W. Craig. Three uses of the Herbrand-Gentzen Theorem in Relating Model Theory and Proof Theory. *J. Symbolic Logic*, 22(3):269–285, 1957.

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

[7] L. de Moura and N. Bjorner. Z3: An Efficient SMT Solver. In *Proc. of TACAS*, pages 337–340, 2008.

[8] B. Dutertre and L. de Moura. A Fast Linear-Arithmetic Solver for DPLL(T). In *Proc. of CAV*, pages 81–94, 2006.

[9] H. Ganzinger and K. Korovin. Theory Instantiation. In *Prof. of LPAR*, pages 497–511, 2006.

[10] S. Gulwani, B. McCloskey, and A. Tiwari. Lifting Abstract Interpreters to Quantified Logical Domains. In *Proc. of POPL*, pages 235–246, 2008.
[11] A. Gupta, C. Popeea, and A. Rybalchenko. Predicate Abstraction and Refinement for Verifying Multi-Threaded Programs. In *Proc. of POPL*, pages 331–344, 2011.
[12] A. Gupta, C. Popeea, and A. Rybalchenko. Solving Recursion-Free Horn Clauses over LI+UIF. In *APLAS*, pages 188–203, 2011.
[13] A. Gupta and A. Rybalchenko. InvGen: An Efficient Invariant Generator. In *Proc. of CAV*, pages 634–640, 2009.
[14] N. Halbwachs and M. Peron. Discovering Properties about Arrays in Simple Programs. In *Proc. of PLDI*, pages 339–348, 2008.
[15] M. Heizmann, J. Hoenicke, and A. Podelski. Nested Interpolants. In *Proc. of POPL*, pages 471–482, 2010.
[16] T. Henzinger, T. Hottelier, and L. Kovács. Valigator: A Verification Tool with Bound and Invariant Generation. In *Proc. of LPAR*, LNCS, pages 333–342, 2008.
[17] T. Henzinger, T. Hottelier, L. Kovács, and A. Rybalchenko. Aligators for Arrays (Tool Paper). In *Proc. of LPAR*, pages 348–356, 2010.
[18] K. Hoder, A. Holzer, L. Kovács, and A. Voronkov. Vinter: A Vampire-Based Tool for Interpolation. In *Proc. of APLAS*, 2012. To appear.
[19] K. Hoder, L. Kovács, and A. Voronkov. Interpolation and Symbol Elimination in Vampire. In *Proc. of IJCAR*, pages 188–195, 2010.
[20] K. Hoder, L. Kovács, and A. Voronkov. Case Studies on Invariant Generation Using a Saturation Theorem Prover. In *Proc. of MICAI*, pages 1–15, 2011.
[21] K. Hoder, L. Kovács, and A. Voronkov. Invariant Generation in Vampire. In *Proc. of TACAS*, 2011.
[22] K. Hoder, L. Kovács, and A. Voronkov. Playing in the Grey Area of Proofs. In *Proc. of POPL*, pages 259–272, 2012.
[23] R. Jhala and K. L. McMillan. A Practical and Complete Approach to Predicate Refinement. In *Proc. of TACAS*, pages 459–473, 2006.
[24] J. Knoop, L. Kovács, and J. Zwirchmayr. Symbolic Loop Bound Computation for WCET Analysis. In *Proc. of PSI*, pages 224–239, 2011.
[25] J. Knoop, L. Kovács, and J. Zwirchmayr. r-TuBound: Loop Bounds for WCET Analysis (Tool Paper). In *Proc. of LPAR*, pages 435–444, 2012.
[26] K. Korovin. iProver - An Instantiation-based Theorem Prover for First-order Logic. In *Proc. of IJCAR*, pages 292–298, 2008.
[27] K. Korovin and A. Voronkov. Integrating Linear Arithmetic into Superposition Calculus. In *Proc. of CSL*, pages 223–237, 2007.
[28] L. Kovács. Aligator: A Mathematica Package for Invariant Generation. In *Proc. of IJCAR*, pages 275–282, 2008.
[29] L. Kovács. Reasoning Algebraically About P-Solvable Loops. In *Proc. of TACAS*, pages 249–264, 2008.
[30] L. Kovács and A. Voronkov. Finding Loop Invariants for Programs over Arrays Using a Theorem Prover. In *Proc. of FASE*, pages 470–485, 2009.
[31] L. Kovács and A. Voronkov. Interpolation and Symbol Elimination. In *Proc. of CADE*, pages 199–213, 2009.
[32] K. L. McMillan. An Interpolating Theorem Prover. In *Proc. of TACAS*, pages 16–30, 2004.
[33] K. L. McMillan. Quantified Invariant Generation Using an Interpolating Saturation Prover. In *Proc. of TACAS*, pages 413–427, 2008.
[34] A. Mine. The Octagon Abstract Domain. In *Proc. of RE*, pages 310–319, 2001.
[35] A. Rybalchenko and V. Sofronie-Stokkermans. Constraint Solving for Interpolation. In *Proc. of VMCAI*, pages 346–362, 2007.
[36] S. Srivastava and S. Gulwani. Program Verification using Templates over Predicate Abstraction. In *Proc. of PLDI*, pages 223–234, 2009.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2013-45614-105023-81

