Report from Dagstuhl Seminar 18341

Formalization of Mathematics in Type Theory

Edited by

Andrej Bauer[1], Martín H. Escardó[2], Peter L. Lumsdaine[3], and Assia Mahboubi[4]

1 **University of Ljubljana, SI,** andrej.bauer@fmf.uni-lj.si
2 **University of Birmingham, GB,** m.escardo@cs.bham.ac.uk
3 **University of Stockholm, SE,** p.l.lumsdaine@math.su.se
4 **INRIA – Nantes, FR,** assia.mahboubi@inria.fr

Abstract

Formalized mathematics is mathematical knowledge (definitions, theorems, and proofs) represented in digital form suitable for computer processing. The central goal of this seminar was to identify the theoretical advances and practical improvements needed in the area of formalized mathematics, in order to make it a mature technology, truly useful to a larger community of students and researchers in mathematics. During the seminar, various software systems for formalization were compared, and potential improvements to existing systems were investigated. There have also been discussions on the representation of algebraic structures in formalization systems.

Seminar August 19–24, 2018 – http://www.dagstuhl.de/18341
2012 ACM Subject Classification Theory of computation → Logic and verification, Theory of computation → Type theory
Keywords and phrases formal methods, formalized mathematics, proof assistant, type theory
Digital Object Identifier 10.4230/DagRep.8.8.130
Edited in cooperation with Auke B. Booij

1 Executive Summary

Andrej Bauer
Martín H. Escardó
Peter L. Lumsdaine
Assia Mahboubi

We and all the participants were delighted to benefit from Dagstuhl's inspiring environment.

Proof assistants are receiving increased attention from users with a background in mathematics, as opposed to their traditional users from theoretical computer science/logic/program verification, and this was the major focus of the meeting. This is true in particular of proof assistants based on dependent types, probably due in part to the advent of homotopy type theory, developed in the proof assistants Coq, Agda and Lean.

The audience of the seminar was thus rather unusual in composition, and featured several experienced researchers used to attending seminars at the Mathematisches Forschungsinstitut Oberwolfach, and visiting Schloss Dagstuhl for the first time. In order to foster discussion and fuse collaborations, we adopted a different format from the standard string of slide-based talks: talks in the morning, so that people get to know the work of each other, and working

in groups in the afternoon. At the end of each day, before dinner, each group presented a summary of the outcomes of their meetings to all participants, which allowed inter-group discussion and collaboration. This had been tried before by some of the organizers, in the course of Dagstuhl seminar 16112, and worked just as well in our case.

Working group topics were proposed by the audience on the first day, by giving short presentations of a few minutes and writing topics in the board. Some were quite specialized and homogeneous (e.g. the cubical type theory group), and allowed people to have a focussed collaborative brainstorming on a specific open problem of the field. Some were more open-ended, and allowed people to confront various approaches to the same issue/concept in different systems (different proof assistants, computer algebra systems, etc.).

Some people did applied work, such as trying to compute the so-called Brunerie number from an existing proof in homotopy type theory, in order to identity and fix inefficiency problems in proofs assistants based on cubical type theory. Some people used their spare time to solve the "Dagstuhl dinner" problem. Details of the topics discussed are in the reports produced by each group.

This was a rather productive meeting, and people from different scientific backgrounds not only met but talked together effectively, solving and identifying problems to work on collaboratively in future.

2 Table of Contents

3 Overview of Talks

3.1 Deriving on Steroids – for proof assistants

Jacques Carette (McMaster University – Hamilton, CA)

Joint work of Jacques Carette, Russell O'Connor
Main reference Jacques Carette, Russell O'Connor: "Theory Presentation Combinators", in Proc. of the Intelligent Computer Mathematics – 11th International Conference, AISC 2012, 19th Symposium, Calculemus 2012, 5th International Workshop, DML 2012, 11th International Conference, MKM 2012, Systems and Projects, Held as Part of CICM 2012, Bremen, Germany, July 8-13, 2012. Proceedings, Lecture Notes in Computer Science, Vol. 7362, pp. 202–215, Springer, 2012.
URL http://dx.doi.org/10.1007/978-3-642-31374-5_14

Developing large theory graphs is a lot of work – even without the proofs. It turns out that the contents of mathematics is highly structured, and that structure can be used to alleviate the development of theories. This naturally leads to theory presentation combinators, which transform existing knowledge into new, in a scalable manner.

