Programming and Reasoning in Higher Dimensional Type Theory (Case for Support)

Thorsten Altenkirch, Eugenia Chang, Neil Ghani and Ondrej Rypacek

Summary

Type Theory is the one of the most promising approaches to formally certified software and mathematics, being the base of tools like the proof assistant Coq `coq` and the dependently typed functional programming language Agda `Agda`. In the research proposed here we are going to investigate and develop further a novel extension of the type-theoretic approach based on a new connection between geometry (Homotopy theory) and reasoning proposed by the Field medallist Vladimir Voevodsky. In a nutshell this *higher dimensional type theory* allows us to treat structures as first class citizens by viewing equivalence of structures as equality. We anticipate that this innovation will have substantial impact on the feasibility of large scale formal development by supporting the replacability of components without affecting the rest of the development. Our work will draw on and create new connections between fundamental mathematical theories like Homotopy theory and higher dimensional category theory on the one side and formally supported software engineering on the other.

Part 1: Track Record

Thorsten Altenkirch

Thorsten Altenkirch received his PhD from the University of Edinburgh in 1993. He has been a research assistant at the University of Edinburgh and at Chalmers University in Gothenburg, Sweden and has held lecturing positions at the University of Munich, Germany, and the University of Nottingham. Since October 2006 he has been a Reader in Computer Science at the University of Nottingham and in October 2008 he founded the Functional Programming Laboratory together with Graham Hutton. Altenkirch and Hutton are now jointly leading the laboratory.

Altenkirch's research interests are in Type Theory, Category Theory, Functional Programming and Quantum Programming. Among his main achievements is the first formalised normalisation proof for System F `alti:tlca93` , his work on normalisation by evaluation `alti:lics01`, extensional equality within intensional Type Theory `alti:lics99` , the development of container theory `alti:cont-tcs`, on monads `alti:fossacs10` and on generic programming `alti:wcgp02` . He has been on the program committee of a number of

conferences and workshops, recently CiE 2008, TPHOLs 2009 and ITP 2010. He has organised TYPES meetings in 1998 and 2006 and has coedited the proceedings; and he has also been the program co-chair of PLPV 2009. He organised a Dagstuhl seminar on dependently typed programming in 2004 and has since organised two further workshops on this topic (DTP 2008 and DTP 2010). He gave an invited lecture series on Type Theory at the University of Tallinn in 2003 and he was a visiting Professor at the Universitè Denis Diderot, Paris in July 2007. He gave invited talks at the first workshop on quantum programming languages (QPL 2003), the workshop on normalisation by evaluation (NBE 2009) and at the special session on proof theory of CiE 2010.

He has been the Principal Investigator on two successfully completed EPSRC grants : *Modelling Irreversible Quantum Computation* (GR/S30818/01), *Observational Equality For Dependently Typed Programming* (EP/C512022/1); and has been co-investigator on *Theory and Applications Of Containers* (EP/C511964/1). He is currently the principal investigator of two ESPRC projects: *Reusability and Dependent Types* (EP/G034109/1), networked with Oxford and Strathclyde, Nottingham is the lead site and *Theory And Applications of Induction Recursion* (EP/G03298X/1), networked with Strathclyde and Swansea. On the European level he has been active in the the TYPES and APPSEM coordination actions and has been involved in the QICS STREP. He also hosted a Marie-Curie fellowship (2006 - 2008). He is regularly teaching courses at the Midland Graduate School and has been the co-investigator of the associated EPSRC grant: *Midlands Graduate School in the Foundations of Computer Science* (GR/T06087/01). He has supervised 5 successfully completed PhD projects at Nottingham, two of them in the area of quantum programming `jjgthesis,asg:thesis`. He has also lectured on the subject of quantum computing at third year undergraduate level, masters level, and to research students (at the Midlands Graduate School).

Eugenia Chang

Neil Ghani

Ondrej Rypacek

Nottingham - host organisation

The School of Computer Science at the University of Nottingham is a research-led School in one of the leading Universities in the UK. The School was ranked 8th in the last Research Assessment Exercise, and the Functional Programming Lab within the School is one of four major research groups, with an international reputation for its work on formally-based approaches to software construction and verification. The FP lab currently comprises 4 academic staff (Thorsten Altenkirch, Venanzio Capretta, Graham Hutton, and Henrik Nilsson), 2 post-doctoral fellows, and 9 PhD students. To date the group has received £1.5M of EPSRC funding over 14 projects, and has 12 completed PhD students.

The Functional Programming Lab provides a highly stimulating research environment for researchers and PhD students with weekly research meetings and frequent seminars.

Part 2: Proposed Research

Recently, the Field medallist Vladimir Voevodsky of the Institute of Advanced Study in Princeton became one of the latest proponents of formally developed, computer checked Mathematics. Voevodsky is now using the interactive proof system Coq to support his work and encourages his colleagues to follow suit. However, while impressed with the potential of systems like Coq based on Type Theory Voevodsky also proposed an important extension of the type theoretic approach based on his background in Homotopy theory: Univalent Type Theory. `voevodsky,awodey`. Univalent Type Theory enables us to view mathematical structures as first class citizens and identify equivalent structures as if they were equal. While this is clearly important for the development of Mathematics it also is essential for the development of a large corpus of reusable and certified software allowing us to replace one abstract module by another equivalent one without having to repeat the effort of certification. While this is a long standing issue in the use of abstract datatypes the novelty of Univalent Type Theory lies in the possibility to view equivalent structures as equal which is not supported by any existing approach.