Name of applicant
Kovacs, Laura

Date of birth
800426-2906

Title of research programme
GenPro: Generating and Proving Program Properties via Symbol Elimination

Appendix B

Curriculum vitae

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Appendix B – Curriculum Vitae

1. Higher education degree

- 2002 July: B.Sc. in Mathematics and Computer Science, West University of Timişoara, Romania;
- 2004 February: M.Sc. in Computer Science, West University of Timişoara, Romania;
- 2007 October: Ph.D. in Computer Science with highest distinction, Research Institute for Symbolic Computation (RISC-Linz), Johannes Kepler University Linz, Austria.

2. Doctoral degree

- 2007 October: Ph.D. in Computer Science with highest distinction, RISC-Linz, Johannes Kepler University Linz, Austria; PhD thesis title: *Automated Invariant Generation by Algebraic Techniques for Imperative Program Verification in Theorema*; Supervisor: Prof. Tudor Jebelean.

3. Postdoctoral positions

- 2009-2010: Postdoctoral scientist in the Programming Methodology group of Prof. Peter Müller, ETH Zürich, Switzerland;
- 2007-2009: Postdoctoral scientist in the Models and Theory of Computation group of Prof. Thomas A. Henzinger, EPFL Lausanne, Switzerland.

4. Docent level

- 2012 November: Habilitation (Docent) in Computer Science, Vienna University of Technology (TU Vienna), Austria; Habilitation thesis title: *Symbol Elimination in Program Analysis*;
- 2013 April: Associate Professor (Docent) in Computer Science, Chalmers University of Technology, Sweden.

5. Present position

- 2013 April - present: Associate Professor (Docent) in Computer Science, Chalmers University of Technology, Sweden; Tenured position; Percentage of research in the position: 80%.

6. Previous positions

Beside the postdoctoral positions above, the previous positions of Kovács include:

- 2010-2013: Austrian Science Fund (FWF) Hertha-Firnberg research fellow and Assistant Professor, TU Vienna, Austria;
- 2003-2007: Ph.D. researcher at RISC-Linz, Austria.

Invited positions:

- 2012 June: Visiting professor, University Joseph Fourier, Grenoble, France;
- 2007 September: Research visitor, Tsukuba University, Japan. Host: Prof. Tetsuo Ida.

7. Deductible time

No deductible times are applicable.

8. Supervision

- Main PhD supervisor at TU Vienna of Dipl.-Ing. Jakob Zwirchmayr (2010-2013, co-supervisor: Prof. Jens Knoop); expected date of defence: 2013 October;
- Main PhD supervisor at TU Vienna for MSc. Ioan Drãgan (2011-2014); expected date of defence: end of 2014.

9. Prizes, Honors, and Distinctions

- 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 International Computer Science Symposium in Russia (CSR), Russia;
- 2007 September: Japan Society for the Promotion of Science fellowship, Tsukuba University, Japan;
- 2005: Best student presentation award, 2nd South-East European Workshop on Formal Methods (SEEFM), Macedonia.

10. Consulting Activities for the Software and Hardware Industry

- Dassault Aviation, France – consulting on theorem proving and program analysis (2009 - 2010);
- Intel Haifa, Israel – consulting on hardware verification (December 2010, restarted in October 2012).

11. Selected Scientific Services in the Research Community

A complete list of activities is available at `http://www.chalmers.se/cse/SV/kontakt/personal/laura-kovacs`. Here we report on the most important ones.

