Report from Dagstuhl Seminar 23401

Automated mathematics: integrating proofs, algorithms and data

Andrej Bauer[*1], **Katja Berčič**[*2], **Florian Rabe**[*3], **Nicolas Thiéry**[*4], **and Jure Taslak**[†5]

1 **University of Ljubljana, SI & Institute for Mathematics, Physics and Mechanics, Ljubljana, SI**
2 **University of Ljubljana, SI & Institute for Mathematics, Physics and Mechanics, Ljubljana, SI**
3 **Universität Erlangen-Nürnberg, DE**
4 **University Paris-Saclay – Orsay, FR**
5 **University of Ljubljana, SI**

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 23401 "Automated mathematics: integrating proofs, algorithms and data". The seminar brought together system developers, library authors, and users from key branches of computer-supported mathematics: formalized mathematics, symbolic computation, and mathematical databases. We addressed issues that were common to all areas of computer-supported mathematics: library management, dependencies and interoperability between software components, quality and correctness assurances, searching for information, and usability by end users. Early on in the week, we formed working groups that worked on specific tasks, as described in this report. Each day was divided into a morning talk session and an afternoon period devoted to working in groups. To keep everyone well-informed, we gathered each day before dinner for an informal "show & tell" session.

Seminar October 1–6, 2023 – https://www.dagstuhl.de/23401
2012 ACM Subject Classification Mathematics of computing → Mathematical software; Computing methodologies → Symbolic and algebraic manipulation; Computing methodologies → Symbolic and algebraic manipulation; Computing methodologies → Symbolic and algebraic manipulation
Keywords and phrases mathematical knowledge management, mathematical software, formalized mathematics, computer algebra, databases of mathematical structures
Digital Object Identifier 10.4230/DagRep.13.10.1

1 Executive Summary

Andrej Bauer
Katja Berčič
Florian Rabe
Nicolas Thiéry

Modern mathematical software and large datasets of mathematical knowledge allow new approaches to solving mathematical problems, and support new kinds of mathematical exploration. In the past, lack of cooperation led to each project developing its own standards

* Editor / Organizer
† Editorial Assistant / Collector

and techniques. Both developers and users experienced the resulting incompatibilities and fragmentation as serious usability issues and major obstacle to wider adoption, and none of the projects can address these successfully on their own. To make further progress that will eventually result in truly comprehensive and advanced computer mathematical assistant systems, the developers of individual software projects must start working together and tackle questions of interoperability, software engineering, and knowledge management on a large scale.

Concretely, this Dagstuhl Seminar brought together system developers, library authors, and users from three key areas of computer-supported mathematics: theorem provers, symbolic computation, and databases of mathematical structures. All three areas develop large formal mathematical libraries, but they do so in fundamentally different and incompatible ways. Theorem provers optimize for precise definitions and automation of proof support, often employing complex and abstract representation languages that capture exactly the semantics of the mathematical concepts of interest. Computer algebra systems, on the other hand, prioritize the efficiency of computation, which usually requires hardware-near representations that can be optimized for speed. Mathematical datasets finally employ general purpose database languages, which are optimized for indexing and fast querying, often requiring non-trivial encodings of mathematical objects in terms of concrete data. Over several decades of mostly independent development, these different communities have built software systems that are as impressively large as they are different from each other.

Work Groups

Broadly speaking, the seminar participants employed three independent approaches towards system integration.

Firstly, *direct integration* builds individual bridges between (usually) two systems. These tend to be more ad hoc but enable a problem-driven approach that delivers a specific practically needed integration solution. Multiple work groups were formed that tackled individual bridges.

Secondly, *ontology-based integration* uses a central representation that acts as an interoperability layer. The ontology describes mathematical concepts abstractly without a commitment to any of the three flavors of systems. Two work groups investigated this approach:

- Alignments: This work group worked towards building a central ontology of mathematical concepts. It surveyed existing approaches and judged the feasibility of major future approaches. Of special interests was the difficulty of *library alignment*, the task of connecting the central ontology to the individual libraries.
- Knowledge graphs: This work group investigated services that can be built on top of a central ontology. Of particular interest were knowledge graph techniques, which use concepts as the nodes and alignments as some of the edges.

Thirdly, for the special goal of integrating datasets, a work group on building an *index of datasets* started a major push towards cataloging the many existing datasets, which are distributed all over the internet, often without active maintenance. This is a necessary step towards more systemic integration with each other and deduction and computation systems.

Outcomes

Overall, we observed that the field has made major progress over the last 10 years. Direct integrations that would have been very expensive in the past, often prohibitively so, have become feasible targets for short meetings as within a Dagstuhl Seminar. This can be attributed to increased awareness in the community of interoperability needs that has led to better interface design. Nonetheless, integrations are still brittle, and a major incentive problem remains: it is difficult for two communities to maintain bridges between their systems.

Ontology-based integration had developed little momentum in the past because of the high cost of additionally maintaining the central ontology. Here, the seminar showed that the time is right for a major push towards this and initiated a community-driven ontology curation project.

The work on building an index of datasets kick-started a dataset curation project. This project has already attracted the attention of outside researchers and has led to the founding of the Mathbase project.

2 Table of Contents

3 Overview of Talks

3.1 Who finds the short proof? Searching for Wormholes in Proof-Space

Christoph Benzmüller (Universität Bamberg, DE)

Joint work of Christoph Benzmüller, David Fuenmayor, Alexander Steen, Geoff Sutcliffe
Main reference Christoph Benzmüller, David Fuenmayor, Alexander Steen, Geoff Sutcliffe: "Who Finds the Short Proof?", Logic Journal of the IGPL, p. jzac082, 2023.
URL https://doi.org//10.1093/jigpal/jzac082

In my talk I presented recently published results that had not been communicated at conferences or workshops before: An exploration of Boolos' Curious Inference using higher-order automated theorem provers (ATPs). Surprisingly, only suitable shorthand notations had to be provided by hand for ATPs to find a short proof. The higher-order lemmas required for constructing a short proof are automatically discovered by the ATPs. Given the observations and suggestions in this paper, full proof automation of Boolos' and related examples now seems to be within reach of higher-order ATPs.

