# Generating Theorems by Generating Proof Structures

Christoph Wernhard

University of Potsdam, Germany

**The Addressed Task: Generating Theorems.** We address generating theorems, together with proofs, from a given set of axioms, without proof goal, aiming at value from a mathematical point of view or as lemmas for automated proving. This occurs as subtask in automated proving and is also a classic objective of automated reasoning, studied, e.g., with *Scott's challenge* [35]. Modern variations include lemma generation [19], premise selection, and, more generally, learning and AI methods for automated reasoning [5].

**Technical Approach: Generating Proof Terms.** Our techniques are based on proof terms and condensed detachment. On the one hand this links to *Metamath* [16] and formulas-as-types; on the other hand it ties in with proof search through enumeration of proof structures, interwoven with formula unification, as in the connection method [3, 4] and in clausal tableaux [13]. Redundancies are avoided by having each structure (proof) only once in the enumeration, and by eliminating sets of structures early on through failure of unification for a substructure. Commonly, the approach is employed "top-down", utilizing the unifiability restriction with a ground-instantiated goal theorem. Compared to saturation-based provers, this has the serious drawback that proven subgoals are not re-used in *fresh copies*. However, proof structures can also be enumerated "bottom-up" without a given goal [22, 19, 28], and as DAGs with copying formulas for each incoming edge [8, 27]. We utilize this for generating theorems by generating proof terms, based on *inductive level characterizations*.

**Inductive Level Characterizations.** A *level characterization* is the specification of a partition of the set of all proof terms into a series $L_0, L_1, L_2, \ldots$ of disjoint subsets, called *levels*. Utilizing a level characterization to organize practical reasoning is facilitated by a so-called *inductive level characterization*, an inductive specification that can be read in two ways as high-level specification of an algorithm to generate proof terms level-by-level: (1) Top-down, embedded in iterative deepening, and (2) bottom-up. In both cases, the practical generation of levels is interwoven with computing via formula unification *most general theorems (MGTs)* proven by the proof terms. Proof terms where this unification fails, i.e., which have no defined MGT (are not *well-typed*, in the formulas-as-types view) are excluded early on. Bottom-up processing on the basis of inductive level characterizations shares with saturation-based provers two key features that are of general importance for proving and generating theorems.

1. *Caching* to avoid recomputations.
2. A certain *simple way to incorporate restrictions*: The actually computed levels can be restricted to proof terms that meet heuristic criteria, e.g., a limit on the size of the MGT.

The *compacted size* (or *DAG-size*) of a proof term is the number of inner nodes of the unique minimal DAG that represents the term. Level generation by compacted size, i.e., such that level $L_i$ is the set of proof terms with compacted size $i$, seems appealing for two reasons:

1. The compacted size of a proof term is often significantly smaller than its tree size (the number of inner nodes of the term as tree).
2. Generation by compacted size justifies incorporating *combinators*[1] like axioms into the proof term enumeration [27]. The combinators restructure the proof term such that new possibilities of factoring shared subterms emerge, reducing the compacted size. At the same time, they can be handled in determining MGTs like ordinary axioms.

[1] In the sense of combinatory logic [23, 6]. For a practice-oriented introduction see [18].

Differently from tree size or height, an *inductive* level characterization by *compacted size* seems not straightforward. However, if we give up completeness, i.e., accept that there are proof terms that are not in any $L_i$, the *PSP* (for *Proof-SubProof*) *level* [31, 32] provides an inductive level characterization by compacted size. Generation by PSP level was applied to solve an ATP challenge problem [19]. Bottom-up (and also top-down) proof term generation with inductive level characterizations is implemented in our highly configurable system *SGCD* [28, 29].

