Generating Theorems by Generating Proof Structures

Christoph Wernhard

University of Potsdam

AITP 2026

Aussois, September 3, 2026

Generating theorems together with proofs from given axioms

- Without given goal theorem
- Aiming at theorems that are
  - “Mathematically interesting”, or
  - Useful as lemmas for first-order provers
- A subtask of first-order ATP
- A classic objective of automated reasoning, with modern variations:
  - Lemma generation
  - Premise selection
  - Learning and AI methods for automated reasoning

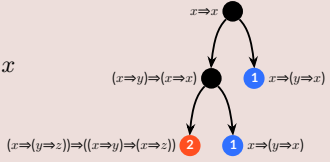
Condensed detachment
Modus ponens + unification [*Meredith, 1956*]
Basic simple type theory [*Hindley, 1990*]
Developments in ATP [*Wos, McCune, Veroff, 1990s*]

First-order proving by enumerating proof structures
[*Connection method, clausal tableaux*]
[*W, Bibel, 2021*]

First-order Horn logic as meta layer on axiomatized object logics

Proof terms that prove a most general theorem

$\mathsf{D}(\mathsf{D}(2,1),1) : x \Rightarrow x$



Proof assistant, mathematical library
Metamath Proof Explorer (set.mm)
44.000 theorems

Proof compression, breaking proofs into lemmas: DAG and tree grammar compression of proof structures

Condensed Detachment in a Nutshell

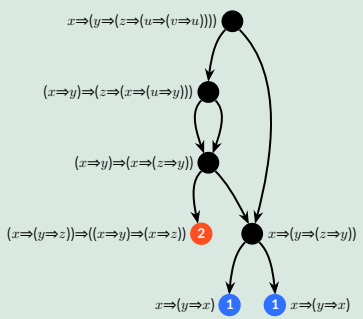
- **First-order meta level** with a unary predicate P (“*provable*”)
- **Object-level formulas: first-order terms**
- **Proof terms**:
  $$d, e := AxiomId \mid \mathsf{D}(d, e)$$
- **The most general theorem (MGT)** of
  - $AxiomId$ is $\mathsf{P}(AxiomFormula)$
  - $\mathsf{D}(d, e)$ is the **hyperresolvent** of
    $$\mathsf{P}(y) \leftarrow (\mathsf{P}(x \Rightarrow y) \wedge \mathsf{P}(x))$$
    with $\mathsf{MGT}(d)$ and $\mathsf{MGT}(e)$
- The MGT may be **undefined** (if unification fails)

Axiom system of *set.mm* for classical propositional logic

Id	*Axiom*	*Names*
1	$x \Rightarrow (y \Rightarrow x)$	**K**, *Simp*
2	$(x \Rightarrow (y \Rightarrow z)) \Rightarrow ((x \Rightarrow y) \Rightarrow (x \Rightarrow z))$	**S**, *Frege*
3	$(\mathsf{n}(x) \Rightarrow \mathsf{n}(y)) \Rightarrow (y \Rightarrow x)$	*Transp*

Example proof term

$\mathsf{D}(\mathsf{D}(2, \mathsf{D}(1, 1)), \mathsf{D}(2, \mathsf{D}(1, 1))), \mathsf{D}(1, 1))$



The proof term as DAG grammar

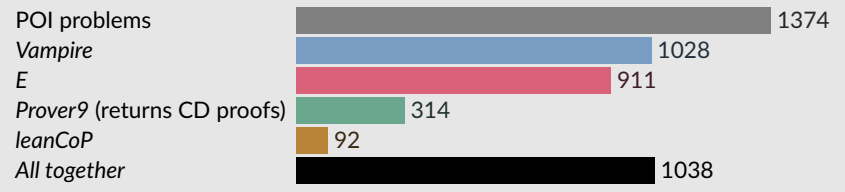
$$\begin{aligned} 4 &\rightarrow \mathsf{D}(1, 1) \\ 5 &\rightarrow \mathsf{D}(2, 4), \\ \mathit{Start} &\rightarrow \mathsf{D}(\mathsf{D}(5, 5), 4) \end{aligned}$$

Hypothesis: The **"mathematically interesting"** theorems that follow from the 3 axioms for propositional logic are the **theorems in *set.mm***

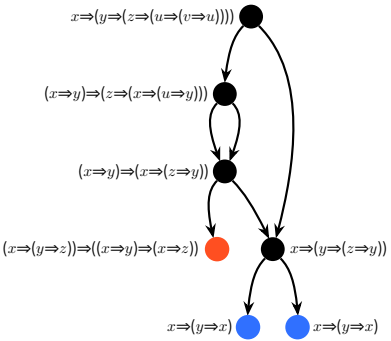
We **extract these from *set.mm*** and perform some standardization to obtain

