hausdorff CENTER FOR MATHEMATICS

UNIVERSITÄT BONN

Workshop on

“Mathematical Language and Practical Type Theory”

1 to 4 February 2020

organized by
Thomas Hales (Pittsburgh), Peter Koepke (Bonn)

List of Abstracts

Kevin Buzzard (Imperial College, London)

At the coal face

Abstract: I have been teaching undergraduate algebraic geometry this term in the mathematics department at Imperial College London. I will talk about my experience trying to persuade undergraduate mathematicians that formalising some of the material they are being taught is worthwhile, possible, and fun. I will also talk about the truth of the matter, which is that currently it is sometimes none of those things. Introducing a controlled natural language into the system as soon as possible will solve some of the problems I encounter.

Merlin Carl (Europa-Universität Flensburg)

Diproche - Automated Proof Checking for Didactical Applications

Abstract: We present the Diproche (Didactical Proof Checking) system (a software inspired by the Naproche (Natural language Proof Checking) system (developed by Koepke, Cramer, Schröder and others) which uses techniques from computational linguistics and automated theorem proving to provide an automated proof checker for proofs written in a fragment of natural mathematical language specifically designed to express solutions to typical proving exercises for beginner students. The aim of Diproche is to serve as an automatic tutor for students learning how to prove. In addition to reporting gaps in proofs, Diproche provides a detailed feedback on linguistic mistakes, failures to meet the goal of the exercise or self-formulated subgoals, type errors and diagnosis of possible formal fallacies. Moreover, users can obtain hints when getting stuck while searching for a solution.

Marcos Cramer (TU Dresden)

Analyzing Mathematical Language Logically and Linguistically: From Naproche to Frames

Abstract: In this talk I will report about my research on analyzing the specialist language of mathematics with a combination of methods from logic and linguistics. The talk will be mostly about the research that I conducted within the Naproche project as a PhD student in Bonn under the supervision of Prof. Peter Koepke. Within the Naproche project we developed a controlled natural

language (CNL) – i.e. a subset of a natural language defined through a formal grammar – for writing mathematical texts. The Naproche system can check the correctness of mathematical proofs written in this CNL. In this talk we will highlight some interesting logical and linguistic problems that had to be solved in the development of the Naproche CNL and the Naproche system: The system uses a formalism from the linguistic theory of presuppositions in order to work with potentially undefined terms, and can handle dynamic quantification, including the phenomenon of implicit dynamic function introduction. Additionally, at the end of the talk I will briefly introduce a new frame-based approach to the analysis of mathematical proof texts.

Tom Hales (University of Pittsburgh)

The formalization of mathematics

Abstract: A formal proof is a mathematical proof that has been checked by computer. The axioms and primitive rules of logic are programmed into a computer, and a proof is not regarded as verified until every step is exhaustively justified by first principles. Examples of proofs that have been formalized by various groups include the Kepler conjecture on sphere packings (2014), the independence of the Continuum Hypothesis (2019), and the Odd-order theorem in finite group theory (2012).
Buoyed up by these successful projects, we are exploring how these tools might bring general benefit to the mathematical community.

Bernhard Fisseni, Deniz Syrikaya (Leibniz-Institut für Deutsche Sprache, Mannheim; Universität Hamburg)

Frames for Mathematical Proofs

Abstract: Frames are a concept in knowledge representation that explains how the receiver, using background information, completes the information conveyed by the sender. This concept is used in different disciplines, most notably in cognitive linguistics and artificial intelligence. We argue that:

- Frames can serve as the basis for describing mathematical proofs.
- Specifically, using frames it is possible to model how mathematicians understand proofs that conform to proof patterns which have not been executed in a fully explicit way.
- Frames can be used to model both (textual) structural properties of proofs and ontological aspects of mathematical knowledge. This distinction is useful.

These claims will be exemplified by discussing inductive proofs and giving a formalization using frames. The application of frames to mathematics is joint work with Marcos Cramer and Bernhard Schröder.

Cezary Kaliszyk, Florian Rabe (University of Innsbruck, University Erlangen-Nuremberg)

Towards an Intuitive Language for Formalizing Mathematics

Abstract: Formalization languages and tools differ greatly from the ones familiar to practitioners, who predominantly use natural language with interspersed unrestricted formulas. Formalizing even basic concepts that are taken for granted in practice often requires significant, non-obvious encoding steps that impede scalability. This gap makes formalization difficult and expensive, sometimes to the point of harming more than helping.

We want to design ILF, a formal language for mathematical statements that provides an intermediate interface for bridging this gap. We envision ILF to be (i) structurally close to human-understandable natural language and thus universally applicable without requiring difficult encoding steps, and (ii) fully formal and machine-understandable to allows for easy integration with verification tools. Despite the obvious potential of such a language, no existing language comes close to that vision: Natural language lacks the formal syntax and semantics, existing intermediate languages lack support for complex type, module, and proof systems, and modern formalization languages are not easily and universally applicable.
ILF is a project in its infancy. In this talk we present some very preliminary ideas and moderate a discussion to collect feedback, design requirements, and interesting case studies and challenge problems.

