Exploring Metamath Proof Structures

Christoph Wernhard [1] Zsolt Zombori [2]

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

AITP 2024

Aussois, September 4, 2024

We Investigate Ways of Structuring Proofs by Lemmas, Relating Two Aspects

1. **Practice by humans** for formalized proofs
   Metamath: 42,500 proven theorems
2. **Mechanizable possibilities**
   A tree compression algorithm: *TreeRePair*

Questions

- Can we **understand human practice** as a form of structure compression?
- Can automated tools **find structurings of interest** that were overlooked by humans?

Our Approach is Driven by Proof Structures, in Contrast to Formulas

- Nice technical foundation: **condensed detachment**, by Carew A. Meredith – mid 1950s
  - Associates simple proof structures and formulas proven by them
  - Foundation of Metamath

Metamath

- Simple and flexible computer-processable language for **verifying, archiving, and presenting proofs**
- Contributors include **Norman Megill** (founder, since early 1990s), David A. Wheeler, Mario Carneiro
- ***set.mm (Metamath Proof Explorer)***: mathematics from scratch, starting from ZFC set theory axioms
  - Currently 42,500 proven theorems
  - **A single text file ("database")**
- Technical basis is substitution – or **condensed detachment**
  [Norman Megill: *A Finitely Axiomatized Formalization of Predicate Calculus with Equality*, 1995]
- Not tied to any particular set of axioms, instead, axioms are defined in the database
- Syntax is similarly defined via substitution rules in the database
- Specification and introduction: **Metamath book** (available as free PDF)
  [Megill & Wheeler: *Metamath – A Computer Language for Mathematical Proofs*, 2nd ed., 2019]
- **Many tools** support Metamath, instead of requiring a "canonical" tool
- Ranks well in "Formalizing 100 Theorems":
  Isabelle 90; HOL Light 87; Coq 79; Lean 76; **Metamath 74**; Mizar 69; nqthm/ACL2 45

Condensed Detachment (CD): Background

- CD was invented by **Carew A. Meredith** in the mid 1950s
- CD problems were used a lot in ATP in the 1990s, e.g. by Larry Wos
- See [Dolph Ulrich: *A Legacy Recalled and a Tradition Continued*, 2001]

NOTES ON THE AXIOMATICS OF THE PROPOSITIONAL CALCULUS

C. A. MEREDITH and A. N. PRIOR

In this paper the proofs, unless otherw
the bracketed notes introducing each item
The proofs are all compressed by Meredit
the most general result (i.e. without any un
iables) of detaching the formula n, or so
formula m, or some substitution in it.

1. *Łukasiewicz's Deduction Shortened*. (Thi
of Łukasiewicz's proof that *CCCpqrCCrpC*
seems worth including, as Łukasiewicz's c
and not easily obtainable.)

1. *CCCpqrCCrpCsp*
2. *CCCpqpCrp* = DDD1D111n
3. *CCCpqrCqr* = DDD1D1D121n
4. *CpCCpqCrq* = D31
5. *CCCpqCrsCCCqtsCrs* = DDD1D1D1
6. *CCCpqCrsCCpsCrs* = D51
7. *CCpCqrCCpsrCqr* = D64
8. *CCCCCpqrtCspCCrpCsp* = D71
9. *CCpqCpq* = D83
10. *CCCCrpCtpCCCpqrsCuCCCpqrs* = D
11. *CCCCpqrCsqCCCqtsCpq* = DD10.10.n
12. *CCCCpqrCsqCCCqtpCsq* = D5.11
13. *CCCCpqrsCCsqCpq* = D12.6
14. *CCCpqrCCrpp* = D12.9
15. *CpCCpqq* = D3.14
16. *CCpqCCCprqq* = D6.15

*17. *CCpqCCqrCpr* = DD.13D.16.16.13
*18. *CCCpqpp* = D14.9
*19. *CpCqp* = D33

Conquering the Meredith Single Axiom*

LARRY WOS
Mathematics and Computer Science Division, Argonne National Laboratory Argonne, IL 60439-4801, U.S.A. e-mail: wos@mcs.anl.gov

Abstract. For more than three and one-half decades, beginning in the early 1960s, a heavy emphasis

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

His 38-step proof relies mainly on condensed detachment but also in part on substitution and on detachment; for the appropriate clause and inference rule to capture condensed detachment, see the input file in the Appendix. (Be warned that proof length, and proof itself, in the literature can be misleading.) Currently, whether there exists a shorter single axiom for this area of logic remains an open question, one that might be profitably studied with the methodology featured in this article.

