VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kansliets noteringar Kod	Dnr
2014-48243-115404-23	2014-4864

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
1252	1255	1381	1385	1561

APPLICANT

Name(Last name, First name) Abel, Andreas	**Date of birth** 740624-6491	**Gender** Male
Email address andreas.abel@gu.se	**Academic title** PhD	**Position** Senior lecturer
Phone +46317721731	**Doctoral degree awarded (yyyy-mm-dd)** 2006-07-14	

WORKING ADDRESS

University/corresponding, Department, Section/Unit, Address, etc.
Göteborgs universitet
Dept. of Computer Science and Engineering
Computer Science and Engineering
Rännvägen 6b
41296 Göteborg, Sweden

ADMINISTRATING ORGANISATION

Administrating Organisation
Göteborgs universitet

DESCRIPTIVE DATA

Project title, Swedish (max 200 char)
Termineringscertifiering för program och bevis med beroende typer

Project title, English (max 200 char)
Termination Certificates for Dependently-Typed Programs and Proofs via Refinement Types

Abstract (max 1500 char)
Proof assistants are software systems that allow users to construct large proofs which are checked step-by-step by the computer. Coq, a proof assistant based on dependent types, has recently enabled prize-winning formalizations of mathematical proofs and safety-critical software projects whose soundness can no longer be rigorously asserted by human reviewers. In dependently-typed proof assistants, such as Coq and Agda, a proof is represented as a functional program which needs to be checked for termination. The current termination checkers are too sensitive to refactorings of a program and do not work well with higher-order functions. Further, they do not produce certificates of termination, and have never been rigorously verified themselves.

This project aims to improve on the current state-of-the-art by fulfilling the following goals:

* Develop a new, more powerful, type-based termination check that produces a termination certificate, but is backwards compatible. This checker will operate on a refinement type layer put on top of the type system of the proof assistant, which will be constructed automatically for standard cases, or manually for advanced uses of type-based termination.

* Prove the correctness of the refinement type system and its compatibility with the new Homotopy Type Theory.

* Implement the refinement type system for the proof assistant Agda.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-48243-115404-23

Name of Applicant
Abel, Andreas

Date of birth
740624-6491

Abstract language
English

Keywords
termination, dependent types, refinement types, sized types,

Review panel
NT-2

Project also includes other research area

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

Kod	2014-48243-115404-23
Name of Applicant	Abel, Andreas
Date of birth	740624-6491

ENCLOSED APPENDICES

A, B, C, N, S

APPLIED FUNDING: THIS APPLICATION

Funding period (planned start and end date)

2015-01-01 -- 2019-12-31

Staff/ salaries (kSEK)

Main applicant	% of full time in the project	2015	2016	2017	2018	2019
Andreas Abel	40	517	530	540	551	562
Other staff						
PhD student	80	578	592	643	697	737
	Total, salaries (kSEK):	1095	1122	1183	1248	1299

Other project related costs (kSEK)	2015	2016	2017	2018	2019
Computer equipment and literature	28	3	29	3	30
Travel and conferences	129	130	152	134	205
Publication costs			17		27
Total, other costs (kSEK):	157	133	198	137	262

Total amount for which applied (kSEK)

2015	2016	2017	2018	2019
1252	1255	1381	1385	1561

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)

POPULAR SCIENCE DESCRIPTION

Popularscience heading and description (max 4500 char)

För forskningsfronten framåt inom datorstödd teorembevisning

När Fermat 1636 skrev om sin stora sats, ?Jag har ett i sanning underbart bevis för detta påstående, men marginalen är alltför smal för att rymma det? (i sin kopia av boken Arithmetica) så hade han helt klart rätt: marginalen var verkligen för smal för att rymma beviset. Det tog 350 år, och generationer av matematiker, innan beviset blev fullständigt.

Vissa bevis är inte bara svåra att hitta men också svåra att kontrollera. Ett till synes oskyldigt påstående är Fyrfärgssatsen som säger att det behövs högst fyra färger för att färglägga en karta på ett sådant sätt att inga angränsande regioner har samma färg. Problemet formulerades 1852 men det dröjde till 1976 innan Appel & Haken först lyckades bevisa det. De reducerade problemet till 1936 fall som de sedan använde ett program för att kontrollera. Beviset möttes av skepsis eftersom bevis anses vara en sekvens påståenden direkt sammanlänkade av enkla logisk regler, och inte beroende av datorberäkningar.

Det visade sig senare att det finns ett bevis av fyrfärgssatsen som är en ?enkel? påståendekedja, men det beviset är för långt för någon matematiker att kontrollera. År 2005 lyckades Gonthier med hjälp av datorprogrammet Coq konstruera ett sådant bevis. Coq hjälper vid beviskonstruktion och kontrollerar att alla steg är korrekta. Coq följer ?bara? de logiska regler som är inprogrammerade i systemet. Ett matematiskt bevis i Coq ser själv ut som ett datorprogram. Detta är ingen slump; de berömda logikerna Curry & Howard upptäckte att bevis och program har många likheter. I Coq kan man skriva program och bevis i samma språk.

Det finns dock en viktig skillnad mellan program och bevis. En oändlig följd av påståenden räknas inte som ett bevis men vissa program kan (och ska) köra i all oändlighet. Bevisassistenter som Coq använder sig av en termineringskontroll som garanterar att varje bevis och program stannar inom ändlig tid. Denna kontroll är en viktig del av Coq: om den är felaktig kan oändliga, ogiltiga bevis accepteras. Förs att vara på den säkra sidan får termineringskontrollen bara acceptera program och bevis vars terminering kan garanteras. En berömd sats inom datavetenskapen, stopproblemet, säger att en perfekt termineringskontroll inte kan implementeras, men vi vill i varje fall komma nära.

Vi har som mål att förbättra termineringskontrollen i bevisassistenter. För det första vill vi göra termineringskontrollen intelligentare så att den kan känna igen fler terminerande program. Vi använder oss av en ny teknik som kallas typ-baserad termineringskontroll. Den använder ett programs typ, som är en abstrakt beskrivning av programmets beteende, för att guida termineringskontrollen. Med denna teknik kan man kontrollera program som består av abstrakta byggstenar som kallas högre ordningens funktioner.

För det andra kommer den nya termineringskontrollen inte bara att svara ja eller nej, utan kommer att ge en formell motivering till varför programmet verkligen terminerar. Denna motivering är i form av ett termineringsmått och beskriver vilken kvantitet som blir mindre varje gång programmet går igenom en loop. För eller senare blir detta mått noll och då stannar programmet. När termineringsmåttet har hittats är det lätt att verifiera att programmet verkligen terminerar och att svaret var korrekt. Man behöver då inte lita blint på termineringskontrollen (som ofta är ganska komplex).

Sammanfattningsvis kommer detta projekt att demonstrera att den nya metoden är korrekt, dvs att om ett giltigt mått hittas så kommer beviset som konstruerats av bevisassistenten att vara korrekt. För att visa detta måste varje bevisregel som används kontrolleras i en så kallad modell av bevisassistenten.

Projektet kommer att påverka det dagliga arbetet för datavetare och matematiker som använder bevissassistenter för att kontrollera sina bevis. En bättre termineringskontroll medför att de kommer att kunna skriva sina bevis på ett naturligare sätt. Eftersom även termingsmått produceras kommer de att kunna lita mer på sina bevis och sova bättre om natten. Och eftersom arbetet bygger på en solid modellkonstruktion hoppas vi kunna övertyga fler matematiker att använda bevisassistenter. De kommer inte längre behöva skriva kryptiska anteckningar i marginalen!