References

1 Benzmüller, C., Fuenmayor, D., Steen, A. and Sutcliffe, G. Who Finds the Short Proof? Logic Journal of the IGPL. 2023. Doi: 10.1093/jigpal/jzac082
2 Benzmüller, C. and Brown, C. The curious inference of Boolos in MIZAR and OMEGA. In Matuszewski, R., Zalewska, A., editor(s), From Insight to Proof – Festschrift in Honour of Andrzej Trybulec, volume 10(23), of Studies in Logic, Grammar, and Rhetoric, pages 299-388. The University of Bialystok, Polen, 2007.
3 Boolos, G. A Curious Inference. Journal of Philosophical Logic 16(1):1-12, 1987.

3.2 A catalogue of mathematical datasets

Katja Berčič (University of Ljubljana, SI)

Main reference Katja Bercic: "Towards a Census of Relational Data in Mathematics", in Proc. of the Conference on "Lernen, Wissen, Daten, Analysen", Berlin, Germany, September 30 – October 2, 2019, CEUR Workshop Proceedings, Vol. 2454, pp. 207–217, CEUR-WS.org, 2019.
URL https://ceur-ws.org/Vol-2454/paper_40.pdf

In addition to well-known mathematical databases, such as the OEIS, LMFDB and the House of Graphs, there is a multitude of smaller projects. Any of these might be interesting to researchers from other areas, including machine learning. However, the smaller dataset projects are typically hard to find: just the right keywords are usually necessary (but not necessarily sufficient) to find them with standard search engines and even in scientific research data repositories (like Zenodo). As a community, we should build a collaborative, and at least partly automated, index of mathematical datasets. Personal projects, such as my own mathdb.mathhub.info, can serve as a starting point.

3.3 Formal verification of mathematical algorithms when the definitions are out of reach

Alex Best (King's College – London, GB)

Joint work of Alex Best, Sander Dahmen, Sacha Huriot-Tattegrain

I'd be happy to report on and discuss some ongoing work (with S. Dahmen and S. Huriot-Tattegrain) implementing a tricky mathematical algorithm in the field of number theory / arithmetic geometry in a proof assistant (this computes certain quantities appearing in the famous BSD conjecture many of which are recorded in the LMFDB). This algorithm, known as Tate's algorithm in the field, is quite involved with many subcases and a non-trivial proof of terminations, and takes many pages to describe when expressed in paper form.

One thing that makes this an interesting is that to give a formal definition of the quantities the algorithm actually computes is still out of reach, nevertheless the steps of this algorithm as described in the literature can be implemented and termination can be proved with some non-trivial tracking of the state involved, but in what ways is this implementation more trustworthy than one in a regular programming language? I'd like to consider what guarantees implementing such code in a the strict setting of a proof assistant can give us, compared to ordinary code, even when the gold standard isn't yet attainable.

The direct relationship between the mathematical theory and the code in this case means that this implementation very clearly states what assumptions are made about the ring we are working in, and in fact makes this the most general implementation available, this then begs the question: how can we best integrate mathematical code written using proof assistants into existing CASes? So it may be useful to users not running proof assistants themselves.

3.4 Learning from "invisible mathematics"

Jacques Carette (McMaster University – Hamilton, CA)

Recently Andrej Bauer coined the term "invisible mathematics" for those aspects of mathematics which are readily apparent when doing them mechanically (especially, but not only, in proof assistants) but which is essentially invisible in traditional paper-math. Three main examples were given, and their formalization discussed.

Here we aim to give more such examples – but the aim is not on how to formalize them, but rather to observe their consequences. In particular, some instances of "invisible mathematics" readily reveal the cognitive load on learners of some (visible) mathematics. Rather than seeing parts of the "de Bruijn factor" as an impediment, we instead regard it as a learning opportunity.

3.5 Proving an Execution of an Algorithm Correct?

James H. Davenport (University of Bath, GB)

We have previously [1] asked the question "Do I believe the output from my (complicated, optimised, unverified) computer algebra system?". If it gives me a positive answer, e.g. "the answer to $\int f$ is g", then we can check that $g' = f$, etc. But what if it says "there is no answer"? In particular, if we ask to factor the polynomial F, then we can check that the factors multiply to F, but what about the (implicit) statement that these factors are irreducible? Currently, the user just has to believe the algebra system. We ask, and partially answer, the question "could the algebra system also produce a certificate, or the hints to construct a certificate, of irreducibility of the factors" (based on the work the algebra system has already done). This was taken up by one working group.

References

1 James Harold Davenport. Proving an Execution of an Algorithm Correct? In Dubois and Kerber [2], pages 255–269.

2 Catherine Dubois and Manfred Kerber, editors. *Proceedings CICM 2023*, volume 14101 of *Springer Lecture Notes in Computer Science*, 2023.

3.6 Extracting Mathematical Concepts from Text

Valeria de Paiva (Topos Institute – Berkeley, US)

Joint work of Valeria de Paiva, Lucy Horowitz, Jacob Collard, Eswaran Subramahnian, Pavel Kovalev, Qiyue Gao, Lawrence Moss
Main reference Lucy Horowitz, Valeria de Paiva: "MathGloss: Building mathematical glossaries from text", CoRR, Vol. abs/2311.12649, 2023.
URL https://doi.org//10.48550/ARXIV.2311.12649
Main reference Jacob Collard, Valeria de Paiva, Eswaran Subrahmanian: "Parmesan: mathematical concept extraction for education", CoRR, Vol. abs/2307.06699, 2023.
URL https://doi.org//10.48550/ARXIV.2307.06699
Main reference Jacob Collard, Valeria de Paiva, Brendan Fong, Eswaran Subrahmanian: "Extracting Mathematical Concepts from Text", CoRR, Vol. abs/2208.13830, 2022.
URL https://doi.org//10.48550/ARXIV.2208.13830
Main reference Valeria de Paiva, Qiyue Gao, Pavel Kovalev, Lawrence S. Moss: "Extracting Mathematical Concepts with Large Language Models", CoRR, Vol. abs/2309.00642, 2023.
URL https://doi.org//10.48550/ARXIV.2309.00642

We describe the project Network Mathematics we are developing at the Topos Institute, Berkeley, CA. We hope to take advantage of the tremendous recent progress in Natural Language Processing (NLP), including LLMs, transformers, etc, to extract information from mathematical texts. Also we think mathematical language is what mathematicians use when communicating with each other, so we should leverage mathematical English to increase the accessibility of mathematics to mathematicians, students and the general public.