Invited Lectures. Kovács gave invited talks at 4 international conferences and workshops, including the Int. Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2011. Kovács gave 22 invited lectures at institutions ranging from industry to academia, including Microsoft Research, Intel, IBM, AdaCore, SRI International, Helsinki Institute for Information Technology, and Verimag Grenoble. She was a tutorial speaker at 5 international conferences, including the Int. Conference on Automated Deduction (CADE), 2011; the Mexican International Conference on Artificial Intelligence (MICAI), 2011.

Guest Editorship of Journals. Special Issue of the J. of Logic and Algebraic Programming on "Automated Specification and Verification of Web Systems", 2012; Special Issue of the J. of Symbolic Computation on "Invariant Generation and Advanced Techniques for Reasoning about Loops", 2011 and 2009; Special Issue of the J. of Applied Logic on "Automated Specification and Verification of Web Systems", 2011.

Conference, Seminar and Training School Organization. Dagstuhl Seminar 12461 on "Games and Decisions for Rigorous Systems Engineering", 2012; ICNPAA 2012 World Congress: 9th International Conference on Mathematical Problems in Engineering, Aerospace and Sciences, TU Vienna, 2012; ARiSE/VCLA Winter School on Verification, TU Vienna, 2012.

Program Committee Chair. Int. Symposium on Symbolic Computation in Software Science (SCSS), 2013; Int. Symposium on Symbolic and Numeric Algorithms for Scientific Computing - Logic and Programming track (SYNASC), 2012 and 2013; Int. Workshop on Automated Specification and Verification of Web Systems (WWV), 2011 and 2010; Int. Workshop on Invariant Generation (WING), 2010 and 2009.

Memberships in Program Committees. Kovács was a program committee member of 19 international conferences and workshops, including the Int. Conference on Certified Programs and Proofs (CPP), 2012; Int. Conference on Computer Science Logic (CSL), 2012; Int. Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR), 2009-2012; Int. Conference on Formal Engineering Methods (ICFEM), 2011; Int. Symposium on Symbolic and Algebraic Computation (ISSAC), 2010.

Selected Journal and Conference Referee. Kovács has additionally acted as a reviewer for 1 book chapter, 1 2 international journals, and 11 international conferences and workshops. This list includes reviewing for 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 Computer Aided Verification (CAV), 2013; Int. Conference on Programming Language Design and Implementation (PLDI), 2010 and 2009.

Memberships in Professional Organizations. Founding member of the Austrian Society for Rigorous Systems Engineering (ARiSE), since 2010.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2013-45614-105023-81

Name of applicant
Kovacs, Laura

Date of birth
800426-2906

Title of research programme
GenPro: Generating and Proving Program Properties via Symbol Elimination

Appendix C

List of publications

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Appendix C – Publication List

Computer science is a very conference-oriented field. Therefore, compared to other fields, there is significantly more importance on conference papers. All conference papers listed below have been subject to high standards of peer reviews from top tier conferences, many of them with very low acceptance rates (some below 20%) such as POPL, TACAS, CADE, and LPAR.

As requested by VR, only publications for the last eight years, that is April 2005 - April 2013, are listed below. The five publications that are the most relevant to the project are publications numbered as 12, 22, 26, 27, and 31. The publications are highlighted and marked using an asterisk (∗). Citations of these articles are given according to Google Scholar (GS).

According to Google Scolar, my total citation index is 488, and my h-index is 12 (source `http://scholar.google.at/`, as of April 9, 2013). My complete list of publications is available at:`http://www.chalmers.se/cse/SV/kontakt/personal/laura-kovacs`.

1 Peer-Reviewed Journal Articles

Note: Co-authored publications present work done in an equally distributed joint collaboration.

[1] Bruno Buchberger, Adrian Craciun, Tudor Jebelean, Laura Kovács, Temur Kutsia, Koji Nakagawa, Florina Piroi, Nikolaj Popov, Judith Robu, Markus Rosenkranz, and Wolfgang Windsteiger. Theorema: Towards Computer-Aided Mathematical Theory Exploration. *Journal of Applied Logic*, 4(4):470–504, 2006.

[2] Adalbert Kovács and Laura Kovács. The Lagrange Interpolation Formula in Determining the Fluid's Velocity Potential through Profile Grids. *Bulletins for Applied and Computer Mathematics*, CVIII(2252):126–135, 2005.

[3] Laura Kovács, Tudor Jebelean, and Adalbert Kovács. Practical Aspects of Algebraic Invariant Generation for Loops with Conditionals. *Bulletins for Applied and Computer Mathematics*, CVIII(2251):116–125, 2005.