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod	2014-48243-115404-23
Name of applicant	Abel, Andreas
Date of birth	740624-6491

Title of research programme

Termination Certificates for Dependently-Typed Programs and Proofs via Refinement Types

Appendix A

Research programme

Appendix A: Research Program

Termination Certificates for Dependently-Typed Programs and Proofs via Refinement Types

Andreas Abel

1 Purpose and Aims

Proof assistants are software systems that allow their users to construct large proofs whose correctness is checked step-by-step by the computer. Coq, a proof assistant based on dependent types, has recently enabled prize-winning formalizations of mathematical proofs and safety-critical software projects whose correctness can no longer be rigorously asserted by human reviewers. Dependently-typed proof assistants, such as Coq and Agda, built on the Curry-Howard isomorphism, which states that constructing a proof of a proposition corresponds to writing a functional program of the respective dependent type. A proof by induction corresponds to a terminating recursive program. Coq and Agda come with a termination checker that looks at the code of a recursive program and tries to verify that at each recursive call, the program arguments become structurally smaller.

This structural termination checker is sensitive to refactorings of a program and does not work well with coding styles that use advanced abstraction principles such as higher-order functions. Further, it does not produce a certificate of termination that can be verified easily, with local checks only. Finally, the termination checkers of Coq and Agda are complicated and have never been fully verified. This makes them the weakest link in current proof assistant implementation.

An alternative to structural termination checking is type-based termination checking, which works on the precise types of programs rather than on their code. This technique is much more powerful when dealing with higher-order functions, and it produces termination certificates that can be verified by a standard type-checking pass of the program. However, current formulations of type-based termination are not backwards-compatible, meaning that existing Coq and Agda developments that rely on the structural termination check cannot be used with type-based termination.

This project aims to improve on the current state-of-the-art by fulfilling the following goals:

1. **Develop a type-based termination check that is backwards compatible.** In contrast to previous versions of type-based termination, the type system of the proof assistant will not be modified to incorporate rules for termination. Instead, a second type layer will be put on top refining the underlying types; the type-based termination check will operate on the refinement type layer. Refinement types for existing terminating programs will be constructed automatically. Only in order to use the additional power of type-based termination, refinement types have to be added manually.

2. **Prove the correctness of the refinement type system.** This involves the construction of a model of the type system to prove that every program accepted by the type-based termination check terminates indeed. The new termination check should also

be compatible with Homotopy Type Theory, a new foundation of mathematics, developed in an initiative spearheaded by Fields medalist Vladimir Voevodsky.

3. **Implement the refinement type system for the proof assistant Agda** developed at Gothenburg University and carry out case studies demonstrating the increased power of the termination checker.

2 Survey and Introduction to the Project

With the increasing complexity of structures investigated in mathematics and constructed in information technology, the need for machine-checked verification of properties has fueled the development of *proof assistants*. They allow their users to construct large proofs of logical propositions whose correctness is checked step-by-step by the computer. A prominent proof assistant, *Coq* [39], has recently enabled several huge verifications that were hopeless to carry out without machine help: in mathematics, Georges Gonthier [34] finished a proof of the prominent *Four Color Theorem* in 2005 and, in a team effort, the *Feit-Thompson Theorem* in 2012 [42][1]. In computer science, Xavier Leroy, leading the *CompCert* project, completed the verification of a realistic compiler for a subset of the C language [43]. For their achievements, Gonthier was awarded the *2011 Grand Prize in Computer Science of the EADS* (European Aeronautic Defence and Space) [41] and Leroy the *Microsoft Research 2012 Verified Software Milestone Award* [38].

2.1 Type Theory

The proof assistant Coq is based on a variant of *Type Theory (TT)*, a logical foundation of mathematics and reasoning developed by Per Martin-Löf [45] and others. TT is based on the observation, coined *de Bruijn-Curry-Howard-Correspondence* or *Propositions-as-Types* paradigm, that writing a computer program and proving a mathematical theorem are very similar activities, to the point that one can have single formal language for both programming and proving. TT thus naturally supports integrated development of programs and their verification. Further, it capitalizes on the established means of programming language technology to structure large developments.

In the last years, Type Theory has gained wider recognition as a foundation for mathematical reasoning and has even been covered in popular science media [20]. This interest has been sparked by the 2002 Fields medalist Vladimir Voevodsky who turned to machine-assisted reasoning after discovering that some of his published results have been incorrect [56]. Realizing that the concept of equality in Type Theory can be interpreted as homotopy, he launched a program to develop *Homotopy Type Theory (HoTT)* [55] as a foundation for formalized mathematics (2012-2013). HoTT, by interpreting *isomorphic types as equal*, sheds new light on the dispute about the right notion of equality in TT that has lasted for decades [36, 35, 17, 18]. If HoTT can deliver a solution of the equality issue, it has the potential to become the "new TT". Currently, libraries for reasoning in HoTT are being developed in (patched versions of) Coq and Agda.

2.2 Agda

Agda [49] is the second-most popular proof assistant based on type theory. While it does not have the industrial strength of Coq yet, it attracts users due to the well-designed lean syntax, a flexible pattern-matching facility, a means of interactive program development and

[1]The proof of the Feit-Thompson Theorem required 170000 lines of code, 15000 definitions, and 4300 theorems.

a powerful termination checker. As of now, close to 100 research papers have been written on proofs and programs developed in Agda [16]. For the last 5 years (2009-2013) around 15 papers using Agda have been published each year. Agda is used as a tool to teach logic and program verification at several universities, the Agda wiki lists 15 Agda courses [15]. In average, there are 1000 messages per year on the Agda mailing list (2009-2013) [14].

2.3 Termination and Logical Consistency

Even a formalized machine-checked proof can be incorrect if the proof checker contains unsound principles or bugs. It is thus highly desirable to build a proof assistant on a small trusted foundation with few, perspicuous proof rules. Coq is built on the *Calculus of (Co)Inductive Constructions (CIC)* which is described in the Coq Reference Manual [40]. However, this description does not include the full rules for the termination checker [21], which is needed to check proofs by induction and coinduction. Indeed, to improve usability, the termination checker has been extended with heuristics that have not been rigorously verified yet. One particular heuristics of the implemented termination checker which is inconsistent with Homotopy Type Theory has been discovered and fixed very recently [52, 30].

Agda's termination checker [7], which I developed in the 1990s, suffers from the same incompatibility with Homotopy Type Theory [6]. As the structural termination order (roughly, a subterm order on program arguments) is not preserved by isomorphisms, a structural termination checker does not harmonize with Homotopy Type Theory which interprets isomorphic types as equal.

2.4 Type-Based Termination

An alternative to the prevailing *structural* termination check is *type-based termination* [2], which has been developed independently in the areas of functional programming [37] and Type Theory [46, 19, 31]. The central idea is to refine data types to *sized types* that carry information about the size of the data structures. The type system is thus able to track how functions modify the size of the data structures they manipulate. In call graphs of mutually recursive functions, it can be checked—based on the size information in the type system—that on each call cycle the size of function arguments decreases with regard to a well-founded order. Type-based termination has several advantages over structural termination:

1. Type-based termination naturally provides *termination certificates*. In fact, type-based termination is formulated in terms of typing rules which can be checked locally. A fully type-annotated program, which is the result of a type-inference or type-reconstruction process, already contains all the evidence for termination. No separate oracle for termination, which might contain bugs, is needed; this substantially reduces the size of the trusted code base of a proof assistant.

2. Since a type is an *abstraction* of program behavior, a termination checker based on types is less dependent on the specific coding of a function. Indeed, the prevailing termination checkers for Coq and Agda require a certain formulation of a function to validate its termination, which demands experience and patience from the user and often causes confusion, as witnessed by recurring postings on the respective mailing lists.

