Mathematical Knowledge Bases as Grammar-Compressed Proof Terms: Exploring Metamath Proof Structures

Christoph Wernhard [1] Zsolt Zombori [2]

[1]University of Potsdam [2]HUN-REN Alfréd Rényi Institute of Mathematics and Eötvös Loránd University

Deduktionstreffen 2025

Stuttgart, August 1, 2025

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 457292495; 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; Based upon work from the action CA20111 EuroProofNet supported by COST

Mathematical Knowledge Bases as Grammar-Compressed Proof Terms

1. Introduction
2. Towards Understanding – An Adequate Theory
3. Experiments: A Bird's Eye View on *set.mm*
4. Experiments: Machine Compression vs. Human Structuring
5. Some Specific Application Possibilities and Related Work
6. Conclusion

1. Introduction

2. Towards Understanding – An Adequate Theory

3. Experiments: A Bird's Eye View on *set.mm*

4. Experiments: Machine Compression vs. Human Structuring

5. Some Specific Application Possibilities and Related Work

6. Conclusion

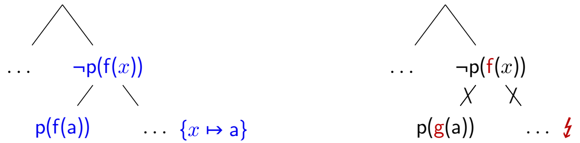
Approach, Goals and Expectations

We develop a **formal and automated** combination of

- **Structure-generating ATP**
- **Condensed detachment (CD)**
- **ITP with *Metamath***
- **Grammar-based tree compression**

- These are unified by the notion of **proof term**
- Scaling up ATP; **tightly integrating ATP and ITP** (and its mathematical KBs)
- **Proofs of lemmas $\hat{=}$ productions of grammars that compress proof terms**
- **Mapping between abstraction levels $\hat{=}$ lossless compression of proof terms** (same language)
- Basis for
  - **Dataset-oriented methods**: statistical, complex networks, machine learning
  - Investigating and analyzing **how proofs are/can be structured**
    - by humans / by machine – for humans and for machine processing
  - Learning to guide proof search in ATP from ITP proofs
- A framework that is **powerful yet sparse**, reduced to essentials – useful for research

- **Enumeration of proof structures** (Prawitz, connection method, clausal tableaux, PTTP)
  - In contrast to generating consequence formulas (resolution, saturation-based techniques)
  - Enumeration is **restricted by unification** of formulas associated with nodes



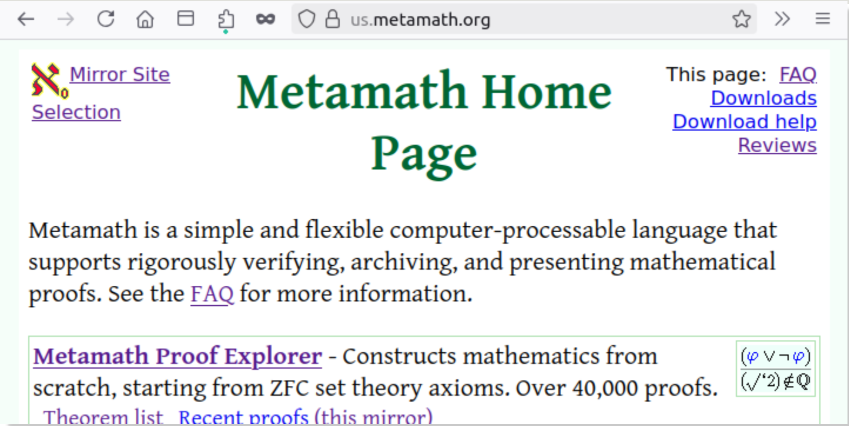
- "Conventionally" only **tree** structures are considered, with "global" (rigid) formula variables – but
  - **DAGs** where sub-proofs are re-used with "local" formula variables give much shorter proofs
    - Like 10^4 vs. 10^{23}
    - Connection structure calculus [Eder, 1989]
    - SGCD: proves **LCL073-1**, short proof of LCL038-1 [W, 2023,2024]
    - CCS: related to combinators [W, 2022]

- **Łukasiewicz, Tarski** since the 1920s: **first-order axiomatizations of propositional logics**
  - Formal proofs with the method of substitution and detachment
- **Carew A. Meredith** in the mid 1950s refined this with **condensed detachment**
  - Implicit **most general unifiers** instead of explicit substitutions
  - **Proof terms, with a DAG representation**

  D(A, B) proves the conclusion y
  if A proves the major premise $(x \Rightarrow y)$
  and B proves the minor premise x

1. *CCCpqrCCrpCsp*
2. *CCCpqpCrp* = DDD1D111n
3. *CCCpqrCqr* = DDD1D1D121n
4. *CpCCpqCrq* = D31
5. *CCCpqCrsCCCqtsCrs* = DDD1D1D1D141n
6. *CCCpqCrsCCpsCrs* = D51

- Formulas-as-types
  [Hindley, D. Meredith: *Principal Type-Schemes and Condensed Detachment*, 1990]
- CD problems were used a lot in ATP in the 1990s, around *OTTER*
  [Ulrich: *A Legacy Recalled and a Tradition Continued*, 2001]
- Renewed interest: fresh views on structure-generating ATP
  [W, Bibel: *Investigations into Proof Structures*, 2021,2024]



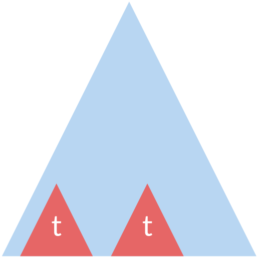
- By **Norman Megill**, started early 1990s; contributors include David A. Wheeler, Mario Carneiro
- ***Metamath Proof Explorer*** aka ***set.mm***: the largest *Metamath* DB; available as a single text file
- "Formalizing 100 Theorems": Isabelle 92; HOL Light 89; Coq 79; Lean 79; **Metamath 74**; Mizar 69

- "Metavariable mathematics" – use of **metavariables over an object logic**
- **Simplest** framework that allows essentially all of mathematics to be expressed with absolute rigor
  - All statements treated as mere **sequences of symbols**, constant and variable tokens

(	ph	->	(	ps	->	ph	)	)

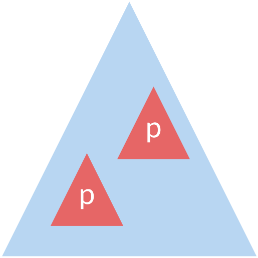
  - Metamath just knows how to **substitute strings of symbols for the variables**, based on instructions you provide it in a proof, subject to constraints you specify for the variables
  - Based on CD: [Megill: *A Finitely Axiomatized Formalization of Predicate Calculus w. Equality*, 1995]
- **No particular set of axioms**, axioms are defined in a DB
- **Almost no hard-wired syntax**; syntax also defined via substitution rules in the DB
  - Parsing is done **within proofs**, based on declarations in the DB
  - It is easy to **strip off the "syntactic" parts** from proofs; tools by default do not show them
- **Specification** and introduction: **Metamath book** (free PDF)
  [Megill, Wheeler: *Metamath – A Computer Language for Mathematical Proofs, 2nd. ed*, 2019]
- No single canonical tool: **many verifiers and proof assistants**, with *metamath.exe* as a reference
  - *metamath.exe* verifies *set.mm* in **7.5 s**, an optimized system in **0.2 s**

