VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kansliets noteringar Kod	Dnr
2014-40564-116615-37	2014-5484

2014
Project Research Grant

Area of science
Natural and Engineering Sciences
Announced grants
Research grants NT April 9, 2014
Total amount for which applied (kSEK)

2015	2016	2017	2018	2019
815	850	1441	1537	

APPLICANT

Name(Last name, First name)
Ruemmer, Philipp

Date of birth
780925-7418

Gender
Male

Email address
philipp.ruemmer@it.uu.se

Academic title
PhD

Position
Researcher

Phone
018 4713156

Doctoral degree awarded (yyyy-mm-dd)
2008-12-22

WORKING ADDRESS

University/corresponding, Department, Section/Unit, Address, etc.
Uppsala universitet
Institutionen för informationsteknologi
Datorteknik
Box 337
75105 Uppsala, Sweden

ADMINISTRATING ORGANISATION

Administrating Organisation
Uppsala universitet

DESCRIPTIVE DATA

Project title, Swedish (max 200 char)
Effektiv Hantering av Kvantifierare i SMT-Lösare

Project title, English (max 200 char)
Satisfiability Modulo Theories and Quantifiers

Abstract (max 1500 char)
Satisfiability Modulo Theories (SMT) is a paradigm for constructing solvers for logical constraints modulo various background theories. Due to their efficiency and flexibility, SMT solvers are widely used as search engines in domains such as verification, program analysis, test-case generation, or planning. SMT solvers are primarily designed, however, for solving quantifier-free constraints; this implies that SMT solvers are incomplete for important languages, including first-order logic, and limits applicability in domains where infinite or unbounded objects are considered.

This project is about the design of a new SMT framework with systematic and efficient support for quantifiers, by combining SMT technology with methods from the first-order theorem proving world. In the scope of the project, the new SMT framework (i) will be developed and formally defined, (ii) equipped with a relevant set of background theories, (iii) adapted to support extraction of Craig interpolants, and (iv) evaluated by means of reference implementation and a case study. The project results will be applicable in a number of domains, including software verification, analysis of hybrid or cyber-physical systems, and computer-aided mathematical reasoning.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-40564-116615-37

Name of Applicant
Ruemmer, Philipp

Date of birth
780925-7418

Abstract language
English

Keywords
SMT, theorem proving, satisfiability, theories, quantifiers

Review panel
NT-2, NT-1

Project also includes other research area

Classification codes (SCB) in order of priority
10201, 10206, 10103

Aspects

Continuation grant
Application concerns: New grant
Registration Number:

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
2014-40564-116615-37

Name of Applicant
Ruemmer, Philipp

Date of birth
780925-7418

ENCLOSED APPENDICES

A, B, C, D, N, S

APPLIED FUNDING: THIS APPLICATION

Funding period (planned start and end date)
2015-01-01 -- 2018-12-31

Staff/ salaries (kSEK)

Main applicant	% of full time in the project	2015	2016	2017	2018	2019
Philipp Ruemmer	25	229	235	241	247	
Other staff						
Doktorand 1 (Peter Backeman)	80			564	610	
Doktorand 2	80	484	513	534	578	
Total, salaries (kSEK):		713	748	1339	1435	

Other project related costs (kSEK)	2015	2016	2017	2018	2019
Premises	25	25	25	25	
Travel costs	64	64	64	64	
Other costs (minor equipment, books, etc)	13	13	13	13	
Total, other costs (kSEK):	102	102	102	102	

Total amount for which applied (kSEK)

2015	2016	2017	2018	2019
815	850	1441	1537	

ALL FUNDING

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

Project title	Applicant	Funded 2014	Funded 2015	Applied 2015
Multicore Embedded Systems: Timing Predictability and Performance	Wang Yi			4749

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

Funding source	Total	Proj.period	Applied 2015
Microsoft Research	900	2013-2016	

Project title	Applicant
Microsoft Research PhD Scholarship: First-Order Satisfiability Modulo Theories	PhD student: Peter Backeman; Supervisor: Philipp Ruemmer

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-40564-116615-37

Name of Applicant
Ruemmer, Philipp

Date of birth
780925-7418

POPULAR SCIENCE DESCRIPTION

Popularscience heading and description (max 4500 char)

Villkorslösare är ett viktigt hjälpmedel inom många olika IT-områden, t.ex. för verktyg att analysera och verifiera programvaror, generering av testfall, schemaläggning av arbete, operationsanalys eller datorassisterad resonerande kring matematiska påståenden. Inom alla dessa områden är det nödvändigt att lösa stora och komplicerade sökproblem, t.ex. för att schemalägga personal på ett sjukhus.

För att lösa ett sådant problem behöver man ställa upp alla villkor som måste uppfyllas, t.ex. "ett arbetspass är åtta timmar", "en anställd jobbar 40 timmar under en vecka", "en avdelning behöver åtta personer på plats", o.s.v. Dessa formuleras som matematiska formler som en villkorslösare sedan försöker hitta en lösning till; varje sådan lösning representerar ett arbetschema som uppfyller alla villkor.

De senaste åren har villkorslösare utvecklats som är oerhört effektiva på att lösa stora formler med upp till flera miljoner variabler och villkor. SMT (Satisfiability Modulo Theories) är en modern arkitektur för att bygga villkorslösare. Det är en påbyggnad av SAT-lösare (boolean SATisfiability), där man utökat funktionaliteten genom att inte bara ta formlerna i beaktande, utan även bakomliggande teorier såsom aritmetik eller funktioner. SMT-lösare har blivit mycket framgångsrika under de senaste åren, och har inom många områden lett till nya algoritmer och lösningar för hitintills svåra problem.

Vissa villkor kan dock vara svårare att hantera än andra, särskilt villkor som ska uppfyllas "för alla" eller "för vissa" värden (t.ex. "Det måste finnas ett schema för alla möjliga beläggningar på en avdelning"). Sådana villkor uttrycks ofta med hjälp av kvantifierare och är en utmaning för dagens lösare. De flesta SMT-lösare kan i grund och botten inte hantera kvantifierare, utan använder sig av heuristiker att instansiera formler som innehåller kvantifierare, tills alla möjligheter är uttömda; denna metod är tidskrävande och fungerar inte för alla formler. Andra lösare för formler inom första ordningens logik kan hantera kvantifierare på ett effektivt sätt, men stöder inte att ta teorier, såsom aritmetik, i beaktande.

På grund av detta har en effektiv hantering av både teorier och kvantifierare tillsammans blivit identifierad som en av de huvudsakliga utmaningarna under utvecklingen av lösare. Det här projektet syftar till att lyfta SMT-paradigmen till första ordningens logik genom att inkludera kvantifierare som första klassens "medborgare" i lösare. Detta kommer att möjliggöra effektivare beräkningar med villkor som innehåller både teorier och kvantifierare.

Målet är att designa ett nytt ramverk för SMT med ett systematiskt och effektivt stöd för kvantifierare genom att kombinera teknik från SMT tillsammans med metoder i första ordningens logik. Det nya ramverket kommer att kunna hantera olika teorier och implementeras på ett modulärt sätt, som är de-facto standard i dagens SMT-lösare, med målet att uppnå samma prestanda som SMT-lösare. Samtidigt kommer det nya ramverket att kunna hantera mer komplicerade villkor än existerande lösare, som har stor betydelse i en mängd olika domäner, bl.a. verifiering av mjukvara, analys av hybrida eller cyberfysiska system såväl som i datorassisterat matematikresonemang. Det kommer också att undersökas hur så kallade Craig interpolanter kan genereras utifrån villkor eller formler inom det nya ramverket; Craig interpolanter har visat sig vara ett viktigt verktyg inom verifiering de senaste åren.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-40564-116615-37

Name of applicant
Ruemmer, Philipp

Date of birth
780925-7418

Title of research programme
Satisfiability Modulo Theories and Quantifiers

Appendix A

Research programme

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

Satisfiability Modulo Theories and Quantifiers

Philipp Rümmer, *Uppsala University*

April 9, 2014

Abstract

Satisfiability Modulo Theories (SMT) is a paradigm for constructing solvers for logical constraints modulo various background theories. Due to their efficiency and flexibility, SMT solvers are widely used as search engines in domains such as verification, program analysis, test-case generation, or planning. SMT solvers are primarily designed, however, for solving *quantifier-free* constraints; this implies that SMT solvers are incomplete for important languages, including first-order logic, and limits applicability in domains where infinite or unbounded objects are considered.

This project is about the design of a new SMT framework with *systematic* and *efficient* support for quantifiers, by combining SMT technology with methods from the first-order theorem proving world. In the scope of the project, the new SMT framework (i) will be developed and formally defined, (ii) equipped with a relevant set of background theories, (iii) adapted to support extraction of Craig interpolants, and (iv) evaluated by means of reference implementation and a case study. The project results will be applicable in a number of domains, including software verification, analysis of hybrid or cyber-physical systems, and computer-aided mathematical reasoning.

1 Introduction

SMT solvers (solvers for Satisfiability Modulo Theories) are efficient solvers for complex logical constraints modulo background theories such as arithmetic, functions, arrays, or algebraic datatypes [17]. Over the last 10 years, SMT solvers have become standard search engines for a wide variety of applications, including hardware and software verification, test-case generation, and planning, and have in many domains significantly pushed forward the frontier of tractable problems. The success of SMT solvers is due to their efficiency, ease of use, and flexibility with respect to supported datatypes and theories.