   The limited capability of structural termination checkers to deal with abstraction is especially crippling for higher-order functional programming. Already the use of simple program combinators like function composition or the monadic bind operation leads to rejection of perfectly terminating programs by checkers that rely on the untyped program syntax only. In contrast, type-based termination is *compositional* and does not limit the use of higher-order and abstract programming patterns [4].

3. As argued by Giménez et al. [31, 23], a termination criterion woven into the type system simplifies its proof of *soundness* considerably. Logical systems such as Type Theory are proven correct by exhibiting a model that interprets every type as the set of its terminating programs, and—viewed from the other side of the propositions-as-types paradigm—every proposition as the set of its proofs [54, 32, 29, 44, 57, 33, 28]. Construction of such models requires high expertise and technical skills already for systems without recursion [53, 47, 48, 9, 13] and it is not obvious where to fit in the rules for termination checking. Indeed, no article has been published that rigorously proves the correctness of the termination checkers for Coq or Agda.

   Type-based termination, however, fits well into state-of-the-art model constructions for advanced type systems [2, 24, 12]. Soundness proofs in the presence of dependent types have been given by Xi [58] for Dependent ML, Blanqui [25] for his Calculus of Algebraic Constructions, and Sacchini for a sized variant of the Calculus of Inductive Constructions [50].

4. Type-based termination is conjectured to be *compatible with Homotopy Type Theory*; at least, the counterexamples given for structural termination do not apply [5]. Besides the absence of counterexamples, there are high-level reasons for soundness: Sized types, being a more fine-grained structure on types, allow fewer isomorphisms, and in particular rule out the isomorphisms that can—via a type-cast—delude a structural termination checker into attributing a smaller structural size to a function argument. Actually *proving* the soundness of type-based termination for Homotopy Type Theory is a large enterprise, which is part of this project proposal.

2.5 Concerns of the Research Community

Given all the advantages, why has the Type Theory community not whole-heartedly embraced type-based termination?

Concern 1 *"It is complicated".* What is meant here is that in addition to the advanced concepts of dependent types, equality, and universes of TT, another dimension is introduced that the user has to familiarize himself with: sizes. With the prevailing structural checkers, termination is independent of the type system, and the user can just write his code and hope it passes the termination check. However, to reconcile *rejected* code with the structural termination checker, substantial reformulations and construction of explicit termination arguments [26, 27] are required. While the additional power of type-based termination would be appreciated, the initial costs of using it are currently considered too high.

Concern 2 *It is invasive.* In the current formulation of type-based termination, data types have to be introduced as sized types from the beginning. However, large libraries of Coq- and Agda-code exist that do not use sized types. Currently, a user that needs a sized version of a datatype cannot use the existing libraries for the non-sized version.

Concern 3 *It is touching the foundations.* TT, as a foundation for mathematics and reasoning, is backed by the test of time, by implementations, and by scientist with a strong reputation. Attempts to change TT at its core are met with some (understandable) reservation by the community.

My proposal to use a refinement type system on top of TT, as described in the next section, aims to address these concerns.

3 Project Plan

I propose to develop a type-based termination criterion for TT in form of a refinement type system placed on top of TT. To counter the concerns with current type-based termination, the language of TT remains intact as such as *underlying type system*, but data types are annotated with size expressions in the *refinement layer*, called *sizedTT*. *Erasure* of sizes takes programs from the refinement layer back to TT. Further, there is a standard *elaboration* of data types into sized types that does not require assistance by the user (addresses Concern 1), and a refinement of TT-programs to programs with size annotations that certify termination. Those size annotations are provided by a variant of the current, structural termination checker. The ambitious goal is that current TT-libraries based on the old termination criterion can be automatically elaborated into sizedTT with very little changes to the source code (addresses Concern 2).

To exploit the full power of type-based termination, a syntax for sizedTT will be provided such that users can make termination measures explicit. Further, in sizedTT they can naturally state size-relations between input and output of functions that can be used to establish termination of programs that call them.

The two-layered approach is important to formulate the correct notion of definitional equality used by the type checker of sizedTT. In particular, two values which are equal in TT but contain different embedded size expression in sizedTT need still to be considered equal, since sizes serve only as termination certificates and must be computationally irrelevant [22, 13].

The goal of the project is a type-based termination criterion that is (1) verified by a semantic soundness proof (addresses Concern 3), (2) consistent with Homotopy Type Theory, and (3) integrated into the dependently-typed language Agda in a backwards-compatible way (addresses concerns 1 and 2). The project is carried out by me (40% of my time) and a PhD student that will devote his full research time (80%) to the project.

3.1 Theoretical Part I: Development of core-sizedTT

As first step, we develop core-sizedTT, a cut down version of sizedTT that allows us to explore the problem and the necessary proof techniques in a simplified, but not over-simplified setting. Core-sizedTT will be a pure functional language with a sized type of natural numbers, dependent function types, size quantification, and a predicative universe of small types. The universe allows us to simulate a distinguishing feature of TT: types defined by case distinction on values. Although they substantially complicate the construction of a model of core-sizedTT, we need to consider universes from the beginning to not get stuck on a toy version of sizedTT that does not scale. In fact, a dependently-typed refinement of a language without universes has already been implemented by Xi [58], yet his semantic correctness proof cannot be extended to universes.

In parallel with the design of core-sizedTT we construct a model that proves its soundness, i.e., the termination of each well-typed program. I expect that previously researched models of dependent types [11, 13] can be combined with those of sized types [12], although preliminary investigations suggest that it is not without technical challenge, and the concept of erasure has to be integrated at the right spot. The work of Blanqui [25] might provide some inspiration, but does not outright solve our problem, because it is concerned with a calculus where programs do not carry size annotations in the first place, and thus type-checking is undecidable. Also, Sacchini's solution [50, 51] cannot be reused since it uses proof-theoretically heavy machinery which we must avoid to gain the desired acceptance by the community.

For the PhD student, work on a soundness proof is a good opportunity to study TT and its models. I expect this phase of the project to be completed within 1.5 years and leading to a publication in a major venue.

3.2 Theoretical Part II: Development of full sizedTT

As second step, core-sizedTT will be extended to full sizedTT by addition of arbitrary sized types, in particular, dependent inductive and coinductive data types, and an equality type. Simultaneously, the model for core-sizedTT will be extended to full sizedTT. This will be beneficial for the Agda language which, as of now, lacks a rigorous description and soundness proof.

At this point, we need to ensure consistency with HoTT, which will require the study of models of HoTT. My colleague Thierry Coquand is actively researching HoTT with a team of PhD students and researchers; being in same research group as him will be of great value to succeed. Still, it is unclear whether we will be able to formally prove the consistency with HoTT. Models of HoTT are just being developed, and it is not clear how far they have progressed until them. If we cannot deliver the full result, we will at least make sure that sizedTT is faithful to TT and no obvious contradictions to HoTT exist.

Due to the extended study of HoTT needed for this phase, the duration is hard to estimate, but I expect it to take 1 year, yielding another publication.

3.3 Practical Part: Integration into Agda

Backed by a solid soundness proof, we are ready to integrate sizedTT into Agda. This involves (1) the design of a concrete syntax for refinement types, (2) an integration of sized types into the internal syntax of Agda, (3) an elaboration procedure that adds size annotations, and (4) a revamping of the current termination checker to produce certifying termination measures. Although I have good familiarity with the Agda code base due to 3.5 years of intensive contributions to Agda (1200 patches), I expect this phase to take full 2 years. A piece of software like Agda (260 modules, 70000 loc) is hard to modify, even more so in a backwards-compatible way. I expect two publications out of this: one describing the evidence-producing termination checker, and one case-study that demonstrates how to apply the new Agda to practical problems, such as formalizations of theorems from computer science, or interesting dependently typed programs.