DAG: sharing repeated **subtrees**



$$\begin{aligned} start &\rightarrow \mathsf{a}(\mathsf{b}(t), t) \\ t &\rightarrow \mathsf{c}(\mathsf{d}) \end{aligned}$$

Grammar: sharing repeated **tree patterns** (connected subgraphs of the tree)



$$\begin{aligned} start &\rightarrow \mathsf{a}(p(\mathsf{b}(\mathsf{c})), p(\mathsf{d}(\mathsf{e}))) \\ p(V) &\rightarrow \mathsf{f}(\mathsf{g}(V)) \end{aligned}$$

1. Introduction

2. Towards Understanding – An Adequate Theory

3. Experiments: A Bird's Eye View on *set.mm*

4. Experiments: Machine Compression vs. Human Structuring

5. Some Specific Application Possibilities and Related Work

6. Conclusion

- We distinguish two vocabularies: for **formulas** and for **proof terms**
  - We call variables in proof terms ***parameters*** and write them $V_1, V_2, \ldots$

$$\mathtt{mptnan}(V_1, \mathtt{D}(\mathtt{xornan}, V_2))$$

- A proof term proves its **most general theorem (MGT)**

$$\mathtt{mptnan}(V_1, \mathtt{D}(\mathtt{xornan}, V_2)) \ : \ \mathtt{IsTheorem}(\mathtt{n}(x)) \leftarrow \mathtt{IsTheorem}(y) \land \mathtt{IsTheorem}(\mathtt{wxo}(y, x))$$

  - The MGT is a definite clause with **a body atom for each parameter in the proof term**
  - For brevity, we drop the single unary predicate `IsTheorem` (in *Metamath* it its written ⊢):

$$\mathtt{mptnan}(V_1, \mathtt{D}(\mathtt{xornan}, V_2)) \ : \ \mathtt{n}(x) \leftarrow y \land \mathtt{wxo}(y, x)$$

  - The MGT is based on **presuppositions (axioms, earlier proven lemmas)**

$$\begin{array}{lcl} \mathtt{mptnan} & :: & \mathtt{n}(y) \leftarrow x \land \mathtt{n}(\mathtt{wa}(x, y)) \\ \mathtt{xornan} & :: & (\mathtt{wxo}(x, y) \Rightarrow \mathtt{n}(\mathtt{wa}(x, y))) \\ \mathtt{D} & :: & y \leftarrow (x \Rightarrow y) \land x \end{array}$$

  - Depending on the presuppositions, the MGT of a proof term may be **undefined**

The *CDDC* Inference System to Specify the "Proves" Relation

Presupposition Application

$$\text{APP}\ \frac{p :: A \leftarrow B_1 \wedge \ldots \wedge B_n \quad d_1 : B'_1 \leftarrow \boldsymbol{R}_1 \quad \ldots \quad d_n : B'_n \leftarrow \boldsymbol{R}_n}{p(d_1, \ldots, d_n) : (A \leftarrow \boldsymbol{U})\sigma}$$

where

- the first premise is in the presupposition base $\mathcal{B}$
- σ = mgu($\{\{B_1, B'_1\}, \ldots, \{B_n, B'_n\}, \{\boldsymbol{U}, \boldsymbol{R}_1, \ldots, \boldsymbol{R}_n\}\}$)
- premises have disjoint sets of variables achieved by renaming
- variables $\boldsymbol{U}$ do not occur in the premises

Parameter Recording

$$\text{PAR}\ \frac{}{V_i : u_i \leftarrow \boldsymbol{U}}$$

Instantiation

$$\text{INS}\ \frac{d : A \leftarrow \boldsymbol{R}}{d : (A \leftarrow \boldsymbol{R})\sigma}$$

σ subject to *Metamath*-specific constraints

Legend

- $p :: F$ – presupposition-statement
- $d : F$ – proves-statement
- For each parameter $V_i \in \{V_1 \ldots V_k\}$ there is a dedicated associated formula variable u_i. $\boldsymbol{U} \overset{\text{def}}{=} u_1 \wedge \ldots \wedge u_k$

The Most General Theorem (MGT) of a Proof Term

$$\textsc{App}\ \frac{p :: A \leftarrow B_1 \wedge \ldots \wedge B_n \quad d_1 : B'_1 \leftarrow \boldsymbol{R}_1 \ \ldots \ d_n : B'_n \leftarrow \boldsymbol{R}_n}{p(d_1, \ldots, d_n) : (A \leftarrow \boldsymbol{U})\sigma,}$$

where σ = mgu($\{\{B_1, B'_1\}, \ldots, \{B_n, B'_n\}, \{\boldsymbol{U}, \boldsymbol{R}_1, \ldots, \boldsymbol{R}_n\}\}$)

$$\textsc{Par}\ \frac{}{V_i : u_i \leftarrow \boldsymbol{U}} \qquad \textsc{Ins}\ \frac{d : A \leftarrow \boldsymbol{R}}{d : (A \leftarrow \boldsymbol{R})\sigma}$$

Definition. If, for presupposition base $\mathcal{B}$, **there is an $\{\textsc{App}, \textsc{Par}\}$-deduction of a proves-statement**

$$d[V_1, \ldots, V_k] : A \leftarrow B_1 \wedge \ldots \wedge B_k,$$

we say that mgt$_{\mathcal{B}}(d[V_1, \ldots, V_k])$ ***is defined*** and

$$\text{mgt}_{\mathcal{B}}(d[V_1, \ldots, V_k]) = A \leftarrow B_1 \wedge \ldots \wedge B_k$$

Example.

$$\frac{\mathtt{D} :: y \leftarrow (x \Rightarrow y) \wedge x \qquad \dfrac{\mathtt{ax\text{-}1} :: (x \Rightarrow (y \Rightarrow x))}{\mathtt{ax\text{-}1} : (x' \Rightarrow (y' \Rightarrow x'))}\ \textsc{App} \qquad \dfrac{\mathtt{ax\text{-}1} :: (x \Rightarrow (y \Rightarrow x))}{\mathtt{ax\text{-}1} : (x'' \Rightarrow (y'' \Rightarrow x''))}\ \textsc{App}}{\mathtt{D}(\mathtt{ax\text{-}1}, \mathtt{ax\text{-}1}) : (y' \Rightarrow (x'' \Rightarrow (y'' \Rightarrow x'')))}\ \textsc{App}$$

