Exploring Metamath Proof Structures: Progress Report

Christoph Wernhard[1] and Zsolt Zombori[2,3]

[1] University of Potsdam, Germany
[2] HUN-REN Alfréd Rényi Institute of Mathematics, Hungary
[3] Eötvös Loránd University, Budapest, Hungary

1 Formal Math as Grammar-Compressed Proof Terms

The very essence of formalized mathematics may be characterized as starting from a few axioms and using a few inference rules to derive theorems. A verifier can then check a given derivation and a theorem prover can search for a derivation of a given conjectured theorem. However, derivations can be very large and also, even for limited size, the number of possible derivations is very large. Thus, "something else" is assumed that allows to distinguish theorems of interest and to guide proof search. Forms of this "something else" are often sought outside of the axiomatic approach. In automated proving for example with heuristics, notions of redundancy and saturation, or coupling with machine learning. In interactive proving for example by assigning names to distinguished formulas. Here we consider a particular form of this "something else", which is already inherent in variations of the axiomatic approach: *compression of proof structures*. It provides a perspective on the structuring of proofs into manageable lemmas that is not driven by formulas but rather by proof structures considered as compressed trees. Lemma formulas come second, determined by substructures of the compression.

Metamath [15, 13, 14], a successful computer language for archiving, verifying, and studying mathematical proofs can be seen in this way. It continues a thread from Łukasiewicz and Tarski who investigate axiomatizations of propositional logics within a first-order meta-level framework [11], via Meredith, who refines this with his *condensed detachment* by the implicit use of most general unifiers instead of explicit substitutions and a view on proof structures as terms with a DAG representation [16, 17, 6, 12, 21, 27].

Metamath proofs are in essence grammar-compressions [9, 8] of proof terms built from just two primitive inference rules: condensed detachment (modus ponens with unification) and condensed generalization (quantifier introduction). A non-cyclic tree grammar with a single production for each nonterminal provides a compressed representation of a set of proof terms. Repeated *patterns*, such as the two occurrences of $\mathsf{g}(\mathsf{h}(\_))$ in $\mathsf{f}(\mathsf{g}(\mathsf{h}(\mathsf{a})), \mathsf{g}(\mathsf{h}(\mathsf{b})))$ can be factored by non-terminals with parameters, such as in the grammar $\{\mathsf{Start} \to \mathsf{f}(\mathsf{p}(\mathsf{a}), \mathsf{p}(\mathsf{b})),\ \mathsf{p}(V) \to \mathsf{g}(\mathsf{h}(V))\}$. The case with no parameter represents the factoring of a subtree as in a DAG.

With this form of proof term compression we can model the way in which mathematical knowledge is structured in *Metamath*. A grammar production corresponds to the proof of a theorem. The theorem formula is, from the first-order perspective (the "meta-level" at which mathematics is expressed in *Metamath*) a definite clause. The length of its body is the number of parameters of the corresponding production. The production's nonterminal serves as theorem name, and as function symbol in other proof terms where the theorem is used as lemma. Given a grammar, axiom clauses and a proof term built from nonterminals, axiom symbols and variables, a most general clause proven by the proof term can be uniquely determined.

The largest *Metamath* database is the *Metamath Proof Explorer*, also called *set.mm*, with about 44,000 theorems on various mathematical topics, developed with Zermelo-Fraenkel set theory. Such a knowledge base now appears as a single tree grammar with a production for each theorem. The expanded value of a nonterminal is a large tree whose inner nodes represent applications of the two primitive inference rules, and whose leaves represent instances of axioms.

2 Theses

Our work starts from the *observation* that condensed detachment (as related to structure-generating theorem proving [27]), *Metamath* proof structures [15] and grammar-based tree compression [8] combine in a natural way, with proof terms as central notion. This leads to a framework of striking simplicity, where a single representation mechanism covers proofs by automated systems as well as proofs by the interactive system, *Metamath*. Moreover, different abstraction levels appear as representations of the same structures, in the same representation mechanism, formally related through lossless compression. In general, this framework opens fresh perspectives on many issues that arise in applying automated first-order theorem proving on mathematical problems and in the integration of automated with interactive theorem proving accompanied by mathematical knowledge bases. Specifically, we investigate the following theses.