The following timetable summarizes the work and publication plan, including the licentiate thesis (lic) of the PhD candidate. I expect a minimum of four high-quality publications that will allow the PhD student to write a cumulative thesis.

year 1		year 2		year 3		year 4		year 5	
core-sizedTT			sizedTT		Agda integration				buffer
		paper 1	paper 2	lic			paper 3	paper 4	thesis

4 Significance

A successful completion of this project will be beneficial for the TT research community, the growing user community of TT-based proof assistants in research and education, and potentially industrial users of TT-based verification techniques.

- Researchers in Type Theory will benefit from our specification and soundness proof of sizedTT, which will promote Agda as a trustworthy proof assistant. Currently, the theory behind Agda is underspecified, which is criticized by scientific users of Agda with interest in foundations. Since Coq is based on a variation of TT, we expect our theoretical results to be transferable to Coq to a large extent.

- Users of Agda will benefit from the more principled, more robust, and more powerful termination checker we will develop, and their productivity will increase. We expect that this will also provide incentive for the Coq developers to switch to type-based termination, such that even the larger Coq community is influenced.
- Better proof assistants make formal verification of real-world software and large mathematical proofs more feasible. A better termination checker which allows more direct encodings will increase the attractiveness of proof assistants.

5 Preliminary Results

The current termination checker [7] of Agda has been implemented and maintained by me. Through my studies and publications (2003-2013) I have gained expertise in both sized non-dependent types [1, 2, 12] and dependent non-sized types and their models [10, 13] and initial experience on computational irrelevance which is a fundamental property of size expressions mixed into program code. I have implemented a prototypical dependently-typed language with sized types, MiniAgda [3, 4], and a prototypical extension of Agda by sized types [8]. (See the detailed discussion in Section 2.4.) While being a step into the right direction, both systems lack the solid theoretical backing I am planning to provide through this project, and both systems fall short of another main project goal: providing type-based termination in a *backward-compatible* way.

6 International Collaboration

In my work on type-based termination, I have recently collaborated [12] with Brigitte Pientka (McGill University, Montreal, Canada). Further, I hold active research connections to James Chapman [8] and Tarmo Uustalu at the Institute of Cybernetics in Tallinn, Estonia, and Lars Birkedal at Aarhus University, Denmark. I am intensively collaborating with the other Agda developers; locally, here at Gothenburg University, with Ulf Norell and Nils Anders Danielsson, and worldwide through the half-annual Agda Implementor's Meetings with participants from three continents. Finally, I am well-connected in the TYPES community (PC member 2014), including researchers at INRIA, Paris (Hugo Herbelin, Matthieu Sozeau, Frédéric Blanqui), in Nottingham (Thorsten Altenkirch), and in Munich (Martin Hofmann, Ulrich Schöpp). But since I joined the CSE department (Chalmers and Gothenburg University) last September, I have all the expertise on Type Theory next door (Thierry Coquand, Peter Dybjer, Bengt Nordström).

7 Independent Line of Research

While this project continues work done during my PhD, type-based termination has always been my own research which I pursued independently of my PhD supervisor Martin Hofmann (who has not worked on this topic).

References

[1] Andreas Abel. Termination and guardedness checking with continuous types. In M. Hofmann, editor, *Proc. of the 6th Int. Conf. on Typed Lambda Calculi and Applications, TLCA 2003*, volume 2701 of *Lect. Notes in Comput. Sci.*, pages 1–15. Springer, June 2003.

[2] Andreas Abel. *A Polymorphic Lambda-Calculus with Sized Higher-Order Types*. PhD thesis, Ludwig-Maximilians-Universität München, 2006.

[3] Andreas Abel. MiniAgda: Integrating sized and dependent types. In Ana Bove, Ekaterina Komendantskaya, and Milad Niqui, editors, *Wksh. on Partiality And Recursion in Interactive Theorem Provers (PAR 2010)*, volume 43 of *Electr. Proc. in Theor. Comp. Sci.*, pages 14–28, 2010. doi:10.4204/EPTCS.43.2.

[4] Andreas Abel. Type-based termination, inflationary fixed-points, and mixed inductive-coinductive types. *Electr. Proc. in Theor. Comp. Sci.*, 77:1–11, 2012. Proceedings of FICS 2012.

[5] Andreas Abel. Message on the coq-club mailing list, January 2014. URL: https://sympa.inria.fr/sympa/arc/coq-club/2014-01/msg00099.html.

[6] Andreas Abel. Structural termination order incompatible with univalence, April 2014. URL: http://code.google.com/p/agda/issues/detail?id=1023.

[7] Andreas Abel and Thorsten Altenkirch. A predicative analysis of structural recursion. *J. Func. Program.*, 12(1):1–41, January 2002. doi:10.1017/S0956796801004191.

[8] Andreas Abel and James Chapman. Normalization by evaluation in the delay monad: A case study for coinduction via copatterns and sized types. In *International Workshop on Mathematically Structured Functional Programming, colocated with ETAPS 2014, 5-13 April 2014, Grenoble, France*, April 2014.

[9] Andreas Abel and Thierry Coquand. Untyped algorithmic equality for Martin-Löf's logical framework with surjective pairs. *Fundam. Inform.*, 77(4):345–395, 2007. TLCA'05 special issue. URL: http://fi.mimuw.edu.pl/abs77.html#15.

[10] Andreas Abel, Thierry Coquand, and Peter Dybjer. Normalization by evaluation for Martin-Löf Type Theory with typed equality judgements. In *Proc. of the 22nd IEEE Symp. on Logic in Computer Science (LICS 2007)*, pages 3–12. IEEE Computer Soc. Press, 2007.

[11] Andreas Abel, Thierry Coquand, and Miguel Pagano. A modular type-checking algorithm for type theory with singleton types and proof irrelevance. *Logical Meth. in Comput. Sci.*, 7(2:4):1–57, May 2011. URL: http://arxiv.org/pdf/1102.2405.

[12] Andreas Abel and Brigitte Pientka. Wellfounded recursion with copatterns: A unified approach to termination and productivity. In Greg Morrisett and Tarmo Uustalu, editors, *Proc. of the 18th ACM SIGPLAN Int. Conf. on Functional Programming, ICFP'13*, pages 185–196. ACM Press, 2013.

[13] Andreas Abel and Gabriel Scherer. On irrelevance and algorithmic equality in predicative type theory. *Logical Meth. in Comput. Sci.*, 8(1:29):1–36, 2012. TYPES'10 special issue.

[14] Agda mailing list, 2014. URL: https://lists.chalmers.se/pipermail/agda/.

[15] Courses using Agda, April 2014. URL: http://wiki.portal.chalmers.se/agda/pmwiki.php?n=Main.Courses.

[16] Papers using Agda, April 2014. URL: http://wiki.portal.chalmers.se/agda/pmwiki.php?n=Main.Publications.

[17] Thorsten Altenkirch. Extensional equality in intensional type theory. In *Proc. of the 14th IEEE Symp. on Logic in Computer Science (LICS'99)*, pages 412–420, 1999.

[18] Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. Observational equality, now! In Aaron Stump and Hongwei Xi, editors, *Proc. of the Wksh. Programming Languages meets Program Verification, PLPV 2007*, pages 57–68. ACM Press, 2007. doi:10.1145/1292597.1292608.