Experience shows that 3 main combinators arise naturally: extension, renaming, and combination. Extending adds a new concept to a theory; here we use the *tiny theories* approach, which is to always add a single concept at a time. Renaming is necessary as mathematical conventions for things which are "the same" nevertheless use different symbols in different contexts. First-class renaming allows this "sameness" to be tracked automatically. Lastly, combination is a generalization of union which takes care of necessary gluings when we want to merge two theories with a common ancestry. In other words, for theory presentations, the "diamond problem" is actually a blessing.

This approach appears to scale well. It also has a denotational theory that is quite familiar, as it re-uses the category of contexts, fibrations and pullbacks as its main ingredients.

From there, the ideas of "deriving" from Haskell really kick in: it is straightforward to notice that many constructions from Universal Algebra lift immediately to this setting. Thus one can automatically derive new theories, such as that of homomorphisms, from existing theories. Further more, one can also derive term languages and accompanying functions, automatically from the signature of a theory. This can also be staged, so that meta-programming comes into scope, so that simplistic optimizing compilers for terms of a theory's language can be automatically derived.

3.2 Classical Analysis with Coq

Cyril Cohen (INRIA Sophia Antipolis, FR) and Assia Mahboubi (INRIA – Nantes, FR)

Joint work of Reynald Affeldt, Cyrill Cohen, Damien Rouhling, Assia Mahboubi, Pierre-Yves Strub

In this talk I presented an ongoing effort to develop a Coq formal library, MathComp-Analysis [1], about classical real analysis. Almost all existing proof assistants on the market have been used to investigate the formalization of real, and sometimes also complex, analysis. A survey by Boldo et al. reviews the different approaches and the breadth of the existing developments [2].

Our motivation for designing yet another formal analysis library is twofold. First, we rely on strong classical axioms, so as to get closer to the logical formalism used in classical

mathematics. Second, we design it along the formalization methodology put into practice in the MATHEMATICAL COMPONENTS libraries [3]. The latter libraries are essentially geared towards algebra and this work aims at providing an extension for topics in analysis.

The main original contributions lie in the effort put in the infrastructure of MATHCOMP-ANALYSIS: automation, notations, etc... I presented more in details two mechanisms to do asymptotic reasoning: one to simplify proofs about filters and another to deal with Bachmann-Landau notations.

References

1 Reynald Affeldt, Cyril Cohen, Assia Mahboubi, Damien Rouhling, and Pierre-Yves Strub. Analysis library compatible with Mathematical Components. https://github.com/math-comp/analysis/releases/tag/0.1.0 (last accessed: 2018/10/01), 2018.
2 Sylvie Boldo, Catherine Lelay, and Guillaume Melquiond. Formalization of real analysis: a survey of proof assistants and libraries. *Mathematical Structures in Computer Science*, 26(7):1196–1233, 2016.
3 Assia Mahboubi and Enrico Tassi. *Mathematical Components*. Available at: https://math-comp.github.io/mcb/, 2016. With contributions by Yves Bertot and Georges Gonthier.

3.3 Isabelle/HOL Demo

Manuel Eberl (TU München, DE)

I demonstrated the interactive theorem prover Isabelle in the logic HOL by showing how to prove the infinitude of primes in it. I also showed some of its more specialized tactics, like those for approximation of real numbers or real limits, and the code generation feature.