In other application areas, however, the potential of SMT solvers is limited by the scope of supported logical features. In particular, SMT solvers are primarily designed for the case of *quantifier-free* formulae; in many state-of-the-art solvers, quantifiers $\forall, \exists$ [24] are handled with the help of simple heuristics, or entirely unsupported. Quantifiers are crucial whenever infinite/unbounded objects have to be considered, which are, among others, prevalent in software verification or computer-aided mathematical reasoning. For instance, to express that an array a is sorted, a quantified formula $\forall i.\ (1 \leq i < a.len \rightarrow a[i] \leq a[i+1])$ is needed.

Heuristic quantifier support in SMT is based on techniques such as e-matching [18, 36], where pattern matching is used to guess large numbers of ground instances $\phi(t)$ of quantified formulae $\forall x.\, \phi(x)$ that might be relevant for the task at hand. Such techniques have several disadvantages, including *inefficiency,* since often many more instances are produced than necessary; *incompleteness,* since there is no guarantee that the right instances are found; and *instability,* since seemingly innocent syntactic modifications of a formula can easily lead the heuristics astray. As a result, current SMT solvers are mostly ineffective when non-trivial quantifier reasoning is needed, for instance for formulae in first-order logic.

We propose a new, systematic approach to integration of quantifier reasoning into SMT. Most current SMT solvers are based on the DPLL(T) architecture [39], developed as an extension of the Davis-Putnam-Logemann-Loveland method [14], with quantifiers mainly added as an afterthought. In the scope of the project, we will lift DPLL(T) to the first-order level, by including quantifiers as first-class objects, together with sophisticated algorithms for handling quantifiers from first-order theorem proving. The resulting framework **DPLL^1(T)** will be similarly modular as DPLL(T), in the sense that a variety of datatypes and background theories

can be integrated efficiently, but also offer systematic handling of quantifiers.

In previous work of the applicant, the feasibility of the general approach has been demonstrated, in a restricted setting with only linear integer arithmetic as single built-in theory, and excluding techniques like conflict-driven learning that are important for scalability in SMT. A practical implementation was developed that was adopted in a number of software verification systems, and that won in the TFA division of the theorem proving competition CASC in 2012. The existing research results are promising and will be a starting point for the proposed project, but have to be extended significantly to obtain theorem provers that are as broadly useful as today's SMT solvers (see Sect. 3.4).

Significance. Efficient reasoning in the presence of **theories and quantifiers in combination** has been identified as a **grand challenge** in the theorem proving field [47, 38, 10]. Due to the importance of logic as the "calculus" of computer science, new algorithms and tools addressing the problem have the potential of becoming key enablers in upcoming domains, in particular in (i) software verification, (ii) analysis of hybrid or cyber-physical systems, (iii) computer-aided mathematical reasoning, (iv) semi-automatic higher-order theorem proving, and (v) knowledge representation and reasoning.

Case study. A case study on analysis of **parameterised timed systems with complex datatypes** will be used to evaluate the project outcomes. Analysis of such systems is crucial, among others, for engineering of **cyber-physical systems**, an area where improved analysis techniques are important and can have huge impact. The combination of complex datatypes (like arrays and arithmetic) and parameterisation (e.g., unboundedly many threads) requires quantifiers to capture relevant system invariants; existing approaches for parameterised systems usually assume finite datatypes. To build an analyser that can handle complex datatypes, we will adapt previously developed predicate abstraction-based algorithms for parameterised timed systems [26], and use the project outcomes for synthesising and checking quantified predicates.

2 Project goals

The overall goal of the project is to develop a new framework for constructing solvers and theorem provers, with (i) the capability to handle **formulae with quantifiers,** in a complete manner similar to first-order theorem provers; (ii) the ability to reason about formulae defined over various **background theories,** similar to the functionality of SMT solvers; and (iii) the **scalability** of SAT/SMT solvers when handling large formulae with complex Boolean structure.

Solver framework. The first outcome of the project will be the overall (theoretic) framework DPLL^1(T), a modular architecture consisting of a core engine for Boolean and first-order reasoning, together with a set of satellite solvers for background theories. The framework will enable the definition of theories using a standardised interface, and the implementation of theory solvers as separate well-structured modules. In contrast to normal SMT solvers on the basis of DPLL(T) [39], our new architecture DPLL^1(T) will support quantifiers and variables as first-class concepts integrated into the core engine. We will include symbolic methods to handle quantified formulae, in particular tableau-style *free variables* and delayed instantiation techniques like *first-order unification.*

As a result, DPLL^1(T) will provide a number of features that are beyond the scope of existing SMT solvers, including (i) complete reasoning about problems in first-order logic (without theories), as well as various other fragments (with theories), for instance about formulae with only universal quantifiers [42]; (ii) robust and systematic reasoning about formulae with quantifiers in general; (iii) parametric analysis, e.g., derivation of values for which a parameterised formula is valid; and (iv) extended forms of quantifier elimination.

The excellent performance of SAT/SMT solvers critically depends on sophisticated techniques for *conflict analysis,* in particular the ability to *learn lemmas* whenever the search process has reached a dead end and has to backtrack. The integration of advanced conflict analysis techniques into DPLL^1(T) will be a central theme of our work.

Theory definition. In order to make the DPLL^1(T) framework practically useful, it has to be instantiated for theories that cover relevant applications, in particular in the area of verification. The second project goal is (i) to investigate which theories are suitable for DPLL^1(T), and (ii) to formally define a relevant set of base theories within DPLL^1(T), and examine essential properties such as computational complexity. The most important theories (for the areas of application listed in Sect. 1) in-

clude different kinds of arithmetic (linear integer, rational, floating-point, bit-vector), sets and other collections, and algebraic datatypes.

A further relevant case is the theory of arrays [31], which is frequently used in verification. In contrast to the theories listed above, arrays do not admit quantifier elimination [24], which means that they can be handled in exactly the same way as, for instance, linear arithmetic theories.

In the normal DPLL(T) framework, it is possible to combine multiple theories thanks to the concept of Nelson-Oppen-style theory combination [37]; this technique is instrumental for achieving the flexibility offered by SMT solvers. We will investigate how similar combination methods can be defined in the DPLL^1(T)- framework, and which theories are DPLL^1(T)-combinable.

Proofs and Craig interpolation. The use of Craig interpolants in program verification has been pioneered by McMillan, starting with a complete SAT-based model checking procedure for finite-state transition systems (in particular targeting hardware designs) [32], which was subsequently adapted to software verification (e.g., [34, 30]). SMT solvers can today compute Craig interpolants quite effectively in the quantifier-free case. Interpolation with quantifiers, however, remains a challenge: even if it is possible to compute interpolants in principle [13], it is usually difficult to compute "good" interpolants that are useful for a model checker.

The third project goal is to extend DPLL^1(T) in such a way that Craig interpolants can be extracted efficiently, with focus on interpolants that are effective for applications. This will be done by first defining a suitable notion of *proof* that is generated as result of a successful run of a solver on an unsatisfiable formula. Craig interpolants are computed by recursively processing such proofs.

Explicit proofs, witnessing unsatisfiability of a formula, are also relevant in the context of *certification:* since SMT solvers are complicated and highly optimised programs, it is challenging to ensure their overall correctness. Generated proofs can be checked independently (using *proof checkers,* whose implementation is much simpler than that of solvers), so that verification of solver implementations becomes less pressing.

Implementation and evaluation. An important aspect of the project is to evaluate efficiency and scalability of the new algorithms. To this end, an open-source reference implementation of the DPLL^1(T) framework will be developed, including a representative set of theory solvers, as well as functionality for proof generation and Craig interpolation. By supporting standard solver interfaces, integration into existing systems will be possible. The effectiveness of our methods will be evaluated by an extensive case study in the area of parameterised timed systems analysis. We will also use existing benchmark libraries to evaluate performance.

3 State of the art

Automatic reasoning in the presence of theories and/or quantifiers has traditionally been investigated in two research areas: *first-order theorem proving* and *SMT solving.* The first line of research focuses on complete and efficient handling of logic with *quantifiers,* while the latter concentrates on reasoning *modulo theories* and efficient handling of formulae with complex Boolean structure. Both communities have in the past been quite disjoint, but are today showing signs of slowly growing together.

3.1 First-order theorem proving

Research in theorem proving in first-order logic goes back to the seminal work by Hilbert and Gentzen in the 1920s and 1930s, and led to the development of fundamental reasoning methods that remain standard until today: *tableaux* in the 1950s–1960s, the *Davis-Putnam-Logemann-Loveland (DPLL)* method in 1962 [14], and *resolution* in 1965. Efficient first-order theorem provers (without theories) are today mainly based on resolution, and the related calculus of *superposition,* and to a lesser degree on tableaux, or alternative calculi such as *model evolution*.