Handling Parameters in Proof Terms

$$\text{App}\ \frac{p :: A \leftarrow B_1 \wedge \ldots \wedge B_n \quad d_1 : B_1' \leftarrow \boldsymbol{R}_1 \ \ldots \ d_n : B_n' \leftarrow \boldsymbol{R}_n}{p(d_1, \ldots, d_n) : (A \leftarrow \boldsymbol{U})\sigma,}$$

where $\sigma = \text{mgu}(\{\{B_1, B_1'\}, \ldots, \{B_n, B_n'\}, \{\boldsymbol{U}, \boldsymbol{R}_1, \ldots, \boldsymbol{R}_n\}\})$

$$\text{Par}\ \frac{}{V_i : u_i \leftarrow \boldsymbol{U}} \qquad \text{Ins}\ \frac{d : A \leftarrow \boldsymbol{R}}{d : (A \leftarrow \boldsymbol{R})\sigma}$$

Rule Par, **parameter recording**, effects that for all occurrences of V_i in the proof term the head of the clause that is "proven" by the V_i is identified with the corresponding variable u_i in $\boldsymbol{U}$

(Recall that $\boldsymbol{U} = u_1 \wedge \ldots \wedge u_k$ where $V_1, \ldots, V_k$ are the parameters under consideration)

Example.

$$\dfrac{\mathtt{D} :: y \leftarrow (x \Rightarrow y) \wedge x \qquad \dfrac{}{V_1 : u_1' \leftarrow u_1'}\ \text{Par} \qquad \dfrac{\mathtt{ax\text{-}1} :: (x_1 \Rightarrow (x_2 \Rightarrow x_1))}{\mathtt{ax\text{-}1} : (x_1 \Rightarrow (x_2 \Rightarrow x_1)) \leftarrow u_1''}\ \text{App}}{\mathtt{D}(V_1, \mathtt{ax\text{-}1}) : y \leftarrow ((x_1 \Rightarrow (x_2 \Rightarrow x_1)) \Rightarrow y)}\ \text{App}$$

$$\begin{aligned} & \text{mgu}(\{\{(x \Rightarrow y), u_1'\},\ \{x, (x_1 \Rightarrow (x_2 \Rightarrow x_1))\}\},\ \{u_1, u_1', u_1''\}\}) \\ = \ & \{u_1 \mapsto ((x_1 \Rightarrow (x_2 \Rightarrow x_1)) \Rightarrow y),\ \ldots\} \end{aligned}$$

A Subtlety with Nonlinear Proof Terms

- A proof term is ***nonlinear*** if it has **multiple occurrences of the same parameter**
  - 30% on the proofs in *set.mm* are nonlinear
- A body atom of the MGT is **constrained simultaneously w.r.t. each occurrence** of the corresponding parameter V_i in the proof term
  - Leads for **nonlinear proof terms** to difference between MGT determined
    (1) **from proof term with parameters and MGTs of substituting proof terms**
    (2) **from proof term after substituting**
  - (1) may even be undefined for defined (2)

Proposition. Assume

- $\mathrm{mgt}_{\mathcal{B}}(d[V_1, \ldots, V_k]) \;=\; A \leftarrow B_1 \land \ldots \land B_k$
- $\mathrm{mgt}_{\mathcal{B}}(d_1) = B'_1, \ldots, \mathrm{mgt}_{\mathcal{B}}(d_k) = B'_k$, where $d_1, \ldots, d_k$ are ground
- $\sigma = \mathrm{mgu}(\{\{B_1, B'_1\}, \ldots, \{B_k, B'_k\}\})$ is defined

Then

- If d is **linear**, then $\quad \mathrm{mgt}_{\mathcal{B}}(d[d_1, \ldots, d_k]) = A\sigma$
- In the general case, **also for nonlinear** d $\quad A\sigma \geq \mathrm{mgt}_{\mathcal{B}}(d[d_1, \ldots, d_k])$

The Two Primitives of *Metamath* (*set.mm*) Proofs

[Megill: *A Finitely Axiomatized Formalization of Predicate Calculus with Equality*, 1995]

- *Condensed detachment* $\mathtt{D} :: y \leftarrow (x \Rightarrow y) \land x$ In *set.mm*: `ax-mp`, switched parameters
- *Condensed generalization* $\mathtt{G} :: \forall(y, x) \leftarrow x$ In *set.mm*: `ax-gen`

A *Metamath* Proof as a Tree Grammar

- Describes a (typically large) proof term built from `D`, `G` and axiom names
- One production per nonterminal; no cyclic dependencies between nonterminals
- **Nonterminals are theorem names in lemma role**

Example.

$$
\begin{array}{lcl}
\mathtt{mptxor}(V_1, V_2) & \rightarrow & \mathtt{mptnan}(V_1, \mathtt{D}(\mathtt{xornan}, V_2)) \\
\mathtt{mptnan}(V_1, V_2) & \rightarrow & \mathtt{D}(\mathtt{imnani}(V_2), V_1) \\
\mathtt{xornan} & \rightarrow & \mathtt{simprbi}(\mathtt{xor2}) \\
\mathtt{imnani}(V_1) & \rightarrow & \mathtt{mpbir}(V_1, \mathtt{imnan}) \\
 & \vdots & \\
\mathtt{mp2}(V_1, V_2, V_3) & \rightarrow & \mathtt{D}(\mathtt{D}(V_3, V_1), V_2) \\
\mathtt{a2i}(V_1) & \rightarrow & \mathtt{D}(\mathtt{ax\text{-}2}, V_1) \\
\mathtt{a1i}(V_1) & \rightarrow & \mathtt{D}(\mathtt{ax\text{-}1}, V_1)
\end{array}
$$

- **We consider productions ordered "bottom-up" and successively enrich the presupposition base**

Definition.

$$\text{grammar-mgt}_{\mathcal{B},G}(p_i(\boldsymbol{V}_i)) \stackrel{\text{def}}{=} \text{mgt}_{\mathcal{B}'}(d_i[\boldsymbol{V}_i]]), \text{ where}$$
$$\mathcal{B}' = \mathcal{B} \cup \bigcup_{j=1}^{i-1} \{p_j :: \text{grammar-mgt}_{\mathcal{B},G}(p_j(\boldsymbol{V}_j))\}.$$

- In case an involved MGT is undefined, we say the grammar-MGT for each nonterminal is undefined
- The subtlety concerning **nonlinear** proof terms and the MGT transfers to the grammar-MGT
  - Let $\text{val}_G(p_i(\boldsymbol{V}_i))$ denote the expansion of nonterminal $p_i(\boldsymbol{V}_i)$ w.r.t. grammar G

Proposition. Assume $\text{grammar-mgt}_{\mathcal{B},G}(p_i(\boldsymbol{V}_i))$ is defined. Then $\text{mgt}_{\mathcal{B}}(\text{val}_G(p_i(\boldsymbol{V}_i)))$ is defined, and