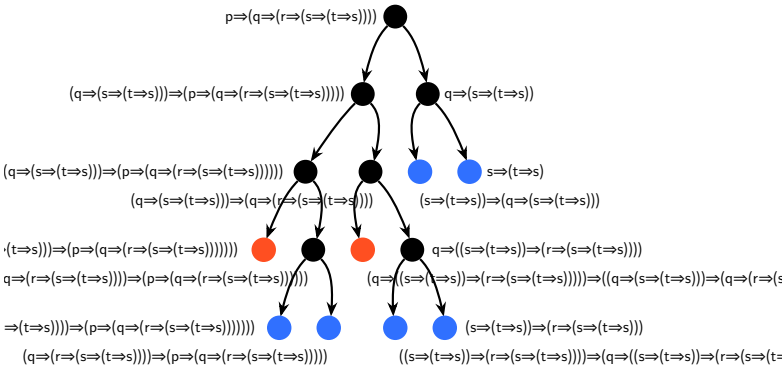
the set of the **1,374 *POI* theorems** (*Propositional-Only Implication form*)

Can **first-order provers** prove POI theorems from the axioms and the detachment clause?

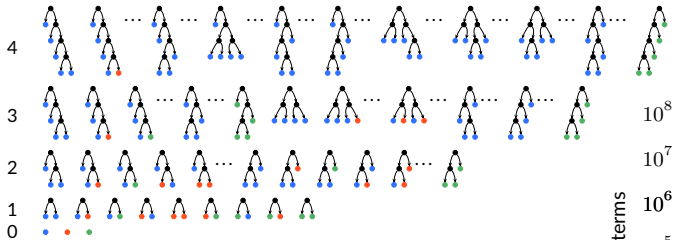


- Typical: **“top-down” for instantiated goal, rigid variables**
- **Interwoven with unification of formulas**: eliminating sets of structures early on through failure of unification for a substructure
- **Each structure (proof) only once** in the enumeration





Generating Proof Terms in Levels by Tree Size

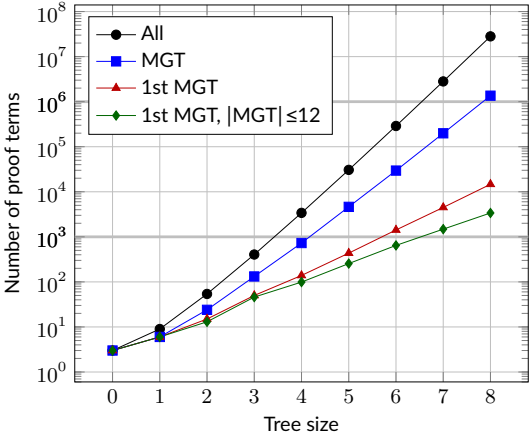


Inductive Level Characterization by Tree Size

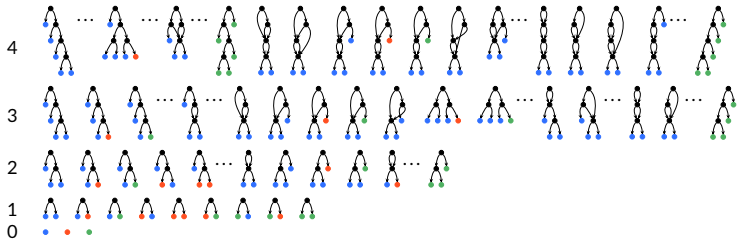
$$\mathcal{T}_0 \stackrel{\text{def}}{=} Ax$$
$$\mathcal{T}_{n+1} \stackrel{\text{def}}{=} \{\mathsf{D}(d,e) \mid \exists i{\leq}n \,.\, d \in \mathcal{T}_i,\ e \in \mathcal{T}_{n-i}\}$$

Suggests “bottom-up” algorithm, which supports

1. **Caching**
2. **Incorporation restrictions**
   - Proof terms with MGT (via incorporating unification)
   - Keep just 1st found of proof terms with same MGT
   - Limit to proof terms with MGT size ≤ 12