[4] Laura Kovács, Nikolaj Popov, and Tudor Jebelean. A Verification Environment for Imperative and Functional Programs in the Theorema System. *Annals of Mathematics, Computing and Teleinformatics (AMCT), TEI Larissa, Greece*, 1(2):27–34, 2005.

2 Peer-Reviewed Conference Contributions

Note: All papers have been published in proceedings.
Co-authored publications present work done in an equally distributed joint collaboration.
The co-authored publications (since 2007) have their authors listed in alphabetical order.

2.1 Refereed Papers in Conference Proceedings

[5] Armelle Bonenfant, Hugues Cassé, Marianne De Michiel, Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. FFX: A Portable WCET Annotation Language. In *Proceedings of the ACM International Conference on Real-Time and Network Systems (RNTS 2012)*, pages 91–100. ACM, 2012.

[6] Kryštof Hoder, Andreas Holzer, Laura Kovács, and Andrei Voronkov. Vinter: A Vampire-Based Tool for Interpolation. In *Proceedings of Asian Symposium on Programming Languages and Systems (APLAS 2012)*, volume 7705 of *LNCS*, pages 148–146, 2012.

[7] Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. r-TuBound: Loop Bounds for WCET Analysis (Tool Paper). In *Proceedings of International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-18)*, volume 7180 of *LNCS*, pages 435–444, 2012.

[8] Adalbert Kovács and Laura Kovács. A Hodographic Approximation Method for Analyzing the Fluid Motion Through Network Profiles. In *Proceedings of 23rd International DAAAM Symposium*, volume 23 of *DAAAM International*, pages 125–128.

[9] Adalbert Kovács, Laura Kovács, and Levente Kovács. The Boundary Element Method in the Study of Non-Stationary Movements Through Network Profiles. In *Proceedings of 13th International Conference on Mathematics and its Applications (ICMA)*, Scientific Bulletin of Politehnica University Timişoara (ISSN 1224-6069), pages 241–248.

[10] Laura Kovács and Adalbert Kovács. Symbol Elimination and its Applications in Program Verification. In *Proceedings of 13th International Conference on Mathematics and its Applications (ICMA)*, Scientific Bulletin of Politehnica University Timişoara (ISSN 1224-6069), pages 329–334.

[11] Laura Kovács, Béla Paláncz, and Levente Kovács. Solving Robust Glucose-Insulin Control by Dixon Resultant Computations. In *Proceedings of International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2012)*, IEEE Computer Society 978-0-7695-4934-7/12, pages 53–61, 2012.

[12] *** Kryštof Hoder and Laura Kovács and Andrei Voronkov**. **Playing in the Grey Area of Proofs**. In **Proceedings of ACM SIGACT-SIGPLAN International Symposium on Principles of Programming Languages (POPL)**, volume **47** of **ACM SIGPLAN Notices**, pages **259–272**, **2012**. **(GS): 4**.

[13] Kryštof Hoder, Laura Kovács, and Andrei Voronkov. Case Studies on Invariant Generation Using a Saturation Theorem Prover. In *Proceedings of the Mexican International Conference on Artificial Intelligence (MICAI)*, volume 7094 of *LNAI*, pages 1–15, 2011.

[14] Kryštof Hoder, Laura Kovács, and Andrei Voronkov. Invariant Generation in Vampire. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 6605 of *LNCS*, pages 60–64, 2011.

[15] Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. Symbolic Loop Bound Computation for WCET Analysis. In *Proceedings of the International Conference on Perspectives of System Informatics (PSI)*, volume 7162 of *LNCS*, pages 224–239, 2011.

[16] Adalbert Kovács and Laura Kovács. Analyzing the Fluid Motion Through Network Profiles Using the Boundary Element Method. In *Proceedings of 22nd International DAAAM Symposium: Intelligent Manufactoring and Automation*, volume 2 of *DAAAM International*, pages 1147–1148, 2011.

[17] Laura Kovács and Adalbert Kovács. Aligator: Experiments and Limitations. In *Proceedings of 22nd International DAAAM Symposium: Intelligent Manufactoring and Automation*, volume 2 of *DAAAM International*, pages 1145–1146, 2011.

[18] Laura Kovács, Georg Moser, and Andrei Voronkov. On Transfinite Knuth-Bendix Orders. In *Proceedings of the International Conference on Automated Deduction (CADE)*, volume 6803 of *LNAI*, pages 384–399. Springer, 2011.