**A Benchmark Extracted from *set.mm* and Two Novel Techniques.** As benchmark theorem set we extract a foundational fragment from the *Metamath* Database *set.mm*[2] [14, 15]: From the best-curated two-thirds (28,673 theorems) we take those 1,374 theorems that depend only on the axiomatization of classical propositional logic with three axioms. We identify 554 of these theorems as "easy-to-generate", which means, proofs for them appear in the outputs of straightforward heuristically restricted bottom-up runs of *SGCD* on the three axioms. With two novel techniques, we generate proofs for 180 more of these theorems:

1. *Lemma synthesis via DAG compression of proof terms.* The underlying hypothesis is that a formula is a valuable lemma if it is proven with a proof term that has a significant compressing effect on some set of previously generated proof terms. We start with a set of proof terms obtained by straightforward heuristically restricted bottom-up runs of *SGCD* on the three axioms. We apply DAG compression to the set and select proof terms from the DAG, basically according to their compressing effect. In a subsequent run of *SGCD* the axioms are supplemented with the MGTs of these selected proof terms.
2. *Combinatory proof schemas.* These allow to incorporate combinators into proof terms, similar to axioms, but for their compressing effect, and in a restricted way such that only specified combinators in specified contexts can become constituents of proof terms.

For several benchmark theorems for which we could generate proofs, the proving problem is hard for provers. Supplied to provers, our lemmas obtained with technique 1 significantly increase solution rates, e.g., for *Vampire* [12] from 74% to 94%, and for *leanCoP* [17] from 7% to 44%.

**Related Work and Outlook.** Our approach is centered on *proof structures* as terms that are enumerated and compressed, differently from most works with similar aims, which are centered on formulas, e.g., [35, 25, 9, 1, 2]. An early exception is [21, 7], where the compressing effect of subproofs in a proof DAG serves to identify important steps. This and other measures based on proof graphs were later investigated with large *HOL light* corpora for lemma synthesis and premise selection [10].

Although our experiments are only with the axioms for propositional logic from *set.mm*, the techniques and their implementation are ready to be applied with other axioms, arbitrary first-order Horn formulas. The modeling of predicate logic in *set.mm* seems related to equality reasoning, raising the question whether inductive level characterizations allow here to transfer techniques from saturation-based provers. Overall, essential for our approach are just proof terms with an MGT. This should also work for resolution, if proofs are constrained or transformed such that final steps are for goal clauses ([24], [11, Lemma 4.1.3]). Another way towards resolution would be simulating it through combinators, as indicated in [27] and related to [8].

For machine learning approaches to ATP that are based on proof structures, e.g., [19, 20, 33], our goal is to provide a foundation, implemented infrastructure, and "symbolic yardstick".

**Artifacts and Full Paper.** The presented techniques are implemented in *SWI-Prolog* [34] with *CD Tools* [26]. Code for the experiments and further artifacts, e.g., problem files in TPTP format, are accessible from `http://cs.christophwernhard.com/cdtools/exp-thgen/`. A full version of this work is published as [29] (preprint with supplementary appendices: [30]).

[2] `https://github.com/metamath/set.mm`, from July 5, 2025 (commit 011fba3).

**Acknowledgments.** Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 457292495. The author gratefully acknowledges the computing time granted by the Resource Allocation Board and provided on the supercomputer Emmy/Grete at NHR-Nord@Göttingen as part of the NHR infrastructure. The calculations for this research were conducted with computing resources under the project `nhr_bb_test_proof_structures`.

# References

[1] Alhessi, Y., Einarsdóttir, S.H., Granberry, G., First, E., Johansson, M., Lerner, S., Smallbone, N.: Lemmanaid: Neuro-symbolic lemma conjecturing. CoRR **abs/2504.04942** (2025). https://doi.org/10.48550/ARXIV.2504.04942

[2] Axelrod, G., Dubhashi, D., Johansson, M., Silvi, A., Smallbone, N., , Stucki, S.: Learning to generate abstractions for an equational solver. In: Douglas, M.R., Hales, T.C., Kaliszyk, C., Schulz, S., Urban, J. (eds.) AITP 2025 (Informal Book of Abstracts) (2025), online: https://aitp-conference.org/2025/abstract/AITP_2025_paper_18.pdf, accessed May 8, 2026

[3] Bibel, W.: Automated Theorem Proving. Vieweg (1987), first edition 1982

[4] Bibel, W., Otten, J.: From Schütte's formal systems to modern automated deduction. In: Kahle, R., Rathjen, M. (eds.) The Legacy of Kurt Schütte, chap. 13, pp. 215–249. Springer (2020). https://doi.org/10.1007/978-3-030-49424-7_13