- If G is **linear**, then $\text{grammar-mgt}_{\mathcal{B},G}(p_i(\boldsymbol{V}_i)) = \text{mgt}_{\mathcal{B}}(\text{val}_G(p_i(\boldsymbol{V}_i)))$
- In the general case, **also for nonlinear** G $\text{grammar-mgt}_{\mathcal{B},G}(p_i(\boldsymbol{V}_i)) \geq \text{mgt}_{\mathcal{B}}(\text{val}_G(p_i(\boldsymbol{V}_i)))$

Taking User-Specified Instantiation into Account

- In *Metamath* theorem statements may be **user-specified strict instances of the proven MGT**
- We model this by **associating with each production** an explicitly given definite clause F_i such that

$$F_i \geq \text{shallow-mgt}_K(p_i(\boldsymbol{V}_i)),$$

where the ***shallow-MGT*** of $p_i(\boldsymbol{V}_i)$ is defined as

$$\text{shallow-mgt}_K(p_i(\boldsymbol{V}_i)) \quad \stackrel{\text{def}}{=} \quad \text{mgt}_{\mathcal{B}'}(d_i[\boldsymbol{V}_i]), \text{ where } \mathcal{B}' = \mathcal{B} \cup \bigcup_{j=1}^{i-1} \{p_j :: F_j\}$$

$\texttt{mptxor}(V_1, V_2)$	$\rightarrow$	$\texttt{mptnan}(V_1, \texttt{D}(\texttt{xornan}, V_2))$	F_n
$\texttt{mptnan}(V_1, V_2)$	$\rightarrow$	$\texttt{D}(\texttt{imnani}(V_2), V_1)$	F_{n-1}
$\texttt{xornan}$	$\rightarrow$	$\texttt{simprbi}(\texttt{xor2})$	F_{n-2}
$\texttt{imnani}(V_1)$	$\rightarrow$	$\texttt{mpbir}(V_1, \texttt{imnan})$	F_{n-3}
	$\vdots$		
$\texttt{mp2}(V_1, V_2, V_3)$	$\rightarrow$	$\texttt{D}(\texttt{D}(V_3, V_1), V_2)$	F_3
$\texttt{a2i}(V_1)$	$\rightarrow$	$\texttt{D}(\texttt{ax-2}, V_1)$	F_2
$\texttt{a1i}(V_1)$	$\rightarrow$	$\texttt{D}(\texttt{ax-1}, V_1)$	F_1

- The MGT variations are related by

$$F_i \;\geq\; \text{shallow-mgt}_K(p_i(\boldsymbol{V}_i)) \;\geq\; \text{grammar-mgt}_{\mathcal{B},G}(p_i(\boldsymbol{V}_i)) \;\geq\; \text{mgt}_{\mathcal{B}}(\text{val}_G(p_i(\boldsymbol{V}_i)))$$

1. Introduction
2. Towards Understanding – An Adequate Theory
3. Experiments: A Bird's Eye View on *set.mm*
4. Experiments: Machine Compression vs. Human Structuring
5. Some Specific Application Possibilities and Related Work
6. Conclusion

The *CD Tools* Environment

- For experimenting with condensed detachment ...
- Written in ***SWI-Prolog***
- Extends the *PIE* (*Proving*, *Interpolating*, *Eliminating*) environment [W, 2016; 2020]
    - Provides interfaces to *TPTP* and many first-order provers
- Includes structure-generating provers for CD and Horn problems: *SGCD*, *CCS* [W 2022; Rawson, W, Zombori, Bibel 2023; W 2024; W, Bibel 2024]
- **New: *Metamath* interface**, written from scratch in *SWI-Prolog*
    - Also **proofs are translated to Prolog terms**, with various options
        - Raw form preserves *Metamath*'s compression through factorized terms
        - With and without *Metamath*'s "syntactic" steps
        - Compatible with other proof terms in *CD Tools*
    - Prolog fact base generated from *set.mm* in 2 min; after compilation it loads in 0.5 s
- **New: methods and support for grammar-based tree compression**

The SetCore KB: The First 60% of *set.mm* for Experimenting

Topic	*1st Thm*
Propositional calculus	1
Predicate calculus	1,744
Zermelo-Fraenkel set theory	2,650
The axiom of replacement	5,086
The axiom of choice	9,916
Tarski-Grothendieck set theory	10,157
Real and complex numbers	10,304
Elementary number theory	15,391
Basic structures	16,243
Basic category theory	16,695
Basic order theory	17,238
Basic algebraic structures	17,517
Basic linear algebra	19,918
Basic topology	20,936
Basic real and complex analysis	23,316
Basic real and complex functions	23,897
Elementary geometry	25,398
Graph theory	25,912
Last Thm	27,235

Topic	*1st Thm*
Guides, miscellanea, examples	27,236
Deprecated material	27,321
70 mathboxes	29,111
Last Thm	43,920

Structural Properties of the KB SetCore (I)

$\|G\|$	N(G)	ref$_G(p)$						$\|p\|$			$\|\text{val}_G(p)\|$	
		med	avg	max	0	1	min	med	avg	max	med	max
1,824,835	27,233	3	53	63,198	16%	20%	0	12	67	21,651	$3{\times}10^{54}$	$5{\times}10^{1880}$

- $|G|$: Size of G (sum of number of edges of the RHSs)
- N(G): Number of productions of G
- ref$_G(p)$: Number of occurrences of the nonterminal p in RHSs
  i.e., occurrences of p as direct premise in another theorem's proof
- $|p|$: Size of the production for p
- $|\text{val}_G(p)|$: Size of the value (expansion) of the LHS for p
  - *These values are gigantic*

Structural Properties of the KB SetCore (II)

$\mathsf{sav}_G(p)$					
min	med	avg	max	<0	0
-366	33	796	3,981,585	12%	10%

- $\mathsf{sav}_G(p)$, the ***save-value of*** p

$$\mathsf{sav}_G(p) \overset{\text{def}}{=} |G'| - |G|,$$

  where G' is G after **eliminating** p (unfolding p in all RHSs and removing $p's$ production)
- **Indicates contribution of the production to grammar size reduction**
- It is 0 if the size remains unchanged and negative if the size is increased
- For a linear production it is $\mathsf{ref}_G(p) \times (|P| - \mathsf{arity}(p)) - |P|$ [Lohrey et al., 2013]
- Subcolumns relate here to the multiset of the values for just those p with $\mathsf{ref}(p) > 0$
- *22% have a save-value ≤ 0. Apparently they serve to break apart a larger proof.*
  Do these have further features that may guide automated breaking apart?