* This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

CD: Recently Aroused Interest

Background

- **Explicit proof structures** in a simple and convenient form
- Basis for **fresh views on structure-oriented ATP** (Prawitz, connection method, clausal tableaux), which operates by enumerating proof structures, constrained by unificication of associated formulas

Recent CD-Related Efforts

- **Lemmas**: proof structure as useful information for **Machine Learning** (unit lemmas) [Rawson, W, Z & Bibel TABLEAUX 2023], related AITP 2022-2024 contributions
- **Structure-driven provers**: *SGCD*, solves LCL073-1 and gives a short proof of LCL038-1 [Rawson, W, Z & Bibel TABLEAUX 2023], [W AReCCA 2023], [W & Bibel JAR 2024]
- *CCS* integrates **combinator compression into proof search** [W PAAR 2022]
- **Abstract framework, reductions/regularities**, relations with the connection method [W & Bibel CADE 2021; JAR 2024]
- ***CD Tools*** – Implemented Prolog system with utilities, interfaces, provers and a Metamath interface `http://cs.christophwernhard.com/cdtools/`

The Metamath Interface of *CD Tools*

- Written from scratch in **SWI-Prolog**, included in *CD Tools*
- Formulas: Metamath's sequence representation is parsed to terms
  Yields the same first-order formulas as *mm-hammer* [Carneiro, Brown & Urban 2023]
- Also **proofs are translated to Prolog terms**, with various options
  - Raw form preserves Metamath's compression through factorized terms
  - With and without Metamath's syntax processing steps
  - Forms compatible with *CD Tools*, which offers utilities to process and reduce CD proofs
- Prolog fact base generated from *set.mm*: 120 s; after compilation it loads in 0.5 s
- Not yet addressed
  - Disjoint variable conditions
  - Special handling for `df-cleq`, `weq`, `wceq`, `wcel`, `df-clel`, `wel`, `ax-prv1` and `ax-tgoldbachgt`
  - Translation of proofs to Metamath format (requires introduction of syntax processing steps)

CD: A Technical First-Order Meta-Level Perspective

- CD is traditionally used to establish **completeness of axiomatizations of propositional logics**, by reasoning on a **first-order "meta-level"**
  - **A single unary first-order predicate** P, for *provable* (written $\vdash$ in Metamath)
  - **Operators of the object logic are first-order function symbols**, e.g. $\Rightarrow$

$$
\models \begin{array}{ll}
\forall pq\,[\mathsf{P}(p \Rightarrow q) \land \mathsf{P}(p) \rightarrow \mathsf{P}(q)] \land & \text{Detachment – a Horn clause} \\
\forall pqrs\,\mathsf{P}((((p \Rightarrow q) \Rightarrow r) \Rightarrow ((r \Rightarrow p) \Rightarrow (s \Rightarrow p)))) & \text{Axiom \textit{Łukasiewicz}} \\
\forall pq\,\mathsf{P}(p \Rightarrow (q \Rightarrow p)) \land & \text{Axiom \textit{Simp}} \\
\forall pq\,\mathsf{P}(((p \Rightarrow q) \Rightarrow p) \Rightarrow p) \land & \text{Axiom \textit{Peirce}} \\
\forall pqr\,\mathsf{P}((p \Rightarrow q) \Rightarrow ((q \Rightarrow r) \Rightarrow (p \Rightarrow r))) & \text{Axiom \textit{Syll}}
\end{array}
$$

- Metamath goes further
  - "Condensed generalization": $\forall px\,[\mathsf{P}(p) \rightarrow \mathsf{P}(\mathbb{\forall}(x, p))]$
  - Syntax is handled with the same mechanism as proving

CD: Proof Terms (DG-Terms), The Proves Relation, The Most General Theorem (MGT)

Definition. The ***proves*** relation is defined inductively:

1.	$\boxed{c}$	*proves*	$P\sigma$	if c is an axiom name with formula P
2.	$\boxed{\mathsf{D}(A, B)}$	*proves*	$Q\sigma$	if A *proves* $\boxed{P \Rightarrow Q}$ and B *proves* P
3.	$\boxed{\mathsf{G}(A)}$	*proves*	$\boxed{\forall(x, P)\sigma}$	if A *proves* P

- If DG-term A proves some formula at all, then there is a most general formula P proven by A, called ***the most general theorem (MGT)*** of A
- The MGT of a DG-term for an axiom assignment is **unique** up to renaming of variables
- Type theory view: D is application, the MGT is the principal type

Definition. The ***proves*** relation is defined inductively:

1.	c	*proves*	$P\sigma$	if c is an axiom name with formula P
2.	D(A, B)	*proves*	$Q\sigma$	if A *proves* $P \Rightarrow Q$ and B *proves* P
3.	G(A)	*proves*	$\forall(x, P)\sigma$	if A *proves* P

Prolog View: Computing the MGT

```
mgt(c₁, AxiomFormula₁).
...
mgt(cₖ, AxiomFormulaₖ).
mgt(d(A,B), Q) :-
        mgt(A, (P=>Q)),
        mgt(B, P).
mgt(g(A), forall(X,P)) :-
        mgt(A, P).
```

Example

```
mgt('ax-1', (P=>(Q=>P))). % clause for axiom ax-1

?- mgt(d('ax-1', 'ax-1'), F).
F = (P=>(Q=>(R=>Q))).
```

The Horn MGT of a DG-Term with Variables

Definition. The ***Horn MGT*** of a DG-term $A[v_1, \ldots, v_n]$ **with variables** $v_1, \ldots, v_n$ is the most general Horn clause

$$(P_1 \land \ldots \land P_n) \rightarrow Q$$

s.t. for all σ it holds that if $A_1, \ldots, A_n$ are DG-terms s.t.

$$A_1 \textit{ proves } P_1\sigma \; \ldots \; A_n \textit{ proves } P_n\sigma, \quad \text{then} \quad A[A_1, \ldots, A_n] \textit{ proves } Q\sigma$$

Computing the Horn MGT

```
mgt(V, P) :-
        var(V), !, V = P.
mgt(c1, AxiomFormula1).
...
mgt(ck, AxiomFormulak).
mgt(d(A,B), Q) :-
        mgt(A, (P=>Q)),
        mgt(B, P).
mgt(g(A), forall(X,P)) :-
        mgt(A, P).
```

Example

```
?- mgt(d('ax-1', d(V, 'ax-1')), F).
V = ((P=>(Q=>P))=>R),
F = (S=>R).
```

The Horn MGT of D(ax-1, D(v, ax-1)) is the Horn clause $((p \Rightarrow (q \Rightarrow p)) \Rightarrow r) \rightarrow (s \Rightarrow r)$.

```
?- mgt('ax-1', F).
F = (P=>(Q=>P)).
?- mgt(d('ax-1', d('ax-1', 'ax-1')), F).
F = (P=>(Q=>(R=>(S=>R)))).
```

Extending the Definition of *proves*.

1.	c	*proves*	$P\sigma$	if c is an axiom name with formula P
2.	$\mathsf{D}(A, B)$	*proves*	$Q\sigma$	if A *proves* $P \Rightarrow Q$ and B *proves* P
3.	$\mathsf{G}(A)$	*proves*	$\forall(x, P)\sigma$	if A *proves* P
4.	$f(A_1, \ldots, A_n)$	*proves*	$Q\theta\sigma$	if f is a ***Horn lemma name*** with clause $(P_1 \wedge \ldots \wedge P_n) \rightarrow Q$ and A_1 *proves* $P_1\theta$ $\ldots$ A_n *proves* $P_n\theta$

- Case 4 could (theoretically) cover cases 1–3
  1. f with arity 0
  2. f with Horn clause $((p \Rightarrow q) \wedge p) \rightarrow q$
  3. f with Horn clause $p \rightarrow \forall(x, p)$
- In Metamath proofs, previously proven theorems appear as Horn lemma names