[5] Blaauwbroek, L., Cerna, D.M., Gauthier, T., Jakubuv, J., Kaliszyk, C., Suda, M., Urban, J.: Learning guided automated reasoning: A brief survey. In: Capretta, V., Krebbers, R., Wiedijk, F. (eds.) Logics and Type Systems in Theory and Practice - Essays Dedicated to Herman Geuvers on The Occasion of His 60th Birthday. LNCS, vol. 14560, pp. 54–83. Springer (2024). https://doi.org/10.1007/978-3-031-61716-4_4

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

[7] 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

[8] Eder, E.: A comparison of the resolution calculus and the connection method, and a new calculus generalizing both methods. In: Börger, E., Kleine Büning, H., Richter, M.M. (eds.) CSL '88. LNCS, vol. 385, pp. 80–98. Springer (1989). https://doi.org/10.1007/BFb0026296

[9] 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

[10] 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

[11] Kleine Büning, H., Lettmann, T.: Propositional Logic – Deduction and Algorithms. Cambridge University Press (1999)

[12] Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer (2013). https://doi.org/10.1007/978-3-642-39799-8_1

[13] Letz, R.: Tableau and Connection Calculi. Structure, Complexity, Implementation. Habilitationsschrift, TU München (1999), https://web.archive.org/web/20230604101128/https://www2.tcs.ifi.lmu.de/~letz/habil.ps, accessed May 8, 2026

[14] 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

[15] Megill, N.D.: Home Page – Metamath, online: https://us.metamath.org/, accessed May 8, 2026

[16] 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

[17] Otten, J.: Restricting backtracking in connection calculi. AI Communications **23**(2-3), 159–182 (2010). https://doi.org/10.3233/AIC-2010-0464

[18] Peyton Jones, S.L.: The Implementation of Functional Programming Languages. Prentice Hall (1987), https://simon.peytonjones.org/slpj-book-1987/

[19] 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

[20] Rawson, M., Zombori, Z., Doré, M., Wernhard, C.: Project proposal: Forward reasoning in hindsight. In: Douglas, M.R., Hales, T.C., Kaliszyk, C., Schulz, S., Urban, J. (eds.) AITP 2024 (Informal Book of Abstracts) (2024), online: https://aitp-conference.org/2024/abstract/AITP_2024_paper_6.pdf, accessed May 8, 2026

[21] 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)

[22] Schumann, J.M.P.: DELTA — A bottom-up preprocessor for top-down theorem provers. In: CADE-12. LNCS (LNAI), vol. 814, pp. 774–777. Springer (1994). https://doi.org/10.1007/3-540-58156-1_58

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

[24] Slagle, J.R., Chang, C., Lee, R.C.T.: Completeness theorems for semantic resolution in consequence-finding. In: Walker, D.E., Norton, L.M. (eds.) IJCAI'69. pp. 281–286. William Kaufmann (1969), http://ijcai.org/Proceedings/69/Papers/028.pdf

[25] 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

[26] Wernhard, C.: CD Tools, online: http://cs.christophwernhard.com/cdtools/, accessed May 8, 2026

[27] 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

[28] 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

[29] Wernhard, C.: Generating theorems by generating proof structures. In: Biere, A., Lutz, C., Negri, S. (eds.) IJCAR 2026. pp. 41–61. LNCS (LNAI), Springer (2026). https://doi.org/10.1007/978-3-032-32589-1_3

[30] Wernhard, C.: Generating theorems by generating proof structures (extended version). CoRR **abs/2602.15511** (2026), https://arxiv.org/abs/2602.15511

[31] Wernhard, C., Bibel, W.: Learning from Łukasiewicz and Meredith: Investigations into proof structures. In: Platzer, A., Sutcliffe, G. (eds.) CADE 28. LNCS (LNAI), vol. 12699, pp. 58–75. Springer (2021). https://doi.org/10.1007/978-3-030-79876-5_4

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

[33] Wernhard, C., Zombori, Z.: Are proof structures machine-learnable? a study with dag-structured proof terms (2026), in preparation

[34] 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

[35] Wos, L.: The resonance strategy. Computers Math. Applic. **29**(2), 133–178 (1995). https://doi.org/10.1016/0898-1221(94)00220-F