Theories in resolution have been considered in several lines of research. Among others, [45] works with theory constraints that determine possible resolution steps, but suffers from high nondeterminism in search, and has ultimately not resulted in efficient theorem provers. A related method is *hierarchic superposition* [2], where theory constraints are attached to clauses. Hierarchic superposition has recently led to implementations [1, 6], with ongoing work investigating further improvements; however, the method is only complete on inputs that are "sufficiently complete." A calculus combining superposition with linear arithmetic is [27], but under restrictions that prevent quantification over numbers. A pragmatic route is

taken in SPASS+T [41], where resolution is combined with an SMT solver for theory reasoning; this combination does not lead to completeness on non-trivial fragments with theories.

Theories in tableaux are handled by the general framework in [7], by distinguishing between foreground and background provers. This framework relies on the ability to solve unification problems modulo theory, which is a hard problem in itself. An approach to embed algebraic constraints in tableaux is described in [40], where quantifier elimination in real arithmetic, via an external procedure, is used for theory reasoning; uninterpreted functions or predicates are not handled.

Model evolution is a specific approach to lift DPLL(T) to the first-order level, and turned out successful for first-order theorem proving without theories. There were proposals to integrate theories in model evolution [4, 5], yet no practical implementations were obtained.

In general it can be observed that only few of the proposed combinations of first-order calculi and theories have led to practical tools. In cases where implementations emerged, scalability on complex problems is usually worse than with SMT solvers.

3.2 Satisfiability modulo theories

SMT solvers based on the DPLL(T) architecture [39] can handle ground problems modulo various theories efficiently, but only offer heuristic quantifier handling. *E-matching* is today used in most SMT solvers, based on techniques that go back to the Simplify prover [18] and The Stanford Pascal Verifier [36]; since then, various refinements of the e-matching approach have been published [20, 15].

Model-based quantifier instantiation (MBQI) [21] is a method used by the Z3 solver, and works by iteratively eliminating spurious models by adding further instances of quantifier formulae. MBQI yields stronger completeness results than e-matching, but is still incomplete for inputs in first-order logic. There is also work on integrating superposition into DPLL(T) and Z3 [16]. The resulting system is complete for first-order logic, but does not address the problem of combining quantifier and theory reasoning.

3.3 Craig interpolation

Given two formulae A and C such that A implies C, written $A \Rightarrow C$, an interpolant is a formula I such that $A \Rightarrow I$, $I \Rightarrow C$, and all of I's non-logical symbols occur in both A and C. Interpolants exist for any two first-order formulae A and C such that $A \Rightarrow C$. Over the past years, Craig interpolation has been identified as a general and practical approximation method that is useful for many analysis procedures.

Most often, Craig interpolants are computed by extending a theorem prover, SMT solver, or decision procedure to generate proofs, which can then be recursively processed in order to compute an interpolant (e.g., [33, 12, 23]). Interpolation procedures for various calculi and theories have been developed, including first-order resolution and superposition [35, 28], Gentzen-style sequent calculi and tableaux [12], rational arithmetic, in combination with uninterpreted functions [33], and quantifier-free Presburger arithmetic [12, 29, 23].

There is relatively little work on computation of Craig interpolants in the presence of both theories and quantifiers; in fact, most interpolation methods used in applications only consider quantifier-free problems. In [13], interpolants are computed in the context of incomplete instantiation-based quantifier methods, like the e-matching method common in SMT, by introducing quantifiers corresponding to the instantiations in the interpolants. An interpolation procedure for the calculus from [16], integrating DPLL(T) and superposition, is developed in [11]. Some other interpolation procedures for superposition [35, 28] have been applied to theories encoded with the help of axioms; for many theories, however, this is not a practical approach.

3.4 Contributions by the applicant

The practicality of the research outlined in this proposal has been demonstrated for the special case of integer arithmetic by the system PRINCESS [42, 43],[1] developed by the applicant. PRINCESS has been adopted as reasoning engine in a number of software verification systems, and has performed well in competitions: it won in the TFA division at CASC 2012,[2] outperforming theorem provers like SPASS+T and SMT solvers like Z3.[3] In 2013, PRINCESS was runner-up in TFA, but won in the category TFI specific for integer problems. The main underlying calculus of PRINCESS

[1] `http://philipp.ruemmer.org/princess.shtml`

[2] `http://www.cs.miami.edu/~tptp/CASC/J6/`

[3] Z3 did not enter in 2012, but competed in 2011, and came second to the tool that ended up on the last place in 2012.

is a free-variable tableau calculus, which is extended with constraints to enable backtracking-free proof expansion, and positive unit hyper-resolution for lightweight instantiation of quantified formulae. Linear integer arithmetic is handled using a set of built-in proof rules resembling the Omega test, which altogether yields a calculus that is complete for full Presburger arithmetic, for first-order logic, and for a number of further fragments.

While PRINCESS performs well on a variety of problems, it still has a number of significant restrictions compared to state-of-the-art SMT solvers, including (i) missing support for theories other than linear integer arithmetic; (ii) no interface for modular integration of new theories; and (iii) limited scalability when applied to large formulae, or formulae with complex Boolean structure, since no conflict-driven learning methods are available. The DPLL^1(T) architecture to be developed in the proposed project will address such issues.

4 Planned work

The project is divided into four sub-projects, corresponding to the project goals, and will yield both theoretic results and implementations.

SP1: The framework DPLL^1(T)

The definition of the overall DPLL^1(T) architecture is at the heart of the project, and crucial for all sub-projects. DPLL^1(T) will combine concepts of the DPLL(T) framework, which is common in today's SMT solvers, with symbolic free-variable techniques developed in the context of first-order theorem provers. The resulting procedure will be able to reason efficiently about formulae with complex Boolean structure, similarly to DPLL(T), and at the same time handle quantified formulae in a complete and systematic way. Since the DPLL^1(T) architecture is modular, background theories can be added on demand.

By itself, DPLL(T) [39] performs systematic depth-first search for models satisfying a given input formula, complemented by powerful methods for constraint propagation and conflict-driven learning. Free variables [24] add the possibility of *delayed instantiation* of quantified formulae $\forall x.\, \phi(x)$: instead of generating concrete instances $\phi(t)$, *symbolic* instances $\phi(X)$ are considered, for fresh free variables X. After adding $\phi(X)$, search can continue, until at a later point it becomes clear which term the variable X should represent. This approach reduces the total number of instances of quantified formulae that have to be generated, and is able to construct instances that are hard to find using heuristic approaches.

Possible valuations of free variables will in DPLL^1(T) be represented with the help of *constraints* [22, 42, 43] modulo background theories. Constraints enable non-destructive management of variable valuations, thus reducing the need for backtracking, and are also a key integration point for theory reasoning.

General definition. The definition of the DPLL^1(T) will be split into three levels:

- The *calculus* level, where the framework is represented as a set of declarative tableau-style rules. Theories are kept abstract, and can later be plugged into the calculus. The central notion of *soundness* (if a formula can be proven, it is valid; if a model of a formula is found, the formula is satisfiable) is justified on the calculus level.

- The *transition system* level, which abstractly represents operational steps to search for models of a formula, or equivalently construct an (implicit) proof that no model exists. On this level, the concepts of *backtracking* and *learning* are captured.

- The *proof search* level, where strategies are defined to apply the transition rules. The notion of *completeness* (if a formula is unsatisfiable, a search strategy will terminate and report that no models exist) is considered on this level,[4] and established by notions of *fairness.*

The architectural definitions are complemented by an abstract interface for theory solvers. Solvers are invoked by the DPLL^1(T) framework in two situations: (i) for theory-specific exploration and construction of satisfying assignments, as part of proof search; and (ii) to handle constraints describing instantiations of free variables, which includes checking the satisfiability of constraints, computing the conjunction/intersection of multiple constraints, and projecting constraints to a subset of the included variables.

Learning in presence of quantifiers. SMT solvers use learning to avoid repeated exploration of the

[4] Since first-order logic modulo theories is usually too expressive to admit complete calculi, the notion of completeness has to be relaxed suitably.

same or similar parts of the search space (of possible models of a formula). Learning is *conflict-driven:* whenever depth-first search arrives at a dead end, i.e., at a conflict, the solvers extracts a simple explanation for the conflict. This explanation, encoded as a *conflict clause,* is again a logical formula, and in particular a consequence of the overall formula to be proven; it is therefore possible to learn the conflict clause and proactively avoid similar conflicts in future search. Conflict-driven learning is important for scalability on problems with complex Boolean structure.

The integration of conflict-driven learning into DPLL^1(T) is one of the main technical challenges in the project. The addition of free variables gives search a more global character, since variable valuations from different branches may have to be considered in combination: search is no longer purely depth-first. In this setting, we will realise learning as a combination of different techniques, in particular: (i) *conflict clauses from sub-trees:* instead of considering single branches of depth-first search, conflict clauses can be derived from whole sub-trees of the constructed tableau-style proof tree, provided that a local conflict for the sub-tree can be identified; (ii) *iterative bounded exploration:* the principle of pure depth-first search can be restored by bounding the number of variables introduced on each search branch. To ensure completeness, it is necessary to iteratively increase the bound.

Model construction. In practice, the ability to construct *models* or *satisfying assignments* for a formula is often more useful than to conduct a proof that no models exist. Generation of models is, however, theoretically limited by computability: for instance, in first-order logic there are complete procedures for the construction of proofs, but no algorithms or semi-algorithms exist to find models of all satisfiable formulae [24].