Metamath Proofs in Prolog Term Representation

```
MM> show proof a1i /lemmon /renumber
1 a1i.1        $e |- ph
2 ax-1         $a |- ( ph -> ( ps -> ph ) )
3 1,2 ax-mp    $a |- ( ps -> ph )
MM> show proof a2i
1 a2i.1        $e |-
2 ax-2         $a |-
3 1,2 ax-mp    $a |-
MM> show proof mpd
1 mpd.1        $e |-
2 mpd.2        $e |-
3 2 a2i        $p |-
4 1,3 ax-mp    $a |- ( ph -> ch )
MM> show proof mpi
1 mpi.1        $e |-
2 1 a1i        $p |-
3 mpi.2        $e |-
4 2,3 mpd      $p |-
```

Translation to DGH-Terms

Proof macro	Horn MGT
`a1i(X) -> d('ax-1', X)`	$p \rightarrow (q \Rightarrow p)$
`a2i(X) -> d('ax-2', X)`	$(p \Rightarrow (q \Rightarrow r)) \rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))$
`mpd(X, Y) -> d(a2i(Y), X)`	$[(p \Rightarrow q) \land (p \Rightarrow (q \Rightarrow r))] \rightarrow (p \Rightarrow r)$
`mpi(X, Y) -> mpd(a1i(X), Y)`	$[p \land (q \Rightarrow (p \Rightarrow r))] \rightarrow (q \Rightarrow r)$

Expansion to DG-Terms

```
a1i(X) -> d('ax-1', X)
a2i(X) -> d('ax-2', X)
mpd(X, Y) -> d(d('ax-2', Y), X)
mpi(X, Y) -> d(d('ax-2', Y), d('ax-1', X))
```

Some Statistics for *set.mm*

- 42,548 proven theorems
- For 10% the stated theorem is a **strict instance** of the Horn MGT of the (unexpanded) proof
- For 59% the associated Horn clause has a **non-empty body**
- Fully expanding the proofs to obtain a DG-term can yield quite large results, e.g. for `peano5`
  - $1,415$ different theorems used in total
  - DG-term has $7.52e \times 10^{46}$ inner nodes, DAG representation has $42,830$ inner nodes

Proof macro	Horn MGT
`a1i(X) -> d('ax-1', X)`	$p \rightarrow (q \Rightarrow p)$
`a2i(X) -> d('ax-2', X)`	$(p \Rightarrow (q \Rightarrow r)) \rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))$
`mpd(X, Y) -> d(a2i(Y), X)`	$[(p \Rightarrow q) \wedge (p \Rightarrow (q \Rightarrow r))] \rightarrow (p \Rightarrow r)$
`mpi(X, Y) -> mpd(a1i(X), Y)`	$[p \wedge (q \Rightarrow (p \Rightarrow r))] \rightarrow (q \Rightarrow r)$

Expansion to DG-term
`mpd(X, Y) -> d(d('ax-2', Y), X)`
`mpi(X, Y) -> d(d('ax-2', Y), d('ax-1', X))`

Correspondences – Ways to Understand these “Proof Macros”

- Proof of a Horn lemma formula
- DGH-term with variables
- Rewrite rule for DGH-terms
- Structural building block for proof search
- **Tree grammar rule of a compressed tree representation**

Proof macro	Horn MGT
`a1i(X) -> d('ax-1', X)`	$p \rightarrow (q \Rightarrow p)$
`a2i(X) -> d('ax-2', X)`	$(p \Rightarrow (q \Rightarrow r)) \rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))$
`mpd(X, Y) -> d(a2i(Y), X)`	$((p \Rightarrow q) \land (p \Rightarrow (q \Rightarrow r))) \rightarrow (p \Rightarrow r)$
`mpi(X, Y) -> mpd(a1i(X), Y)`	$(p \land (q \Rightarrow (p \Rightarrow r))) \rightarrow (q \Rightarrow r)$

Expansion to DG-term
`mpd(X, Y) -> d(d('ax-2', Y), X)`
`mpi(X, Y) -> d(d('ax-2', Y), d('ax-1', X))`

Tree Compression Algorithm *TreeRePair*: Background

- By [Lohrey, Maneth & Mennicke, 2010, 2013]
- Originally addressed XML compression
- Adaptation to trees of **RePair** for strings [Larsson & Moffat, 1999]
- The compressed tree is represented by an SLCF **tree grammar**
  - Straight-line: each nonterminal has exactly one production; acyclic
  - Nonterminals with parameters (rank ≥ 0)