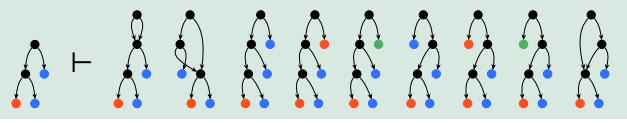


Generating Proof Terms by Compacted Size (DAG Size) and by PSP Level

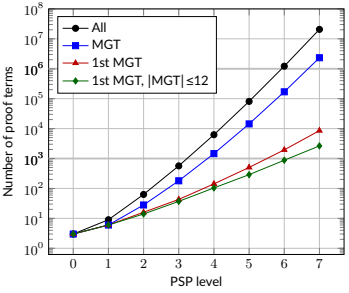


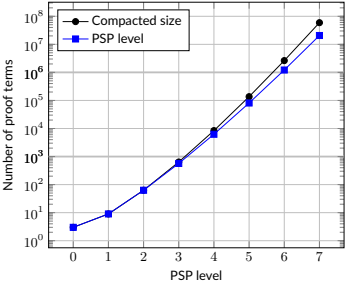
Inductive Level Characterization by PSP (*Proof-SubProof*) Level

$$\begin{aligned}\mathcal{P}_0 &\stackrel{\text{def}}{=} Ax\\ \mathcal{P}_{n+1} &\stackrel{\text{def}}{=} \{\mathsf{D}(d,e) \mid d \in \mathcal{P}_n,\ e \in \mathsf{Subterms}(d) \cup Ax\}\\ &\quad \cup \{\mathsf{D}(e,d) \mid d \in \mathcal{P}_n,\ e \in \mathsf{Subterms}(d) \cup Ax\}\end{aligned}$$



- Incomplete variation of compacted size
- Supports **caching** and **restrictions** through inductive level characterication



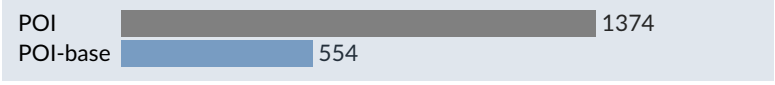


POI Base: An Easy-to-Generate Subset of POI

Idea: Take as **basis for experiments** those POI theorems that are **"easy to generate"** by straightforward proof term enumeration with straightforward heuristic restrictions

System: ***SGCD*** (*Structure-Generating theorem proving for Condensed Detachment*)
Part of ***CD Tools***, written in **SWI-Prolog**

Define ***POI-base*** as the set of those 554 *POI* theorems for which proofs are in the results of 7 "straightforward" *SGCD* runs



POI-base includes theorems of

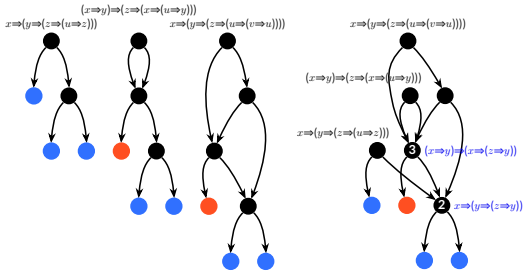
- all 92 problems proven by *leanCoP*
- all 314 problems proven by *Prover9*

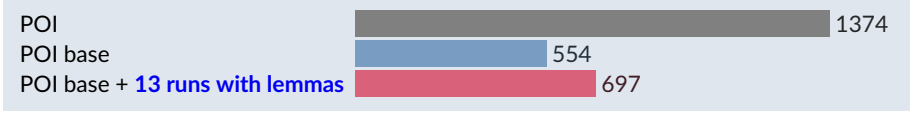
Hypothesis: A formula is a **valuable lemma** if it is proven by a proof term with a **significant compressing effect** on some **large set of proof terms** with defined MGT

$$
\begin{aligned}
4 &\rightarrow \mathrm{D}(1,1) && \text{save value: } 2\\
5 &\rightarrow \mathrm{D}(2,4) && \text{save value: } 3\\
\mathit{Start} &\rightarrow \mathrm{D}(1,4)\\
\mathit{Start} &\rightarrow \mathrm{D}(5,5)\\
\mathit{Start} &\rightarrow \mathrm{D}(5,\mathrm{D}(5,4))
\end{aligned}
$$

