Proof theory and computational algebra

Thomas Powell
University of Bath

Dagstuhl Seminar 22702
New Perspectives in Symbolic Computation and Satisfiability Checking

15 February 2022

These slides will be available at
`https://t-powell.github.io/talks`

My research area in diagram:

$$\begin{tikzcd}
\boxed{\text{proof}} \arrow[r, dotted, "\textit{interpretation}"] & \boxed{\text{program}}
\end{tikzcd}$$

More precisely: Use of techniques from proof theory to extract programs that are interesting because they are either:

- New.
- Formally verified.

Some History

Where it all started

K. Goedel invented one of the first functional programming languages: System T.

$$\begin{tikzcd} \boxed{\mathrm{PA} \vdash 0 = 1} \arrow[r, dotted, "\textit{interpretation}"] & \boxed{\text{System T} \vdash 0 = 1} \end{tikzcd}$$

$$\text{Consistency}(\text{System T}) \implies \text{Consistency}(\text{PA})$$

The birth of applied proof theory

G. Kreisel (*A Survey of Proof Theory II, 1971*):

> *"What more do we know when we know that a theorem can be proved by limited means than if we merely know that it is true?"*

In other words, the **proof** of a theorem gives us much more information than the mere **truth** of that theorem.

Applied proof theory is a branch of logic that uses proof theoretic techniques to exploit this phenomenon.

$$\begin{tikzcd}
\boxed{\text{complex mathematical proof}} \arrow[r, dotted, "\textit{interpretation}"] & \boxed{\text{interesting/useful program}}
\end{tikzcd}$$

Everyone does applied proof theory

PROBLEM. Give me an upper bound on the nth prime number p_n.

1. What is p_n? I know it exists because of Euclid...
2. Specifically, given $p_1, \ldots, p_{n-1}$, I know that $N := p_1 \cdot \ldots \cdot p_{n-1} + 1$ contains a *new* prime factor q, and so $p_n \leq q \leq N$.
3. In other words, the sequence $\{p_n\}$ satisfies

$$p_n \leq p_1 \cdot \ldots \cdot p_{n-1} + 1 \leq (p_{n-1})^{n-1}$$

4. By induction, it follows that e.g. $p_n < 2^{2^n}$.

This is a simple example of applied proof theory in action! From the **proof** that there are infinitely many primes, we have inferred a **bound** on the nth prime.

… but it's not always that simple

Theorem (Littlewood 1914)

The functions of integers

(a) $\psi(x) - x$, *and*

(b) $\pi(x) - li(x)$

change signs infinitely often, where $\pi(x)$ *is the number of prime* $\leq x$, $\psi(x)$ *is the is logarithm of the l.c.m. of numbers* $\leq x$ *and* $li(x) = \int_0^x (1/\log(u))du$.

The original proof is utterly nonconstructive, using among other things a **case distinction on the Riemann hypothesis**. At the time, no numerical value of x for which $\pi(x) > li(x)$ was known.

In 1952, Kreisel analysed this proof and extracted recursive bounds for sign changes (*On the interpretation of non-finitist proofs, Part II*):

> *"Concerning the bound … note that it is to be expected from our principle, since if the conclusion … holds when the Riemann hypothesis is true, it should also hold when the Riemann hypothesis is nearly true: not all zeros need lie on* $\sigma = \frac{1}{2}$, *but only those whose imaginary part lies below a certain bound … and they need not lie on the line* $\sigma = \frac{1}{2}$, *but near it"*

Where we are today

Modern applied proof theory

A field in the intersection of formal logic, mathematics and computer science. Took off in the 1990s/early 2000s, and now an established area of research, roughly split into two branches:

1. Using proof theoretic methods to derive new theorems in mathematics (often known as “proof mining”)
2. Using formal version of these techniques to synthesise correct-by-construction programs.

1. “Proof mining”

Input. A proof of an existence theorem from a research paper.

Output. A strengthening of this theorem with additional information “hidden” in the proof, such as:

- An algorithm or computable bound for finding an object.
- A qualitative strengthening of the theorem e.g. abstract generalisation.