1. Compression of proof structures is a suitable approach to lemma synthesis. Effects on proof structure provide indicators for the quality and importance of a lemma.
2. For lemma synthesis not only compression "from scratch", i.e., of a fully expanded proof structure, can be useful, but also further compression applied to already compressed structures. These may stem from a human-made knowledge base, where lemma synthesis could suggest improvements on the structuring, and from automated systems, where, e.g., a structuring that is better suited for humans is sought.
3. The structuring of mathematical knowledge by human experts, exemplified with *Metamath* knowledge bases, is by itself worth systematic investigation for understanding human reasoning. One may ask how far the human structuring can be modeled with mechanical compression methods and which other systematic principles are behind it that are not justified by a compressing effect.
4. A mathematical knowledge base with proofs in the same format as that of automated systems is helpful in advancing automated theorem proving. Proofs from the knowledge base provide examples of the ultimately desired results of proof search. "Learning" from these, i.e., detecting and investigating their features and transferring the resulting insights into proof search seems necessary for substantial progress in automated theorem proving.

3 Towards Addressing the Theses

Establishing our theses in full would be an ambitious long-term project. Nevertheless we have developed a formal and implemented framework and used it to perform first exemplary experiments towards verifying them.

Formal Framework. We developed a "proof theory" that meets the requirements arising from our approach. It is a generalization of condensed detachment that is capable of representing *Metamath* proofs and also the introduction of new lemmas with compression techniques. Proofs are considered as explicit objects, proof terms. The "proves" relation between proof terms and formulas is specified by means of an inference system. A central concept is the notion of *most general theorem (MGT)*, the unique most general formula proven by a proof term, which is determined via unification. Condensed detachment is generalized to proof terms with parameters. The MGT is then a definite clause with a body atom for each parameter. Observed subtleties concerning parameters with multiple occurrences (nonlinear proof terms) are addressed. *Proof grammars* provide a grammar-compressed representation of proof terms. Each grammar production represents the proof of a lemma. MGTs corresponding to lemmas can be determined directly from the grammars, without need for decompression. We also take into account that *Metamath* allows theorems to be user-specified strict instances of their proof's MGT.

Implemented System. We extended *CD Tools*, which is written in *SWI-Prolog* [30], embedded in *PIE* [23, 24], and centered around experimenting with condensed detachment [25, 26, 18] with support for grammar-based tree compression and a *Metamath* interface. The central tree compression method is *TreeRePair* [9], which we implemented in *SWI-Prolog* operating on a DAG representation of the term to be compressed. Through *SWI-Prolog*'s internal structure sharing it handles terms of gigantic size, as long as their DAG size is moderate. As an alternative to grammars, *CD Tools* also supports a combinator [20, 3] representation of compressed proof terms, with mappings to and from the grammar representation. The *Metamath* interface allows to read-in *Metamath* databases such as *set.mm* and convert formulas as well as proofs into Prolog terms. For proofs, various formats are supported, including the format supported generally for condensed detachment proofs by *CD Tools*. To generate new inputs for *Metamath*, formulas can be printed in *Metamath* syntax and proofs can be supplemented by inferred "syntactic" proof parts that meet specified *disjoint variable restrictions* [13].[1]

First Exemplary Experiments. We started with inspecting the proof grammar for the first 60% of *set.mm*, its best curated part, which excludes deprecated material and user-specific contributions. We considered structural properties, including properties from the literature on grammar compression, as well as formula-related properties. Among the findings are that 28% of the productions are nonlinear, 12% of the productions increase the grammar size, 10% have no effect on the grammar size, for 8.4% the theorem statement is a strict instance of the MGT, and 3.9% of the theorem statements are redundant due to subsumption. We then applied a similar analysis to a small subset of *set.mm* that allowed to compare the human structuring of *set.mm* with a machine structuring "from scratch", obtained with a compression workflow involving *TreeRePair*, nonlinear compression and further techniques. Its result is 30% larger than the human compression, suggesting that our machine compression techniques can still be improved. An indicator for the usefulness of the lemmas found by machine is that for about one third of them their formula appears as a theorem in *set.mm*. In a further experiment, we apply the machine compression workflow to the given structuring of *set.mm*, broken into 18 large chunks, one for each mathematical topic. This machine compression applied on top of an existing human structuring achieved a further compression of 4–30%.