Lemma Synthesis Workflow

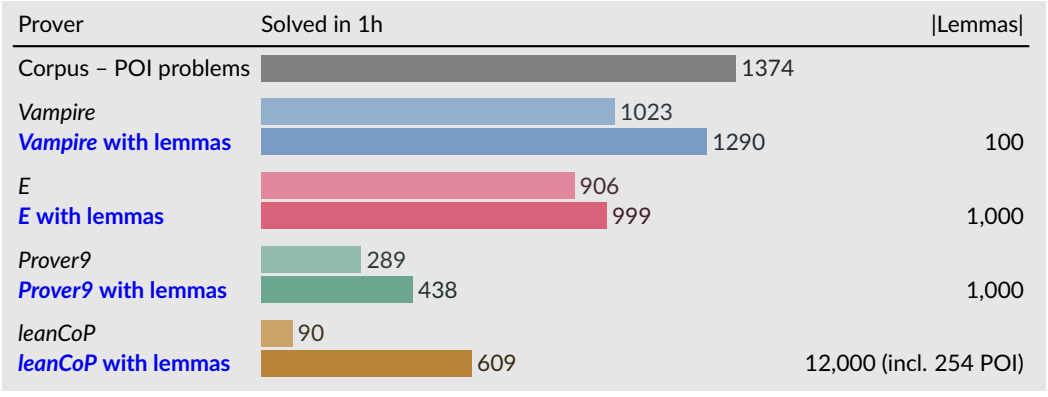
1. Generate a **base set** of a million proof terms with *SGCD*
2. Compress the base set into its **minimal DAG**, represented as **DAG grammar**
3. **Order** productions according to **save value** and MGT size
4. Perform subsumption reduction wrt. MGTs
5. Take a **prefix with configured length** (100–12,000)
   - Expanded productions: proof terms
   - Their MGTs: lemma theorems
6. Run *SGCD* again **with the MGTs as additional axioms**





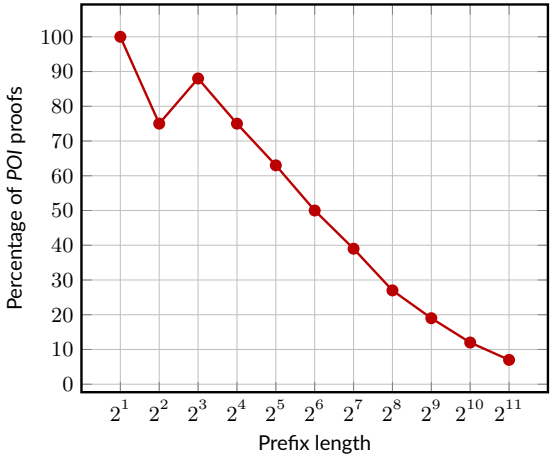
Can our **synthesized lemmas help first-order provers** with the proving problems for the *POI* theorems?

- We tried 7 lemma sets from the previous experiment, supplemented as **additional axioms**:



Can we identify subsets of **generated proofs with a large fraction of proofs of POI theorems**?

- Example for one of the obtained lemma sequences, ordered by save value



Hypothesis: We can **generate compressed proofs** by **incorporating combinators** in proof term enumeration

- Combinators restructure proof terms such that **repeated patterns** become **repeated subterms**, factored in the DAG
- **For MGT computation, they are handled like axioms**

Combinator B

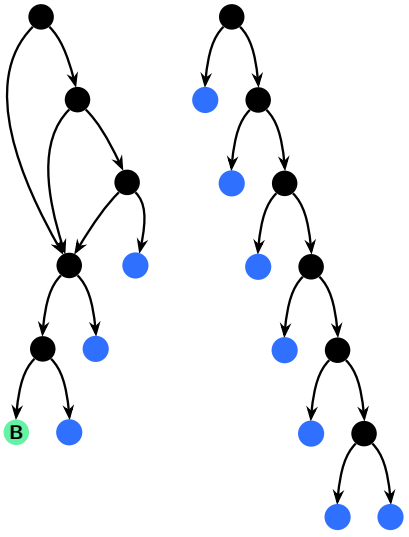
Reduction rule: $\mathsf{D}(\mathsf{D}(\mathsf{D}(\mathbf{B}, X), Y), Z) \longrightarrow \mathsf{D}(X, \mathsf{D}(Y, Z))$

Principal type: $\mathbf{B} : (x \Rightarrow y) \Rightarrow ((z \Rightarrow x) \Rightarrow (z \Rightarrow y))$

Example

$$\mathsf{D}(\mathsf{D}(\mathsf{D}(\mathbf{B}, 1), 1), \mathsf{D}(\mathsf{D}(\mathsf{D}(\mathbf{B}, 1), 1), \mathsf{D}(\mathsf{D}(\mathsf{D}(\mathbf{B}, 1), 1), 1)))$$
$$\xrightarrow{*} \mathsf{D}(1, \mathsf{D}(1, \mathsf{D}(1, \mathsf{D}(1, \mathsf{D}(1, \mathsf{D}(1, 1))))))$$

$$4 \rightarrow \mathsf{D}(\mathsf{D}(\mathbf{B}, 1), 1)$$
$$\mathit{Start} \rightarrow \mathsf{D}(4, \mathsf{D}(4, \mathsf{D}(4, 1)))$$



