Learning to Identify Useful Lemmas from Failure

Michael Rawson [1]   Christoph Wernhard [2]   Zsolt Zombori [3,4]

[1]TU Wien [2]University of Potsdam [3]Alfréd Rényi Institute of Mathematics [4]Eötvös Loránd University

AITP2023

Aussois, France, September 3-8, 2023

Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 457292495, by the North-German Supercomputing Alliance (HLRN), by the ERC grant CoG ARTIST 101002685, by the Hungarian National Excellence Grant 2018-1.2.1-NKP-00008, the Hungarian Artificial Intelligence National Laboratory Program (RRF-2.3.1-21-2022-00004), the ELTE TKP 2021-NKTA-62 funding scheme and the COST action CA20111.

National Research, Development and Innovation Office
Hungary
PROJECT FINANCED
FROM THE NRDI FUND

Learning to Identify Useful Lemmas from Failure

Lemmas to Aid Automated Proof Search

Explore the benefit of identifying/using lemmas to aid proof search

- Lemmas can make the proof shorter
- Lemmas can make selecting the next inference harder
- Ideally, we would like to identify just a few relevant lemmas
- Similar to premise selection, but we assume no given premise set

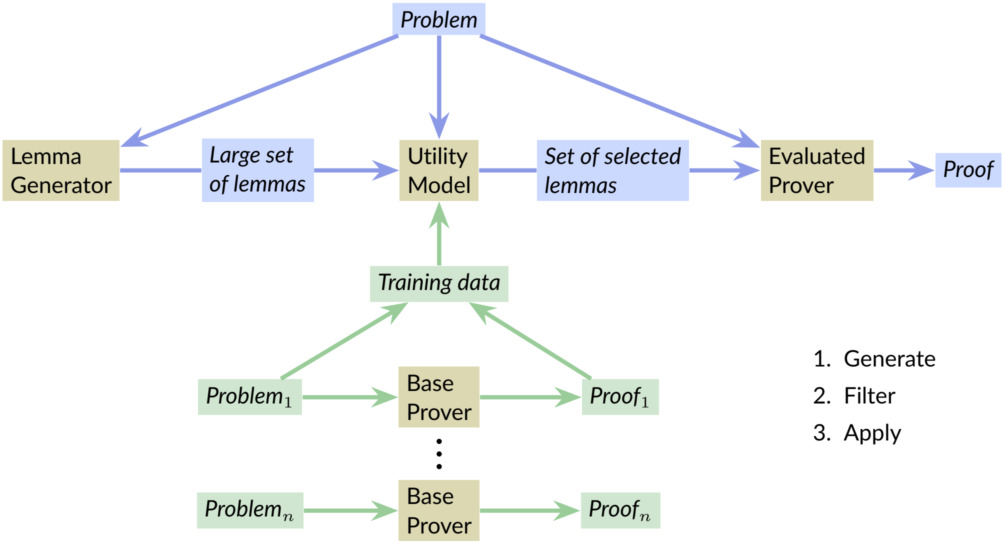
Rawson, Wernhard, Zombori, Bibel. Lemmas: Generation, Selection, Application. To appear at TABLEAUX2023

Dataset

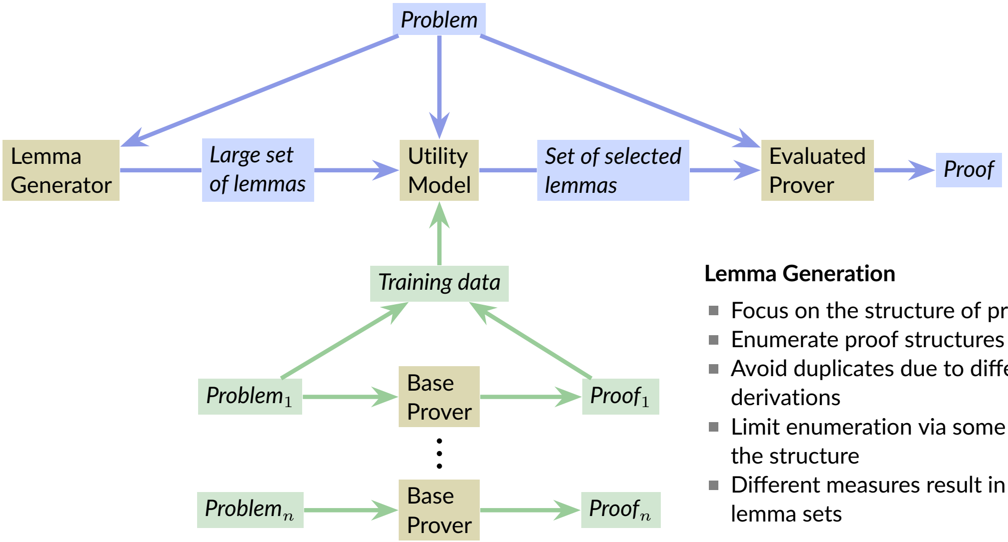
Restrict attention to Condensed Detachment (CD) problems