1 Background

Type Theory ala Martin Löf is at the same time a programming language and a logical systems based on the propositions as as types principle. The basic notions are Π-types generalizing the notion of a function type from functional programming to a situation where the domain type can *depend* on the actual input and on the logical side covering the intuitionistic explanation of the notions of implication and universal quantification. On the other hand Σ-types generalize the notion of a product type (or record) in functional programming and on the logical side allow us to model conjunction and existential quantification. A 3rd central component are equality types which assign to any two values the type of proofs that these values are equal. Unlike in conventional logic, Type Theory allows us to talk about properties of proofs, e.g. we can ask the question whether two propositions (i.e. types) aren't only logically equivalent but whether they are actually isomorphic (i.e. computationally equivalent). Other components of Type Theory are inductive and coinductive types which allow us to construct trees with finite or potentially infinite depth and notion of a universe, such as the universe of small sets corresponding to inaccessible cardinals in set theory.

Many interesting questions in relation to Type Theory center around the notion of equality. E.g. can we prove the principle of functional extensionality, i.e. that two functions are equal if they are pointwise equal? Maybe surprisingly, this principle is not provable in Intensional Type Theory which is the basis of most implementations of Type Theory (e.g. Agda, Coq). However, this shortcoming can be addressed using Observational Type Theory which is one of the main outcomes of EPSRC project XXX based on earlier work by Altenkirch `alti:lics99` . Another question related to equality is the question whether any two proofs of equality are themselves equal (uniqueness of equality proofs)? It was shown by Hofmann and Streicher that this is not provable in standard Type Theory using a groupoid interpretation of Type Theory `groupoid-model`. Later

Lumsdaine and independently Garner and de Berg `lumsdaine,garner-deberg` showed that equality in Type Theory give rise to a weak ω-groupoid, a structure well known in higher-dimensional category.

Voevodsky and Awodey developed an interpretation of Type Theory using homotopy theory `voevodsky,awodey`. In this interpretation types are viewed as (special) topological spaces, elements as points and equality proofs as paths or homotopies between elements. The homotopy interpretation gives a very intuitive geometric explanation for the unprovability of uniqueness of equality proofs. It also lead Voevodsky to postulate the univalence axiom, which states that weakly equivalent types should be equal. In particular isomorphic sets such as natural numbers and lists of natural numbers are equated as a consequence of the univalence axiom. Clearly, the univalence axiom is incompatible with uniqueness of equality proofs since in general there is more than one way to show that two isomorphic sets are equal. The univalence axiom implies the principle of extensionality — indeed it can be viewed as a strong extensionality principle which identifies indistinguishable types.

Type Theory has an increasing influence on the development of certified software and mathematical theories in particular through the Coq system. While Coq in practice maintains a separation of logic and proof, newer developments like the Agda system introduce Type Theory as a total functional programming language with a particular expressive type system and thus makes Type Theory accessible to interested programmers. The availability of extensionality principles (such as functional extensionality and univalence) is here of practical importance because it is essential for an structured development of a complex deliverables allowing us to replace one module by another, behaviourially equivalent one. Unlike in conventional logic where it is enough just to postulate an axiom this is not sufficient in Type Theory because this may stop computation. Hence extensionality principles come with a canonicity problem which need to be addressed before these principles can be used in practice.

1.1 Higher dimensional category theory

1.2 Homotopy Type Theory

2 Programme and Methodology

In the research proposed here we are going to explore Univalent Type Theory and in particular investigate its potential impact on Computer Science in particular the efficient development of certified software. Identifying equivalence of structures with equality raises well known coherence issues which have been investigated in the context of higher dimensional category theory `eugenia` which forms one of the foundations of our research (WP1). We conjecture that this background is useful when developing Univalent Type Theory addressing thorny issues such as the canonicity problem `harper,voevodsky,coquand` and the formulation of higher dimensional quotients (WP2). The Agda system `agda` which is at the same time a programming language and a interactive proof system is ideally suited to make these concepts available to interested researchers. We will use the Agda system both as a tool to develop the theory (WP3) but also as a target to turn theory into practice by developing software tools based on Agda to support the use of Univalent Type Theory for certification (WP4). Finally

we plan to conduct a number of case studies evaluating the potential impact of Univalent Type Theory in Computer Science (WP5) and Computer Aided Mathematics (WP6).

WP1 : Higher dimensional category theory

Research challenges

- Identify a workable, constructive definition of a weak ω-groupoid
- Investigate models of Type Theory (CWFs, locally cartesian closed categories) in a higher dimensional setting,
- Relate this to existing approaches to ω-categories and groupoids based on contractible operads
- Relate the notion the simplicial approach to ω-groupoid.

WP2 : Foundations of Higher Dimensional Type Theory

Research challenges

- Develop a feasible judgemental presentation of higher order, higher dimensional Type Theory
- Identify a notion of higher dimensional model and show that standard constructions give rise or are closed under this notion,
- Investigate the problem of weak and strong canonicity in the presence of extensionality and univalence.

WP3 : Formalizing higher dimensional structures

Research challenges

- Make the notions of WP1 and WP2 sufficiently precise so that they can be formalized and mechanically checked in Agda.
- Formalize core aspects of higher dimensional category theory in Agda.
- Formalize core aspects of higher dimensional type theory in Agda.
- Identify the requirements on the Metatheory we need to represent our notion (e.g. induction-recursion or induction-induction related to project XXX).

WP4 : Implement higher dimensional type theory

Research challenges

- Develop the implementation of a higher dimensional core language in a functional language (e.g. Haskell).
- Study the potential of modifying the Agda system to support higher dimensional concepts.

WP5 : Applications in Computer Science

Research challenges

WP6 : Applications in Formal Mathematics

Research challenges

3 Relevance to Beneficiaries

4 Dissemination and Exploitation

References

[1] T. Altenkirch. Extensional equality in intensional type theory. In *14th Symposium on Logic in Computer Science*, pages 412 – 420, 1999.