Combinator compression as DAG grammar

$4 \rightarrow \mathsf{D}(\mathsf{D}(\mathbf{B}, 1), 1)$
$Start \rightarrow \mathsf{D}(4, \mathsf{D}(4, \mathsf{D}(4, 1)))$

Equivalent tree grammar

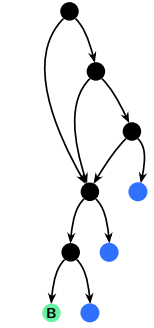
$\mathsf{r}(X) \rightarrow \mathsf{D}(1, \mathsf{D}(1, X))$
$Start \rightarrow \mathsf{r}(\mathsf{r}(\mathsf{r}(1)))$

Observation:

λ to SKI translation is linear [Kiselyov, 2018]

Thus, if the principal types of the combinators (i.e., those of **K** and **S**) are theorems, then **tree compression (and combinator compression) can improve at most linearly over DAG compression**

In *set.mm* we actually have **K** and **S** as axioms 1 and 2



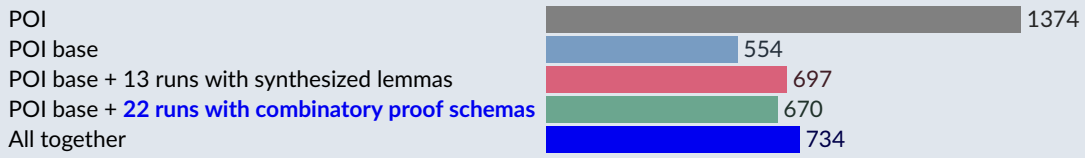
B ≡ S(KS)K

Experiments with Combinatory Proof Schemas

We **incorporate combinators in a restricted way to proof term generation**:

- Proof terms are then **built from *proof patterns***, other function symbols than D, defined as a combinator applied to a specified number of arguments
  - Actually, in two variations: **combinatory proof schemas** subsumes both
  - For identifying combinators that seem useful in proof schemas we analyzed *set.mm*
  - Primitive operators from [Peyton Jones: *The Implementation of Functional Programming Languages*, 1987]
  - Examples of used pattern sets: $\{\mathbf{I}/2, \mathbf{B}^*/3, \mathbf{C}/2, \mathbf{C}'/3, \mathbf{S}'/3\}$ $\{\mathbf{B}^*/3, \mathbf{C}'/3\}$
- Generation is **by compacted size with an adaptation of the PSP level**

Final results



Conclusion

Our approach is centered on proof terms in contrast to formulas

- Differently from most works with similar aims [Wos: *The resonance strategy*, 1995] [Jirí Vyskocil et al.: *Automated proof compression by invention of new definitions*, 2010] [Hetzl: *Applying tree languages in proof theory*, 2012] [Alhessi et al.: *Lemmanaid: Neuro-symbolic lemma conjecturing*, 2025] [Axelrod et al.: *Twitch: Learning abstractions for equational theorem proving*, 2026]
- An Exception [Schulz: *Analyse und Transformation von Gleichheitsbeweisen*, 1993]: **save values in proof DAG to identify important steps**; taken up in [Kaliszyk, Urban: *Learning-assisted th. pr. with millions of lemmas*, 2015]
- A reimagination of proving by proof structure enumeration (connection method, clausal tableaux)

A direction for future work: utilizing further the available lemmas/proofs of *set.mm*

- Comparing features of **human-made vs machine-generated lemmas/proofs** (e.g. with grammar compressions)
- Basis for machine learning for ATP (collaboration with Zsolt Zombori)

On generality and generalization possibilities

- Techniques and implementation are **ready for other axioms, arbitrary first-order Horn formulas**
- Predicate logic in *set.mm* needs a second function in proof terms: the unary G, **condensed generalization**, and axioms that seem to benefit from adapting **equality handling** of saturation-based provers
- Essential for our approach are **proof terms with a most general theorem**; possibilities for **resolution**:
  - Completeness for consequence finding [Lee, 1967]
  - Proof permutations [Kleine Büning, Richter, 1989]
  - Simulation by connection structure calculus [Eder, 1989], and with combinators [W, 2022]

References I

[Alhessi et al., 2025] Alhessi, Y., Einarsdóttir, S. H., Granberry, G., First, E., Johansson, M., Lerner, S., and Smallbone, N. (2025).
Lemmanaid: Neuro-symbolic lemma conjecturing.
CoRR, abs/2504.04942.