[19] Régis Blanc, Thomas A. Henzinger, Thibaud Hottelier, and Laura Kovács. ABC: Algebraic Bound Computation for Loops. In *Proceedings of the International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR-16)*, volume 6355 of *LNAI*, pages 103–118, 2010.

[20] Thomas A. Henzinger, Thibaud. Hottelier, Laura Kovács, and Andrey Rybalchenko. Aligators for Arrays (Tool Paper). In *Proceedings of the International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR-17)*, volume 6397 of *LNCS*, pages 348–356, 2010.

[21] Thomas A. Henzinger, Thibaud Hottelier, Laura Kovács, and Andrei Voronkov. Invariant and Type Inference for Matrices. In *Proceedings of the International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 5944 of *LNCS*, pages 163–179, 2010.

[22] *** Kryštof Hoder and Laura Kovács and Andrei Voronkov**. **Interpolation and Symbol Elimination in Vampire**. In **Proceedings of the International Joint Conference on Automated Reasoning (IJCAR)**, volume **6173** of **LNCS**, pages **188–195**, **2010**. **(GS): 21**.

[23] Laura Kovács. A Complete Invariant Generation Approach for P-solvable Loops. In *Proceedings of the International Conference on Perspectives of System Informatics (PSI)*, volume 5947 of *LNCS*, pages 242–256, 2009.

[24] Laura Kovács and Adalbert Kovács. Deciding Properties of Affine Loops. In *Proceedings of the International Symposium of Mathematics and its Applications (ISMA)*, Scientific Bulletins of the Politehnica, University of Timişoara, Transactions on Mathematics and Physics, pages 401–406, 2009.

[25] Laura Kovács and Adalbert Kovács. Solving the Four Fundamental Problems from the Theory of Profile Grids via BEM. In *Proceedings of the International Symposium of Mathematics and its Applications (ISMA)*, Scientific Bulletins of the Politehnica, University of Timişoara, Transactions on Mathematics and Physics, pages 393–400, 2009.

[26] *** Laura Kovács and Andrei Voronkov**. **Finding Loop Invariants for Programs over Arrays Using a Theorem Prover**. In **Proceedings of the International Conference on Fundamental Approaches to Software Engineering (FASE)**, volume **5503** of **LNCS**, pages **470–485**, **2009**. **(GS): 63**.

[27] *** Laura Kovács and Andrei Voronkov**. **Interpolation and Symbol Elimination**. In **Proceedings of the International Conference on Automated Deduction (CADE)**, volume **5663** of **LNCS**, pages **199–213**, **2009**. **(GS): 23**.

[28] Thomas A. Henzinger, Thibaud Hottelier, and Laura Kovács. Valigator: A Verification Tool with Bound and Invariant Generation. In *Proceedings of the International Conference on Logic for Programming, Artifical Intelligence and Reasoning (LPAR-15)*, LNCS, pages 333–342, 2008.

[29] Laura Kovács. Aligator: A Mathematica Package for Invariant Generation (System Description). In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR)*, volume 5195 of *LNCS*, pages 275–282, 2008.

[30] Laura Kovács. Invariant Generation for P-solvable Loops with Assignments Only. In *Proceedings of Computer Science in Russia (CSR)*, volume 5010 of *LNCS*, pages 349–359, 2008. **Best Paper Award in the Application Track**.

[31] *** Laura Kovács**. **Reasoning Algebraically About P-solvable Loops**. In **Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)**, volume **4963** of **LNCS**, pages **249–264**, **2008**. **(GS): 26**.

[32] Laura Kovács. Aligator: a Package for Reasoning about Loops. In *Proceedings of the International Conference on Logic for Programming, Artificial Intelligence and Reasoning – Short Papers (LPAR-14)*, pages 5–8, 2007.

[33] Laura Kovács, Nikolaj Popov, and Tudor Jebelean. Combining Logic and Algebraic Techniques for Program Verification in Theorema. In *Proceedings of the International Conference on Leveraging Applications of Formal Methods, Verification and Validation (ISOLA)*, pages 59–67, 2006.

[34] Laura Kovács and Tudor Jebelean. An Algorithm for Automated Generation of Invariants for Loops with Conditionals. In *Proceedings of the International Conference on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, IEEE Computer Society, pages 245–249, 2005.

2.2 Refereed Papers in Workshop Proceedings

[35] Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. An Evaluation of WCET Analysis using Symbolic Loop Bounds. In *Proceedings of the International Workshop on Worst-Case Execution Time Analysis (WCET)*, pages 93–103, 2011.