The main challenge when constructing models is to prevent indefinite instantiation of quantified formulae, leading to non-termination of a solver. In the DPLL^1(T) setting with free variables, this necessitates *stopping criteria* that detect situations in which further instantiations are useless, combined with techniques to extract corresponding models of the input formula. In addition, we plan to investigate how techniques for automatic induction, developed in previous work of the applicant [25], can be used to infer infinite models of formulae.

Free-variable congruence closure. Congruence closure (CC) is the standard method used in SMT solvers for reasoning about equations and function symbols, and works by constructing and saturating a graph connecting those terms for which equality follows from asserted equations. CC is highly efficient, and can be combined with other background theories (e.g., arithmetic or arrays) in an elegant way. CC is in SMT usually implemented as part of a theory solver for the theory EUF of equality and uninterpreted functions [39]; in contrast, in DPLL^1(T) equality and functions are integrated into the core engine. The presence of free variables in DPLL^1(T) requires combination of CC with concepts developed in the field of *first-order unification:* it is no longer sufficient to check whether two terms are equal, but also necessary to derive whether some valuation of relevant variables exists that makes terms equal. Effective methods for checking unifiability are crucial to curb the nondeterminism caused by free variables.

SP2: Theory definition

Over the past years, significant engineering effort has led to the development of various fast theory solvers, tailored to the integration in DPLL(T). Such solvers offer a simple and unified interface, with main operations to *assert* new theory constraints, *check satisfiability* of asserted constraints, and *backtrack* by removing constraints. The need to implement all three operations efficiently has implications for the design of decision procedures in theory solvers: in particular, such procedures tend to work *incrementally*, so that constraints can be added and removed with little overhead.

SP2 is less concerned with the development of new decision procedures and algorithms, but rather with the reformulation of existing procedures in the form of theory solvers for DPLL^1(T). In addition to the functionality needed already in DPLL(T), solvers in DPLL^1(T) also have to handle free variables in constraints (possibly with the help of *first-order unification* [24], or similar approaches, to discover valuations that make a constraint solvable), and have to provide functions to reason about overall constraints that describe such valuations. SP2 will investigate how such functionality can be added to existing theory solver designs without sacrificing efficiency.

Arithmetic theories. The theory of *linear integer* arithmetic (LIA) has already been considered in the context of PRINCESS [42], resulting in an effective solution on the basis of the Omega test. The

existing results will be ported to the new framework DPLL^1(T); importantly, constraint handling, and projection of constraints, can be implemented with the help of quantifier elimination [24]. Similar solvers will be possible for the case of *linear rational* arithmetic (LRA), and *mixed linear* arithmetic (the combination of LIA and LRA), which have comparable theoretic properties as LIA.

In many applications, also bounded arithmetic datatypes are relevant, in particular *bit-vectors* and *floating-point* arithmetic, even including nonlinear operations. Since those theories satisfy only weaker mathematical laws (than LIA or LRA), reasoning often reduces to solving intricate combinatorial or Boolean problems. We plan to integrate existing decision procedures for bounded datatypes into DPLL^1(T); to deal with free variables, we will develop new iterative methods to extract increasingly tight bounds on satisfying valuations.

The theory of arrays [31] is highly important for applications, but requires a different integration approach than arithmetic theories since the concept of quantifier elimination is not available. We will handle arrays, and other theories with similar properties, by restricting the set of symbols that are allowed to occur in constraints on free variables. In this way it is possible to preserve the ability to project constraints, which is important for the overall functioning of DPLL^1(T) model search. Like in existing SMT solvers, the theory solver for arrays will work as an extension of the congruence closure procedure developed in SP1.

SP3: Proofs and Craig interpolation

The ability to extract Craig interpolants for valid implications $A \Rightarrow C$ is an important prerequisite to make DPLL^1(T) useful for applications, e.g., model checking. Following the paradigm commonly used in interpolation procedures, in SP3 we will extend DPLL^1(T) by adding functionality to generate explicit proofs, and then define algorithms to extract Craig interpolants from proofs. This requires an extension of the theory solver interface defined in SP1, since proofs need to record theory-specific reasoning steps, and also interpolation procedures are specific to individual theories.

Explicit proofs. Craig interpolation has in the past been defined for different kinds of proofs, in particular for resolution proofs [28, 11] and for sequent-style proofs [33, 12]. Resolution is commonly used as proof format in SMT solvers, since generated conflict clauses can naturally be justified with the help of resolution steps, enabling proof generation with comparatively little overhead. However, splitting and free variables, present in the DPLL^1(T) framework, cannot easily be encoded in resolution proofs. We therefore plan to define proofs for DPLL^1(T) as an extension of Gentzen-style sequent proofs, including proof rules to effectively infer conflict clauses.

The DPLL^1(T) proof format will be defined as a collection of proof rules: (i) Gentzen-style rules capturing the DPLL^1(T) *calculus* level defined in SP1, in particular for Boolean connectives and quantifiers; (ii) rules to manage constraints describing possible valuations of free variables; (iii) rules to derive conflict clauses, similar to resolution rules; and (iv) theory-specific proof rules, which are contributed by theory solvers. The semantics and soundness of all rules will be established formally, ensuring that (well-formed) proofs can only be derived for formulae that actually hold.

The generation of proofs within DPLL^1(T) will be defined by annotating the *transition system* rules from SP1 accordingly. For practical purposes, a textual representation of proofs will be defined, on the basis of proposals that have been made in the SMT community [9].

Craig interpolants are computed by recursively annotating proofs with *partial interpolants,* which at the root of the proof reduce to interpolants for the input formula. Annotation is defined by deriving interpolating versions of all proof rules, together with procedures for computing *theory interpolants* for theory-specific reasoning steps, integrated into theory solvers. The main technical challenge are the interpolating rules for quantifiers, free variables, and constraints describing valuations of free variables. Those rules will be handled by extracting witness terms from constraints, enabling translation of free-variable proofs to instantiation-based (ground) proofs. Such witness terms can be derived *explicitly,* with the help of witness-producing quantifier elimination, or *implicitly* by reduction to a cut in the proof. We will evaluate these different approaches in the project.

In general, interpolants for quantified problems again need to contain quantifiers. For practical reasons, it is still essential to keep extracted interpolants simple, avoiding occurrences of quantifiers to the extent possible, so that subsequent processing is not unduly slowed down. This will be done by devising methods for *proof simplification,* with

the goal of transforming proofs in such a way that simple interpolants emerge.

Proof simplification can lead to simpler interpolants, but cannot change the overall shape of interpolants; in order to obtain interpolants that are useful for applications, it can be necessary to drastically reconstruct proofs. In recent work, we have developed general techniques to guide solvers towards proofs that give rise to interpolants with desirable properties [44], with the possibility of including domain-specific knowledge. Those methods will be adapted to interpolation problems with quantifiers.

SP4: Implementation and evaluation

In order to evaluate the results of SP1–SP3, we will develop an open-source implementation of our DPLL^1(T) framework, based on experience and code from previous implementations, including the solver PRINCESS. The development of efficient solvers on the basis of frameworks like DPLL(T) or DPLL^1(T) tends to be a non-trivial engineering problem; in particular, suitable datastructures for expressions, solver state, and indexing, as well as search heuristics for Boolean reasoning are needed. A main goal of SP4 is to evaluate how existing solutions from the SAT/SMT field can be applied in our new framework DPLL^1(T).

Interfaces. For the purpose of integration into other systems, our new solver will offer different interfaces: (i) the SMT-LIB format and command language [3], which is the standard textual interface used by SMT solvers; (ii) the TPTP format [46], which is the format used by first-order theorem provers; (iii) an API that enables integration of the solver in form of a library. In particular, this will make it possible to use the new solver as a drop-in replacement for existing SMT solvers.

Evaluation of our implementation, and the project results will be done in several ways: (i) by integration into deductive verification systems [19, 8], which are tailored to analysis of complex functional properties of software programs and use both theories and quantifiers extensively; (ii) by integration into software model checkers, in particular the Horn solver Eldarica [25], which rely on the ability of solvers to check satisfiability of large numbers of (smaller) formulae quickly, but also need Craig interpolants; (iii) with the help of a case study on parameterised timed systems analysis, extending previous work of the applicant [26], and using the Eldarica model checker as engine; and (iv) with the help of existing benchmark libraries and competitions, including SMT-LIB and TPTP/CASC.

5 Past achievements

The project leader has published in major conferences relevant for theorem proving, SMT solving, and verification, including IJCAR, CADE, CAV, FMCAD, TACAS, DAC, VMCAI, LPAR, FMCO, as well as in many workshops in the area. In 2013, the project leader received the Oscar award (Oscarspriset) of Uppsala University, acknowledging contributions to the field of program correctness. The theorem prover PRINCESS developed by the project leader won a first price at the CASC competition in 2012, and was runner-up in 2013. Together with coauthors, the applicant developed the first proof-based, practical interpolation procedure for Presburger arithmetic [12].