Structural Properties of the KB SetCore (III)

arity(p)				voccs(v)				vmult$_G(v)$		
avg	max	0	nl$_G$	min	med	avg	max	min	med	max
2	28	45%	28%	0	1	7	2,445	0	16,640	7×10^{1795}

- arity(p): Arity of p
  - *Maximum is 28, but average just 2, where 45% have arity 0 as in DAG compression*
- nl$_G$: Percentage of productions that are nonlinear
- voccs(v): Number of occurrences of variable v in the RHS
  - *Although median is 1, some have ≥ 2,000 occurrences. Do these productions play special roles?*
  - *Minimum 0 indicates LHS-only variables. What is their purpose?*
- vmult(v) : Number of occurrences of variable v in val$_G(p)$ for the production p that has v in its LHS

Formula Properties of the KB SetCore

$\lvert F\rvert$				height(F)						
min	med	avg	max	min	med	avg	max	$\gtrdot$mgt	$\doteq$	$\geq$
0	10	14	193	0	4	4	20	8.38%	3.08%	3.91%

- $\lvert F\rvert$: Size of the clause (sum of tree size of atom arguments)
- height(F): Height of the clause (maximal height of its atoms)
  - *$\lvert F\rvert$ and height(F) have large differences between maximum and average*
- $\gtrdot$mgt: **Percentage of theorem clauses that are a *strict* instance of the corresponding shallow-MGT**
  - *The portion is significant*
- $\doteq$: **Percentage of theorem clauses that would be removed if *duplicates* were deleted** such that only a single copy is retained (modulo renaming of variables and clause body permutations)
- $\geq$: Like $\doteq$ but w.r.t. **subsumption**
  - *This redundancy might have reasons: different theorem names in different application contexts; shorter or otherwise preferable proof of a strictly subsumed theorem*

1. Introduction
2. Towards Understanding – An Adequate Theory
3. Experiments: A Bird's Eye View on *set.mm*
4. Experiments: Machine Compression vs. Human Structuring
5. Some Specific Application Possibilities and Related Work
6. Conclusion

- Background: ***Re-Pair*** algorithm for grammar-based string compression [Larsson, Moffat, 2000]
  - Recursively replace a most frequent *digram* (pair fg of consecutive symbols) with a fresh nonterminal h, defined with production $h \to fg$

- ***TreeRePair*** adapts it to trees [Lohrey, Maneth, Mennicke: *XML Tree Struct. Compr. using RePair*, 2013]
  - A ***digram is now a pattern*** characterized by a parent symbol f with arity $n \geq 1$, child symbol g with arity $m \geq 0$ and index i. The defining production with fresh nonterminal h is

$$h(V_1, \ldots, V_{n-1+m}) \to f(V_1, \ldots, V_{i-1}, g(V_i, \ldots, V_{i+m}), V_{i+m+1}, \ldots, V_{n-1+m})$$

Example.

Illustration	*Digram*	*Occurrences*
f(g(e, e), f(g(e, e), e))	$\mathsf{f}(\mathsf{g}(V_1, V_2), V_3)$	2
f(g(e, e), f(g(e, e), e))	$\mathsf{g}(\mathsf{e}, V_1)$	2
f(g(e, e), f(g(e, e), e))	$\mathsf{g}(V_1, \mathsf{e})$	2
f(g(e, e), f(g(e, e), e))	$\mathsf{f}(V_1, \mathsf{f}(V_2, V_3))$	1
f(g(e, e), f(g(e, e), e))	$\mathsf{f}(V_1, \mathsf{e})$	1

TreeRePair [Lohrey et al. 2013] – Two Phases

1. **Replacement phase**

   Loop, maintains a *main term* initialized with input term
   The (initially possibly large) *main term* may internally be represented as DAG

   - Identify digrams with multiple occurrences
   - Select one or more digrams according to heuristic criteria (e.g., arity, no. of occurrences)

     $\mathsf{f}(\mathsf{g}(\mathsf{e}, \mathsf{e}), \mathsf{f}(\mathsf{g}(\mathsf{e}, \mathsf{e}), \mathsf{e}))$
   - Generate productions with fresh nonterminals for the selected digrams

     $\mathsf{h}(V_1, V_2, V_3) \rightarrow \mathsf{f}(\mathsf{g}(V_1, V_2), V_3)$
   - In the *main term*, fold into these productions (rewrite with them from RHS to LHS) – configurable

     $\mathsf{f}(\mathsf{g}(\mathsf{e}, \mathsf{e}), \mathsf{f}(\mathsf{g}(\mathsf{e}, \mathsf{e}), \mathsf{e})) \Longrightarrow \mathsf{f}(\mathsf{g}(\mathsf{e}, \mathsf{e}), \mathsf{h}(\mathsf{e}, \mathsf{e}, \mathsf{e})) \Longrightarrow \mathsf{h}(\mathsf{e}, \mathsf{e}, \mathsf{h}(\mathsf{e}, \mathsf{e}, \mathsf{e}))$

   Output: *Proof grammar* with the fresh productions and a production $(\mathit{Start} \rightarrow \mathit{FinalMainTerm})$

2. **Pruning phase**

   Productions whose save-value is ≤0 are eliminated by unfolding them in all RHSs – configurable

Our Proof Compression Workflow

	Processing stage	*Kind*	*Source*	$\|G\|$	$\mathrm{N}(G)$
	Initial set of trees			$5{\times}10^{22}$	17
	Initial set of trees as DAG			21,472	927
1.	TreeRePair replacement phase	Structural	[Lohrey et al., 2013]	9,739	4,153
2.	TreeRePair pruning phase	Structural	[Lohrey et al., 2013]	3,683	905
3.	Nonlinear compression	Structural		3,204	604
4.	Same-value reduction	Structural		3,174	593
5.	MGT-based reduction	Formula-related		3,017	534

- **Nonlinear compression:** introduce nonlinear productions for RHS occurrences with repeated arguments
- **Same-value reduction:** eliminate multiple nonterminals with same expansion
- **MGT-based reduction:** eliminate productions for which the grammar-MGT is subsumed by that of another production
- Subtleties
  - Configuration such that productions of specified **top-level theorems** are preserved
  - Consideration of parameters modulo permutation

The KBs MINISET and MINITRP

	Processing stage	Kind	Source	$\lvert G \rvert$	$\mathrm{N}(G)$
	Initial set of trees			$5{\times}10^{22}$	17
	Initial set of trees as DAG			21,472	927
1.	TreeRePair replacement phase	Structural	[Lohrey et al., 2013]	9,739	4,153
2.	TreeRePair pruning phase	Structural	[Lohrey et al., 2013]	3,683	905
3.	Nonlinear compression	Structural		3,204	604
4.	Same-value reduction	Structural		3,174	593
5.	MGT-based reduction	Formula-related		3,017	534

A Small Manageable Extract from *set.mm*

- *Theorem Sampler* highlights 44 theorems from *set.mm*
- We chose those 17 where expansion and our grammar-compression workflow succeeded in 60 s

The MINISET KB – Human-Expert Proof Structuring

- Productions for the proofs the 17 theorems, supplemented by productions from *set.mm* for all theorems that are directly or indirectly referenced by these

The MINITRP KB – Machine Proof Structuring

- Result of our compression workflow for the set of the expanded proofs of the 17 theorems

Structural Properties of MiniTrp vs. MiniSet (I)