[36] Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. Practical Experiments with Symbolic Loop Bound Computation for WCET Analysis. In *Proceedings of the 28th Workshop of the GI-Working group on Programming Languages and Computing Concepts*, 2011. Technical Report of the Computer Science Faculty of the Christian-Albrechts University Kiel.

[37] Laura Kovács. Invariant Generation with Aligator. In *Proceedings of Austrian-Japanese Workshop on Symbolic Computation in Software Science (SCCS)*, number 08-08 in RISC-Linz Report Series, pages 123–136, 2008.

[38] Laura Kovács. Automated Invariant Generation by Algebraic Techniques for Imperative Program Verification in Theorema. In *Proceedings of the International Workshop on Invariant Generation (WING)*, number 07-07 in RISC Report Series, pages 56–69, 2007.

[39] Laura Kovács and Tudor Jebelean. Finding Polynomial Invariants for Imperative Loops in the Theorema System. In *Proceedings of the Verify Workshop, IJCAR, The 2006 Federated Logic Conferences (FLoC)*, pages 52–67, 2006.

[40] Laura Kovács, Nikolaj Popov, and Tudor Jebelean. A Verification Environment for Imperative and Functional Programs in the Theorema System. In *Proceedings of the South-East European Workshop on Formal Methods (SEEFM)*, 2005.

2.3 Invited Papers in Conference Proceedings

[41] Laura Kovács and Andrei Voronkov. First-Order Theorem Proving and Vampire. In *Proceedings of the International Conference on Computer Aided Verification (CAV)*, LNCS, 2013. 16 pages, To appear.

[42] Laura Kovács. Symbol Elimination in Program Analysis. In *Proceedings of the International Conference on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, volume P3964 of *IEEE Computer Society*, page 12, 2012.

[43] Laura Kovács and Andrei Voronkov. Finding Loop Invariants for Programs over Arrays using a Theorem Prover. In *Proceedings of the International Conference on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, volume P3964 of *IEEE Computer Society*, page 10, 2009.

2.4 Refereed Abstracts in Conference and Workshop Proceedings

[44] Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. An Evaluation of WCET Analysis using Symbolic Loop Bounds (abstract/presentation). In *Proceedings of the 16th Colloquium on Programming Languages and Programming Foundations (KPS)*, page 200, 2011. Westfälische Wilhelms University Münster.

[45] Jens Knoop, Laura Kovács, and Jakob Zwirchmayr. An Evaluation of WCET Analysis using Symbolic Loop Bounds (extended Abstract). In *Proceedings of the Annual Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS)*, page 119, 2011. Lednice, Czech Republic.

3 Review Articles, Book Chapters, Books

3.1 Textbooks

[46] Adalbert Kovács, Gheorghe Ţigan, Laura Kovács, and Constantin Milici. *Computer Assisted Mathematics (in Romanian)*. "Politechnica" Publisher, Timişoara, 3rd edition, 2012.

3.2 Edited Volumes

[47] Nikolaj Bjorner, Krishnendu Chatterjee, Laura Kovacs, and Rupak M. Majumdar, editors. *Games and Decisions for Rigorous Systems Engineering (Dagstuhl Seminar 12461)*, volume 2 of *Dagstuhl Reports*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.

[48] Nikolaj Bjørner and Laura Kovács, editors. *Special Issue on Invariant Generation and Advanced Techniques for Reasoning about Loops*, volume 47 of *Journal of Symbolic Computation*, 2012.

[49] Laura Kovács and Temur Kutsia, editors. *Special Issue on Automated Specification and Verification of Web Systems*, volume 10 of *Journal of Applied Logic*, 2012.

[50] Laura Kovács, Rosario Pugliese, and Francesco Tiezzi, editors. *Proceedings of the 7th International Workshop on Automated Specification and Verification of Web Systems*, volume 61 of *EPTCS*, 2011.

[51] Nikolaj Bjørner and Laura Kovács, editors. *Proceedings of the International Workshop on Invariant Generation (WING)*. IJCAR, University of Edinburgh, 2010.

[52] Martin Giese, Andrew Ireland, and Laura Kovács, editors. *Special Issue on Invariant Generation and Advanced Techniques for Reasoning about Loops*, volume 45 of *Journal of Symbolic Computation*, 2010.