3.4 A Coq Formalization of Digital Filters

Diane Gallois-Wong (Laboratoire de Recherche en Informatique – Orsay, FR)

Joint work of Boldo, Sylvie; Hilaire, Thibault
Main reference Diane Gallois-Wong, Sylvie Boldo, Thibault Hilaire: "A Coq Formalization of Digital Filters", in Proc. of the Intelligent Computer Mathematics – 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings, Lecture Notes in Computer Science, Vol. 11006, pp. 87–103, Springer, 2018.
URL http://dx.doi.org/10.1007/978-3-319-96812-4_8

Digital filters are small iterative algorithms, used as basic bricks in signal processing and control systems. Therefore, they have numerous application domains, including communication, automotive, robotics, aeronautics, etc. They are usually studied as mathematical objects using real numbers. However, to be used in practice, they need to be implemented, which implies finite precision arithmetic (floating- or fixed-point numbers) and rounding errors. Moreover, propagation of these rounding errors through iteration makes these errors potentially critical but also hard to study. That is why we aim at providing a formal analysis of the rounding errors in digital filters, using the Coq proof assistant. In our current formalization, we define three algorithms used to implement digital filters, called realizations.

We prove that they are equivalent, so that we can focus on one of them for the rest of the error analysis. Then, we formally prove two theorems that are essential to the error analysis: the theorem of the error filter, that characterizes the final error between the implemented filter and the ideal one using infinite precision, and the Worst-Case Peak-Gain theorem, that bounds the output corresponding to a bounded input.

3.5 An overview of UniMath

Daniel R. Grayson (Urbana, US)

In this 30 minute talk we give a brief overview of UniMath, the formalization project started by Voevodsky, Ahrens, and me, that aims to formalize a substantial body of mathematics in the univalent foundations, building on the original formalization by Voevodsky from 2009, the "Foundations". The code (174K lines) is in Coq and is hosted at http://unimath.org/. My personal goal for the next year is to formalize a preprint of mine on algebraic K-theory, so when I finally submit it for publication, I can include the formalization to make the job of the referee easier.

We give a tour of the code, showing how interaction with it works in ProofGeneral in emacs, touching upon univalence, the implementation of groups, and Voevodsky's resizing axioms.

3.6 Formalization of Smooth Manifolds in Isabelle/HOL

Fabian Immler (Carnegie Mellon University – Pittsburgh, US) and Bohua Zhan (Chinese Academy of Sciences – Beijing, CN)

We formalize the definition and basic properties of smooth manifolds in Isabelle/HOL. Concepts covered include partition of unity, tangent and cotangent spaces, and the fundamental theorem for line integrals. We also construct some concrete manifolds such as spheres and projective spaces. The formalization makes extensive use of the existing libraries for topology and analysis. The existing library for linear algebra is not flexible enough for our needs. We therefore set up the first systematic and large scale application of "types to sets". It allows us to automatically transform the existing (type based) library of linear algebra to one with explicit carrier sets.

3.7 Heuristics for rewrite search

Scott Morrison (Australian National University – Canberra, AU)

Joint work of Scott Morrison, Keeley Hoek

I gave a demo of my recent work formalizing category theory in Lean, both discussing my goals as mathematician visiting the world of interactive theorem proving, and showing off some fun graph visualizations of an algorithm for proving equational lemmas using rewriting guiding by edit distance heuristics.

In general, I have been trying to understand how far writing mathematics in a modern interactive theorem prover is from the usual experience of writing and explaining mathematics to other humans. (Of course, the answer for now is "too far".) Category theory is an interesting and easy test case, as frequently in proofs and constructions there are quite considerable verifications which ought to be undertaken (checking functors are functorial, natural transformations natural) but which are very frequently omitted in human mathematics. I've been trying to write a category theory library working within the constraint that none of these verifications may be performed with human assistance (and ideally, should be kept entirely out of human sight). Of course, this requires developing at the same time a certain amount of automation particular to the domain of the mathematics being formalized. It is essentially for this reason that I've chosen to work in Lean: it seems to have the most flexible and easy to learn mechanism for writing new automation amongst modern theorem provers.