	$\|G\|$	N(G)	$\mathrm{ref}_G(p)$ med	$\mathrm{ref}_G(p)$ avg	$\mathrm{ref}_G(p)$ max	$\mathrm{ref}_G(p)$ 1	$\mathrm{ref}_G(p)$ min	$\|p\|$ med	$\|p\|$ avg	$\|p\|$ max	$\|\mathrm{val}_G(p)\|$ med	$\|\mathrm{val}_G(p)\|$ max
MiniSet	2,302	690	2	3	68	45%	0	3	3	68	46,647	5.06×10^{22}
MiniTrp	3,017	534	3	5	85	0%	1	3	6	288	11,034	1.29×10^{20}
MiniDag	21,472	927	2	7	966	0%	0	8	23	1,694	17,171,018	5.06×10^{22}

- *$|G|$ is for MiniTrp 30% larger than for MiniSet. What mechanical techniques are missing for a comparable compression rate?*
- We also include MiniDag, the minimal DAG compression of set of the 17 expanded proofs
  - *DAG compression already brings the gigantic tree sizes down to feasibility for machines. Pattern-based grammar compression reduces the size further by a factor of about 7–10*

Structural Properties of MiniTrp vs. MiniSet (II)

	$\|\text{sav}_G(p)\|$						arity(p)			
	min	med	avg	max	<0	0	avg	max	0	nl_G
MiniSet	-5	0	3	358	31%	29%	1	5	40%	2.17%
MiniTrp	0	4	25	7,063	0%	0%	1	7	48%	2.43%
MiniDag	1	18	71	16,140	0%	0%	0	0	100%	0.00%

- *Save-values are noticeable larger in MiniTrp than in MiniSet*

Formula Properties of MiniTrp vs. MiniSet

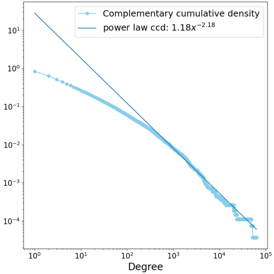
	N(G)	$\|F\|$ min	$\|F\|$ med	$\|F\|$ avg	$\|F\|$ max	height(F) min	height(F) med	height(F) avg	height(F) max	Set	MS
MiniSet	690	0	5	6	48	0	3	3	13		
MiniTrp	534	0	6	7	74	0	3	3	11	34%	29%
MiniDag	927	0	7	10	53	0	4	5	13	21%	18%

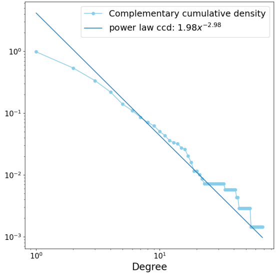
- Set: **34% of the formulas in MiniTrp are also in *set.mm***
- MS: **29% of the formulas in MiniTrp are also in MiniSet**
  - (not counting the 17 top-level theorems and modulo body permutations)
  - *These are automated rediscoveries of lemma formulas from human structuring*
  - *The 5% difference between both values represents formulas of MiniTrp that are in set.mm and potentially useful for proving the 17 top-level theorems, but were not used to prove them in the human structuring*

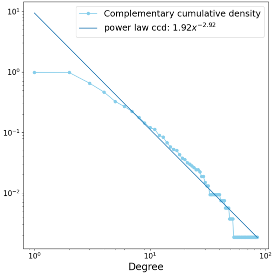
Definition. PDNet(G), the ***proof dependency network*** of G, is a directed graph:

- node = nonterminal
- **edge $p \to q$: occurrence of q in the RHS for p** ("q occurs as a direct premise of p")

- Then $\text{ref}_G(p)$ is the **in-degree** of node p in PDNet(G)
- Many real-world networks are ***scale-free***, i.e., exhibit **power law degree distributions** (roughly: "a large fraction of wealth falling into a small fraction of the nodes"); often a power law holds only for the tail of the distribution [Newman: *Networks*, 2018]
- **SetCore, MiniSet and MiniTrp have power-law in-degree distributions!**







Lemma Synthesis by Compressing Human Structurings Further

- A way to **combine given human structuring with machine compression**
- **Compressing a given *grammar***: take set of RHSs; add root; apply tree compression workflow

KB	$\lvert G \rvert$
MiniSet	2.302
MiniSet compressed further	1,831
MiniTrp	3,017

- **Scales up**: experiments on **subsets of SetCore for mathematical topics**
  - Given grammar vs. union of further compressions: 7% reduction
  - Reduction per topic: from 4% (*Basic Algebraic Structures*) to 30% (*Tarski-Grothendieck Set Theory*)
- **Yields some often used and thus apparently useful new lemmas**
  For example, for *Axiom of Choice*

```
lemma905(A) -> ad2antrr(syl(A, necon2ai(mtbii(sdom0, breq2)))).
$e |- ( A -> B ~< C ) $.
$p |- ( ( ( A /\ D ) /\ E ) -> (/) =/= C ) $.
```

Lemma Synthesis by Compressing Human Structurings Further: Reduction per Topic

$G_{\mathcal{T}}^{\mathsf{TRP}}$ is the result of compressing grammar $G_{\mathcal{T}}$ for topic $\mathcal{T}$ further

Topic $\mathcal{T}$	N($G_{\mathcal{T}}$)	N($G_{\mathcal{T}}^{\mathsf{TRP}}$)	$\lvert G_{\mathcal{T}}\rvert$	$\lvert G_{\mathcal{T}}^{\mathsf{TRP}}\rvert$	Size reduction
Propositional calculus	1,743	1,843	5,760	5,191	10%
Predicate calculus	904	1,050	4,357	3,942	10%
Zermelo-Fraenkel set theory	2,436	2,964	16,343	14,887	9%
Axiom of replacement	4,831	5,753	121,521	115,476	5%
Axiom of choice	240	785	21,693	16,551	24%
Tarski-Grothendieck set theory	147	356	4,986	3,469	30%
Real and complex numbers	5,087	5,986	174,933	163,507	7%
Elementary number theory	852	1,716	96,692	88,497	8%
Basic structures	452	804	10,456	8,925	15%
Basic category theory	543	1230	57,242	51,440	10%
Basic order theory	280	516	7,278	6,112	16%
Basic algebraic structures	2,401	3,318	169,352	163,547	4%
Basic linear algebra	1,018	1,843	88,757	79,573	10%
Basic topology	2,380	3,296	171,054	162,217	5%
Basic real and complex analysis	581	1,451	193,877	182,005	6%
Basic real and complex functions	1,501	2,378	499,438	459,803	8%
Elementary geometry	[514]	–	[139,192]	–	–
Graph theory	1,324	2,193	41,904	37,701	10%
Total	26,720	37,482	1,685,643	1,562,843	7%

Mathematical Knowledge Bases as Grammar-Compressed Proof Terms

1. Introduction
2. Towards Understanding – An Adequate Theory
3. Experiments: A Bird's Eye View on *set.mm*
4. Experiments: Machine Compression vs. Human Structuring
5. **Some Specific Application Possibilities and Related Work**
6. Conclusion