[Axelrod et al., 2026] Axelrod, G., Johansson, M., and Smallbone, N. (2026).
Twitch: Learning abstractions for equational theorem proving.
In Biere, A., Lutz, C., and Negri, S., editors, *IJCAR 2026*, LNCS (LNAI), pages 62–79. Springer.

[Eder, 1989] Eder, E. (1989).
A comparison of the resolution calculus and the connection method, and a new calculus generalizing both methods.
In Börger, E., Kleine Büning, H., and Richter, M. M., editors, *CSL '88*, volume 385 of *LNCS*, pages 80–98. Springer.

[Hetzl, 2012] Hetzl, S. (2012).
Applying tree languages in proof theory.
In Dediu, A.-H. and Martín-Vide, C., editors, *LATA 2012*, volume 7183 of *LNCS*, pages 301–312.

[Kaliszyk and Urban, 2015] Kaliszyk, C. and Urban, J. (2015).
Learning-assisted theorem proving with millions of lemmas.
J. Symb. Comput., 69:109–128.

References II

[Kiselyov, 2018] Kiselyov, O. (2018).
λ to SKI, semantically.
In Gallagher, J. P. and Sulzmann, M., editors, *FLOPS 2018*, volume 10818 of *LNCS (LNTCS)*, pages 33–50.

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

[Lee, 1967] Lee, R. C.-T. (1967).
A completeness theorem and computer program for finding theorems derivable from given axioms.
PhD thesis, University of California, Berkeley, CA.

[Megill,] Megill, N. D.
Metamath Proof Explorer Home Page.
Online: `https://us.metamath.org/mpeuni/mmset.html`, accessed May 8, 2026.

[Megill, 1995] Megill, N. D. (1995).
A finitely axiomatized formalization of predicate calculus with equality.
Notre Dame J. of Formal Logic, 36(3):435–453.

References III

[Peyton Jones, 1987] Peyton Jones, S. L. (1987).
The Implementation of Functional Programming Languages.
Prentice Hall.

[Prior, 1956] Prior, A. N. (1956).
Logicians at play; or Syll, Simp and Hilbert.
Australasian Journal of Philosophy, 34(3):182–192.

[Rawson et al., 2023] Rawson, M., Wernhard, C., Zombori, Z., and Bibel, W. (2023).
Lemmas: Generation, selection, application.
In Ramanayake, R. and Urban, J., editors, *TABLEAUX 2023*, volume 14278 of *LNAI*, pages 153–174.

[Schulz, 1993] Schulz, S. (1993).
Analyse und Transformation von Gleichheitsbeweisen.
Projektarbeit in informatik, Fachbereich Informatik, Universität Kaiserslautern.
(German Language).

[Ulrich, 2001] Ulrich, D. (2001).
A legacy recalled and a tradition continued.
J. Autom. Reasoning, 27(2):97–122.

References IV

[Vyskocil et al., 2010] Vyskocil, J., Stanovský, D., and Urban, J. (2010).
Automated proof compression by invention of new definitions.
In Clarke, E. M. and Voronkov, A., editors, *LPAR-16*, volume 6355 of *LNCS*, pages 447–462. Springer.

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

[Wernhard, 2024] Wernhard, C. (2024).
Structure-generating first-order theorem proving.
In Otten, J. and Bibel, W., editors, *AReCCa 2023*, volume 3613 of *CEUR Workshop Proc.*, pages 64–83. CEUR-WS.org.

[Wernhard, 2026] Wernhard, C. (2026).
Generating theorems by generating proof structures.
In Biere, A., Lutz, C., and Negri, S., editors, *IJCAR 2026*, LNCS (LNAI), pages 41–61. Springer.

[Wernhard and Bibel, 2021] Wernhard, C. and Bibel, W. (2021).
Learning from Łukasiewicz and Meredith: Investigations into proof structures.
In Platzer, A. and Sutcliffe, G., editors, *CADE 28*, volume 12699 of *LNCS (LNAI)*, pages 58–75. Springer.

References V

[Wernhard and Bibel, 2024] Wernhard, C. and Bibel, W. (2024).
Investigations into proof structures.
J. Autom. Reasoning, 68(24).

[Wernhard and Zombori, 2025] Wernhard, C. and Zombori, Z. (2025).
Mathematical knowledge bases as grammar-compressed proof terms: Exploring Metamath proof structures.
CoRR, abs/2505.12305.

[Wos, 1995] Wos, L. (1995).
The resonance strategy.
Computers Math. Applic., 29(2):133–178.