Do we hope one day to have a non-trivial portion of research mathematics performed with the aid of computers? (Here I mean the actual research, not merely post hoc formalization.) If so, I think it will be necessary that 'writing tactics' becomes easy enough that it is within reach of end users, not just developers of the interactive theorem provers. For now, of course, it is not easy enough, but I have been encouraged by working in Lean, and observing my (mathematics) students coming to grips with Lean and writing tactics in Lean. (The biggest obstacle may just be that nearly all mathematicians, and still most mathematics students, aren't at all familiar with functional programming and working with monads! Dependent type theories themselves are no obstacle.)

In my demo I showed two related recent pieces of work. One was an algorithm for proving equational goals via rewriting, using heuristics based on edit distance to explore the graph of possible rewrites by a given set of lemmas. (I think the audience enjoyed the graphical visualizations of the proof searches!) Along with a student Keeley Hoek, we're now incorporating classification techniques, using a support vector machine to dynamically reweight the tokens appearing in expressions as the search proceeds. This is early work, but tentatively it appears that this can help focus the search on the key steps, avoiding needlessly exploring minor irrelevant rewrites. We're new to this field, and hoping to learn more about previous work in this direction, and especially hoping to come up with heuristics for generating "stepping stone" intermediate goals based on analyzing partial search graphs.

The second was an illustration of how this algorithm can elide many of the "boring" proofs in a basic category theory library. I quickly showed some examples of proofs of the Yoneda lemma from other interactive theorem provers, followed by the very short proofs in my library in Lean. These successfully rely on some basic automation, and the heuristics for rewrite searches described above, to allow us to just write the statements a mathematician

would write, omitting all the easy verifications. As an example, we can reduce the entire definition of the Yoneda functor itself to

```
def yoneda : C ⇒ ((C^op) ⇒ (Type v₁)) := λ' X, λ' Y : C, Y → X
```

with two functoriality and one naturality statement being synthesized behind the scenes. (I found some formalizations of this statement that occupied more than a page.) Obviously this is an extreme example, but it illustrates my goal that automation should strive to meet the mathematician, rather than the other way round, when possible.

Participating in the Dagstuhl seminar was really exciting for me – it was a great opportunity to make contact with the community around interactive theorem provers, and it was great that mathematicians new to the field were made so welcome!

3.8 Cubical Agda Demo

Anders Mörtberg (Chalmers University of Technology – Göteborg, SE)

In this short demo I showed a cubical version of the Agda proof assistant implemented by Andrea Vezzosi. This system allows for a direct proof of functional extensionality and also the univalence axiom. Another exciting aspect is that it allows the user to define higher inductive types with good computational behavior. This was illustrated by live proving that the torus is equivalent to the product of two circles.

3.9 Semi-formal verification as a routine tool

Neil Strickland (University of Sheffield, GB)

Proof assistants are rarely used by working mathematicians for new research. Many people have thought about what proof assistants can currently do, and how one should work from there towards what mathematicians need. In this talk we look at the problem from a different angle. There are other mathematical software systems that are very widely used by researchers, including Sage, Mathematica and Maple. In particular, the author has made extensive use of Maple for a kind of semi-formal verification of some kinds of mathematical arguments. In this talk we describe this experience, and discuss how to narrow the gap with fully formal verification. It would be helpful if proof assistants could interface in some way with systems such as Maple, and we also discuss some issues related to this.

3.10 Formal Abstracts

Floris van Doorn (University of Pittsburgh, US)

Joint work of Tom Hales, Floris van Door
URL https://github.com/formalabstracts/formalabstracts

Formal Abstracts is a ambitious project to build a large database of theorems and definitions from almost all branches of mathematics in both human-readable and machine-readable form. It will serve as a database for machine-learning projects, can be used to semantically search the database of mathematical definitions and theorems, could be used for exploring mathematics, and to translate mathematics between languages, preserving the semantics. In this talk I gave an overview of the goals and concrete plans of the Formal Abstracts project.