Examples of such Grammar Rules

```
a1i(X) -> d('ax-1', X)
a2i(X) -> d('ax-2', X)
mpd(X, Y) -> d(a2i(Y), X)
mpi(X, Y) -> mpd(a1i(X), Y)
```

TreeRePair

- RePair for strings: **replace most frequent digram (2 consecutive letters) by new nonterminal**

xabcabdabc	
$\mathrm{x}A\mathrm{c}A\mathrm{d}A\mathrm{c}$	$A \to \mathrm{ab}$
$\mathrm{x}BA\mathrm{d}B$	$B \to A\mathrm{c}$
	$\mathit{Start} \to \mathrm{x}BA\mathrm{d}B$

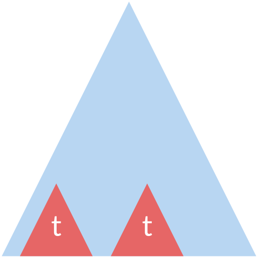
- TreeRePair: **digrams are triples ⟨*parent-symbol, child-index, child-symbol* ⟩**

f(g(e, e), f(g(e, e), e)	f(g(_, _), _)	$\langle \mathrm{f}, 1, \mathrm{g}\rangle$
f(g(e, e), f(g(e, e), e)	g(e, _)	$\langle \mathrm{g}, 1, \mathrm{e}\rangle$
f(g(e, e), f(g(e, e), e)	g(_, e)	$\langle \mathrm{g}, 2, \mathrm{e}\rangle$

f(g(e, e), f(g(e, e), e)	
$A(\mathrm{e}, \mathrm{e}, A(\mathrm{e}, \mathrm{e}, \mathrm{e}))$	$A(x_1, x_2, x_3) \to \mathrm{f}(\mathrm{g}(x_1, x_2), x_3)$
$B(\mathrm{e}, B(\mathrm{e}, \mathrm{e}))$	$B(x_1, x_2) \to A(\mathrm{e}, x_1, x_2)$
$C(C(\mathrm{e}))$	$C(x_1) \to B(\mathrm{e}, x_1)$
	$\mathit{Start} \to C(C(\mathrm{e}))$

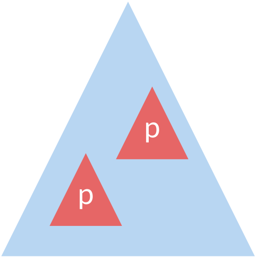
TreeRePair: Grammar Compression Generalizes DAG Compression

DAG: sharing repeated **subtrees**




```
biimp -> syl(d(simplim,'df-bi'),simplim)
simplim -> con1i('pm2.21')
```

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




```
'pm3.48' -> jaao(imim2i(orc),imim2i(olc))
imim2i(X) -> a2i(a1i(X))
```

Experiments with TreeRePair on Metamath Proofs

Basic Idea

- For a **set of theorems from Metamath**, take all their **fully expanded proofs** (DG-terms)
- **Apply TreeRePair to compress the set of trees into a grammar**
- **The grammar represents DGH-terms that prove Horn lemmas**
- **Inspect the generated lemmas:**
  - Are they among the theorems present in Metamath?
    Can usefulness in terms of human practice be explained by compression?
  - Are they new?
    Do they suggest overlooked economic ways to structure the proofs?

But

- Fully expanded proofs can be really large: `peano5`: $7.52e \times 10^{46}$ (tree), $42,830$ (DAG), $1,415$ lemmas

Approach

- Process the proof set **in accumulating subsets** and **don't expand all lemmas** – strategies:
  - Don't expand if it was a rediscovered Metamath theorem
  - Expand only a few steps with limit of expansion size

Outline of the Results of Two Experimental Runs

Our Variation of TreeRePair

- Implemented in SWI-Prolog
- Adapted to our needs: accumulative processing of sets of proof structures
- Heuristic possibilities:
  - Controlling frequency threshold of digrams to be chosen as nonterminal
  - Controlling choice among equally frequent digrams

Experiment *Peano*

- The full set consists of the proofs of the 1,415 theorems involved in proving `peano5`
- We generate a well compressing grammar for all proofs in the set; takes 32 s
- We **rediscover around 50%** of the 1,415 processed theorems
- There are also a **handful of novel lemmas, 23**, depending on configuration