[53] Laura Kovács and Temur Kutsia, editors. *Proceedings of the International Workshop on Automated Specification and Verification of Web System (WWV)*. Vienna University of Technology, 2010.

[54] Andrew Ireland and Laura Kovács, editors. *Proceedings of the Second International Workshop on Invariant Generation (WING)*. ETAPS, University of York, 2009.

3.3 Editorial Papers

[55] Nikolaj Bjørner and Laura Kovács. Foreword to the Special Issue on Invariant Generation and Advanced Techniques for Reasoning about Loops. *Journal of Symbolic Computation*, 47(12):1413–1415, 2012.

[56] Laura Kovács and Temur Kutsia. Editorial to the Special Issue on Automated Specification and Verification of Web Systems. *Journal of Applied Logic*, 10(1):1, 2012.

[57] Martin Giese, Andrew Ireland, and Laura Kovács. Introduction to the Special Issue on Invariant Generation and Advanced Techniques for Reasoning about Loops. *Journal of Symbolic Computation*, 45(11):1097–1100, 2010.

3.4 Theses

[58] Laura Kovács. *Symbol Elimination in Program Analysis*. Habilitation Thesis, Vienna University of Technology, Austria, November 2012.

[59] Laura Kovács. *Automated Invariant Generation by Algebraic Techniques for Imperative Program Verification in Theorema*. PhD thesis, with highest distinction. RISC, Johannes Kepler University Linz, Austria, October 2007. RISC Technical Report No. 07-16. Supervisor: Prof. Tudor Jebelean.

4 Patents

I have been consulting Intel Haifa in program verification but I am not yet ready to create a patent in the area of my research interest.

5 Open-Access Computer Programs

ALIGATOR
`http://mtc.epfl.ch/software-tools/Aligator/`
– a software package for generating loop invariants of programs over scalars and arrays
– developer since 2007

VAMPIRE
`http://vprover.org/`
– developing program analysis and theory reasoning in the Vampire theorem prover
– developer since 2009

r-TuBound
`http://costa.tuwien.ac.at/tubound.html`
– a static analysis tool for the WCET analysis of programs
– developer since 2010

ABC
`http://mtc.epfl.ch/software-tools/ABC`
– a software tool for computing upper bounds on loop iterations
– developer since 2009

VALIGATOR
`http://mtc.epfl.ch/software-tools/Aligator/Valigator/`
– a verification tool with invariant and bound inference
– developer since 2008

6 Popular Science Articles/Presentations

There are no popular science articles.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2013-45614-105023-81

Name of applicant
Kovacs, Laura

Date of birth
800426-2906

Title of research programme

Appendix N

Budget and research resources

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Appendix N – Budget and Research Resources

1. Project Budget

The proposed project will be carried out at the Department of Computer Science of the Chalmers University of Technology. The project will use the general infrastructure, namely offices, seminar rooms, secretarial and networking support of the department. The requested funding from Swedish Research Council - VR has the *total amount of SEK 5,659,000*. This funding covers direct costs including personnel, travels, and materials, as presented in Table 1. Costs in Table 1 are given in kSEK. Personnel costs are calculated with an increase of about 3% pro year; premises and direct IT costs are calculated according to the VR regulations.

Cost Item	**2014**	**2015**	**2016**	**2017**	**2018**	**Total (in kSEK)**
PI - Laura Kovacs, 20%	278	288	299	309	320	**1,494**
PhD Student, 80%	626	648	671	695	720	**3,360**
Travel Cost	70	70	70	70	70	**350**
Laptops	20			20		**40**
PhD Examination Costs			10		40	**50**
Premises	58	60	62	64	67	**311**
Direct IT costs	10	10	11	11	12	**54**
Total (in kSEK)	*1,062*	*1,076*	*1,123*	*1,169*	*1,229*	**5,569**

Table 1: The GenPro Budget.

In the sequel, we briefly describe each cost item below.

PI Salary

We request funding covering 20% of the PIs salary for 5 years, with a *total amount of SEK 1,494,000*. The rest of the PI's salary will be covered by Chalmers.

PhD Student Salary

We request funding covering 80% of a PhD student positions for 5 years, with a *total amount of SEK 3,360,000* . The rest of the PhD student's salary will be covered by Chalmers. Our project involves both theoretical and practical research directions, and provide sufficient material for a doctoral dissertation.

Travel Cost

We request travel costs for 5 years, with a *total amount of SEK 350,000*. These costs include travels and conference fees as charged by main conferences in the area of the proposal.