To work on these goals, we have three main preliminary subprojects:

1. MathGloss (together with Lucy Horowitz, Chicago) – a collection of glossaries of college math, connected to each other via WikiData.
2. Parmesan (together with Jacob Collard and Eswaran Subramahnian, NIST) provides semantic search on the field of Category Theory, using different knowledge bases at different levels (research math, wiki math and textbook math).

3. MathChat (together with Gao, Kovalev and Moss, Indiana) using generative AI (chatGPT and GTP-4) to extract concepts from text.

All these subprojects need to grow and need to include more sources. Also, further work needs to be done to structure the extracted concepts into a proper ontology, to obtain a proper Knowledge Graph for mathematics.

3.7 (Re)Verification of Proofs

Catherine Dubois (ENSIIE – Evry, FR)

Main reference Guillaume Burel, Guillaume Bury, Raphaël Cauderlier, David Delahaye, Pierre Halmagrand, Olivier Hermant: "First-Order Automated Reasoning with Theories: When Deduction Modulo Theory Meets Practice", J. Autom. Reason., Vol. 64(6), pp. 1001–1050, 2020.
URL https://doi.org//10.1007/S10817-019-09533-Z

The talk gives a quick overview of some tools developed around Dedukti to verify, re-verify or cross-verify proofs, more precisely, Zenon Modulo, iProverModulo, Archsat, and Ekstrakto. The three first ones directly produce Dedukti proofs that can be checked by the Dedukti checker. The latter reconstructs a Dedukti proof from a proof trace by reproving each step using a Dedukti producing tool and combining the proofs of the steps to get a proof of the original formula. In the talk, we also point out two projects: BWare and ICSPA. The first one aimed at developing a mechanized framework for automated verification of AtelierB proof obligations where Zenon Modulo and iProvermodulo were developed or used. ICSPA is a project in progress where the objectives are to improve confidence in the proofs realized in the context of B/Event-B and TLA+ by formally and independently verifying these proofs and also to enable sharing and reusing proofs and models between B/Event-B and TLA+ using lambda-PI calculus modulo theory and Dedukti.

3.8 Understanding the Symmetries of Bin Packing Problems Inspired by Application Deployment in the Cloud

Madalina Erascu (West University of Timisoara, RO)

Automated deployment of component-based applications in the Cloud consists in the allocation of virtual machines (VMs) offers from various Cloud Providers such that the constraints induced by the interactions between components and by the components hardware/software requirements are satisfied and the performance objectives are optimized (e.g. costs are minimized). It can be formulated as a constraint optimization problem, hence, in principle, the optimization can be carried out automatically. In the case the set of VM offers is large (several hundreds), the computational requirement is huge, making the automatic optimization practically impossible with the current general optimization modulo theory (OMT) and mathematical programming (MP) tools. We overcame the difficulty by methodologically analyzing the particularities of the problem with the aim of identifying search space reduction methods. Some of these methods are exploiting the symmetries of the general Cloud deployment problem leading to symmetry breakers. However, little is know/understood about the symmetries of this problem which could lead to symmetry breakers which are not experimentally constructed. We present some of the symmetries of a simplistic formulation of automated deployment with the hope that the audience:

- can bring input regarding approaches applied for similar problems (bin-packing)
- is interested in taking part in a small working group to discuss if invariant theory of finite groups (see, e.g., Chapter 7 of [1]) could bring some insights towards the formalization of symmetries.

References

1 Cox, D.A., Little, J., O'Shea, D.. *Invariant Theory of Finite Groups*. In: Ideals, Varieties, and Algorithms. Undergraduate Texts in Mathematics. Springer, Cham, 2015

3.9 House of Graphs: A searchable database of interesting graphs and more

Jan Goedgebeur (KU Leuven, BE)

Joint work of Jan Goedgebeur, Gunnar Brinkmann, Kris Coolsaet, Sven D'hondt, Gauvain Devillez, Hadrien Mélot
Main reference Kris Coolsaet, Sven D'hondt, Jan Goedgebeur: "House of Graphs 2.0: A database of interesting graphs and more", Discret. Appl. Math., Vol. 325, pp. 97–107, 2023.
URL https://doi.org//10.1016/J.DAM.2022.10.013

We will present the House of Graphs (`https://houseofgraphs.org/`), which is a database of graphs. The House of Graphs hosts complete lists of graphs of various graph classes (e.g. cubic graphs, fullerenes, trees, etc.), but its main feature is a searchable database of so called "interesting" graphs, which includes graphs that already occurred as extremal graphs or as counterexamples to conjectures. We will highlight the features of the website and demonstrate how users can perform queries on this database and how they can add new interesting graphs to it.

3.10 Mostly Automated Proof Repair for Verified Libraries

Kiran Gopinathan (National University of Singapore, SG)

Joint work of Kiran Gopinathan, Mayank Keoliya, Ilya Sergey
Main reference Kiran Gopinathan, Mayank Keoliya, Ilya Sergey: "Mostly Automated Proof Repair for Verified Libraries", Proc. ACM Program. Lang., Vol. 7(PLDI), pp. 25–49, 2023.
URL https://doi.org//10.1145/3591221

The cost of maintaining formally specified and verified software is widely considered prohibitively high due to the need to constantly keep code and the proofs of its correctness in sync—the problem known as proof repair. One of the main challenges in automated proof repair for evolving code is to infer invariants for a new version of a once verified program that are strong enough to establish its full functional correctness.

In this work, we present the first proof repair methodology for higher-order imperative functions, whose initial versions were verified in the Coq proof assistant and whose specifications remained unchanged. Our proof repair procedure is based on the combination of dynamic program alignment, enumerative invariant synthesis, and a novel technique for efficiently pruning the space of invariant candidates, dubbed proof-driven testing, enabled by the constructive nature of Coq's proof certificates.