Experiment *1600*

- The full set consists of the proofs of the first 1,600 theorems in *set.mm*
- 40% of the 1,600 rediscovered; 33 novel lemmas; takes 40 s

Experiment *Peano*: Newly Found Lemmas – Proofs and Horn MGTs (I)

```
lemma9(X,Y,Z) -> d(Z,d(Y,X)).
$e |- A $.
$e |- ( A -> B ) $.
$e |- ( B -> C ) $.
$p |- C $.
```

Double modus ponens inference mp2b, which is not used for peano5

```
lemma5486 -> sylbi(ordeleqon,
      jaoi(nsyl2(mtbii(d(ordirr, ordom), baib(elom)),
                 alrimiv(com12(con1d(syld(ord(mpbid(limomss,
                                                   sylancr(ordom,
                                                           limord,
                                                           ordsseleq))),
                                         biimprcd(limeq)))))),
           mpbiri(mpbir3an(ordon, onn0, eqcomi(unon), 'df-lim'), limeq)))
$p |- ( Ord _om -> Lim _om ) $.
```

If ω is an ordinal, it is a limit ordinal (ω is the class of natural numbers)
This is limom weakened by the precondition
Appears in the MM proof of limom immediately before the last step

Experiment *Peano*: Newly Found Lemmas – Proofs and Horn MGTs (II)

```
lemma2808(X) -> '3bitr4g'(exbidv(anbi1d(X)),dfclel,dfclel).
$e |- ( A -> ( B = C <-> B = D ) ) $.
$p |- ( A -> ( C e. E <-> D e. E ) ) $.
```

If we know that given A, B equals C exactly when B equals D, then
it follows that given A, C and D are members of the same sets

```
lemma3104(X) -> bitr4i(X,albii(nbn(noel))).
$e |- ( A <-> A. B ( C <-> D e. (/) ) ) $.
$p |- ( A <-> A. B -. C ) $.
```

If we know that A is equivalent to forall B.(C is equivalent to D in empytset), then
it follows that A is equivalent to forall B.(not C)

```
lemma3108(X,Y) -> eqtr4di(eqtrd(a1i(X),abbidv(Y)),X).
$e |- A = { B | C } $.
$e |- ( D -> ( C <-> C ) ) $.
$p |- ( D -> A = A ) $.
```

Let A be the set of elements satisfying formula C; Then, if D implies that C is equivalent to itself, then D implies that A = A
Not clear where this is really used/useful

Experiment *Peano*: Newly Found Lemmas – Proofs and Horn MGTs (III)

```
lemma3193(X,Y) -> eqcomd(eqtrdi(abbi2dv(X),eqcomi(Y))).
$e |- ( A -> ( B e. C <-> D ) ) $.
$e |- E = { B | D } $.
$p |- ( A -> E = C ) $.
```

Given A, if we know that B is a member of C exactly when B satisfies some property D and we also know that E is the set of B satisfying property D, then given A, it follows that E and C are the same set

```
lemma4618(X) -> sseqtrri(sstri(sseqtrri(X,'df-pr'),ssun1),'df-tp').
$e |- A C_ ( { B } u. { C } ) $.
$p |- A C_ { B , C , D } $.
```

If A is a subset of {B} union {C}, then it is a subset of {B,C,D}

Experiment *1600*: Newly Found Lemmas – Proofs and Horn MGTs (I)

```
lemma14(X,Y,Z) -> d(d(Z,Y),X).
$e |- A $.
$e |- B $.
$e |- ( B -> ( A -> C ) ) $.
$p |- C $.

lemma17(X) -> bicomi(con2bii(X)).
$e |- ( A <-> -. B ) $.
$p |- ( -. A <-> B ) $.

lemma35(X) -> lemma14(impsingle,d(impsingle,X),impsingle).
$e |- ( ( A -> B ) -> ( ( A -> C ) -> ( D -> C ) ) ) $.
$p |- ( E -> ( ( A -> C ) -> ( D -> C ) ) ) $.

lemma79 -> adantl(id).
$p |- ( ( A /\ B ) -> B ) $.
```