The project leader is well-connected in the SMT and theorem proving community, and in an ideal position to carry out the research proposed here. In 2014, the project leader is co-chairing the 12th International Workshop on Satisfiability Modulo Theories (SMT), and lecturing at the 4th SAT/SMT Summer School. In 2010 and 2012, the project leader was co-chair of the workshop on Logics for System Analysis (LfSA). The project leader is also frequently invited to seminars or workshops in the area (including 6 invitations to Dagstuhl seminars between 2007 and 2014), and has been serving on programme committees of relevant events, such as the Conference on Formal Methods in Computer-Aided Design (FMCAD), the Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), the Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR), and the Symposium on Frontiers of Combining Systems (FroCoS).

The project leader started research on theorem provers, SMT solvers, and decision procedures during his PhD studies, in the context of deductive verification systems for Java but as a mostly independent line of research. Development on the theorem prover PRINCESS started during PhD studies, but did not overlap with research of the PhD advisers. Research in deduction was continued and extended during post-doc work in Oxford, in particular starting work on Craig interpolation. Since moving to Uppsala in 2011, the connection to embedded and timed systems analysis was strengthened.

6 Research environment

The proposed project will be carried out at Uppsala University, in collaboration with leading groups in the areas of embedded systems and verification:

- Wang Yi
  Real-time model checking
- Bengt Jonsson
  Verification of embedded, real-time, concurrent, and distributed systems, testing
- Parosh Abdulla
  Automata-based methods for reasoning and verification, including SMT
- Pierre Flener
  Constraint programming, testing

7 Collaboration

In addition, the applicant is collaborating with leading national and international groups in the areas of verification and deduction, including:

- Wolfgang Ahrendt
  (Chalmers, Gothenburg)
- Byron Cook
  (University College L., Microsoft Research)
- Alastair Donaldson
  (Imperial College, London)
- Reiner Hähnle
  (TU Darmstadt)
- Daniel Kroening
  (University of Oxford)
- Viktor Kuncak
  (EPFL Lausanne)
- Jérôme Leroux
  (CNRS, Bordeaux)
- André Platzer
  (Carnegie-Mellon University, Pittsburgh)
- Christoph M. Wintersteiger
  (Microsoft Research Cambridge)

We are also involved in the project CERTAINTY (FP7) on *Certification of Real-time Applications Designed for Mixed Criticality*, ending in 2014.

8 Budget and current funding

The project leader is (permanently) employed as researcher by Uppsala University, and holds the VR grant "Scalable Floating-Point Reasoning for Embedded Systems Analysis" (2011-6310) which ends in 2014. In the scope of the grant, decision procedures for the theory of floating-point arithmetic are developed; such procedures can also be integrated into the framework of the proposed project (SP3). The previous project also led to new methods for constraint-based analysis of software/systems, which can be applied in combination with the outcomes of the proposed project.

The project leader received a Microsoft PhD grant, funding one PhD student during 2013–2016 on the project "First-Order Satisfiability Modulo Theories." The project proposed here has a broader scope than the PhD grant, but there is overlap in SP1 and SP4 (further details in Appendix N). It is planned that a second PhD student focuses on SP2–SP4, and the project leader will be active in all SPs.

The project leader is also co-applicant of the VR framework application "Multicore Embedded Systems: Timing Predictability and Performance." The topic of the framework project does not overlap with the project proposed here; it is planned that the project leader will be active 25% in the framework project, and 25% in the present project.

References

[1] Ernst Althaus, Evgeny Kruglov, and Christoph Weidenbach. Superposition modulo linear arithmetic SUP(LA). In *FroCos*, 2009.

[2] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Theorem proving for hierarchic first-order theories. In *ALP*, 1992.

[3] Clark Barrett, Aaron Stump, and Cesare Tinelli. The SMT-LIB Standard: Version 2.0. Technical report, Department of Computer Science, The University of Iowa, 2010.

[4] Peter Baumgartner, Alexander Fuchs, and Cesare Tinelli. ME(LIA) - model evolution with linear integer arithmetic constraints. In *LPAR*, 2008.

[5] Peter Baumgartner and Cesare Tinelli. Model evolution with equality modulo built-in theories. In *CADE*, volume 6803 of *LNCS*. Springer, 2011.

[6] Peter Baumgartner and Uwe Waldmann. Hierarchic superposition with weak abstraction. In *CADE*, volume 7898 of *LNCS*. Springer, 2013.

[7] Bernhard Beckert. Equality and other theories. In Marcello D'Agostino, Dov Gabbay, Reiner Hähnle, and Joachim Posegga, editors, *Handbook of Tableau Methods*. Kluwer, Dordrecht, 1999.

[8] Bernhard Beckert, Reiner Hähnle, and Peter H. Schmitt, editors. *Verification of Object-Oriented Software: The KeY Approach*. 2007.

[9] Frédéric Besson, Pascal Fontaine, and Laurent Théry. A flexible proof format for SMT: a pro-

posal. In *PxTP*, 2011.

[10] Sascha Böhme and Michal Moskal. Heaps and data structures: A challenge for automated provers. In *CADE*, 2011.

[11] Maria Paola Bonacina and Moa Johansson. On interpolation in automated theorem proving. Technical Report 86/2012, Dipartimento di Informatica, Università degli Studi di Verona, 2012.

[12] Angelo Brillout, Daniel Kroening, Philipp Rümmer, and Thomas Wahl. An interpolating sequent calculus for quantifier-free Presburger arithmetic. In *IJCAR*, 2010.

[13] Juergen Christ and Jochen Hoenicke. Instantiation-based interpolation for quantified formulae. In *Decision Procedures in Software, Hardware and Bioware*, number 10161 in Dagstuhl Seminar Proceedings, 2010.

[14] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[15] Leonardo Mendonça de Moura and Nikolaj Bjørner. Efficient e-matching for SMT solvers. In *CADE*, volume 4603 of *LNCS*. Springer, 2007.

[16] Leonardo Mendonça de Moura and Nikolaj Bjørner. Engineering DPLL(T) + saturation. In *IJCAR*, volume 5195 of *LNCS*. Springer, 2008.

[17] Leonardo Mendonça de Moura and Nikolaj Bjørner. Satisfiability modulo theories: introduction and applications. *Commun. ACM*, 54(9):69–77, 2011.

[18] David Detlefs, Greg Nelson, and James B. Saxe. Simplify: A theorem prover for program checking. *Journal of the ACM*, 52(3), 2005.

[19] Jean-Christophe Filliâtre and Claude Marché. The Why/Krakatoa/Caduceus platform for deductive program verification. In *CAV*, 2007.

[20] Yeting Ge, Clark Barrett, and Cesare Tinelli. Solving quantified verification conditions using satisfiability modulo theories. In *CADE*, 2007.

[21] Yeting Ge and Leonardo Mendonça de Moura. Complete instantiation for quantified formulas in satisfiability modulo theories. In *CAV*, 2009.

[22] Martin Giese. Incremental closure of free variable tableaux. In *IJCAR*, 2001.

[23] Alberto Griggio, Thi Thieu Hoa Le, and Roberto Sebastiani. Efficient interpolant generation in satisfiability modulo linear integer arithmetic. In *TACAS*, volume 6605 of *LNCS*. Springer, 2011.

[24] John Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.

[25] Hossein Hojjat, Filip Konecný, Florent Garnier, Radu Iosif, Viktor Kuncak, and Philipp Rümmer. A verification toolkit for numerical transition systems - tool paper. In *FM*, 2012.

[26] Hossein Hojjat, Philipp Rümmer, Pavle Subotic, and Wang Yi. Uniform analysis for communicating timed systems. Technical report, EPFL, 2013.

[27] K. Korovin and A. Voronkov. Integrating linear arithmetic into superposition calculus. In *CSL*, volume 4646 of *LNCS*. Springer, 2007.

[28] Laura Kovács and Andrei Voronkov. Interpolation and symbol elimination. In *CADE*, 2009.

[29] Daniel Kroening, Jérôme Leroux, and Philipp Rümmer. Interpolating quantifier-free Presburger arithmetic. In *LPAR*, 2010.

[30] Daniel Kroening and Georg Weissenbacher. Interpolation-based software verification with Wolverine. In *CAV*, 2011.

[31] John McCarthy. Towards a mathematical science of computation. In C. M. Popplewell, editor, *Information Processing 1962*. North Holland, 1963.

[32] Kenneth L. McMillan. Interpolation and SAT-based model checking. In *CAV*, 2003.

[33] Kenneth L. McMillan. An interpolating theorem prover. *Theor. Comput. Sci.*, 345(1), 2005.

[34] Kenneth L. McMillan. Lazy abstraction with interpolants. In *Proceedings, CAV*, 2006.

[35] Kenneth L. McMillan. Quantified invariant generation using an interpolating saturation prover. In *TACAS*, 2008.

[36] Greg Nelson. Techniques for program verification. Technical Report CSL-81-10, Xerox Palo Alto Research Center, 1981.

[37] Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. Program. Lang. Syst.*, 1(2):245–257, 1979.

[38] Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Albert Rubio. Challenges in Satisfiability Modulo Theories. In *RTA*, pages 2–18, 2007.

[39] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, 2006.

[40] André Platzer. Differential dynamic logic for hybrid systems. *Journal of Automated Reasoning*, 41(2):143–189, 2008.

[41] Virgile Prevosto and Uwe Waldmann. SPASS + T. *ESCoR: FLoC'06 Workshop on Empirically Successful Computerized Reasoning*, 2006.