Benedikt Löwe (Universität Hamburg)

Technology adoption in mathematical research: automated proof-checking as a case study

Abstract: The practical benefits of automated proof-checking as an integrated component of mathematical research are well known: a reduced chance of errors in papers, reducing the work-load of referees in the peer review process, the ability to check the correctness of long proofs, etc. And yet, there is no wide-spread adoption of this technology as an instrument in mathematical research.
In this talk, I would like to encourage the research community that uses methods from the empirical social sciences to study mathematical research practice (for lack of a better community label: ”philosophy of mathematical practice”) to consider this phenomenon in the context of general theories of technology adoption and the wider context of our understanding of mathematical research practice.

Andrei Paskevich (Université Paris-Sud)

ForTheL, a formal language in human clothes

Abstract: ForTheL is a formal language that imitates the natural syntax and structure of human mathematical texts. It is the input language of SAD, a proof assistant intended for automated proof verification. The main motivation behind the design of ForTheL and SAD was to push the mechanized mathematics and automated reasoning towards the natural style of presenting mathematical theories and proofs — not just with the purpose of lowering the entry barrier for non-experts in formalization, but also with the hope of leveraging the added ”richness” of the language to improve the automated treatment. In this talk, we present the syntax and semantics of ForTheL, and show that it allows for terse and comprehensible formal mathematics.

Aarne Ranta (University of Gothenburg)

From Logical to Grammatical Framework

Abstract: Logical Frameworks (LF) are formalisms in which different logics and mathematical theories can be defined in a declarative way. In computer implementations of logics, an LF saves work because many details are taken care of by the framework. However, the resulting logic may look unfamiliar to the user, as it encodes essentially the abstract syntax, typically consisting of LISP-like expressions such as (Conj A B) instead of A & B. To restore the familiar look of expressions, an LF can be extended by concrete syntax rules such as infix declarations and precedence levels. The Grammatical Framework (GF) takes this idea far enough to enable syntax similar to natural language.

The system named GF grew out of ALF (Another Logical Framework) in the 1990s in this way. Its first use case was to convert formalized theorems and proofs from ALF to English. It soon turned out that the one and the same abstract syntax could be rendered in multiple languages. This led to several applications of GF outside the mathematical domain, where LF has been used as a framework for translation interlinguas. The use cases range from software specifications to multilingual web shops. This talk will give a concise definition of GF as an extension of LF and an overview of recent applications, in particular ones related to formalized mathematics.

Bernhard Schröder (Universität Duisburg-Essen)

How proofs are told. Linguistic aspects of proof texts

Abstract: Proof texts show less ambiguity and less vagueness than virtually any other text genre. But for the correct understanding of proof texts the expert reader has to perform interpretation efforts comparable to other texts, most of which efforts usually remain unconscious. The inferences necessary for the interpretation require expert readers and are the main reason why a fully automatic interpretation of proof texts as appearing in text books and research papers is not (yet) feasible. The talk will discuss linguistic sources of interpretation hurdles in proof texts, requirements for a machine-interpretable controlled natural language for proofs, and issues of the effective cognitive organization of proof texts.

Sebastian Ullrich (Karlsruhe Institute of Technology)

Embedding languages into Lean 4

Abstract: One goal of the upcoming version (v4) of the Lean theorem prover is to break down the barrier between built-in and user-defined syntax by means of a new, fully extensible frontend implemented completely in the Lean language. Beyond merely supporting more complex mathematical notations, embedding full domain-specific languages on a high level of abstraction is now becoming a possibility. In this talk, I will discuss first experiments in that regard as well as the parser implementation and macro system underlying this new frontend.

Floris van Doorn (University of Pittsburgh)

Tactics in Lean

Abstract: Lean offers a powerful metaprogramming framework to write custom tactics and tools in the Lean language itself. In the Lean community these tactics and tools are part of the library, and are written as the need arises during the development of the library. The tools that have been written include domain-specific and general purpose automation tactics, commands to simplify routine tasks, and maintenance tools like semantic linting.

Konstantin Verchine (Paris)

How the system SAD was born in Kiev

Abstract: SAD means ”System for Automated Demonstration”. The project was initiated in the early 60th by the academician V. Glushkov under the name ”Evidence Algorithm” and was conceived as a computer assistant to working mathematicians. It quickly became clear that three components must be developed: (1) a convenient language to represent mathematical knowledge, (2) an efficient proof procedure, (3) tools, based on mathematical practice, which reinforce the proof procedure. I'll show some basic ideas that led to the creation of a version of SAD, particularly to the development of ForTeL language. An interesting question is: provided mathematics is an anthropomorphic science, are the "natural" features of the language syntactical sugar only?

Makarius Wenzel (Augsburg, https://sketis.net)

The Isar Proof Language after Two Decades

Abstract: This is a high-level overview of the Isar proof language, which has originally emerged around 1999–2001 for the Isabelle logical framework, and has been substantially revised in 2015/2016. Some aspects of the overall approach are revisited and put into historical perspective: What has worked well, what had to be changed later on, which questions are still pending? The discussion also takes other proof languages into account — notably SSReflect and Lean — that have been influenced by early Isar designs and might still benefit from its refinements in later years.