Success primarily in mathematical analysis, including:

- convex optimization
- approximation theory
- fixpoint theory
- ergodic theory

What proof mining looks like

Theorem (P. and Wiesnet, Numer. Func. Anal. Opt. 2021)

Suppose that $\{A_n\}$ is quasi asymptotically ψ-weakly contractive w.r.t. q and σ, and that the sequence $\{x_n\}$ satisfies

$$x_{n+1} = (1-\alpha_n)x_n + \alpha_n A_n x_n$$

for $\{\alpha_n\}$ a sequence of nonnegative reals such that $\sum_{n=0}^{\infty} \alpha_n = \infty$. Then whenever $\|x_n - q\|$ is bounded above by some $c > 0$, we have $\|x_n - q\| \to 0$, with rate of convergence

$$\|x_n - q\| \leq F^{-1}\left(2\Psi(c) - \sum_{i=0}^{n-2} \alpha_i\right)$$

where $F : (0,\infty) \to \mathbb{R}$ is any strictly increasing and continuous function satisfying

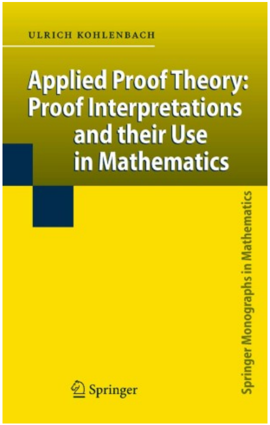
$$F(\varepsilon) \geq 2\Psi\left(\frac{\varepsilon}{2}\right) - \alpha \cdot \sigma\left(\frac{1}{2}\min\left\{\psi\left(\frac{\varepsilon}{2}\right), \frac{\varepsilon}{\alpha}\right\}, c\right)$$

and Ψ is given by

$$\Psi(s) := \int^{s} \frac{dt}{\psi(t)}$$

Resources

- A textbook for the field was published in 2008 (Kohlenbach, Springer):


ULRICH KOHLENBACH
Applied Proof Theory: Proof Interpretations and their Use in Mathematics
Springer
Springer Monographs in Mathematics

- For a survey of more recent results: Kohlenbach, *Proof-theoretic Methods in Nonlinear Analysis* Proc. ICM 2018, World Scientific 2019.

2. Program synthesis

Input. Formalised proof of $\exists x \in X \forall y \in Y\, A(x,y)$ where $A(x,y)$ is a specification relating input and output.

Output. An program $f : X \to Y$ satisfying $\forall x\, A(x, fx)$ which is

- Correct by construction.
- Often comes with a bound on its complexity.

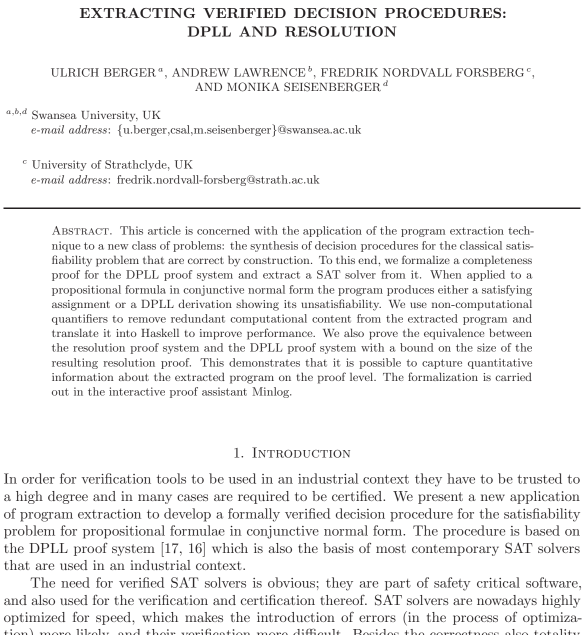
Major case studies carried out in:

- real number computation
- well-quasi order theory
- list sorting
- boolean satisfiability

In terms of proof assistants:

- A specialised proof system Minlog has been developed for extracting programs from proofs.
- Proof interpretations have also been implemented in Coq and Adga.

A relevant example: Berger at al, LMCS 2015



EXTRACTING VERIFIED DECISION PROCEDURES: DPLL AND RESOLUTION

ULRICH BERGER [a], ANDREW LAWRENCE [b], FREDRIK NORDVALL FORSBERG [c], AND MONIKA SEISENBERGER [d]

[a,b,d] Swansea University, UK
e-mail address: {u.berger,csal,m.seisenberger}@swansea.ac.uk

[c] University of Strathclyde, UK
e-mail address: fredrik.nordvall-forsberg@strath.ac.uk

Abstract. This article is concerned with the application of the program extraction technique to a new class of problems: the synthesis of decision procedures for the classical satisfiability problem that are correct by construction. To this end, we formalize a completeness proof for the DPLL proof system and extract a SAT solver from it. When applied to a propositional formula in conjunctive normal form the program produces either a satisfying assignment or a DPLL derivation showing its unsatisfiability. We use non-computational quantifiers to remove redundant computational content from the extracted program and translate it into Haskell to improve performance. We also prove the equivalence between the resolution proof system and the DPLL proof system with a bound on the size of the resulting resolution proof. This demonstrates that it is possible to capture quantitative information about the extracted program on the proof level. The formalization is carried out in the interactive proof assistant Minlog.