[42] Philipp Rümmer. A constraint sequent calculus for first-order logic with linear integer arithmetic. In *LPAR*, volume 5330 of *LNCS*. Springer, 2008.

[43] Philipp Rümmer. E-Matching with free variables. In *LPAR*, volume 7180 of *LNCS*. Springer, 2012.

[44] Philipp Rümmer and Pavle Subotic. Exploring interpolants. In *FMCAD*, pages 69–76, 2013.

[45] M E Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1(4):333–355, 1985.

[46] Geoff Sutcliffe. The TPTP world - infrastructure for automated reasoning. In *LPAR*, 2010.

[47] Cesare Tinelli. Grand challenges for automated reasoning – position statement. *Workshop on Challenges and Novel Applications for Automated Reasoning*, 2003.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-40564-116615-37

Name of applicant
Ruemmer, Philipp

Date of birth
780925-7418

Title of research programme
Satisfiability Modulo Theories and Quantifiers

Appendix B

Curriculum vitae

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

Dr. Philipp Rümmer

Forskare/Researcher
Uppsala University
Department of Information Technology

Box 337
Uppsala University, Department of IT
75105 Uppsala, Sweden
E-Mail: philipp.ruemmer@it.uu.se
http://www.philipp.ruemmer.org

Qualification

09/2004	Received a diploma degree (equivalent to MSc) with distinction in Computing Science Thesis: "Interactive Verification of JCSP Programs" (University of Karlsruhe, Germany)
12/2008	Received degree of Doctor of Philosophy in Computing Science Thesis: "Calculi for Program Incorrectness and Arithmetic" Supervisor: Reiner Hähnle, Wolfgang Ahrendt (Göteborg University, Sweden)

Post-Doctoral Positions

04/2009–12/2010	Research Assistant (Post-Doc) at the Oxford University Computing Laboratory

Present and Previous Positions

05/2012–present	Forskare/Researcher at Uppsala University (50% teaching, 50% research)
01/2011–04/2012	Forskarassistent/Research Fellow at Uppsala University

Interruptions in Research

2014	In total 1.5 months of parental leave
2013	In total 3 months of parental leave

Supervision of PhD Students

since 2013	Supervisor of *Peter Backeman,* research on automated reasoning.
since 2012	Supervisor of *Aleksandar Zeljić,* research on analysis techniques for programs operating on floating-point data.
since 2012	Supervisor of *Pavle Subotić,* research on novel techniques for analysing and model-checking networks of timed automata.

Awards and Grants

2013	Oscarspris of Uppsala University For contributions to the field of program correctness

2013–2016	Scholarship by Microsoft Research, financing one PhD student
2012–2014	Vetenskapsrådet (VR) grant (Projektbidrag – Unga forskare) "Scalable Floating-Point Reasoning for Embedded Systems Analysis"
2010	Microsoft Software Engineering Innovation Foundation (SEIF) Award "Testing Embedded Software with the Z3 SMT Solver"
07/2005	SAP Award: Best final degree in 2004 at Department of Computing Science at the University of Karlsruhe

Professional Activities

2015	Programme Committee of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)
2014	Lecturer at the 4th SAT/SMT Summer School, Semmering, Austria
2014	Programme Chair of the 12th International Workshop on Satisfiability Modulo Theories (SMT)
2014	Programme Committee of the 14st International Conference on Formal Methods in Computer-Aided Design (FMCAD)
2012	Programme Chair of the 2nd Workshop on Logics for System Analysis (LfSA'12, affiliated with CAV, Berkeley)
2011	Lecturer at the UPMARC Multicore Summer School, Stockholm, Sweden
2010	Programme Chair of the Workshop on Logics for System Analysis (LfSA, affiliated with LICS and IJCAR, Edinburgh)
2010–2013	Programme Committee of the conferences FMCAD, FroCoS, FSEN, HVC, iFM, LCTES, LPAR, MEMOCODE, SYNASC, VMCAI, and workshops.

Participation in Projects

from autumn 2013	Supervisor of the project "First-Order Satisfiability Modulo Theories" funded by the Microsoft PhD Scholarship Programme
2012–2014	Principal Investigator of the Vetenskapsrådet (VR) project "Scalable Floating-Point Reasoning for Embedded Systems Analysis"
2011–2014	CERTAINTY, STREP FP7 (Certification of Real Time Applications designed for Mixed Criticality)
2010	Principal Investigator of the Microsoft Software Engineering Innovation Foundation (SEIF) Award on "Testing Embedded Software with the Z3 SMT Solver"
2009–2013	MC substitute member of COST Action IC0901 "Rich-Model Toolkit"
2009–2010	CESAR, Artemis (Cost-efficient methods and processes for safety relevant embedded syst.)
2009–2010	MOGENTES, STREP FP7 (Model-based generation of tests for dependable embedded systems)
2002–2008	KeY (on deductive verification of Java programs)

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-40564-116615-37

Name of applicant
Ruemmer, Philipp

Date of birth
780925-7418

Title of research programme
Satisfiability Modulo Theories and Quantifiers

Appendix C

List of publications

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

Publications by Philipp Rümmer

Citation numbers from Google Scholar (2014-04-06).
The five most cited papers are marked with **Most cited**.

Peer-Reviewed Original Articles

- Byron Cook, Daniel Kroening, Philipp Rümmer, and Christoph M. Wintersteiger. Ranking function synthesis for bit-vector relations. *Formal Methods in System Design*, 43(1):93–120, 2013. Number of citations: 1.

- Angelo Brillout, Daniel Kroening, Philipp Rümmer, and Thomas Wahl. An interpolating sequent calculus for quantifier-free Presburger arithmetic. *Journal of Automated Reasoning*, 47:341–367, 2011. Number of citations: 37 (Google Scholar counted this paper together with our IJCAR 2011 paper, with same name). ✻

- Alastair F. Donaldson, Daniel Kroening, and Philipp Rümmer. Automatic analysis of DMA races using model checking and k-induction. *Formal Methods in System Design*, 39(1):83–113, 2011. Number of citations: 10.

- Wolfgang Ahrendt, Bernhard Beckert, Martin Giese, and Philipp Rümmer. Practical aspects of automated deduction for program verification. *KI - Künstliche Intelligenz*, 24(1):43–49, 2010. Number of citations: 3.

- Reiner Hähnle, Jing Pan, Philipp Rümmer, and Dennis Walter. Integration of a security type system into a program logic. *Theor. Comput. Sci.*, 402(2-3):172–189, 2008. Number of citations: 5.

Peer-Reviewed Conference Contributions

- Aleksandar Zeljić, Christoph M. Wintersteiger, and Philipp Rümmer. Approximations for model construction. In *IJCAR*, 2014. To appear. Number of citations: 0.

- Stephan Arlt, Cindy Rubio-González, Philipp Rümmer, Martin Schäf, and Natarajan Shankar. The gradual verifier. In *NASA Formal Methods*, 2014. To appear. Number of citations: 0.

- Philipp Rümmer and Pavle Subotić. Exploring interpolants. In *Formal Methods in Computer-Aided Design (FMCAD)*, pages 69–76. IEEE, 2013. Number of citations: 1. ✻

- Stephan Arlt, Philipp Rümmer, and Martin Schäf. A theory for control-flow graph exploration. In Dang Van Hung and Mizuhito Ogawa, editors, *Automated Technology for Verification and Analysis (ATVA)*, volume 8172 of *Lecture Notes in Computer Science*, pages 506–515. Springer, 2013. Number of citations: 1.

- Philipp Rümmer, Hossein Hojjat, and Viktor Kuncak. Classifying and solving Horn clauses for verification. In Ernie Cohen and Andrey Rybalchenko, editors, *VSTTE*, volume 8164 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2013. Number of citations: 1.

- Philipp Rümmer, Hossein Hojjat, and Viktor Kuncak. Disjunctive interpolants for Horn-clause verification. In *Computer Aided Verification (CAV)*, volume 8044 of *LNCS*, pages 347–363. Springer, 2013. Number of citations: 8.

- Hossein Hojjat, Radu Iosif, Filip Konečný, Viktor Kuncak, and Philipp Rümmer. Accelerating interpolants. In *Automated Technology for Verification and Analysis (ATVA)*, volume 7561 of *LNCS*, pages 187–202. Springer, 2012. Number of citations: 13.

- Hossein Hojjat, Filip Konečný, Florent Garnier, Radu Iosif, Viktor Kuncak, and Philipp Rümmer. A verification toolkit for numerical transition systems (tool paper). In *16th International Symposium on Formal Methods (FM)*, volume 7436 of *LNCS*, pages 247–251. Springer, 2012. Number of citations: 16.

- Philipp Rümmer. E-Matching with free variables. In *Proceedings, 18th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 7180 of *LNCS*, pages 359–374. Springer, 2012. Number of citations: 3. ✻

- Nannan He, Daniel Kroening, Thomas Wahl, Kung-Kiu Lau, Faris Taweel, Cuong M. Tran, Philipp Rümmer, and Sanjiv S. Sharma. Component-based design and verification in X-MAN. In *ERTS2 2012: Embedded Real Time Software and Systems, Toulouse, France*, 2012. Number of citations: 6.