Metamath source text

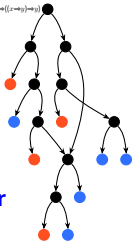
```
$( Principle of identity. Theorem *2.08 of [WhiteheadRussell] p.
101. [...] $)
id $p |- ( ph -> ph ) $=
  wph wph wph wi wph wph wph ax-1 wph wph wph wi ax-1 mpd $.

${
  syl5com.1 $e |- ( ph -> ps ) $.
  syl5com.2 $e |- ( ch -> ( ps -> th ) ) $.
  $( Syllogism inference with commuted antecedents. [...] $)
  syl5com $p |- ( ph -> ( ch -> th ) ) $=
    wph wch wps wth wph wps wch syl5com.1 a1d syl5com.2 sylcom $.
$}

${
  com12.1 $e |- ( ph -> ( ps -> ch ) ) $.
 $( Inference that swaps (commutes) antecedents in an
     implication. Inference associated with pm2.04 [...] $)
  com12 $p |- ( ps -> ( ph -> ch ) ) $=
    wps wps wph wch wps id com12.1 syl5com $.
$}

$( This theorem, sometimes called "Assertion" or "Pon" (for
   "ponens"), can be thought of as a closed form of modus ponens
   ~ ax-mp . Theorem *2.27 of [WhiteheadRussell] p. 104.
   (Contributed by NM, 15-Jul-1993.) $)
pm2.27 $p |- ( ph -> ( ( ph -> ps ) -> ps ) ) $=
    wph wps wi wph wps wph wps wi id com12 $.
```

Tree grammar

$$\begin{aligned}
\texttt{a1i}(X) &\to \mathrm{D}(1, X) \\
\texttt{a2i}(X) &\to \mathrm{D}(2, X) \\
\texttt{mpd}(X, Y) &\to \mathrm{D}(\texttt{a2i}(Y), X) \\
\texttt{id} &\to \texttt{mpd}(1, 1) \\
\texttt{syl}(X, Y) &\to \texttt{mpd}(X, \texttt{a1i}(Y)) \\
\texttt{a1d}(X) &\to \texttt{syl}(X, 1) \\
\texttt{sylcom}(X, Y) &\to \texttt{syl}(X, \texttt{a2i}(Y)) \\
\texttt{syl5com}(X, Y) &\to \texttt{sylcom}(\texttt{a1d}(X), Y) \\
\texttt{com12}(X) &\to \texttt{syl5com}(\texttt{id}, X) \\
\texttt{pm2.27} &\to \texttt{com12}(\texttt{id})
\end{aligned}$$

Expansion of `pm2.27`

$$\mathrm{D}(\mathrm{D}(2, \mathrm{D}(1, \mathrm{D}(2, \mathrm{D}(\mathrm{D}(2, 1), 1)))),\ \mathrm{D}(\mathrm{D}(2, \mathrm{D}(1, 1)), \mathrm{D}(\mathrm{D}(2, 1), 1)))$$



Expansion of `pm2.27` as DAG grammar

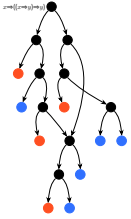
$$\begin{aligned}
4 &\to \mathrm{D}(\mathrm{D}(2, 1), 1) \\
\texttt{pm2.27} &\to \mathrm{D}(\mathrm{D}(2, \mathrm{D}(1, \mathrm{D}(2, 4))), \mathrm{D}(\mathrm{D}(2, \mathrm{D}(1, 1)), 4))
\end{aligned}$$

Tree grammar

$$\begin{aligned}
\texttt{a1i}(X) &\to \mathsf{D}(1, X)\\
\texttt{a2i}(X) &\to \mathsf{D}(2, X)\\
\texttt{mpd}(X, Y) &\to \mathsf{D}(\texttt{a2i}(Y), X)\\
\texttt{id} &\to \texttt{mpd}(1, 1)\\
\texttt{syl}(X, Y) &\to \texttt{mpd}(X, \texttt{a1i}(Y))\\
\texttt{a1d}(X) &\to \texttt{syl}(X, 1)\\
\texttt{sylcom}(X, Y) &\to \texttt{syl}(X, \texttt{a2i}(Y))\\
\texttt{syl5com}(X, Y) &\to \texttt{sylcom}(\texttt{a1d}(X), Y)\\
\texttt{com12}(X) &\to \texttt{syl5com}(\texttt{id}, X)\\
\texttt{pm2.27} &\to \texttt{com12}(\texttt{id})
\end{aligned}$$