Related Works and Fresh Perspectives on Specific Issues. Variations of *grammar-based proof compression* are also considered in [22, 5], however applied to *formulas* involved in proofs, in contrast to *proof structures*. For *structuring of proofs by automated systems*, in [19, 4] it was observed that *structural* properties of proofs (in contrast to formula properties) provide the most useful indicators to identify distinguished lemmas. Some key properties described in [19, 4] turn out as DAG specializations of generic key properties of grammar compressions. In [7], an investigation of *premise selection*, it is observed that relevant lemmas are not only found among named theorems of a knowledge base corpus, but also among lemmas used implicitly in proofs. Moreover, lemmas can be identified at the level of "atomic" kernel inferences, leading to big data, and at the higher level of combinations of "tactics". As outlined above, the approach based on grammar-compressed proof structures integrates these levels, both representing the proof in full detail, and both using the same representation mechanism. Our implemented tools so far provide elements of a *hammer system* [1]. In contrast to the hammer system for *Metamath* described in [2], we operate on a direct formula translation, with the same result as the translation of [2] via higher-order logic. In our system, proof translation from a given grammar representation would not involve a size blowup as described for resolution proofs in [2].

[1] "Syntactic" proof parts are those that are only displayed by `metamath.exe` with the `/all` options. While optionally available in our *Metamath* interface, in the proof terms considered by us for condensed detachment and compression these are also omitted.

Further Information

The *CD Tools* system and code for additional experiments are available from http://cs.christophwernhard.com/cdtools/. An earlier stage of this work was presented at AITP 2024 [28]. For a more comprehensive presentation we refer to the recent report [29].

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 457292495. The authors would like to thank the Federal Ministry of Education and Research and the state governments (https://www.nhr-verein.de/unsere-partner) for supporting this work as part of the joint funding of National High Performance Computing (NHR). Funded by the Hungarian Artificial Intelligence National Laboratory Program (RRF-2.3.1-21-2022-00004) as well as the ELTE TKP 2021-NKTA-62 funding scheme. This article is based upon work from the action CA20111 EuroProofNet supported by COST (European Cooperation in Science and Technology).

References

[1] Blanchette, J.C., Kaliszyk, C., Paulson, L.C., Urban, J.: Hammering towards QED. J. Formaliz. Reason. **9**(1), 101–148 (2016). https://doi.org/10.6092/ISSN.1972-5787/4593

[2] Carneiro, M., Brown, C.E., Urban, J.: Automated theorem proving for Metamath. In: Naumowicz, A., Thiemann, R. (eds.) ITP 2023. LIPIcs, vol. 268, pp. 9:1–9:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2023). https://doi.org/10.4230/LIPIcs.ITP.2023.9

[3] Curry, H., Feys, R.: Combinatory Logic, vol. I. North-Holland (1958)

[4] Denzinger, J., Schulz, S.: Analysis and Representation of Equational Proofs Generated by a Distributed Completion Based Proof System. Seki-Report SR-94-05, Universität Kaiserslautern (1994), http://wwwlehre.dhbw-stuttgart.de/~sschulz/PAPERS/DS94-SR-94-05.ps.gz, revised September 1997

[5] Hetzl, S.: Applying tree languages in proof theory. In: Dediu, A.H., Martín-Vide, C. (eds.) LATA 2012. LNCS, vol. 7183, pp. 301–312 (2012). https://doi.org/10.1007/978-3-642-28332-1_26

[6] Hindley, J.R., Meredith, D.: Principal type-schemes and condensed detachment. Journal of Symbolic Logic **55**(1), 90–105 (1990). https://doi.org/10.2307/2274956

[7] Kaliszyk, C., Urban, J.: Learning-assisted theorem proving with millions of lemmas. J. Symb. Comput. **69**, 109–128 (2015). https://doi.org/10.1016/J.JSC.2014.09.032

[8] Lohrey, M.: Grammar-based tree compression. In: Potapov, I. (ed.) DLT 2015. LNCS, vol. 9168, pp. 46–57. Springer (2015). https://doi.org/10.1007/978-3-319-21500-6_3

[9] Lohrey, M., Maneth, S., Mennicke, R.: XML tree structure compression using RePair. Inf. Syst. **38**(8), 1150–1167 (2013). https://doi.org/10.1016/j.is.2013.06.006, system available from https://github.com/dc0d32/TreeRePair, accessed Jun 30, 2022

[10] Łukasiewicz, J.: Selected Works. North Holland (1970), edited by L. Borkowski

[11] Łukasiewicz, J., Tarski, A.: Untersuchungen über den Aussagenkalkül. Comptes rendus des séances de la Soc. d. Sciences et d. Lettres de Varsovie **23** (1930), English translation in [10], pp. 131–152

[12] McCune, W., Wos, L.: Experiments in automated deduction with condensed detachment. In: Kapur, D. (ed.) CADE-11. LNCS (LNAI), vol. 607, pp. 209–223. Springer (1992). https://doi.org/10.1007/3-540-55602-8_167