3.11 Towards a centralized system for mathematical objects

Dimitri Leemans (UL – Brussels, BE)

Mathematical data are available in all sorts of formats on the web. Often it is difficult for those who did not create the data to use them. Also these data are at risk of disappearing. We advocate for the building of a system consisting of

- a centralised database of mathematical objects, checked by peers,
- a website permitting to extract data from the database and send it to computational software (Magma, gap, etc.) to test conjectures, build more objects, ...

3.12 Machine-learnable Data Sets for Formalized Mathematics (MLFMF)

Matej Petkovic (University of Ljubljana, SI), Andrej Bauer (University of Ljubljana, SI)

Joint work of Andrej Bauer, Matej Petković, Ljupčo Todorovski
Main reference Andrej Bauer, Matej Petkovic, Ljupco Todorovski: "MLFMF: Data Sets for Machine Learning for Mathematical Formalization", CoRR, Vol. abs/2310.16005, 2023.
URL https://doi.org//10.48550/ARXIV.2310.16005

MLFMF is a collection of data sets for benchmarking recommendation systems used to support formalization of mathematics with proof assistants. Each data set is derived from a library of formalized mathematics written in proof assistants Agda or Lean. The collection includes the largest Lean 4 library Mathlib, and some of the largest Agda libraries: the standard library, the library of univalent mathematics Agda-unimath, and the TypeTopology library. Each data set represents the corresponding library in two ways: as a heterogeneous network, and as a list of s-expressions representing the syntax trees of all the entries in the library. The network contains the (modular) structure of the library and the references between entries, while the s-expressions give complete and easily parsed information about every entry.

3.13 Heterogenous search in formal mathematical libraries

Claudio Sacerdoti Coen (University of Bologna, IT)

I will present how I integrated in LambdaPi an indexer and search engine for mathematical formulae up to instantiation/generalization. The search engine allows to query in parallel libraries obtained by various provers (e.g. HOL/Matita/Coq/...) via an encoding into Dedukti/LambdaPi.

The task poses several challenges that are novel compared to search engines developed for a single system:

1. the statements occur encoded in LambaPi and the same statement coming from several systems is encoded in a different way

2. the statement can occur in several shapes, up to rewriting rules
3. in order to look in parallel in several libraries, one need to search up to alignment of constants.

I have solved all three previous challenges exploiting the rewriting engine of LambdaPi in order to:

1. undo the encoding via a non-type preserving transformation,
2. normalize the statements before indexing and the queries as well,
3. rewrite all cosntants to a canonic form (the representative of the equivalence class of aligned formula).

Indexing has been implemented combining substitution trees and information positioning (the one exploited in the Whelp system).

3.14 Proof and Computation with PVS

Natarajan Shankar (SRI – Menlo Park, US)

SRI's Prototype Verification System (PVS) is an interactive proof assistant based on higher-order logic developed at SRI over the last three decades as a unified platform and language for formal specification, mathematical modeling, programming, and proof. It has been used to develop extensive libraries for mathematics and computing and for verification projects spanning fault-tolerant systems, air-traffic control systems, parsers, compilers, separation kernel, data refinement, and hardware. Nearly all of the specification language is efficiently executable with code extraction to Common Lisp and C, and experimental code generators targetting standard ML and Rust. We describe some of our experiments with modeling, proof, and computation focusing on extracting efficient C code from verified definitions.

During the Dagstuhl Seminar, we used PVS to verify two witness formats and executable checkers for graph connectivity. The witness for connectivity is a sequence containing all of the vertices such that each vertex has a neighbor in the preceding part of the sequence. This implies, by induction, that the graph is connected. For disconnectivity, the witness is a k-coloring of the vertices for $k > 1$ such that no vertex has a neighbor of a different color.

3.15 Enumerion, a system for systematic enumeration of finite mathematical structures

Jure Taslak (University of Ljubljana, SI)

Joint work of Jure Taslak, Andrej Bauer

In the talk I presented my recent work in progress on Enumerion, which is a system for systematic enumeration of finite mathematical structures based on dependent type theory and implemented in OCaml. The problem of counting and enumerating discrete finite structures is a classical one. In the history of counting there are however quite a few published mistakes[1].

The idea of this system is to have a systematic way of describing the enumeration problem which is amenable to formalization. After explaining the idea behind the system I showed a short demo of the current capabilities of Enumerion.

References

1 McKay, Brendan D and Meynert, Alison and Myrvold, Wendy. *Small Latin squares, quasigroups, and loops* Journal of Combinatorial Designs

3.16 Alien Coding: Learning Synthesis of OEIS Sequences

Josef Urban (Czech Technical University – Prague, CZ)

Joint work of Thibault Gauthier, Miroslav Olšák, Josef Urban
Main reference Thibault Gauthier, Miroslav Olsák, Josef Urban: "Alien coding", Int. J. Approx. Reason., Vol. 162, p. 109009, 2023.
URL https://doi.org//10.1016/J.IJAR.2023.109009

We introduce a self-learning algorithm for synthesizing programs that provide explanations for OEIS sequences. The algorithm starts from scratch initially generating programs at random. Then it runs many iterations of a self-learning loop that interleaves (i) training neural machine translation to learn the correspondence between sequences and the programs discovered so far, and (ii) proposing many new programs for each OEIS sequence by the trained neural machine translator. The algorithm discovers on its own programs for more than 78000 OEIS sequences, sometimes developing unusual programming methods. We analyze its behavior and the invented programs in several experiments.

3.17 Isabelle as System Platform for the Archive of Formal Proofs (AFP)

Makarius Wenzel (Dr. Wenzel – Augsburg, DE)

Isabelle is usually seen as an interactive proof assistant, mostly for Higher-Order Logic (HOL), but that is somehow accidental. In reality, Isabelle is a system platform for functional programming and formal proofs, with sufficient infrastructure to carry its own weight and gravity. The Archive of Formal Proofs (AFP) is the official collection of Isabelle applications that is maintained together with the base system. That poses ever growing demands on the Isabelle platform. This talk gives an overview of Isabelle software technology, with specific focus on Programming and Scaling, e.g. distributed build clusters for AFP.