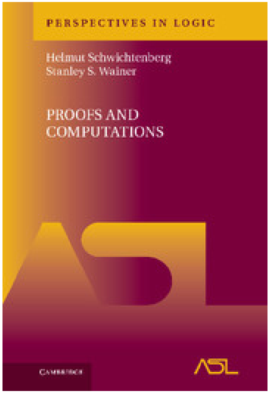
1. Introduction

In order for verification tools to be used in an industrial context they have to be trusted to a high degree and in many cases are required to be certified. We present a new application of program extraction to develop a formally verified decision procedure for the satisfiability problem for propositional formulae in conjunctive normal form. The procedure is based on the DPLL proof system [17, 16] which is also the basis of most contemporary SAT solvers that are used in an industrial context.

The need for verified SAT solvers is obvious; they are part of safety critical software, and also used for the verification and certification thereof. SAT solvers are nowadays highly optimized for speed, which makes the introduction of errors (in the process of optimization) more likely, and their verification more difficult. Besides the correctness also totality

Resources

- The much of the core theory is written up in a 2012 textbook (Schwichtenberg and Wainer, CUP):



- For recent results, see the Minlog homepage: `https://www.mathematik.uni-muenchen.de/~logik/minlog/index.php`

Proof Theory in Computational Algebra?

P., Schuster and Wiesnet, Inf. Comput. 2021

A universal algorithm for Krull's theorem

Thomas Powell[1], Peter Schuster[2], and Franziskus Wiesnet[2,3,4]

[1]Department of Computer Science, University of Bath
[2]Department of Computer Science, University of Verona
[3]Department of Mathematics, Università degli Studi di Trento
[4]Department of Mathematics, Ludwig-Maximilians Universität

April 23, 2021

Abstract

We give a computational interpretation to an abstract formulation of Krull's theorem, by analysing its classical proof based on Zorn's lemma. Our approach is inspired by proof theory, and uses a form of update recursion to replace the existence of maximal ideals. Our main result allows us to derive, in a uniform way, algorithms which compute witnesses for existential theorems in countable abstract algebra. We give a number of concrete examples of this phenomenon, including the prime ideal theorem and Krull's theorem on valuation rings.

Keywords: Krull's theorem, maximal ideals, program extraction, constructive algebra

1 Introduction

Krull's theorem for prime ideals is a fundamental result from abstract algebra. It can be formulated as follows: Let $F \subseteq R$ be an arbitrary subset of some commutative ring R. Then whenever $r \in R$ lies in the intersection of all prime ideals containing F, the element r also lies in the radical ideal $\sqrt{(F)}$ generated by F, or in other words:

$$\bigcap\{P \,:\, F \subseteq P \text{ and } P \text{ a prime ideal}\} \subseteq \sqrt{(F)}.$$

The standard proof of this fact appeals to Zorn's lemma. More specifically, we assume for contradiction that $r \notin \sqrt{(F)}$ and consider an ideal which is maximal among all ideals I such that $F \subseteq I$ but $r \notin I$. We conclude by demonstrating that this maximal ideal must be prime.

A simple example

Theorem

Let R be a commutative ring and $f = \sum_{i=0}^{d} a_i X^i$ and invertible element of $R[X]$. Then the coefficients a_i for $i > 0$ are all nilpotent.

Proof.

An application of Zorn's lemma. $\square$

In P., Schuster, Wiesnet 21 we extract from this proof a concrete program for computing an $e > 0$ such that

$$a_i^e = 0_R$$

Part of a general framework, a range of much more complex algorithms extracted, generally for reasoning about polynomials.

Algorithms based on building computable approximations to maximal ideals via trial-and-error.

Things I'd love to talk about

- Are there interesting proof systems suited to reasoning about proofs and programs in computer algebra, satisfiability checking? We need new proof systems for applied proof theory in abstract algebra. Is there some overlap?

- Can proof theoretic techniques be used as part of a verification strategy for algorithms in computer algebra, RAG, etc.? What is the state-of-the-art for formalising proofs in these areas?

- Are there interesting examples of nonconstructive proofs where proof interpretations can yield e.g.
  - numerical bounds
  - convergence rates
  - new algorithms?