[19] Roberto M. Amadio and Solange Coupet-Grimal. Analysis of a guard condition in type theory (extended abstract). In Maurice Nivat, editor, *Proc. of the 1st Int. Conf. on Foundations of Software Science and Computation Structure, FoSSaCS'98*, volume 1378 of *Lect. Notes in Comput. Sci.*, pages 48–62. Springer, 1998. doi:10.1007/BFb0053541.

[20] Jacob Aron. Mathematicians think like machines for perfect proofs, June 2013. URL: http://www.newscientist.com/article/dn23749-mathematicians-think-like-machines-for-perfect-proofs.html.

[21] Bruno Barras. The syntactic guard condition of coq. Talk at the Journée "égalité et terminaison" du 2 février 2010 in conjunction with JFLA 2010, February 2010. URL: http://coq.inria.fr/files/adt-2fev10-barras.pdf.

[22] Bruno Barras and Bruno Bernardo. The implicit calculus of constructions as a programming language with dependent types. In Roberto M. Amadio, editor, *FoSSaCS*, volume 4962 of *Lect. Notes in Comput. Sci.*, pages 365–379. Springer, 2008. doi:10.1007/978-3-540-78499-9_26.

[23] Gilles Barthe, Maria J. Frade, Eduardo Giménez, Luis Pinto, and Tarmo Uustalu. Type-based termination of recursive definitions. *Math. Struct. in Comput. Sci.*, 14(1):97–141, 2004.

[24] Gilles Barthe, Benjamin Grégoire, and Colin Riba. Type-based termination with sized products. In Michael Kaminski and Simone Martini, editors, *Computer Science Logic, 22nd Int. Wksh., CSL 2008, 17th Annual Conf. of the EACSL*, volume 5213 of *Lect. Notes in Comput. Sci.*, pages 493–507. Springer, 2008. doi:10.1007/978-3-540-87531-4_35.

[25] Frédéric Blanqui. A type-based termination criterion for dependently-typed higher-order rewrite systems. In Vincent van Oostrom, editor, *Rewriting Techniques and Applications (RTA 2004), Aachen, Germany*, volume 3091 of *Lect. Notes in Comput. Sci.*, pages 24–39. Springer, 2004. doi:10.1007/978-3-540-25979-4_2.

[26] Ana Bove and Venanzio Capretta. Modelling general recursion in type theory. *Math. Struct. in Comput. Sci.*, 15(4):671–708, 2005.

[27] Ana Bove, Alexander Krauss, and Matthieu Sozeau. Partiality and recursion in interactive theorem provers: An overview. *Math. Struct. in Comput. Sci.*, 2013. To appear.

[28] Catarina Coquand. A realizability interpretation of Martin-Löf's type theory. In G. Sambin and J. Smith, editors, *Twenty-Five Years of Constructive Type Theory*. Oxford University Press, 1998.

[29] Thierry Coquand and Jean Gallier. A proof of strong normalization for the theory of constructions using a kripke-like interpretation. In Gerard Huet and Gordon Plotkin, editors, *Proceedings of the First Workshop on Logical Frameworks*, May 1990.

[30] Maxime Denes. Propositional extensionality is inconsistent in Coq. Message on the Coq Club mailing list, December 2012. URL: https://sympa.inria.fr/sympa/arc/coq-club/2013-12/msg00119.html.

[31] Eduardo Giménez. Structural recursive definitions in type theory. In K. G. Larsen, S. Skyum, and G. Winskel, editors, *Int. Colloquium on Automata, Languages and Programming (ICALP'98), Aalborg, Denmark*, volume 1443 of *Lect. Notes in Comput. Sci.*, pages 397–408. Springer, 1998. doi:10.1007/BFb0055070.

[32] Jean-Yves Girard. *Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur*. Thèse de Doctorat d'État, Université de Paris VII, 1972.

[33] Healfdene Goguen. *A Typed Operational Semantics for Type Theory*. PhD thesis, University of Edinburgh, August 1994. Available as LFCS Report ECS-LFCS-94-304.

[34] Georges Gonthier. Formal proof – the four colour theorem. *Notices of the AMS*, 55(11):1382–1392, 2008.

[35] Martin Hofmann. *Extensional concepts in intensional type theory*. PhD thesis, University of Edinburgh, July 1995. Available as LFCS Report ECS-LFCS-95-327.

[36] Martin Hofmann and Thomas Streicher. The groupoid model refutes uniqueness of identity proofs. In *Proc. of the 9th IEEE Symp. on Logic in Computer Science (LICS'94)*, pages 208–212. IEEE Computer Soc. Press, 1994.

[37] John Hughes, Lars Pareto, and Amr Sabry. Proving the correctness of reactive systems using sized types. In *Proc. of the 23rd ACM Symp. on Principles of Programming Languages, POPL'96*, pages 410–423, 1996. doi:10.1145/237721.240882.

[38] The Verified Software Initiative. 2012 Microsoft Research Verified Software Milestone Award Citation, 2012. URL: https://sites.google.com/site/verifiedsoftwareinitiative/mrs-award/2012-award.

[39] INRIA. *The Coq Proof Assistant Reference Manual*. INRIA, version 8.4 edition, 2012. URL: http://coq.inria.fr/.

[40] INRIA. Calculus of Inductive Constructions, April 2014. The Coq Proof Assistant Reference Manual. URL: http://coq.inria.fr/doc/Reference-Manual006.html.

[41] Rob Knies. Gonthier earns EADS foundation honor, November 2011. URL: http://research.microsoft.com/en-us/news/features/gonthier-112211.aspx.

[42] Rob Knies. Six-year journey leads to proof of Feit-Thompson Theorem, October 2012. URL: http://phys.org/news/2012-10-six-year-journey-proof-feit-thompson-theorem.html.

[43] Xavier Leroy. Formal verification of a realistic compiler. *Commun. ACM*, 52(7):107–115, 2009.

[44] Zhaohui Luo. *ECC: An Extended Calculus of Constructions*. PhD thesis, University of Edinburgh, 1990. URL: http://www.dur.ac.uk/~{}dcs0zl/publications.html.

[45] Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

[46] Nax Paul Mendler. Recursive types and type constraints in second-order lambda calculus. In *Proc. of the 2nd IEEE Symp. on Logic in Computer Science (LICS'87)*, pages 30–36. IEEE Computer Soc. Press, 1987.

[47] Alexandre Miquel. *Le Calcul des Constructions implicite: syntaxe et sémantique.* PhD thesis, Université Paris 7, December 2001.

[48] Alexandre Miquel and Benjamin Werner. The not so simple proof-irrelevant model of CC. In Herman Geuvers and Freek Wiedijk, editors, *Types for Proofs and Programs, Int. Wksh., TYPES 2002*, volume 2646 of *Lect. Notes in Comput. Sci.*, pages 240–258. Springer, 2003.

[49] Ulf Norell. *Towards a Practical Programming Language Based on Dependent Type Theory*. PhD thesis, Dept of Comput. Sci. and Engrg., Chalmers, Göteborg, Sweden, September 2007.

[50] Jorge Luis Sacchini. *On Type-Based Termination and Pattern Matching in the Calculus of Inductive Constructions*. PhD thesis, INRIA Sophia-Antipolis and École des Mines de Paris, 2011.

[51] Jorge Luis Sacchini. Type-based productivity of stream definitions in the calculus of constructions. In *Logics in Computer Science (LICS 2013), June 25-28, 2013, New Orleans*, 2013.

[52] Daniel Schepler. bijective function implies equal types is provably inconsistent with functional extensionality in Coq. Message on the Coq Club mailing list, December 2013. URL: `https://sympa.inria.fr/sympa/arc/coq-club/2013-12/msg00114.html`.