Expansion of `pm2.27`

$$\begin{aligned}
&\mathsf{D}(\mathsf{D}(2, \mathsf{D}(1, \mathsf{D}(2, \mathsf{D}(\mathsf{D}(2, 1), 1)))),\\
&\quad \mathsf{D}(\mathsf{D}(2, \mathsf{D}(1, 1)), \mathsf{D}(\mathsf{D}(2, 1), 1))))
\end{aligned}$$

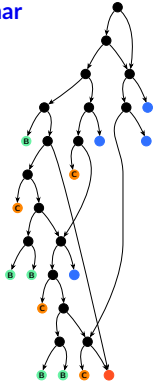


Expansion of `pm2.27` as DAG grammar

$$\begin{aligned}
4 &\to \mathsf{D}(\mathsf{D}(2, 1), 1)\\
\texttt{pm2.27} &\to \mathsf{D}(\mathsf{D}(2, \mathsf{D}(1, \mathsf{D}(2, 4))), \mathsf{D}(\mathsf{D}(2, \mathsf{D}(1, 1)), 4))
\end{aligned}$$

Combinator form as DAG grammar

$$\begin{aligned}
x_1 &\to 1\\
x_2 &\to 2\\
x_3 &\to \mathbf{C}x_2\\
x_4 &\to x_3 11\\
x_5 &\to \mathbf{C}(\mathbf{BB}x_3)x_1\\
x_6 &\to \mathbf{C}x_5 1\\
x_7 &\to \mathbf{C}(\mathbf{BB}x_5)x_2\\
x_8 &\to \mathbf{B}x_7 x_6\\
x_9 &\to x_8 x_4\\
\mathit{Start} &\to x_9 x_4
\end{aligned}$$



Combinator form

$$\begin{aligned}
&\mathbf{B}(\mathbf{C}(\mathbf{BB}(\mathbf{C}(\mathbf{BB}(\mathbf{C}2))1))2)\\
&\quad (\mathbf{C}(\mathbf{C}(\mathbf{BB}(\mathbf{C}2))1)1)(\mathbf{C}211)(\mathbf{C}211)
\end{aligned}$$

We want to **restrict combinators in proof term enumeration** to those from a given set, and in given contexts

A ***proof pattern*** is characterized by a function symbol, its arity and a combinator

Example proof patterns

$\mathsf{p}/2, \mathbf{B}$ $\quad \mathsf{p}(X, Y) \longrightarrow \mathsf{D}(\mathsf{D}(\mathbf{B}, X), Y)$

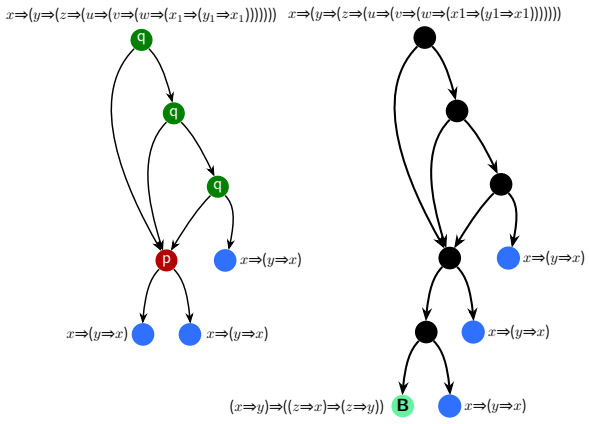
$\mathsf{q}/2, \mathbf{I}$ $\quad \mathsf{q}(X, Y) \longrightarrow \mathsf{D}(\mathsf{D}(\mathbf{I}, X), Y) \longrightarrow \mathsf{D}(X, Y)$

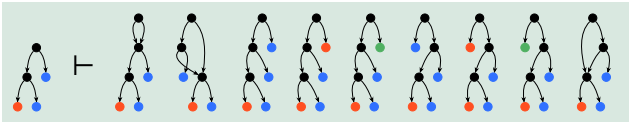
A proof term built from p, q

$$4 \;\rightarrow\; \mathsf{p}(1, 1)$$
$$\mathit{Start} \;\rightarrow\; \mathsf{q}(4, \mathsf{q}(4, \mathsf{q}(4, 1)))$$

Rewriting the patterns (directly on the grammar) gives the combinator compression:

$$4 \;\rightarrow\; \mathsf{D}(\mathsf{D}(\mathbf{B}, 1), 1)$$
$$\mathit{Start} \;\rightarrow\; \mathsf{D}(4, \mathsf{D}(4, \mathsf{D}(4, 1)))$$





Some counterexamples of size 3 and 4