4 Working groups

4.1 Using verified code inside CASes

Alex Best (King's College – London, GB) and Tobias Nipkow (TU München – Garching, DE)

Joint work of Alex Best, Tobias Nipkow, Samuel Lelièvre, Olexandr Konovalov, Mario Carneiro, Assia Mahboubi

Proof assistants are by now able to produce reasonably efficient executable code for a variety of mathematical algorithms. This, together with the high degree of confidence they can provide in the correctness of the code (and therefore in the answers produced), makes using code produced using a proof assistant in an otherwise unverified computer algebra system quite attractive. This could be used to replace existing unverified implementations (or complement them with possibly slower reference implementations) or add new functionality completely.

This working group used the expertise from participants on both the ITP and CAS sides to explore different methods and implementations to pass results of computations from code generated by ITPs to CASes. This was in order to understand what is possible and evaluate whether CASes could feasibly benefit from using such code more widely.

As a proof of concept, Tobias Nipkow took a verified (in Isabelle) implementation (in Haskell) of the Berlekamp and Zassenhaus integer polynomial factorization algorithm (which is available at the AFP `https://www.isa-afp.org/entries/Berlekamp_Zassenhaus.html`) and made it callable from the computer algebra systems Sage (with the help of Samuel Lelièvre) and GAP (with the help of Olexandr Konovalov). Although it was not meant to compete with the standard Sage and GAP functions, it turned out that in the case of GAP it outperformed the standard Factors function on large polynomials.

Alex Best, with the help and input of Mario Carneiro and Assia Mahboubi experimented with linking directly to binary (compiled) objects produced by Lean 4 using Python (with the intention of targeting SageMath). Using the low level CTypes library it was possible to directly call Lean implementations of functions on simple types, such as a `nextPrime` on 64 bit integers. A proof of concept for working at a higher level, using Lean interpreter to look up Lean functions stored memory, without needing prior knowledge of their exported names in compiled code, was also written. This allowed more flexibility as no wrapping function would need to be written on the Lean side, the CAS could call arbitrary code producing answers in a desired format without having to re-run the Lean compiler. Finally initial experiments with using polymorphic functions implemented in Lean on types implemented on the Python side were successful, for example with this paradigm a Python program could provide a list of Python objects and a callback comparison function to a Lean implementation of quicksort (which operates an arbitrary ordered type), this combination would produce correct output subject to the assumption that the python comparison function satisfies the partial order hypothesis used in the Lean code. In this way mixing verified and unverified code could be done with little overhead.

Going forward next steps would be to turn these examples into user facing libraries, to dynamically load and call code from ITP libraries from CASes, we believe that our experiments have shown such libraries can be of benefit to CAS users, even those with little formalization experience.

4.2 Reconceptualization

Jacques Carette (McMaster University – Hamilton, CA), Gilles Dowek (ENS – Gif-sur-Yvette, FR), and Catherine Dubois (ENSIIE – Evry, FR)

Main reference Florian Rabe, Franziska Weber: "Morphism Equality in Theory Graphs", in Proc. of the Intelligent Computer Mathematics – 16th International Conference, CICM 2023, Cambridge, UK, September 5-8, 2023, Proceedings, Lecture Notes in Computer Science, Vol. 14101, pp. 174–189, Springer, 2023.
URL https://doi.org//10.1007/978-3-031-42753-4_12

4.2.1 Motivation

Many concepts in mathematics have evolved over time, perhaps none more so than the concept of "space". While it seemed to have settled on "topological space" for a certain time, it was then rethought later and today it seems that "locale" and "infinity groupoid" are both solid contenders for the modern notion. Many notions have a similarly rich ongoing history: integration (quadrature), function, equality, and so on. Recent examples of such rethinking abound (perfectoid spaces and condensed mathematics, homotopy type theory, and so on).

In parallel, it is equally clear that many concepts have a multitude of equivalent (or quasi-equivalent) formulations. For example, the number of different representations of graphs is quite astounding.

4.2.2 Purpose

The working group was formed to discuss how to enable proof assistants in particular, and mechanized mathematics systems in general, to deal with these issues.

4.2.3 Discussion

We first discussed the simpler issue, that of having multiple presentations and representations. Historically, the first realization of the importance of this is via *change of variables*: working on orbital mechanics in Cartesian coordinates is close to sheer insanity while being rather workable in spherical coordinates. This is a very common theme in mathematics where a change in point-of-view can make a substantial difference in how easy a concept is to handle. This ranges for simple issues like different axiomatizations of what is a group, to larger shifts such as switching from indexed categories to fibered categories (and more recently, using displayed categories for similar aims).

It is worthwhile remembering that Lie Symmetries are a classical example of this phenomenon: they are all about finding a "good" set of coordinates for which a PDE non-trivially simplifies. What makes them different is that one can *compute* what this change of viewpoint needs to be, while in most cases creativity is required.

Alas, current proof assistants are largely hampered by a sub-optimal design decision, namely that concepts are assumed to have a single "canonical" definition. While univalent mathematics promises that one can transport between equivalent concepts, no system has yet to build in features that facilitate this. In other words, there has been no engineering effort made to create user-friendly facilities to accommodate multiple representations. There does exist work [3, 4] that lays out the concept of a **Realm** which is supposed to encompass this very idea: a single concept with potentially many presentations and/or representations. Many engineering and usability hurdles remain.

Beyond convenience, the reason to want concepts to have multiple interfaces and sub-interfaces is to be able to allow more powerful *development by refinement.* Here the idea of sub-interfaces really shines: if we know in advance that certain methods will never be used, it is sometimes the case that significantly more efficient representations and/or algorithms become feasible. For example, if we want a set representation where we can perform union, intersection and membership but neither count the number of elements or perform iteration on all elements, then there is a fast representation.

The overlap with the alignment working group was also discussed: for example, anyone who does constructive mathematics knows that there is no such singular concept as "finite set". Rather, there are a multitude of subtly related concepts all of which are equivalent to "finite set" classically. Whether to lump them all together or not is a thorny issue.