[53] Thomas Streicher. *Semantics of Type Theory*. Progress in Theoretical Computer Science. Birkhäuser Verlag, Basel, 1991.

[54] William W. Tait. Intensional interpretations of functionals of finite type I. *J. Symb. Logic*, 32(2):198–212, 1967.

[55] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `http://homotopytypetheory.org/book/`, Institute for Advanced Study, 2013.

[56] Vladimir Voevodsky. Univalent foundations, March 2014. Talk given at the Institute of Advanced Studies, Princeton, New Jersey. Video available at `http://video.ias.edu/node/6395`. URL: `http://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/2014_IAS.pdf`.

[57] Benjamin Werner. *Une Théorie des Constructiones Inductives*. PhD thesis, Universite Paris 7, 1994.

[58] Hongwei Xi. Dependent types for program termination verification. *J. Higher-Order and Symb. Comput.*, 15(1):91–131, October 2002. `doi:10.1023/A:1019916231463`.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-48243-115404-23

Name of applicant
Abel, Andreas

Date of birth
740624-6491

Title of research programme
Termination Certificates for Dependently-Typed Programs and Proofs via Refinement Types

Appendix B

Curriculum vitae

Appendix B: Andreas Abel's CV

1 Higher education qualifications

1999 Informatik Diplom, Ludwig-Maximilians-Universität München. (Equivalent to Master in computer science.)

2 Doctoral degree

2006 Dr. rer. nat. *(Doctor rerum naturarum)*, Informatik (computer science), Ludwig-Maximilians-Universität München. Title: *A Polymorphic Lambda-Calculus with Sized Higher-Order Types.* Supervisor: Prof. Martin Hofmann, PhD.

3 Postdoctoral positions

2004-2005 Department of Computer Science and Engineering, Chalmers University of Technology.

4 Qualification required for appointment as a docent

2013 *Habilitation zum Privatdozent*, Ludwig-Maximilians-Universität München. Title of habilitation thesis: *Normalization by Evaluation: Dependent Types and Impredicativity*.

5 Present position

2013– Senior lecturer at the Department of Computer Science and Engineering, Gothenburg University. Permanent position. 50% research.

6 Previous positions

2005-2008 Wissenschaftlicher Mitarbeiter (assistant professor), Department of Computer Science, Ludwig-Maximilians-Universität München.

2008-2013 Akademischer Rat auf Zeit (assistant professor), Department of Computer Science, Ludwig-Maximilians-Universität München.

2009-2010 Invited researcher, INRIA Rocquencourt, Paris.

8 Supervision

PhD level Co-supervision of Dulma Rodriguez (graduated 2012), Christoph-Simon Senjak, and Andrea Vezzosi.

Master level Supervised 5 individual projects, 3 of them lead to presentations at international conferences (ICFP 2012, CSL 2008) or publications in international journals (LMCS).

Bachelor level Supervised 7 individual projects, 3 of them lead to presentations at international conferences or workshops (POPL 2013, LFMTP 2011, PTP 2001).

9 Other information of importance to the application

Professional service

Conference PC memberships FoSSaCS 2010, MFCS 2011, ESOP 2012, RTA 2012, ITP 2012, FoSSaCS 2013, TLCA 2013, Haskell Symposium 2013, RTA-TLCA 2014, CPP 2015, FoSSaCS 2015, ICALP 2015 (Track B).

Co-editorship TYPES 2011 (post-proceedings).

Workshop organization and PC chairing LFMTP 2008, PLPV 2013.

Informal workshop organization Agda Implementor's Meeting 2012 (Fischbachau, Germany) and 2014 (Paris).

Workshop PC memberships LFMTP 2007, PLPV 2009, PAR 2010, MSFP 2010, HOR 2012, FICS 2013, WGP 2013, CMCS 2014, MSFP 2014, TYPES 2014, DTP 2014, LFMTP 2014.

Additional refereeing More than 100 reviews; for the conferences APLAS, CADE, CiE, CSL, ESOP, FoSSaCS, FLOPS, FSTTCS, ICALP, ICFP, LiCS, LPAR, MPC, POPL, PPDP, RTA, STACS, and TLCA, for the post-workshop proceedings of CL&C, TFP, and TYPES, and for the journals Fundamenta Informaticæ, HOSC, IGPL, IPL, JAR, JFP, LMCS, MSCS, and TCS.

PhD committee memberships Johan Granström, *Reference and Computation in Intuitionistic Type Theory* (Uppsala, 2009); Cody Roux, *Type Based Termination: Semantics and Generalisations* (LORIA, Nancy, 2011); and Jorge Sacchini, *On Type-Based Termination and Dependent Pattern Matching in the Calculus of Inductive Constructions* (École des Mines de Paris, 2011).

Awards, fellowships, and grants

2012 Invited Presentation at FICS Workshop, Tallinn, Estonia

2009 Invited Researcher (6 months) of INRIA, France

2008 Travel funds for research cooperation with LORIA, Nancy, on *Type-Based Termination*, provided by the *Bayerisch-französisches Hochschulzentrum* (EUR 6350)

2005–2007 SRC Project Research Grant *Typed Lambda Calculi and Applications* (600 kSEK), co-applicant with Prof. Thierry Coquand and Prof. Peter Dybjer

2004–2005 Postdoctoral Fellowship from the Swedish Foundation of Strategic Research (SFF) in the CoVer Project, Chalmers

2000–2001 Fellowship from the Office of Technology in Education, Carnegie-Mellon University, Pittsburgh

1999–2003 Stipend from PhD Program Logic in Computer Science, *Deutsche Forschungsgemeinschaft (DFG)* (German Research Foundation)

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod	2014-48243-115404-23
Name of applicant	Abel, Andreas
Date of birth	740624-6491

Title of research programme

Termination Certificates for Dependently-Typed Programs and Proofs via Refinement Types

Appendix C

List of publications

Appendix C: Publications

Andreas Abel

This list contains publications from the last eight years (2006-2014); plus the five most cited works (actually six because of a draw):

12. *Termination checking with types* —Number of citations: 68
40. My PhD thesis *Type-based termination* —Number of citations: 51
11. *Iteration schemes for higher-order and nested datatypes* —Number of citations: 51
13. *A predicative analysis of structural recursion* —Number of citations: 46
32. *Normalization by evaluation for Martin-Löf Type Theory with typed equality judgements* —Number of citations: 36
38. *Human-readable machine-verifiable proofs for teaching constructive logic* —Number of citations: 36

Citation information taken from Google Scholar (`http://scholar.google.se/citations?hl=en&user=diMIMosAAAA`), which lists my h-index as 16 and my i10-index as 26.

1 Peer-reviewed original articles

The journal articles are listed in reverse chronological order. I published in *Fundamenta Informaticæ* (10), *Journal of Functional Programming* (5,13), *Logical Methods in Computer Science* (1,2,8), *Mathematical Structures in Computer Science* (6), *Science of Computer Programming* (4), *Theoretical Computer Science* (11), *Theoretical Informatics and Applications* (12), and *Electronic Notes in Theoretical Computer Science* (3,7,9).