Experiment *1600*: Newly Found Lemmas – Proofs and Horn MGTs (II)

```
lemma89(X) -> lemma14(impsingle,impsingle,lemma35(D(impsingle,X))).
$e |- ( ( ( A -> ( ( B -> C ) -> ( D -> C ) ) ) -> E ) -> ( ( C -> F ) -> B ) ) $.
$p |- ( A -> ( ( B -> C ) -> ( D -> C ) ) ) $.

lemma205(X,Y) -> impd(syld(X,expd(ancomsd(Y)))).
$e |- ( A -> ( B -> C ) ) $.
$e |- ( A -> ( ( D /\ C ) -> E ) ) $.
$p |- ( A -> ( ( B /\ D ) -> E ) ) $.

lemma266(X,Y) -> bitri(xchbinx(X,Y),bitru(fal)).
$e |- ( A <-> -. B ) $.
$e |- ( B <-> F. ) $.
$p |- ( A <-> T. ) $.

lemma347 -> lemma14(impsingle,
                    lemma89('impsingle-step4'),
                    lemma89(d(impsingle,lemma89('impsingle-step4')))).
$p |- ( ( ( ( ( A -> B ) -> C ) -> D ) -> ( E -> A ) ) ->
        ( ( C -> A ) -> ( E -> A ) ) ) $.
```

Experiment *1600*: Newly Found Lemmas – Proofs and Horn MGTs (III)

```
lemma728 -> com12(con4d(ex(sylc(ancoms(syl2an(id,sylib(olc,con2bii(bicomi(ioran))),id)),
                                sylib(syl(simpl,orc),con2bii(bicomi(ioran))),
                                con3d('pm2.27'))))).
$p |- ( ( ( -. A /\ -. ( -. B /\ -. C ) ) -> ( -. C /\ -. D ) ) -> ( C -> A ) ) $.

lemma734(X) -> '3bitri'('3anbi123i'('3anass',
                                    bitri('3anass',biancomi(bianass(ancom))),
                                    bitri(bitr4i('3ancoma',bitri('3ancoma','3ancomb')),
                                          '3anass')),
                        bicomi(an6),anbi2i(X)).
$e |- ( ( ( A /\ B ) /\ ( C /\ D ) /\ ( E /\ F ) ) <-> G ) $.
$p |- ( ( ( H /\ A /\ B ) /\ ( C /\ I /\ D ) /\ ( F /\ E /\ J ) ) <->
        ( ( H /\ I /\ J ) /\ G ) ) $.
```

Agenda and Speculations

- Further experiments, deeper analysis of the generated lemmas
- Heuristics and refinements of TreeRePair
- Bringing provers and learning into play
- **Combinators** in proof-terms provide a further representation of lemmas; it is variable-free

```
mpi(X, Y) -> d(d('ax-2', Y), d('ax-1', X))
mpi -> d(d(b, d(c, 'ax-2')), 'ax-1')
```

- Can we **categorize lemmas**, e.g., general inference rule or specific for a certain application area, based on compressing effects and occurrences in given sets of proofs?
- Do **mechanically observed redundancies** in human-made proofs (they are at least in many small Metamath proof) have a beneficial purpose?

Conclusion

- We observed a **correspondence** of
  - **Condensed detachment proof structures, generalized to allow variables**
  - **Horn clauses** proven by these structures
  - **Metamath proofs**
  - **Grammar rules representing compressed trees**
- We utilize this for **lemma synthesis purely from proof structure**
  - **A lemma is justified by its compressing effect on the proof structure**
  - The lemma *formula* comes second, it is computed from the proof structure
- We **implemented** all this: our programs can directly read-in Metamath database files
- First **experiments** with Metamath's *set.mm* database show:
  - About one half of the human-made lemmas can be justified by mechanically reproducible compression effects
  - Mechanical compression suggests a few novel lemmas

References I

[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.

[Larsson and Moffat, 1999] Larsson, N. J. and Moffat, A. (1999).
Off-line dictionary-based compression.
In *DCC'99*, pages 296–305. IEEE.

[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.

[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 II

[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.

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

[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.

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

References III

[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.

References IV

[Wernhard and Bibel, 2024] Wernhard, C. and Bibel, W. (2024).
Investigations into proof structures.
J. Autom. Reasoning.
to appear, preprint `https://arxiv.org/abs/2304.12827`.

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