We also discussed the inverse interface problem: given some definitions and proven properties, how to find a good set of axioms that would abstract over that? Many proof developments, in practice, contain natural "layers" where outer layers only depend a *de facto* interface given by a lower layer, but this interface is never made explicit. Can the process of finding these layers in a given development be automated? That would help us create **strong abstraction barriers**. Weak barriers are a real problem as some systems are rather eager to δ-expand, leading to proof scripts which are much lower-level than necessary. As different formalizations may give rise to very different interfaces, the need for Realms re-appears.

Another aspect of automation was discussed: transport by meta-programming. The idea here is to not just transport across isomorphisms "in theory" but to perform it as a meta-program that tries to eliminate the transport altogether by attempting to rephrase everything in terms of the target language. In particular, such transport meta-programs remain very useful even when Univalence is false.

4.2.4 Conclusion

It became clear that, given the scope of the problem and the short amount of time at the seminar, we could not obtain any tangible results beyond clearly documenting the problem. The group then disolved early.

4.3 Formal Verification of Computer Algebra (Factorisatoion)

James H. Davenport (University of Bath, GB), Alex Best (King's College – London, GB), Mario Carneiro (Carnegie Mellon University – Pittsburgh, US), and Edgar Costa (MIT – Cambridge, US)

Of the problems listed by Davenport ([1] and talk here) we chose polynomial factorisation as having least mathematical pre-requisites. For computer algebra, we took FLINT as an easy-to access library of algorithms (specifically `factor_Zassenhaus`). As proof engine we took LEAN (familiarity). We have realised what we need as a certificate, seen that we can extract these data from FLINT (but could FLINT do a little more work to make the verification easier?), and have started formalising the required statements in LEAN. This has also led to identification of improvements to LEAN, and research questions in computer algebra. We are continuing to develop this ideas, in Davenport's case as a challenge to his students' LEAN study group.

References

1 James Harold Davenport. Proving an Execution of an Algorithm Correct? In Dubois and Kerber [2], pages 255–269.

2 Catherine Dubois and Manfred Kerber, editors. *Proceedings CICM 2023*, volume 14101 of *Springer Lecture Notes in Computer Science*, 2023.
3 Catherine Dubois and Manfred Kerber, editors. *Proceedings CICM 2023*, volume 14101 of *Springer Lecture Notes in Computer Science*, 2023.
4 F. Rabe and F. Weber. Morphism Equality in Theory Graphs. In C. Dubois and M. Kerber, editors, *Intelligent Computer Mathematics*, pages 174–189. Springer, 2023.

4.4 Coq, Isabelle and Dedukti as heterogeneous networks

Filip Koprivec (University of Ljubljana, SI), Mario Carneiro (Carnegie Mellon University – Pittsburgh, US), Stefania Dumbrava (ENSIIE – Paris, FR), Matej Petkovic (University of Ljubljana, SI), and Makarius Wenzel (Dr. Wenzel – Augsburg, DE)

The working group worked on pushing the entries (that correspond to formalized mathematical concepts) from Coq, Isabelle and Dedukti systems into graph databases, on which machine learning can be applied.

To unify the schema of the created networks, we
- adjusted the RDF triplets describing the Coq entries,
- wrote an extension of Dedukti parser that extracts the necessary information from abstract syntax trees of the entries,
- and wrote parsing tool for Isabelle XMLs.

The end goal was include this work to the series of the existing networks, extracted from the largest Agda libraries and Mathlib4 library of Lean.

We thank Talia Ringer and Catherine Dobuis for the ideas.

4.5 Object identification using invariant based decision trees

Filip Koprivec (University of Ljubljana, SI) and Matej Petkovic (University of Ljubljana, SI)

Object identification up to isomorphism is in general computationally expensive (group isomorphism, graph isomorphism...) and identifying a given object with a representative from a known dataset is difficult.

But often, one is presented with object together with few pre-computed invariants or some invariants, that can quickly be computed from a given representation. Most of the time, the difficulty of calculating the invariant can be approximated relatively well or one can use the timings produced as a side effect of constructing the whole database of objects and associated invariants.

The working group was mostly focused on the problems where objects are graphs or groups and the preliminary experiments were carried out on the dataset, representing graphs. Every graph (a row in the data table) was represented by the number of invariants (columns in the dataset). Some of the values in the table were missing, i.e., not all the variants were computed for all the graphs. The working group experimented with a modification of decision tree algorithm that uses both information gain of an invariant and its calculation difficulty to compute the next invariant on which to split the dataset. With such an approach, we can speed up the identification of unknown object in an existing database of objects.

The results of experimentation: implemented decision tree variant and data acquisition from house of graphs together with evaluation are available at `https://github.com/Petkomat/invariant-computation-trees`.

4.6 Datasets

Dimitri Leemans (UL – Brussels, BE), Katja Berčič (University of Ljubljana, SI), Jan Goedgebeur (KU Leuven, BE), Darij Grinberg (Drexel Univ. – Philadelphia, US), Samuel Lelievre (University Paris-Saclay – Orsay, FR), Harshit J Motwani (Ghent University, BE), and Tom Kaspar Wiesing (Universität Erlangen-Nürnberg, DE)

Participants in this working group worked on several smaller, but related tasks.

4.6.1 Cataloguing and/or indexing mathematical datasets.

1. Mathrepo: `https://mathrepo.mis.mpg.de/`, `https://ar5iv.labs.arxiv.org/html/2202.04022`
2. MathDB: `https://mathdb.mathhub.info/`

It may be possible to automate a part of the collection process by scraping Zenodo for mathematical datasets. This would probably require a list of mathematical concepts to be used in search queries. We identified a few considerations for catalogue/index projects:

- automatic uploading of datasets to Zenodo for long-term storage,
- accommodating reproducibility (recomputing the datasets, cf. Rescience),
- ratings and tags: independently recomputed, found same results; independently recomputed, found different results; not independently recomputed; formally certified.

4.6.2 Connecting databases and CAS.

Implementing an interface to the House of Graphs database in SageMath, including improving the House of Graph's API. The pull request can be viewed at `https://github.com/sagemath/sage/pull/36409`.