3.11 Structuring principles for specifications in Isabelle

Makarius Wenzel (Augsburg, DE)

URL http://files.sketis.net/Dagstuhl2018/Isabelle_Structure.tar.gz

This is a brief overview of Local Theory Specifications in Isabelle/Pure, which are extensively used in Isabelle/HOL libraries and applications: unnamed contexts, locales, type classes. The included theory document (Isabelle_Structure.tar.gz) is for Isabelle2018.

3.12 The Isabelle Prover IDE after 10 years of development

Makarius Wenzel (Augsburg, DE)

URL https://sketis.net/wp-content/uploads/2018/08/Dagstuhl2018.pdf

The main ideas around Isabelle/PIDE go back to summer 2008. This is an overview of what has been achieved in the past 10 years, with some prospects for the future. Where can we go from here as Isabelle community? (E.g. towards alternative front-ends like Visual Studio Code; remote prover sessions "in the cloud"; support for collaborative editing of large formal libraries.) Where can we go as greater ITP community (Lean, Coq, HOL family)?

4 Working groups

4.1 Dagstuhl's Happy Diner Problem

Auke Booij (University of Birmingham, GB) and Floris van Doorn (University of Pittsburgh, US)

URL https://github.com/fpvandoorn/Dagstuhl-tables/

We have investigated Dagstuhl's Happy Diner Problem to find optimal seating arrangements during the meals at Dagstuhl.

The problem statement: What is the minimum number of meals so that each of the n conference participants can share at least one meal with every other participant when eating at tables of at most k persons? We call this number $T(n, k)$.

In particular, we have an unlimited number of tables, and we do not require that any two participants have a meal together exactly once, or that every table is fully occupied.

During the seminar, we have made progress on this problem using various techniques. This work is being documented via Github [1], and is ongoing.

- We have found several relations between various entries in the table of values $T(n, k)$, yielding both lower bounds and upper bounds for many entries. These relations allow us to fill in many entries in the table without any further exhaustive searches.
- We have manually computed certain entries $T(n, k)$, allowing us to fill in certain regions of the table.
- We collaborated with Michael Trott to use Mathematica's built-in SAT solver to find upper bounds for $T(n, k)$ for certain values of n and k.
- We have compared this problem with various related problems, such as the Oberwolfach problem [2], the Social Golfer problem [3, 4], and finding Kirkman Triple Systems[5]. In some cases, this allowed us to find values $T(n, k)$.

As a result of the work, we have submitted sequences to the Online Encyclopedia of Integer Sequences: A318240 and A318241. We have summarized the results in Table 1.

References

1 Floris P. van Doorn, Auke B. Booij. *Dagstuhl's Happy Diner problem*. Available at: https://github.com/fpvandoorn/Dagstuhl-tables/.
2 Sarah Holliday. *Sarah's Oberwolfach Problem Page*. Available at: http://facultyweb.kennesaw.edu/shollid4/oberwolfach.php. Accessed October 2018.
3 Social Golfer Problem on Wolfram MathWorld. Available at: http://mathworld.wolfram.com/SocialGolferProblem.html. Accessed October 2018.
4 A107431 on the Online Encyclopedia of Integer Sequences. Available at: https://oeis.org/A107431. Accessed October 2018.
5 Dijen K. Ray-Chaudhuri, Richard M. Wilson. *Solution of Kirkman's schoolgirl problem*. In Proc. of Symp. in Pure Math, Vol 19, 1971.

Table 1 Table of solutions $T(n,k)$, or ranges of possible solutions. Bold numbers are optimal solutions in the sense that every conference participants shares a meal with every other participant *exactly* once.