1. (*) Andreas Abel and Gabriel Scherer. On irrelevance and algorithmic equality in predicative type theory. *Logical Methods in Computer Science*, 8(1:29):1–36, 2012. TYPES'10 special issue —*Number of citations: 6*

2. Andreas Abel, Thierry Coquand, and Miguel Pagano. A modular type-checking algorithm for type theory with singleton types and proof irrelevance. *Logical Methods in Computer Science*, 7(2:4):1–57, May 2011 —*Number of citations: 0*

3. Andreas Abel and Thorsten Altenkirch. A partial type checking algorithm for Type:Type. *Electronic Notes in Theoretical Computer Science*, 229(5):3–17, 2011. Proceedings of the Second Workshop on Mathematically Structured Functional Programming (MSFP 2008) —*Number of citations: 6*

4. Andreas Abel. Type-based termination of generic programs. *Science of Computer Programming*, 74(8):550–567, 2009. MPC'06 special issue —*Number of citations: 11*

5. Andreas Abel. Implementing a normalizer using sized heterogeneous types. *Journal of Functional Programming*, 19(3–4):287–310, 2009 —*Number of citations: 17*

6. Andreas Abel. Polarized subtyping for sized types. *Mathematical Structures in Computer Science*, 18:797–822, 2008. Special issue on subtyping, edited by Healfdene Goguen and Adriana Compagnoni —*Number of citations: 6*

7. Andreas Abel. Normalization for the simply-typed lambda-calculus in Twelf. In Carsten Schürmann, editor, *Logical Frameworks and Metalanguages (LFM 04)*, volume 199C of *Electronic Notes in Theoretical Computer Science*, pages 3–16. Elsevier, 2008 —*Number of citations: 6*

8. (*) Andreas Abel. Semi-continuous sized types and termination. *Logical Methods in Computer Science*, 4(2:3):1–33, 2008. CSL'06 special issue —*Number of citations: 24 (together with conference version)*

9. Andreas Abel, Klaus Aehlig, and Peter Dybjer. Normalization by evaluation for Martin-Löf type theory with one universe. In Marcelo Fiore, editor, *Proceedings of the 23rd Conference on the Mathematical Foundations of Programming Semantics (MFPS XXIII), New Orleans, LA, USA, 11-14 April 2007*, volume 173 of *Electronic Notes in Theoretical Computer Science*, pages 17–39. Elsevier, 2007 —*Number of citations: 22*

10. Andreas Abel and Thierry Coquand. Untyped algorithmic equality for Martin-Löf's logical framework with surjective pairs. *Fundamenta Informaticae*, 77(4):345–395, 2007. TLCA'05 special issue —*Number of citations: 7*

11. Andreas Abel, Ralph Matthes, and Tarmo Uustalu. Iteration schemes for higher-order and nested datatypes. *Theoretical Computer Science*, 333(1–2):3–66, 2005 —*Number of citations: **51***

12. Andreas Abel. Termination checking with types. *RAIRO – Theoretical Informatics and Applications*, 38(4):277–319, 2004. Special Issue: Fixed Points in Computer Science (FICS'03) —*Number of citations: **68***

13. Andreas Abel and Thorsten Altenkirch. A predicative analysis of structural recursion. *Journal of Functional Programming*, 12(1):1–41, January 2002 —*Number of citations: **46***

2 Peer-reviewed conference and workshop contributions

From 2006 to 2014, I contributed to the international conferences POPL (16), ICFP (15), LiCS (32), FoSSaCS (19), CSL (24,27,35), TLCA (18,26,34), LPAR (29), APLAS (31), FLOPS (28), MPC (30), and CSR (37), and the international workshops MSFP (14), LFMTP (20,22), PAR (23), and HOR (33).

14. Andreas Abel and James Chapman. Normalization by evaluation in the delay monad: A case study for coinduction via copatterns and sized types. In *International Workshop on Mathematically Structured Functional Programming, colocated with ETAPS 2014, 5-13 April 2014, Grenoble, France*, April 2014 —*Number of citations: 0*

15. (*) Andreas Abel and Brigitte Pientka. Wellfounded recursion with copatterns: A unified approach to termination and productivity. In Greg Morrisett and Tarmo Uustalu, editors, *Proceedings of the Eighteenth ACM SIGPLAN International Conference on Functional Programming, ICFP'13, Boston, MA, USA, September 25-27, 2013*, pages 185–196. ACM Press, 2013 —*Number of citations: 12*

16. Andreas Abel, Brigitte Pientka, David Thibodeau, and Anton Setzer. Copatterns: Programming infinite structures by observations. In Roberto Giacobazzi and Radhia Cousot, editors, *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL'13, Rome, Italy, January 23 - 25, 2013*, pages 27–38. ACM Press, 2013 —*Number of citations: 16*

17. (*) Andreas Abel. Type-based termination, inflationary fixed-points, and mixed inductive-coinductive types. *Electronic Proceedings in Theoretical Computer Science*, 77:1–11, 2012. Proceedings of FICS 2012 —*Number of citations: 7*

18. Andreas Abel and Brigitte Pientka. Higher-order dynamic pattern unification for dependent types and records. In *Typed Lambda-Calculi and Applications 2011*, 2011 —*Number of citations: 9*

19. Andreas Abel. Irrelevance in type theory with a heterogeneous equality judgement. In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures, 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26 - April 3, 2011. Proceedings*, volume 6604 of *Lecture Notes in Computer Science*, pages 57–71. Springer-Verlag, 2011 —*Number of citations: 9*

20. Andreas Abel and Nicolai Kraus. A lambda term representation inspired by linear ordered logic. In Herman Geuvers and Gopalan Nadathur, editors, *6th International Workshop on Logical Frameworks and Meta-languages: Theory and Practice (LFMTP 2011), Nijmegen, The Netherlands, August 26, 2011*, volume 71 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–13, 2011 —*Number of citations: 0*

21. Andreas Abel. Towards Normalization by Evaluation for the $\beta\eta$-Calculus of Constructions. In Matthias Blume, Naoki Kobayashi, and Germán Vidal, editors, *Functional and Logic Programming, 10th International Symposium, FLOPS 2010, Sendai, Japan, April 19-21, 2010. Proceedings*, volume 6009 of *Lecture Notes in Computer Science*, pages 224–239. Springer-Verlag, 2010 —*Number of citations: 9*

22. Andreas Abel and Brigitte Pientka. Explicit substitutions for contextual type theory. In Karl Crary and Marino Miculan, editors, *5th International Workshop on Logical Frameworks and Meta-languages: Theory and Practice (LFMTP 2010), Edinburgh, Scotland, UK, July 14, 2010*, volume 34 of *Electronic Proceedings in Theoretical Computer Science*, pages 5–20, 2010 —*Number of citations: 3*

23. (*) Andreas Abel. MiniAgda: Integrating sized and dependent types. In Ana Bove, Ekaterina Komendantskaya, and Milad Niqui, editors, *Workshop on Partiality And Recursion in Interactive Theorem Provers (PAR 2010), Satellite Workshop of ITP'10 at FLoC 2010*, volume 43 of *Electronic Proceedings in Theoretical Computer Science*, pages 14–28, 2010 —*Number of citations: 29*

24. Andreas Abel. Typed applicative structures and normalization by evaluation for System Fω. In Erich Grädel and Reinhard Kahle, editors, *Computer Science Logic, 23rd international Workshop, CSL 2009, 18th Annual Conference of the EACSL, Coimbra, Portugal, September 7-11, 2009. Proceedings*, volume 5771 of *Lecture Notes in Computer Science*, pages 40–54. Springer-Verlag, 2009 —*Number of citations: 9*

25. Andreas Abel. Extensional normalization in the logical framework with proof irrelevant equality. In Olivier Danvy, editor, *Workshop on Normalization by Evaluation, affiliated to LiCS 2009, Los Angeles, 15 August 2009*, 2009 —*Number of citations: 3*

26. Andreas Abel, Thierry Coquand, and Miguel Pagano. A modular type-checking algorithm for type theory with singleton types and proof irrelevance. In P.-L. Curien, editor, *Typed Lambda Calculi and Applications, 9th International Conference, TLCA 2009, Brasilia, Brazil, July 1-3, 2009, Proceedings*, volume 5608 of *Lecture Notes in Computer Science*, pages 5–19. Springer-Verlag, 2009 —*Number of citations: 12*