4.7 Aligning Mathematical Concepts Across Libraries

Florian Rabe (Universität Erlangen-Nürnberg, DE)

4.7.1 Motivation

Irrespective of the knowledge aspect (such as deduction, computation, data, or documentation), mathematical libraries share several abstract characteristics. Most importantly, they can be seen as a set of fragments each carrying a global identifier and describing a mathematical concept. Here we use *concept* as a generic term to subsume any named type, object, operation, theorem, or similar. For example, in a logical library, these are (mostly) definitions and theorems; in a computer algebra library, they are (mostly) type and function definitions; in a concrete data set, they are the entries of the data set, often rows in a table; and in a document library, they are the articles explaining a concept.

The work group recognized the *alignment problem* [5, 6] as a major roadblock to interoperability across mathematical software systems: The mathematical concepts introduced by the various libraries overlap substantially, but there is no systematic connection between the same mathematical concept described in different libraries. Concretely, we call a pair of identifiers, typically but not necessarily from different libraries, an *alignment* if both are descriptions of the same concept. Then the alignment problem can be formulated as the challenge of (a) collecting alignments for existing large mathematical libraries and (b) leveraging these alignments for knowledge interchange.

The group identified three levels at which the problem can be attacked:

- At the *identifier level*, the alignments are just pairs of identifiers without a machine-checkable guarantee that they correspond to each other in any way.
- At the *expression level*, an alignment (c, d) additionally carries information how terms with head c can be translated to terms with head d. This translation can be quite complex and involve, e.g., changing (which may require computation), adding (which may require inference), omitting, or reordering arguments.
- At the *semantic level*, the alignments are additionally verified for correctness. This can be done, e.g., by translating theorems about c along the alignment and proving the translated theorems in the system of d. Critically, from (i) to (iii), task (a) becomes harder while task (b) becomes easier. But even identifier level alignments are useful, e.g., to cross-reference across libraries or to search for the same query in multiple libraries in parallel.

4.7.2 Results

The group surveyed the available technologies and alignment collection efforts and concluded that, while semantic alignments must be the ultimate goal, only for identifier level alignments is a major community-driven collection effort feasible at this point.

The group compiled the following existing collections of identifier alignments and concept lists:

- the Math Subject Classification (MSC)
- the nLab page titles (`https://ncatlab.org/nlab/`)
- the SMGloM concept and translation library [2]
- the concept list for the undergraduate math curriculum and the alignments into Lean Mathlib (`https://github.com/leanprover-community/mathlib4/blob/master/docs/undergrad.yaml`)
- the concept translation library maintained by Hosgood (`https://thosgood.com/maths-dictionary/`)
- the manual alignments collected for theorem prover libraries in [7]
- the concept list used by SageMath to align computer algebra systems integrated with SageMath (`https://doc.sagemath.org/html/en/reference/categories/index.html`)
- the MathGloss alignments for undergraduate math education [4] (`https://mathgloss.github.io/MathGloss/`)
- the relevant subset of the Wikidata ontology, which includes various alignments to informal libraries
- the nNexus alignments across informal libraries [1]
- the semantic alignments between HOL systems found by machine learning in [3]

The work identified a set of several hundred concepts that can be used as a seed for an alignment library and started using the above resources to compile alignments for it. These resources are collected at `https://github.com/UniFormal/alignments/`.

In order to scalably collect, maintain, and leverage alignment sets in the future, the group makes two recommendations:

- All developers of math libraries should add a feature to their tool that allows tagging definitions with the aligned identifier in a central concept list. The build process of the library should generate the list of alignments between those central concepts and the tagged identifiers in the system's library. This list should be published alongside the library.
- Wikidata is suggested as the central concept list. This is motivated by the observation that Wikidata is a neutral library (in the sense of not being biased towards any research system or community) and the most likely to be scalably maintained and broadly used in the long term.

References

1 D. Ginev and J. Corneli. Nnexus reloaded. In S. Watt, J. Davenport, A. Sexton, P. Sojka, and J. Urban, editors, *Intelligent Computer Mathematics*, pages 423–426. Springer, 2014.

2 D. Ginev, M. Iancu, C. Jucovshi, A. Kohlhase, M. Kohlhase, A. Oripov, J. Schefter, W. Sperber, O. Teschke, and T. Wiesing. The smglom project and system: Towards a terminology and ontology for mathematics. In G. Greuel, T. Koch, P. Paule, and A. Sommese, editors, *Mathematical Software – ICMS*, volume 9725, pages 451–457. Springer, 2016.

3 T. Gauthier and C. Kaliszyk. Aligning concepts across proof assistant libraries. *Journal of Symbolic Computation*, 90:89–123, 2019.

4 L. Horowitz and V. de Paiva. Mathgloss: Linked undergraduate math concepts, 2023. `https://arxiv.org/abs/2311.12649`.

5 C. Kaliszyk, M. Kohlhase, D. Müller, and F. Rabe. A Standard for Aligning Mathematical Concepts. In A. Kohlhase, M. Kohlhase, P. Libbrecht, B. Miller, F. Tompa, A. Naummowicz, W. Neuper, P. Quaresma, and M. Suda, editors, *Work in Progress at CICM 2016*, pages 229–244. CEUR-WS.org, 2016.

6 D. Müller, T. Gauthier, C. Kaliszyk, M. Kohlhase, and F. Rabe. Classification of Alignments between Concepts of Formal Mathematical Systems. In H. Geuvers, M. England, O. Hasan, F. Rabe, and O. Teschke, editors, *Intelligent Computer Mathematics*, pages 83–98. Springer, 2017.

7 D. Müller, C. Rothgang, Y. Liu, and F. Rabe. Alignment-based Translations Across Formal Systems Using Interface Theories. In C. Dubois and B. Woltzenlogel Paleo, editors, *Proof eXchange for Theorem Proving*, pages 77–93. Open Publishing Association, 2017.