Hammering: Proof and Formula Translation

Hammer systems [Blanchette, Kaliszyk, Paulson, Urban: *Hammering towards QED*, 2016]

1. Premise selector
2. Translation module that constructs an ATP problem
3. Proof reconstruction module that converts the ATP proof for the ITP system

- ***Metamath* Hammer:** [Carneiro, Brown, Urban: *Automated Theorem Proving for Metamath*, 2023]
  - Different formula translations via *Metamath Zero*, into higher-order logic
  - From there to definite first-order clauses
  - *Prover9* yields proofs of first-order Horn problems suitable for proof reconstruction, which involves expanding the resolution proof DAG into a tree
- ***CD Tools Metamath* interface**
  - Formula parsing with Prolog DCG grammars generated on the fly from relevant declarations; **same result as first-order translation of *Metamath* Hammer**
  - Syntax declarations can be used to pretty-print formulas in *Metamath* notation
  - Proof terms are by default without the "syntactic parts"
  - To export proofs for *Metamath*, a procedure that **infers a suitable "syntactic part"**, on the basis of declarations, subject to *Metamath*'s inheritance mechanism of *disjoint variable restrictions*

Premise Selection – Levels of Granularity

- **Kaliszyk, Urban: Learning-Assisted Theorem Proving with Millions of Lemmas, 2015**
  - Relevant lemmas not only named theorems, but also **among lemmas used implicitly in proofs**
  - Can be taken into account at **different levels**
    1. "Atomic" kernel inferences, leading to big data
    2. Combinations of "tactics"
- **Here: grammar-compressed proof structures**
  - **A single representation mechanism that integrates both levels**
    1. Fully expanded proof trees of gigantic size
    2. Lossless grammar compressions
       - Can be verified in fractions of a second
       - Provide with each production a distinguished lemma

- **Here: grammar-based tree compression of proof structures**
  - Proof of lemma $\widehat{=}$ production; lemma formula $\widehat{=}$ MGT
  - Structural properties such as $\mathrm{ref}_G(p)$ and $\mathrm{sav}_G(p)$

- **Schulz: Analyse und Transformation von Gleichheitsbeweisen, 1993**
  - Proof represented by graph: our PDNet but edges flipped ("p occurs as a direct premise of q")
  - Procedure awards status "*lemma*" to nodes with estimated high importance
  - Of 7 investigated criteria the **3 most powerful are structure-based**
    - *Frequently used steps* = $\mathrm{ref}_G(p)$
    - *Important intermediate results* = $\mathrm{sav}_G(p)$ for DAGs
    - *Isolated proof segments*: important *for given proof* if used often within it but rarely from outside

- **Grammar-based tree compression – of formulas involved in proofs**
  - [Vyskocil, Stanovský, Urban: *Automated Proof Compression by Invention of New Definitions*, 2010]
    - Lemmas often uninteresting for mathematicians; definitions costly to learn for humans
  - [Hetzl: *Applying Tree Languages in Proof Theory*, 2012]

Proof Term

$$\mathtt{3syl}(V_1, V_2, V_3) \rightarrow \mathtt{D}(\mathtt{D}(\mathtt{ax\text{-}2}, \mathtt{D}(\mathtt{ax\text{-}1}, V_3)), \mathtt{D}(\mathtt{D}(\mathtt{ax\text{-}2}, \mathtt{D}(\mathtt{ax\text{-}1}, V_2)), V_1))$$

Grammar Compression with Lemmas from *set.mm*

$$\begin{array}{lcl}
\mathtt{a1i}(V_1) & \rightarrow & \mathtt{D}(\mathtt{ax\text{-}1}, V_1) \\
\mathtt{a2i}(V_1) & \rightarrow & \mathtt{D}(\mathtt{ax\text{-}2}, V_1) \\
\mathtt{mpd}(V_1, V_2) & \rightarrow & \mathtt{D}(\mathtt{a2i}(V_2), V_1) \\
\mathtt{syl}(V_1, V_2) & \rightarrow & \mathtt{mpd}(V_1, \mathtt{a1i}(V_2)) \\
\mathtt{3syl}(V_1, V_2, V_3) & \rightarrow & \mathtt{syl}(\mathtt{syl}(V_1, V_2), V_3)
\end{array}$$

DAG-Factorized Combinator Term in D-Syntax

$$\begin{array}{lcl}
f_1 & = & \mathtt{D}(\mathbf{C}, \mathtt{ax\text{-}2}) \\
f_2 & = & \mathtt{D}(\mathtt{D}(\mathtt{D}(\mathbf{C_4}, \mathbf{B}), f_1), \mathtt{ax\text{-}1}) \\
\mathtt{3syl} & = & \mathtt{D}(\mathtt{D}(\mathbf{B}, \mathtt{D}(\mathbf{B}, f_2)), f_2)
\end{array}$$

DAG-Factorized Combinator Term

$$\begin{array}{lcl}
f_1 & = & \mathbf{C}\mathrm{a}_2 \\
f_2 & = & \mathbf{C_4}\mathbf{B} f_1 \mathrm{a}_1 \\
\mathtt{3syl} & = & \mathbf{B}(\mathbf{B} f_2) f_2
\end{array}$$

Combinator Term

$$\mathbf{B}(\mathbf{B}(\mathbf{C_4}\mathbf{B}(\mathbf{C}\mathrm{a}_2)\mathrm{a}_1))(\mathbf{C_4}\mathbf{B}(\mathbf{C}\mathrm{a}_2)\mathrm{a}_1)$$

Some Combinators

	λ-Term	Principal Type
B	$\lambda xyz\,.\,x(yz)$	$(p \Rightarrow q) \Rightarrow ((r \Rightarrow p) \Rightarrow (r \Rightarrow q))$
C	$\lambda xyz\,.\,xzy$	$(p \Rightarrow (q \Rightarrow r)) \Rightarrow (q \Rightarrow (p \Rightarrow r))$
$\mathbf{C_4}$	$\lambda xyzu.x(yu)z$	$(p \Rightarrow (q \Rightarrow r)) \Rightarrow ((s \Rightarrow p) \Rightarrow (q \Rightarrow (s \Rightarrow r)))$

1. Introduction
2. Towards Understanding – An Adequate Theory
3. Experiments: A Bird's Eye View on *set.mm*
4. Experiments: Machine Compression vs. Human Structuring
5. Some Specific Application Possibilities and Related Work
6. Conclusion

Structure-generating ATP | CD | ITP with *Metamath* | Grammar-based tree compression

Theoretical Framework

- MGT: unique most general formula proven by a proof term / determined via unification / specified with inference system / generalizing CD to a definite clauses, body atom $\widehat{=}$ parameter in the proof term / subtlety for nonlinear proof terms
- Proof grammar: compressed representation of proof term / lemma name $\widehat{=}$ nonterminal / proof of lemma $\widehat{=}$ grammar production / MGT for lemma proof determinable directly from grammar
- Theorems can be user-specified strict instances of their proof's MGT