For each project year, the travel costs are calculated with SEK 70,000 per year as follows. We expect two conference attendances outside Europe (SEK 20,000 per attendee and conference, SEK 40,000 all together) and two conference attendances in Europe (SEK 11,000 per attendee and conference, SEK 22,000 all together). In addition, we request we request SEK 8,000 for inviting experts to Chalmers to collaborate with us on the project.

Laptops

We require one high-end laptop for the PhD student of our project in year 1, plus one replacement laptop in years 4, with a *total amount of SEK 40,000*. We need high-end laptops for programming and running our computationally expensive systems.

PhD Examination Costs

We request *a total amount of SEK 50,000* covering the examination costs of the PhD student of our project. These costs are split as follows. In year 3, SEK 10,000 will cover the licentiate exam cost of the PhD student (after 2.5 years of PhD research work). In year 5, we ask for SEK 40,000 covering the final PhD exam costs of the PhD student. This amount will also cover the travel costs of PhD examiners.

Premesis

Premesis are calculated according to the VR regulation, and sum up to a *a total amount of SEK 311,000*.

Direct IT costs

Direct IT costs are calculated according to the VR regulation, and sum up to a *a total amount of SEK 54,000*.

2. Other Grants and Resources

I have joined Chalmers as an associated professor only recently, in April 2013. *I have not yet applied for any research grant from the VR.* The current application is my first VR proposal, and I do not apply for other VR grants.

Related projects. Some parts of this project build upon results obtained during the work on my "Interpolation and Symbol Elimination" Project S11410-N23 of the Austrian Science Fund - FWF. Nevertheless, *our project goes far beyond the work done in my S11410-N23 Project.* Our proposal will strengthen the objectives of S11410-N23 by generalizing interpolants to tree (and other) interpolants and studying the relation between quantified interpolants and inductive invariants (WP2). The invariant generation part (WP1) and reasoning with quantifiers and theories (WP3) are new. Further, our project proposes symbol elimination as a unifying framework for invariant generation, interpolation, and reasoning about program properties.

The ending date of my S11410-N23 project is February 2015. The current project would therefore provide an opportunity to build upon the work of S11410-N23, and also to cover new challenging topics and a wider range of applications of symbol elimination to program verification. The VR grant would make it possible for me to start my own independent research group in formal methods at Chalmers. The VR grant would also provide me the opportunity to integrate myself in the Swedish scientific community.

Type of Grant	Applied or Granted	Funding Source	Grant Holder and Project Leader	Grant Period	Total amount in SEK thousands
Project research grant S11410-N23	Granted	Austrian FWF	Laura Kovács	2011-2015	SEK 1,491 (equivalent of EUR 178,668)

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2013-45614-105023-81

Dnr
2013 -

Name of applicant
Kovacs, Laura

Date of birth
800426-2906

Reg date
2013-04-11 10:54:23

Project title
GenPro: Generating and Proving Program Properties via Symbol Elimination

NT

Research grants NT April 11, 2013
Project Grant Junior Researchers
*Nat-Tek generellt
Chalmers tekniska högskola

NT-2

Appendix S

Signatures

2014	2015	2016	2017	2018	
1062	1076	1123	1169	1229	2014-01-01 -- 2018-12-31

Applicant | **Date**

Head of department at host University | **Clarifi cation of signature** | **Telephone**

A signature on the application is required not only from the applicant but also from the authorised representative of the administrating organisation/institution or equivalent (normally the head of the department or establishment where the research is to be conducted). The signature confirms that the department can accommodate the proposed research, position or equipment; that the costing in the application is approved for the department's part, that any proposed experimentation on human or animal subjects has been reported, and that the applicant has reported any secondary occupations and commercial ties (s)he may have, and nothing inconsistent with good research practice has thereby emerged. The applicant must have discussed these conditions with the representative of the administrating organisation/institution or equivalent before the latter approves and signs the application.

NOTE the exceptions!
Please note that a different procedure applies for the following types of grants:

- Infrastructure- Operation Grants
- Infrastructure- Grants for Expensive Equipment
- Infrastructure- Grants for Large Databases
- Framework Grant SIMSAM

For these types of grants the vice-chancellor, instead of the head of department, must sign Appendix S. Thereby (s)he confirms the commitments of the administrative organization. The details are specified in the Call for Proposals, published at www.vr.se.

Vetenskapsrådets noteringar
Kod

2013-45614-105023-81

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se