n / k	2	3	4	5	6	7	8
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**
2	**1**	1	1	1	1	1	1
3	3	**1**	1	1	1	1	1
4	**3**	3	**1**	1	1	1	1
5	5	3	3	**1**	1	1	1
6	**5**	4	3	3	**1**	1	1
7	7	4	3	3	3	**1**	1
8	**7**	4	3	3	3	3	**1**
9	9	**4**	4	3	3	3	3
10	**9**	6	4	4	3	3	3
11	11	6	5	4	3	3	3
12	**11**	6	5	4	3	3	3
13	13	7	5	5	4	3	3
14	**13**	7	5	5	4	4	3
15	15	**7**	5	5	4	4	3
16	**15**	9	**5**	5	4	4	3
17	17	9	6-9	5	4	4	3-4
18	**17**	9	7-9	5-6	4	4	3-4
19	19	10	7-9	5-6	5-6	4	3-4
20	**19**	10	7-9	5-6	5-6	4-6	4
21	21	**10**	8-9	6	5-6	4-6	4-5
22	**21**	12	8-9	6	5-6	4-6	4-5
23	23	12	8-9	6	5-6	4-6	4-5
24	**23**	12	8-9	6	5-6	5-6	4-5
25	25	13	9	**6**	6	5-6	4-5
26	**25**	13	9	7-9	6	5-6	4-5
27	27	**13**	9	7-9	6-7	5-6	5
28	**27**	15-16	**9**	8-9	6-7	5-7	5
29	29	15-16	10-11	8-9	6-7	5-7	5
30	**29**	15-16	11	8-11	6-7	5-7	5

4.2 Interoperability of systems

Mario Carneiro (Carnegie Mellon University – Pittsburgh, US), Gaëtan Gilbert (INRIA – Nantes, FR), Fabian Immler (Carnegie Mellon University – Pittsburgh, US), Maria Emilia Maietti (University of Padova, IT), and Makarius Wenzel (Augsburg, DE)

There was a total of 4 sessions, with slightly varying participants. Some notable topics of discussion:

- General problems of adjusting the logical languages, e.g. Lean vs. HOL, or other Type Theories.
- Questions about alignment of library content, e.g. the translated version of Nat vs. the existing one in the target system.

- An unorthodox approach to import HOL4 + CakeML into Isabelle, based on Isabelle/ML "virtualization" and replay of the original HOL4 theory and proof scripts directly in Isabelle/HOL (backed by concrete experiments by Fabian Immler).

Immler and Wenzel later continued the prototype of "virtual HOL4 inside Isabelle"; this work is likely to become part of future releases of any of these proof assistants.

4.3 Debugging Coq

Gaëtan Gilbert (INRIA – Nantes, FR) and Daniel R. Grayson (Urbana, US)

Gaëtan Gilbert and I looked into the internals of Coq to try to figure out how to make the resizing axioms work in UniMath without disabling all universe checking using the "type in type" option. The main result was a succinct bug report to the Coq team at INRIA, which we hope will be acted upon soon. I also learned from Gaëtan some tricks for debugging Coq in the OCaml debugger, which will help if I have to look more deeply into the problem.

4.4 Structures

Assia Mahboubi (INRIA – Nantes, FR), Yves Bertot (INRIA Sophia Antipolis, FR), Jacques Carette (McMaster University – Hamilton, CA), Cyril Cohen (INRIA Sophia Antipolis, FR), Diane Gallois-Wong (Laboratoire de Recherche en Informatique – Orsay, FR), Georges Gonthier (INRIA Saclay – Île-de-France, FR), Florent Hivert (Laboratoire de Recherche en Informatique – Orsay, FR), Johannes Hölzl (Free University Amsterdam, NL), Scott Morrison (Australian National University – Canberra, AU), Russell O'Connor (Blockstream – Montreal, CA), Claudio Sacerdoti Coen (University of Bologna, IT), Bas Spitters (Aarhus University, DK), Michael Trott (Wolfram Research – Champaign, US), Hoang Le Truong (Universität des Saarlandes, DE), Josef Urban (Czech Technical University – Prague, CZ), and Makarius Wenzel (Augsburg, DE)

The participants of this working group have discussed the representation of algebraic structures both in computer algebra systems and in a collection of different proof assistants. Several short presentations have fostered the discussions, including:

- Canonical Structures in MathComp by Georges Gonthier;
- Structures in Sage by Florent Hivert (based on excerpts of Nicolas M. Thiéry's talks at CICM, July 28th of 2016, Bialystok);
- MathClasses by Bas Spitters;
- Unification hints by Claudio Sacerdoti;
- Type classes/Locales in Isabelle by Makarius Wenzel;
- Type classes in Lean by Johannes Hölzl;
- Soft typing in Mizar by Josef Urban;
- auto2, by Bohua Zhan's.