Compression Techniques Beyond TreeRePair

- Nonlinear compression / proof term specific techniques

Implemented System *CD Tools*, Written in *SWI-Prolog*

- Metamath interface / TreeRePair on DAG representation

First Experiments: dataset-oriented methods and human/machine proof structuring

- Properties of set.mm as a grammar / proof dependencies as complex network – it is scale-free
- Human vs. machine proof structurings / lemma synthesis by compressing set.mm further

1. Introduction
2. Towards Understanding – An Adequate Theory
3. Experiments: A Bird's Eye View on *set.mm*
4. Experiments: Machine Compression vs. Human Structuring
5. Some Specific Application Possibilities and Related Work
6. Conclusion

References

References I

[Albert and Barabási, 2002] Albert, R. and Barabási, A.-L. (2002).
Statistical mechanics of complex networks.
Reviews of Modern Physics, 74(1):47–97.

[Benzmüller et al., 2023] Benzmüller, C., Fuenmayor, D., Steen, A., and Sutcliffe, G. (2023).
Who finds the short proof?
Logic Journal of the IGPL.

[Bibel, 1987] Bibel, W. (1987).
Automated Theorem Proving.
Vieweg.
First edition 1982.

[Bibel and Otten, 2020] Bibel, W. and Otten, J. (2020).
From Schütte's formal systems to modern automated deduction.
In Kahle, R. and Rathjen, M., editors, *The Legacy of Kurt Schütte*, chapter 13, pages 215–249. Springer.

References II

[Blanchette et al., 2016] Blanchette, J. C., Kaliszyk, C., Paulson, L. C., and Urban, J. (2016).
Hammering towards QED.
J. Formaliz. Reason., 9(1):101–148.

[Boolos, 1987] Boolos, G. (1987).
A curious inference.
J. Philos. Logic, 16:1–12.

[Broido and Clauset, 2019] Broido, A. D. and Clauset, A. (2019).
Scale-free networks are rare.
Nature Communications, 10(1):1017.

[Carneiro, 2014] Carneiro, M. (2014).
Conversion of HOL Light proofs into Metamath.
CoRR, abs/1412.8091.

[Carneiro, 2020] Carneiro, M. (2020).
Metamath zero: Designing a theorem prover prover.
In Benzmüller, C. and Miller, B., editors, *CICM 2020*, volume 12236 of *LNCS (LNAI)*, pages 71–88. Springer.

References III

[Carneiro et al., 2023] Carneiro, M., Brown, C. E., and Urban, J. (2023).
Automated theorem proving for Metamath.
In Naumowicz, A. and Thiemann, R., editors, *ITP 2023*, volume 268 of *LIPIcs*, pages 9:1–9:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Clauset et al., 2009] Clauset, A., Shalizi, C. R., and Newman, M. E. J. (2009).
Power-law distributions in empirical data.
SIAM Review, 51(4):661–703.

[Curry and Feys, 1958] Curry, H. and Feys, R. (1958).
Combinatory Logic, volume I.
North-Holland.

[Dahn and Wernhard, 1997] Dahn, I. and Wernhard, C. (1997).
First order proof problems extracted from an article in the Mizar mathematical library.
In Bonacina, M. P. and Furbach, U., editors, *FTP'97*, RISC-Linz Report Series No. 97–50, pages 58–62, Linz. Joh. Kepler Univ.

References IV

[Denzinger and Schulz, 1994] Denzinger, J. and Schulz, S. (1994).
Analysis and Representation of Equational Proofs Generated by a Distributed Completion Based Proof System.
Seki-Report SR-94-05, Universität Kaiserslautern.
Revised September 1997.

[Eder, 1985] Eder, E. (1985).
Properties of substitutions and unification.
J. Symb. Comput., 1(1):31–46.

[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.

[Hindley, 1997] Hindley, J. R. (1997).
Basic Simple Type Theory.
Cambridge University Press.

References V

[Hindley and Meredith, 1990] Hindley, J. R. and Meredith, D. (1990).
Principal type-schemes and condensed detachment.
Journal of Symbolic Logic, 55(1):90–105.

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

[Larsson and Moffat, 2000] Larsson, N. J. and Moffat, A. (2000).
Off-line dictionary-based compression.
Proc. IEEE, 88(11):1722–1732.

[Lohrey, 2015] Lohrey, M. (2015).
Grammar-based tree compression.
In Potapov, I., editor, *DLT 2015*, volume 9168 of *LNCS*, pages 46–57. Springer.

[Lohrey et al., 2013] Lohrey, M., Maneth, S., and Mennicke, R. (2013).
XML tree structure compression using RePair.
Inf. Syst., 38(8):1150–1167.
System available from https://github.com/dc0d32/TreeRePair, accessed Jun 30, 2022.

[Loveland, 1968] Loveland, D. W. (1968).
Mechanical theorem proving by model elimination.
JACM, 15(2):236–251.

[McCune, 2010] McCune, W. (2005–2010).
Prover9 and Mace4.
http://www.cs.unm.edu/~mccune/prover9.

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

References VII

[Megill,] Megill, N. D.
Proof Explorer – Home Page – Metamath: A Theorem Sampler.
Online: `https://us.metamath.org/mpeuni/mmset.html#theorems`, accessed Jan 10, 2025.

[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.

[Meredith and Prior, 1963] Meredith, C. A. and Prior, A. N. (1963).
Notes on the axiomatics of the propositional calculus.
Notre Dame J. of Formal Logic, 4(3):171–187.

[Newman, 2018] Newman, M. (2018).
Networks.
Oxford Univ. Press, second edition.

[Prawitz, 1960] Prawitz, D. (1960).
An improved proof procedure.
Theoria, 26:102–139.

[Prawitz, 1969] Prawitz, D. (1969).
Advances and problems in mechanical proof procedures.
Machine Intelligence, 4:59–71.
Reprinted with author preface in J. Siekmann, G. Wright (eds.): *Automation of Reasoning, vol 2: Classical Papers on Computational Logic 1967–1970, Springer, 1983, pp. 283–297.*

[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.

[Rezus, 2020] Rezus, A. (2020).
Witness Theory – Notes on λ-calculus and Logic, volume 84 of *Studies in Logic*.
College Publications, London.

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

References IX

[Schönfinkel, 1924] Schönfinkel, M. (1924).
Über die Bausteine der mathematischen Logik.
Math. Ann., 92(3–4):305–316.

[Stickel, 1988] Stickel, M. E. (1988).
A Prolog technology theorem prover: implementation by an extended Prolog compiler.
J. Autom. Reasoning, 4(4):353–380.

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

[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.

References X

[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 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.

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

References XI

[Wos, 2001] Wos, L. (2001).
Conquering the Meredith single axiom.
J. Autom. Reasoning, 27(2):175–199.