4.8 Scalability Estimates of Graph Certificates in a Theorem Prover Using SAT Encodings

Kathrin Stark (Heriot-Watt University – Edinburgh, GB), Madalina Erascu (West University of Timisoara, RO), Kazuhiko Sakaguchi (INRIA – Nantes, FR), and Jure Taslak (University of Ljubljana, SI)

4.8.1 Motivation

What is the minimum amount of information needed to certify that a graph has a specific property? For some properties, these certificates are easy to provide. For example, to prove that a graph *is* Hamiltonian, one only requires describing a Hamiltonian path. But for some properties, it is not obvious what an efficient certificate is. For example, to prove that a graph is *not* Hamiltonian, the obvious certificates all get extremely large. SAT encodings are a general way to encode properties, and it is well-known that most graph predicates can be encoded via SAT. But how well does this scale?

This working group consists of experts on SMT solvers, theorem provers, and CoqELPI (Madalina Erascu, Kazuhiko Sakaguchi, Kathrin Stark, Jure Taslak). The aim was to create a proof of concept for end-to-end verified certificates for graph properties to check their scalability.

4.8.2 Previous Work

Certificate checking of (Un)SAT problems has been used in several papers. SMTCoq (Ekici et al. 2017, Keller 2019) is a certified checker for proof witnesses for the SAT solver ZChaff and the SMT solvers veriT and CVC4. Lammich uses the GRAT certificate for a verified SAT solver in Isabelle. Cruz-Filipe et al. implemented two certified LRAT checkers, one of them for the Coq proof assistant.

This working group decided to start off with the work by Cruz-Filipe et al., as this was a relatively lightweight development that was created in a theorem prover the group members were familiar with. The theorem prover further offered the ability to define the SAT encoding manually, making it easier to test scalability.

4.8.3 Overview

For any given graph, the approach consists of three steps. First, an unverified Python program generates a SAT encoding of the desired graph property and solves the encoding using a state-of-the-art SAT solver. This SAT solver also generates an LRAT certificate to demonstrate the correctness of the result. Next, Cruz-Filipe et al.'s approach is used to validate the LRAT certificate. Finally, a separate (constructive) Coq proof shows that the SAT encoding indeed corresponds to the desired abstract graph property. The abstract graph property uses a naive encoding of graphs, paths, and connectivity/Hamiltonicity in Coq. It typically requires a proof by contradiction and hence requires the decidability of the respective property. Using CoqELPI, we could moreover show that the SAT encoding of Python and the one used in Coq coincide. Overall, the construction thus provides a proof of the given property while leaving most of the heavy computational lifting to an external, highly-optimized SAT solver.

For the scalability tests in this seminar we considered two graph properties: connectivity and Hamiltonicity. We randomly generated graphs through a simple python program and checked at which node count the checker suffered a segmentation fault. The tests included both graphs that have and that do not have the respective property.

4.8.4 Results

The first results provided a first idea of scalability and showed that the approach scaled up to graphs that were around an order of magnitude bigger than initially expected.

For a proof of concept, the working group started with certificates for connectivity. The process described above was shown to scale up to 2500 nodes (both for connectivity and the absence of connectivity). An end-to-end correctness proof (with the assumed dependencies) required around 300 lines of code. It seems likely that the code could be shortened significantly by using a suitable graph library.

For certificates of being non-Hamiltonian, the first approaches scaled up to 50 nodes. In the available time, no end-to-end proof was implemented.

4.8.5 Future Work

A full end-to-end proof for being non-Hamiltonian would be an obvious next step. Another interesting direction would be to take a deeper look at graph certificates via SAT encodings in the literature. For this proof of concept, we chose the simplest encoding for non-Hamiltonian graphs. A more efficient SAT encoding should improve scalability accordingly and extend to further graph properties.

Participants

- Andrej Bauer
University of Ljubljana, SI
- Christoph Benzmüller
Universität Bamberg, DE
- Katja Bercic
University of Ljubljana, SI
- Alex Best
King's College – London, GB
- James Boyd
Wolfram Institute – Cambridge, US
- Jacques Carette
McMaster University – Hamilton, CA
- Mario Carneiro
Carnegie Mellon University – Pittsburgh, US
- Edgar Costa
MIT – Cambridge, US
- James H. Davenport
University of Bath, GB
- Valeria de Paiva
Topos Institute – Berkeley, US
- Gilles Dowek
ENS – Gif-sur-Yvette, FR
- Catherine Dubois
ENSIIE – Evry, FR
- Stefania Dumbrava
ENSIIE – Paris, FR
- Madalina Erascu
West University of Timisoara, RO
- Jan Goedgebeur
KU Leuven, BE
- Kiran Gopinathan
National University of Singapore, SG
- Darij Grinberg
Drexel Univ. – Philadelphia, US
- Jyoti Jyoti
Panjab University – Chandigarh, IN
- Michael Kohlhase
Universität Erlangen-Nürnberg, DE
- Olexandr Konovalov
University of St Andrews, GB
- Filip Koprivec
University of Ljubljana, SI
- Dimitri Leemans
UL – Brussels, BE
- Samuel Lelievre
University Paris-Saclay – Orsay, FR
- Assia Mahboubi
INRIA – Nantes, FR
- Harshit J Motwani
Ghent University, BE
- Dennis Müller
Universität Erlangen-Nürnberg, DE
- Tobias Nipkow
TU München – Garching, DE
- Matej Petkovic
University of Ljubljana, SI
- Christian Pfrang
Bay. Min. für Digitales – München, DE
- Florian Rabe
Universität Erlangen-Nürnberg, DE
- Talia Ringer
University of Illinois – Urbana-Champaign, US
- Claudio Sacerdoti Coen
University of Bologna, IT
- Kazuhiko Sakaguchi
INRIA – Nantes, FR
- Natarajan Shankar
SRI – Menlo Park, US
- Kathrin Stark
Heriot-Watt University – Edinburgh, GB
- Jure Taslak
University of Ljubljana, SI
- Nicolas Thiéry
University Paris-Saclay – Orsay, FR
- Josef Urban
Czech Technical University – Prague, CZ
- Makarius Wenzel
Dr. Wenzel – Augsburg, DE
- Tom Kaspar Wiesing
Universität Erlangen-Nürnberg, DE