- Alastair F. Donaldson, Nannan He, Daniel Kroening, and Philipp Rümmer. Tightening test coverage metrics: A case study in equivalence checking using k-induction. In *9th International Symposium on Formal Methods for Components and Objects, Graz, Austria, 2010*, volume 6957 of *LNCS*, pages 297–315. Springer, 2012. Number of citations: 1.

- Alastair F. Donaldson, Leopold Haller, Daniel Kroening, and Philipp Rümmer. Software verification using k-induction. In Eran Yahav, editor, *Proceedings, International Symposium on Static Analysis (SAS)*, volume 6887 of *LNCS*, pages 351–368. Springer, 2011. Number of citations: 15.

- Nannan He, Philipp Rümmer, and Daniel Kroening. Test-case generation for embedded Simulink via formal concept analysis. In Leon Stok, Nikil D. Dutt, and Soha Hassoun, editors, *Proceedings, Design Automation Conference (DAC)*, pages 224–229. ACM, 2011. Number of citations: 15.

- Alastair F. Donaldson, Daniel Kroening, and Philipp Rümmer. SCRATCH: a tool for automatic analysis of DMA races. In *PPOPP*, pages 311–312. ACM, 2011. Number of citations: 4.

- Angelo Brillout, Daniel Kroening, Philipp Rümmer, and Thomas Wahl. Beyond quantifier-free interpolation in extensions of Presburger arithmetic. In Ranjit Jhala and David Schmidt, editors, *Proceedings, International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 6538 of *LNCS*, pages 88–102. Springer, 2011. Number of citations: 14. ✻

- Daniel Kroening, Jérôme Leroux, and Philipp Rümmer. Interpolating quantifier-free Presburger arithmetic. In *17th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 6397, pages 489–503, 2010. Number of citations: 14.

- Angelo Brillout, Nannan He, Michele Mazzucchi, Daniel Kroening, Mitra Purandare, Philipp Rümmer, and Georg Weissenbacher. Mutation-based test case generation for Simulink models. In *Software Technologies Concertation on Formal Methods for Components and Objects 2009, Eindhoven, The Netherlands*, volume 6286, pages 208–227, 2010. Number of citations: 19.

- Angelo Brillout, Daniel Kroening, Philipp Rümmer, and Thomas Wahl. An interpolating sequent calculus for quantifier-free Presburger arithmetic. In *Proceedings, International Joint Conference on Automated Reasoning (IJCAR)*, volume 6173 of *LNCS*, pages 384–399. Springer, 2010. Number of citations: 37. **Most cited**.

- K. Rustan M. Leino and Philipp Rümmer. A polymorphic intermediate verification language: Design and logical encoding. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 6015 of *LNCS*, pages 312–327. Springer, 2010. Number of citations: 57. **Most cited**.

- Byron Cook, Daniel Kroening, Philipp Rümmer, and Christoph M. Wintersteiger. Ranking function synthesis for bit-vector relations. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 6015 of *LNCS*, pages 236–250. Springer, 2010. Number of citations: 28.

- Alastair F. Donaldson, Daniel Kroening, and Philipp Rümmer. Automatic analysis of scratch-pad memory code for heterogeneous multicore processors. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 6015 of *LNCS*, pages 280–295. Springer, 2010. Number of citations: 20.

- André Platzer, Jan-David Quesel, and Philipp Rümmer. Real world verification. In Renate A. Schmidt, editor, *International Conference on Automated Deduction, Montreal, Canada*, volume 5663 of *LNCS*, pages 485–501. Springer, 2009. Number of citations: 31. **Most cited**.

- Philipp Rümmer. A constraint sequent calculus for first-order logic with linear integer arithmetic. In *15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 5330 of *LNCS*, pages 274–289. Springer, 2008. Number of citations: 34. **Most cited**. ✻

- Helga Velroyen and Philipp Rümmer. Non-termination checking for imperative programs. In Bernhard Beckert and Reiner Hähnle, editors, *Tests and Proofs, Second International Conference, TAP 2008, Prato, Italy*, volume 4966 of *LNCS*, pages 154–170. Springer, 2008. Number of citations: 26.

- Christian Engel, Christoph Gladisch, Vladimir Klebanov, and Philipp Rümmer. Integrating verification and testing of object-oriented software. In Bernhard Beckert and Reiner Hähnle, editors, *Tests and Proofs, Second International Conference, TAP 2008, Prato, Italy*, volume 4966 of *LNCS*, pages 182–191. Springer, 2008. Number of citations: 12.

- Bernhard Beckert, Martin Giese, Reiner Hähnle, Vladimir Klebanov, Philipp Rümmer, Steffen Schlager, and Peter H. Schmitt. The KeY System 1.0 (deduction component). In F. Pfenning, editor, *International Conference on Automated Deduction, Bremen, Germany*, volume 4603 of *LNCS*, pages 379–384. Springer, 2007. Number of citations: 20.

- Philipp Rümmer and Muhammad Ali Shah. Proving programs incorrect using a sequent calculus for Java Dynamic Logic. In Yuri Gurevich and Bertrand Meyer, editors, *Tests and Proofs, First International Conference, TAP 2007, Zurich, Switzerland, February 12-13, 2007. Revised Papers*, volume 4454 of *LNCS*, pages 41–60. Springer, 2007. Number of citations: 14.

- Reiner Hähnle, Jing Pan, Philipp Rümmer, and Dennis Walter. Integration of a security type system into a program logic. In *2nd Symposium on Trustworthy Global Computing (TGC), Lucca, Italy*, volume 4661 of *LNCS*, pages 116–131. Springer, 2007. Number of citations: 13.

- Philipp Rümmer. Sequential, parallel, and quantified updates of first-order structures. In *Logic for Programming, Artificial Intelligence and Reasoning, Phnom Penh, Cambodia*, volume 4246 of *LNCS*, pages 422–436. Springer, 2006. Number of citations: 36. **Most cited**.

Book Chapters and Tutorials

- Wolfgang Ahrendt, Bernhard Beckert, Reiner Hähnle, Philipp Rümmer, and Peter H. Schmitt. Verifying object-oriented programs with KeY: A tutorial. In *5th International Symposium on Formal Methods for Components and Objects, Amsterdam, The Netherlands*, volume 4709 of *LNCS*, pages 70–101. Springer, 2007. Number of citations: 11.

- Philipp Rümmer. Construction of proofs. In Bernhard Beckert, Reiner Hähnle, and Peter H. Schmitt, editors, *Verification of Object-Oriented Software: The KeY Approach*, volume 4334 of *LNCS*, chapter 4, pages 177–242. Springer, 2007. Number of citations: 2. (Number of citations of the whole book: 395).

Workshop Papers

- Stephan Arlt, Philipp Rümmer, and Martin Schäf. Joogie: from Java through Jimple to Boogie. In *ACM SIGPLAN International Workshop on the State Of the Art in Java Program Analysis (SOAP)*, 2013. Number of citations: 1.

- Angelo Brillout, Daniel Kroening, Philipp Rümmer, and Thomas Wahl. Program verification via Craig interpolation for Presburger arithmetic with arrays. In *Informal proceedings of 6th International Verification Workshop at FLoC, Edinburgh, Scotland*, 2010. Number of citations: 1.

- Philipp Rümmer and Thomas Wahl. An SMT-LIB theory of binary floating-point arithmetic. In *Informal proceedings of 8th International Workshop on Satisfiability Modulo Theories (SMT) at FLoC, Edinburgh, Scotland*, 2010. Number of citations: 13.

- Alastair F. Donaldson, Daniel Kroening, and Philipp Rümmer. Analysing DMA races in multicore software. In *Informal proceedings, 3rd Workshop on Programming Language Approaches to Concurrency and Communication-Centric Software (PLACES'10) at ETAPS 2010*, 2010. Number of citations: 0.

- Daniel Kröning, Philipp Rümmer, and Georg Weissenbacher. A proposal for a theory of finite sets, lists, and maps for the SMT-LIB standard. In *Informal proceedings, 7th International Workshop on Satisfiability Modulo Theories at CADE 22*, 2009. Number of citations: 7.

- Richard Bubel, Andreas Roth, and Philipp Rümmer. Ensuring the correctness of lightweight tactics for JavaCard dynamic logic. *Electron. Notes Theor. Comput. Sci.*, 199:107–128, 2008. Number of citations: 6.

- Philipp Rümmer. A sequent calculus for integer arithmetic with counterexample generation. In *4th International Verification Workshop (VERIFY'07)*, volume 259 of *CEUR*, 2007. Number of citations: 16.