The objective was to review the different solutions, their assets and their limitations. In fact, a significant part of the discussion was devoted to identifying the requirements of the various stakeholders-roles, namely system builders, library builders, advanced users and users. Making more precise these requirements should help benchmarking the different ways of designing and implementing a graph of structures in a proof assistant, together with the related tools for inference, search, debug, etc.

The second main outcome of the working group is a list of known difficult problems related to the inference of instances of algebraic structures, in various contexts. The items in this list are very diverse in nature, ranging from the algorithmic issues in the inference algorithms implemented by proof assistants, to the design of complex hierarchies like ordered algebraic structures, and to the cost of changing the representation of objects (e.g. dense vs sparse polynomials).

4.5 Cubical Working Group

Anders Mörtberg (Chalmers University of Technology – Göteborg, SE), Carlo Angiuli (Carnegie Mellon University – Pittsburgh, US), Guillaume Brunerie (University of Stockholm, SE), Kuen-Bang (Favonia) Hou (University of Minnesota – Minneapolis, US), Simon Huber (University of Göteborg, SE), Dan Licata (Wesleyan University – Middletown, US), Ian Orton (University of Cambridge, GB), Bas Spitters (Aarhus University, DK), and Jonathan Sterling (Carnegie Mellon University – Pittsburgh, US)

The members of the cubical working group worked on a variety of problems related to cubical type theories. These theories provide computational justifications to the univalence axiom and higher inductive types and there are now multiple implementations based on these new type theories. Many of the authors of these systems were present at the meeting which led to very fruitful collaborations among experts that are not often at the same place – thanks to the organizers and Dagstuhl for providing us with this opportunity.

The main problem that we worked on was to better understand the various computational inefficiencies that seem present in all of the implementations of cubical type theories. For example we noticed that the computation time and memory usage was heavily dependent on how loops were nested when computing winding numbers. With these examples we could benchmark the various systems and get new ideas for how to optimize the particular systems. This led to a variety of new optimizations which increased the performance on multiple of the examples.

We also optimized the proof of one of the key lemmas underlying the most complicated part of the algorithms in all of these systems. In this algorithm we only need a special case of a general lemma and we found a new proof of this special case, which we now call the "Dagstuhl lemma". The lemma was used to optimize the implementation of the cubicaltt proof checker and it was also formally verified in Agda during the meeting.

One of the major open problems in implementing cubical type theory is to define a simple and efficient notion of evaluation which is adequate for open terms. The presence of the diagonal cofibrations in cartesian cubical type theory complicates the question and constrains the potential solutions, leading to a form of evaluation which is executed relative to an evolving equational theory on dimensions. During the seminar, members of the cubical type

theory working group collaborated to work out the invariants and operations of a semantic domain for open computation, which will form the backbone of the algorithm to decide definitional equivalence in implementations of cartesian cubical type theory, such as the redtt proof assistant.

4.6 Subjecting mathematicians to proof assistants

Neil Strickland (University of Sheffield, GB), Sophie Bernard (INRIA Sophia Antipolis, FR), Auke Booij (University of Birmingham, GB), Mario Carneiro (Carnegie Mellon University – Pittsburgh, US), Manuel Eberl (TU München, DE), Martín H. Escardó (University of Birmingham, GB), Daniel R. Grayson (Urbana, US), Nicolai Kraus (University of Nottingham, GB), Scott Morrison (Australian National University – Canberra, AU), Anja Petkovic (University of Ljubljana, SI), Makarius Wenzel (Augsburg, DE), and Bohua Zhan (Chinese Academy of Sciences – Beijing, CN)

URL http://neil-strickland.staff.shef.ac.uk/dagstuhl/

The aim of this working group was to develop examples and documentation explaining the use of proof assistants to working mathematicians. The emphasis is on issues likely to arise when trying to formalize new research, and issues where proof assistants fit poorly with a mathematician's natural expectations and intuitions. During the meeting we gathered a lot of useful information, from presentations of code as well as discussions of conceptual issues. Since the meeting a substantial amount of work has been done towards assembling this information into a useful set of web pages, and the Lean community has also contributed further code and advice. This work is still ongoing.