27. Andreas Abel and Dulma Rodriguez. Syntactic metatheory of higher-order subtyping. In Michael Kaminski and Simone Martini, editors, *Computer Science Logic, 22nd International Workshop, CSL 2008, 17th Annual Conference of the EACSL, Bertinoro, Italy, September 16-19, 2008. Proceedings*, volume 5213 of *Lecture Notes in Computer Science*, pages 446–460. Springer-Verlag, 2008 —*Number of citations: 6*

28. Andreas Abel, Thierry Coquand, and Peter Dybjer. On the algebraic foundation of proof assistants for intuitionistic type theory. In Jacques Garrigue and Manuel V. Hermenegildo, editors, *Functional and Logic Programming, 9th International Symposium, FLOPS 2008, Ise, Japan, April 14-16, 2008. Proceedings*, volume 4989 of *Lecture Notes in Computer Science*, pages 3–13. Springer-Verlag, 2008 —*Number of citations: 5*

29. Andreas Abel. Weak $\beta\eta$-normalization and normalization by evaluation for System F. In Illiano Cervesato, Helmut Veith, and Andrei Voronkov, editors, *15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2008, 22-27 November 2008, Doha, Qatar, Proceedings*, volume 5330 of *Lecture Notes in Artificial Intelligence*, pages 497–511. Springer-Verlag, 2008 —*Number of citations: 8*

30. Andreas Abel, Thierry Coquand, and Peter Dybjer. Verifying a semantic $\beta\eta$-conversion test for Martin-Löf type theory. In Philippe Audebaud and Christine Paulin-Mohring, editors, *Mathematics of Program Construction, 9th International Conference, MPC 2008, Marseille, France, July 15-18, 2008. Proceedings*, volume 5133 of *Lecture Notes in Computer Science*, pages 29–56. Springer-Verlag, 2008 —*Number of citations: 25*

31. Andreas Abel. Mixed inductive/coinductive types and strong normalization. In Zhong Shao, editor, *Programming Languages and Systems, 5th Asian Symposium, APLAS 2007, Singapore, November 29-December 1, 2007, Proceedings*, volume 4807 of *Lecture Notes in Computer Science*, pages 286–301. Springer-Verlag, 2007 —*Number of citations: 13*

32. Andreas Abel, Thierry Coquand, and Peter Dybjer. Normalization by evaluation for Martin-Löf Type Theory with typed equality judgements. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings*, pages 3–12. IEEE Computer Society Press, 2007 —*Number of citations:* ***36***

33. Andreas Abel. Syntactical normalization for intersection types with term rewriting rules. In *Fourth International Workshop on Higher-Order Rewriting, HOR'07, Paris, France, 25 June 2007*, 2007 —*Number of citations: 0*

34. Andreas Abel. Strong normalization and equi-(co)inductive types. In Simona Ronchi Della Rocca, editor, *Typed Lambda Calculi and Applications, 8th International Conference, TLCA 2007, Paris, France, June 26-28, 2007, Proceedings*, volume 4583 of *Lecture Notes in Computer Science*, pages 8–22. Springer-Verlag, 2007 —*Number of citations: 4*

35. Andreas Abel. Semi-continuous sized types and termination. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 21-24, 2006, Proceedings*, volume 4207 of *Lecture Notes in Computer Science*, pages 72–88. Springer-Verlag, 2006 —*Number of citations: 24 (together with journal version)*

36. Andreas Abel. Towards generic programming with sized types. In Tarmo Uustalu, editor, *Mathematics of Program Construction: 8th International Conference, MPC 2006, Kuressaare, Estonia, July 3-5, 2006. Proceedings*, volume 4014 of *Lecture Notes in Computer Science*, pages 10–28. Springer-Verlag, 2006 —*Number of citations: 4*

37. Andreas Abel. Polarized subtyping for sized types. In Dima Grigoriev, John Harrison, and Edward A. Hirsch, editors, *Computer Science - Theory and Applications, First International Computer Science Symposium in Russia, CSR 2006, St. Petersburg, Russia, June 8-12, 2006, Proceedings*, volume 3967 of *Lecture Notes in Computer Science*, pages 381–392. Springer-Verlag, 2006 —*Number of citations: 7*

38. Andreas Abel, Bor-Yuh Evan Chang, and Frank Pfenning. Human-readable machine-verifiable proofs for teaching constructive logic. Technical report, Università degli Studi Siena, Dipartimento di Ingegneria dell'Informazione, 2001. In Proceedings of the Workshop on Proof Transformation and Presentation and Proof Complexities (PTP'01) —*Number of citations: **36***

4 Books and book chapters

39. Andreas Abel. *Normalization by Evaluation: Dependent Types and Impredicativity*. 2013. Habilitation thesis, Department of Computer Science, Ludwig-Maximilians-University Munich —*Number of citations: 0*

40. Andreas Abel. *Type-Based Termination*. Harland media, February 2007. Dissertation, Department of Computer Science, Ludwig-Maximilians-University Munich, 2006, titled *A Polymorphic Lambda Calculus with Sized Higher-Order Types —Number of citations: **51***

6 Open-access computer programs

Agda, 2004– A dependently-typed functional programming language and proof assistant, available from `http://wiki.portal.chalmers.se/agda/pmwiki.php`. I am one of the three main developers and contributed 1200 out of 5500 patches.

MiniAgda, 2007– A prototypical variant of Agda, for testing language extensions, available from `http://www.cse.chalmers.se/~abela/miniagda/`.

Tutch, 2000–2004 A tool for teaching logic `http://www.cse.chalmers.se/~abela/tutch/`. Used in several courses, `http://www.cs.cmu.edu/~fp/courses/15317-f09/software.html`

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod	2014-48243-115404-23
Name of applicant	Abel, Andreas
Date of birth	740624-6491
Title of research programme	

Appendix N

Budget and research resources

Appendix N: Budget and Research Resources

Andreas Abel

Justification of the Budget

Own salary Since I am a new member of the Department of Computer Science and Engineering (CSE) of Gothenburg University, my research time (50%) is currently financed by faculty funds of CSE, but will be sourced from research grants from 2015. I plan to devote 40% of my time on this project, which includes supervision of a PhD student, and thus apply for 40% of my salary for 2015-2019.

PhD student PhD students at CSE do 20% teaching, which is funded by the department, and 80% research, which is funded by research grants. I intend to employ a PhD student to work on the theoretical and practical aspects of this project, and thus apply for 80% of a PhD students salary.

Computer equipment and literature I apply for a new computer equipment for the PhD student (2015 and 2019) and myself (2017), plus a base funding for research literature not available at our library.

Travel and conference In computer science, dissemination and communication of results usually happens at international conferences. I apply for travel expenses and fees for four conferences per year for myself, and of three for the PhD student.

Publication costs These are the standard CSE publication costs for a licentiate thesis (2017) and a PhD thesis (2019) for the PhD student.

Total Research Resources of the Project

Since I am not holder or beneficiary of any grant yet, I apply for 100% funding of the total annual costs of this project from the SRC.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod	2014-48243-115404-23
Dnr	2014 -
Name of applicant	Abel, Andreas
Date of birth	740624-6491
Reg date	2014-04-09 22:16:55

Project title

Termination Certificates for Dependently-Typed Programs and Proofs via Refinement Types

NT

NT-2

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

Appendix S

Signatures

2015	2016	2017	2018	2019
1252	1255	1381	1385	1561

2015-01-01 -- 2019-12-31

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

2014-48243-115404-23