Developed Open Access Computer Programs

- **Eldarica**
  Symbolic model checker for software programs and recursive fixed-point equations
  Contributed significant parts of model checking core
  `http://lara.epfl.ch/w/eldarica`
  `http://www.philipp.ruemmer.org/eldarica-p.shtml`

- **Princess**
  Theorem Prover for First-Order Logic modulo Linear Integer Arithmetic, includes generation of Craig interpolants
  Main developer
  `http://www.philipp.ruemmer.org/princess.shtml`

- **Seneschal**
  Tool for synthesising linear ranking functions for programs expressible in Presburger arithmetic
  Sole developer
  `http://www.philipp.ruemmer.org/seneschal.shtml`

- **KeY**
  Deductive verification system for Java
  Contributed central reasoning components, including decision procedures for arithmetic and first-order fragments
  `http://www.key-project.org/`

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-40564-116615-37

Name of applicant
Ruemmer, Philipp

Date of birth
780925-7418

Title of research programme

Appendix D

Scientific report/account of scientific activity

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

Preliminary Scientific Report for VR 2011-6310 — Appendix D

Scalable Floating-Point Reasoning for Embedded Systems Analysis

Philipp Rümmer, *Uppsala University*

April 9, 2014

This document gives a report of results obtained in the ongoing VR project 2011-6310 (Project Research Grant for Junior Researcher), which is running from 2012-01-01 until 2014-12-31.

1 Scientific results

1.1 Floating-point decision procedures

The first goal of the project is the design of new algorithms for finding satisfying assignments of floating-point constraints, or for showing that no such assignments exist; corresponding solvers are relevant for applications like verification and test-case generation.

We defined a new framework for systematic application of approximations in order to speed up the construction of satisfying assignments [9]; this framework is applicable to floating-point constraints, but also to other theories. Our method is more general than previous techniques in the sense that approximations that are neither under- nor over-approximations (or the combination of both) can be used. An implementation (for floating-point numbers) was done within the Z3 solver, and enabled a speed-up of up to one order of magnitude over previous solvers.

We also continued our work on a standard format for floating-point constraints, extending the SMT-LIB format commonly used to interface SMT solvers. Our previous proposal for such a format was discussed extensively in the SMT community, revised several times, and finalised as an official SMT-LIB theory/logic in 2013 [1].

1.2 Arithmetic Craig interpolation

A second area investigated in the project is the development of efficient algorithms for Craig interpolation; such algorithms are an important building block in model checkers, where they are used to automatically solve fixed-point constraints and compute inductive invariants.

To obtain abstractions that are suitable for verification, model checkers rely on theorem provers or SMT solvers to find the right interpolants in a generally infinite lattice of interpolants for any given interpolation problem. This is in particularly challenging for the derivation of interpolants relating several variables in arithmetic theories. We developed a new semantic and solver-independent approach for systematically exploring interpolant lattices, based on the notion of *interpolation abstraction,* and instantiated the approach for several theories and verification problems [8, 5]. This resulted in interpolation procedures that are able to verify programs and systems that are beyond the scope of other verifiers.

We also investigated extended forms of Craig interpolation, such as tree interpolation and disjunctive interpolation, which are important for inter-procedural program analysis and analysis of concurrent systems. We obtained new complexity results for such problems, and clarified the relationship between Craig interpolation and constraints in form of recursion-free sets of Horn clauses [7, 6].

1.3 Analysis of numerical programs

A third area is the development of new analysis methods for programs and systems operating on numerical data.

We investigated the combination of two previously separate analysis methods, namely predicate abstraction with refinement via Craig interpolation, and acceleration, resulting in a robust analysis procedure that is able to handle systems where each individual methods fails or diverges [2]. This work

Table 1: Resources available for the project 2011-6310

Type of grant	Status	Source	Grant holder	Period	Amount
Project Research Grant for Junior Researcher	Granted	VR	Philipp Rümmer	2012–2014	2 400 kSEK
Salary of project leader (40%)		Uppsala University		2012–2014	1 050 kSEK
COST Action IC0901 "Rich model toolkit"	Granted	COST/ESF	Viktor Kuncak, Cesar Sanchez	2009–2013	$\approx$ 50 kSEK (travel expenses)

was done in the context of a newly defined intermediate format for *numerical transition systems,* intended both as a language for benchmarks and as an ingredient for constructing analysis systems [3].

Building on our results about Horn constraints, we also developed a new constraint-based analysis procedure for timed systems [4]. The new procedure is able to handle parameterised systems (with an unbounded or infinite number of components), and exploit compositionality to achieve better scalability.

2 Relationship to the new project

The proposed new project is about the development of theorem provers and SMT solvers that can efficiently handle theories and quantifiers in combination. The new project is related to the previous/ongoing project, but not overlapping in scope. Results of the previous project will be applicable in combination with outcomes of the new project: in particular, solvers for floating-point constraints developed here (Sect. 1.1) can be used within the solver framework to be designed in the new project. Vice versa, the solver to be developed in the new project can be used in the context of the results in Sect. 1.2, 1.3.

3 Available resources

An overview of funding is given in Table 1.

The funding provided through VR was used to employ one PhD student (Aleksandar Zeljić) from April 2012; the ongoing research of the student is on decision and interpolation procedures for floating-point arithmetic.

The salary of the project leader was provided by the IT Department at Uppsala University.

In 2011, the project leader joined the COST network "Rich model toolkit" as MC substitute member for Sweden; the network subsequently covered expenses for several research visits and travel to meetings, in particular for exchange with Daniel Kroening's group at Oxford University, Viktor Kuncak's group at EPFL Lausanne, and Radu Iosif at VERIMAG Grenoble.

References

[1] Martin Brain, Cesare Tinelli, Philipp Rümmer, and Thomas Wahl. An automatable formal semantics for IEEE-754 floating-point arithmetic. 2014. submitted.

[2] Hossein Hojjat, Radu Iosif, Filip Konečný, Viktor Kuncak, and Philipp Rümmer. Accelerating interpolants. In *Automated Technology for Verification and Analysis (ATVA)*, volume 7561 of *LNCS*, pages 187–202. Springer, 2012.

[3] Hossein Hojjat, Filip Konecný, Florent Garnier, Radu Iosif, Viktor Kuncak, and Philipp Rümmer. A verification toolkit for numerical transition systems - tool paper. In *FM*, pages 247–251, 2012.

[4] Hossein Hojjat, Philipp Rümmer, Pavle Subotic, and Wang Yi. Uniform analysis for communicating timed systems. Technical report, EPFL, 2013.

[5] Jérôme Leroux, Philipp Rümmer, and Pavle Subotić. Guiding Craig interpolation with domain-specific abstractions. 2014. submitted.

[6] Philipp Rümmer, Hossein Hojjat, and Viktor Kuncak. Classifying and solving Horn clauses for verification. In *VSTTE*, pages 1–21, 2013.

[7] Philipp Rümmer, Hossein Hojjat, and Viktor Kuncak. Disjunctive interpolants for Horn-clause verification. In *Computer Aided Verification (CAV)*, volume 8044 of *LNCS*, pages 347–363. Springer, 2013.

[8] Philipp Rümmer and Pavle Subotic. Exploring interpolants. In *FMCAD*, pages 69–76, 2013.

[9] Aleksandar Zeljić, Christoph M. Wintersteiger, and Philipp Rümmer. Approximations for model construction. In *IJCAR*, 2014. to appear.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-40564-116615-37

Name of applicant
Ruemmer, Philipp

Date of birth
780925-7418

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

Budget and Research Resources (Appendix N)

Philipp Rümmer, *Uppsala University*

April 9, 2014

1 Proposed Budget

- **Project leader/applicant** (25% of salary)

  It is planned that the project leader works 25% on the project proposed here, active in all of SP1–SP4, and 25% in the framework project "Multicore Embedded Systems: Timing Predictability and Performance" submitted simultaneously (main applicant: Wang Yi).

- **PhD student 1** (80% of salary for last two years of study, 2017 and 2018)

  The PhD student Peter Backeman started his studies in November 2013, and is funded by a Microsoft PhD grant/scholarship (held by the applicant) until end of 2016. Peter Backeman is supervised by the applicant and working on the related project "First-Order Satisfiability Modulo Theories," which overlaps with SP1 and SP4 of the proposed VR project.

  Since the Microsoft PhD scholarship only covers three years of studies (2013–2016), we apply for VR funding for the last two years (2017 and 2018).

- **PhD student 2** (80% of salary for 2015–2018)

  A newly employed PhD student, whose research topics focus on SP2, SP3 and SP4 of the proposed project.

- **Premises:** 4×25 kSEK

  Cost of office space at Uppsala University.

- **Travel costs:** 4×64 kSEK

  Covering about 2 trips to conferences or research visits per person/year, including travels of the project leader.

- **Minor expenses:** 4×13 kSEK

  Expenses for books, printing, minor equipment like desktop computers, etc.

2 Total research resources of the project

Type of grant	Status	Source	Grant holder	Grant period	Amount
Project Research Grant	Applied	VR	Philipp Rümmer	2015–2018	4 642 kSEK
Microsoft PhD scholarship	Granted	Microsoft Research	Philipp Rümmer	2013–2016	900 kSEK

3 Proportion of annual costs applied for from VR

Year	2015	2016	2017	2018	Total
Proportion	73%	74%	100%	100%	89%

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-40564-116615-37

Dnr
2014 -

Name of applicant
Ruemmer, Philipp

Date of birth
780925-7418

Reg date
2014-04-09 19:22:19

Project title
Satisfiability Modulo Theories and Quantifiers

NT

NT-2, NT-1

Research grants NT April 9, 2014
Project Research Grant
*Övrigt generellt
Uppsala universitet

Appendix S

Signatures

2015	2016	2017	2018	2019
815	850	1441	1537	

2015-01-01 -- 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
- International Recruitment

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

2014-40564-116615-37

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