Our discussions in the working group were organized around the four tasks described below. Some solutions were presented by Sophie Bernard (Coq + ssreflect), Manuel Eberl (Isabelle-HOL) and Bohua Zhan (Isabelle-FOL), and the detailed walk-through of this code was very illuminating. There were also conversations outside the working group in which various people explained useful things; thanks are especially due to Mario Carneiro, Scott Morrison and Makarius Wenzel.

In brief, the tasks were as follows.

- Prove that for any natural number n, there is a prime p with $p > n$.
- Set up the theory of the group of units in a commutative ring.
- Set up the theory of the ideal of nilpotent in a commutative ring, and the corresponding quotient ring.
- Set up the theory of chained preorders (a kind of discrete combinatorial structure on a finite set).

The detailed specification of the tasks covers a number of issues that may cause difficulty: locating standard results in the standard library, confusion between different implementations of the natural numbers, the general framework for abstract algebra, subobjects and quotient objects, finiteness and decidability, and so on.

Participants

- Benedikt Ahrens
University of Birmingham, GB
- Carlo Angiuli
Carnegie Mellon University – Pittsburgh, US
- Andrej Bauer
University of Ljubljana, SI
- Sophie Bernard
INRIA Sophia Antipolis, FR
- Yves Bertot
INRIA Sophia Antipolis, FR
- Auke Booij
University of Birmingham, GB
- Guillaume Brunerie
University of Stockholm, SE
- Jacques Carette
McMaster University – Hamilton, CA
- Mario Carneiro
Carnegie Mellon University – Pittsburgh, US
- Cyril Cohen
INRIA Sophia Antipolis, FR
- Manuel Eberl
TU München, DE
- Martín H. Escardó
University of Birmingham, GB
- Diane Gallois-Wong
Laboratoire de Recherche en Informatique – Orsay, FR
- Gaëtan Gilbert
INRIA – Nantes, FR
- Georges Gonthier
INRIA Saclay – Île-de-France, FR
- Daniel R. Grayson
Urbana, US
- Philipp Haselwarter
University of Ljubljana, SI
- Florent Hivert
Laboratoire de Recherche en Informatique – Orsay, FR
- Johannes Hölzl
Free University Amsterdam, NL
- Kuen-Bang (Favonia) Hou
University of Minnesota – Minneapolis, US
- Simon Huber
University of Göteborg, SE
- Fabian Immler
Carnegie Mellon University – Pittsburgh, US
- Nicolai Kraus
University of Nottingham, GB
- Dan Licata
Wesleyan University – Middletown, US
- Peter L. Lumsdaine
University of Stockholm, SE
- Assia Mahboubi
INRIA – Nantes, FR
- Maria Emilia Maietti
University of Padova, IT
- Anders Mörtberg
Chalmers University of Technology – Göteborg, SE
- Scott Morrison
Australian National University – Canberra, AU
- Russell O'Connor
Blockstream – Montreal, CA
- Ian Orton
University of Cambridge, GB
- Anja Petkovic
University of Ljubljana, SI
- Claudio Sacerdoti Coen
University of Bologna, IT
- Bas Spitters
Aarhus University, DK
- Jonathan Sterling
Carnegie Mellon University – Pittsburgh, US
- Neil Strickland
University of Sheffield, GB
- Michael Trott
Wolfram Research – Champaign, US
- Hoang Le Truong
Universität des Saarlandes, DE
- Josef Urban
Czech Technical University – Prague, CZ
- Floris van Doorn
University of Pittsburgh, US
- Makarius Wenzel
Augsburg, DE
- Bohua Zhan
Chinese Academy of Sciences – Beijing, CN