[13] Megill, N., Wheeler, D.A.: Metamath: A Computer Language for Mathematical Proofs. lulu.com, second edn. (2019), online https://us.metamath.org/downloads/metamath.pdf

[14] Megill, N.D.: Home Page – Metamath, online: https://us.metamath.org/, accessed Jan 10, 2025

[15] Megill, N.D.: A finitely axiomatized formalization of predicate calculus with equality. Notre Dame J. of Formal Logic **36**(3), 435–453 (1995). https://doi.org/10.1305/ndjfl/1040149359

[16] Prior, A.N.: Logicians at play; or Syll, Simp and Hilbert. Australasian Journal of Philosophy **34**(3), 182–192 (1956). https://doi.org/10.1080/00048405685200181

[17] Prior, A.N.: Formal Logic. Clarendon Press, Oxford, 2nd edn. (1962). https://doi.org/10.1093/acprof:oso/9780198241560.001.0001

[18] Rawson, M., Wernhard, C., Zombori, Z., Bibel, W.: Lemmas: Generation, selection, application. In: Ramanayake, R., Urban, J. (eds.) TABLEAUX 2023. LNAI, vol. 14278, pp. 153–174 (2023). https://doi.org/10.1007/978-3-031-43513-3_9

[19] Schulz, S.: Analyse und Transformation von Gleichheitsbeweisen. Projektarbeit in informatik, Fachbereich Informatik, Universität Kaiserslautern (1993), http://wwwlehre.dhbw-stuttgart.de/~sschulz/PAPERS/Sch93-project.ps.gz, (German Language)

[20] Schönfinkel, M.: Über die Bausteine der mathematischen Logik. Math. Ann. **92**(3–4), 305–316 (1924). https://doi.org/10.1007/BF01448013

[21] Ulrich, D.: A legacy recalled and a tradition continued. J. Autom. Reasoning **27**(2), 97–122 (2001). https://doi.org/10.1023/A:1010683508225

[22] Vyskocil, J., Stanovský, D., Urban, J.: Automated proof compression by invention of new definitions. In: Clarke, E.M., Voronkov, A. (eds.) LPAR-16. LNCS, vol. 6355, pp. 447–462. Springer (2010). https://doi.org/10.1007/978-3-642-17511-4_25

[23] Wernhard, C.: The PIE system for proving, interpolating and eliminating. In: Fontaine, P., Schulz, S., Urban, J. (eds.) PAAR 2016. CEUR Workshop Proc., vol. 1635, pp. 125–138. CEUR-WS.org (2016), http://ceur-ws.org/Vol-1635/paper-11.pdf

[24] Wernhard, C.: Facets of the PIE environment for proving, interpolating and eliminating on the basis of first-order logic. In: Hofstedt, P., et al. (eds.) DECLARE 2019. LNCS (LNAI), vol. 12057, pp. 160–177 (2020). https://doi.org/10.1007/978-3-030-46714-2_11

[25] Wernhard, C.: Generating compressed combinatory proof structures — an approach to automated first-order theorem proving. In: Konev, B., Schon, C., Steen, A. (eds.) PAAR 2022. CEUR Workshop Proc., vol. 3201. CEUR-WS.org (2022), https://arxiv.org/abs/2209.12592

[26] Wernhard, C.: Structure-generating first-order theorem proving. In: Otten, J., Bibel, W. (eds.) AReCCa 2023. CEUR Workshop Proc., vol. 3613, pp. 64–83. CEUR-WS.org (2024), https://ceur-ws.org/Vol-3613/AReCCa2023_paper5.pdf

[27] Wernhard, C., Bibel, W.: Investigations into proof structures. J. Autom. Reasoning **68**(24) (2024). https://doi.org/10.1007/s10817-024-09711-8

[28] Wernhard, C., Zombori, Z.: Exploring Metamath proof structures (extended abstract). In: Douglas, M.R., Hales, T.C., Kaliszyk, C., Schulz, S., Urban, J. (eds.) 9th Conference on Artificial Intelligence and Theorem Proving, AITP 2024 (Informal Book of Abstracts) (2024), http://aitp-conference.org/2024/abstract/AITP_2024_paper_14.pdf

[29] Wernhard, C., Zombori, Z.: Mathematical knowledge bases as grammar-compressed proof terms: Exploring Metamath proof structures. CoRR **abs/2505.12305** (2025). https://doi.org/10.48550/arXiv.2505.12305

[30] Wielemaker, J., Schrijvers, T., Triska, M., Lager, T.: SWI-Prolog. Theory and Practice of Logic Programming **12**(1-2), 67–96 (2012). https://doi.org/10.1017/S1471068411000494