Detachment axiom	$\mathsf{P}(\mathsf{i}(x, y)) \land \mathsf{P}(x) \to \mathsf{P}(y)$	
Proper axioms	units	e.g. $\mathsf{P}(\mathsf{i}(\mathsf{i}(\mathsf{i}(x, y), z), \mathsf{i}(\mathsf{i}(z, x), \mathsf{i}(u, x))))$
Goal	negative ground unit	e.g. $\neg\mathsf{P}(\mathsf{i}(\mathsf{a}, \mathsf{i}(\mathsf{b}, \mathsf{a})))$

- Horn, first-order variables, binary function symbol, cyclic predicate dependency
- Generalization to arbitrary Horn problems is possible
- Proofs have a simple regular tree structure (D-terms)
- D-terms are convenient for feature extraction and for structure enumeration

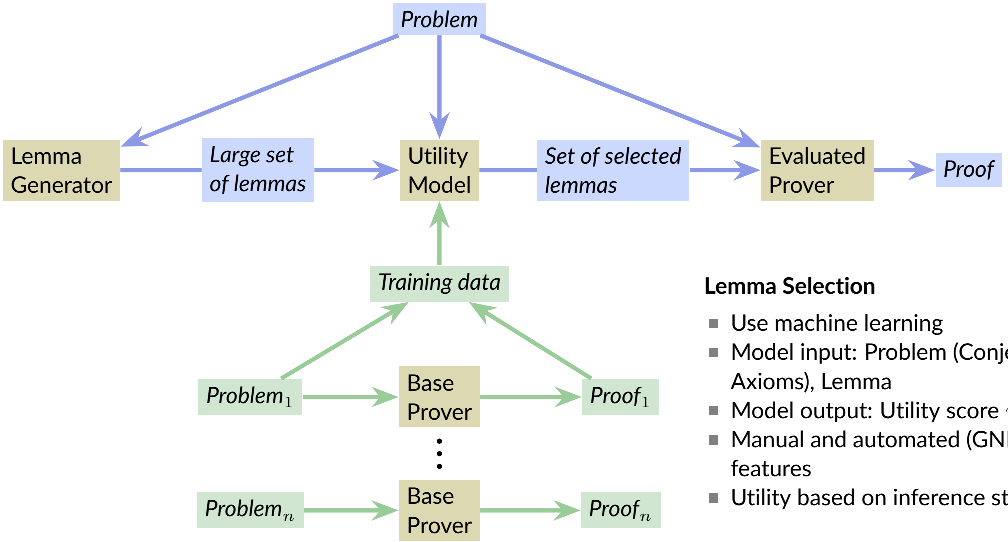


1. Generate
2. Filter
3. Apply



Lemma Generation

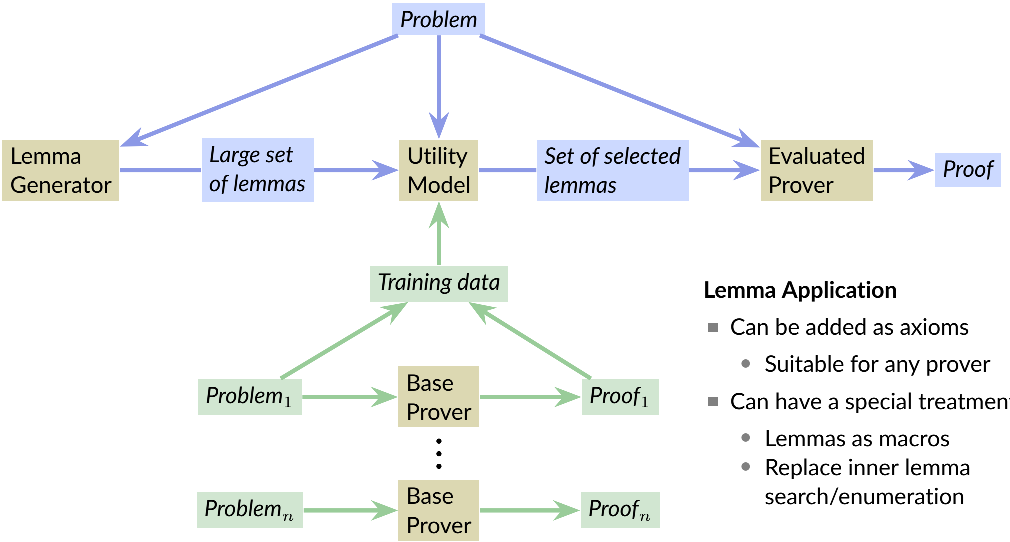
- Focus on the structure of proofs
- Enumerate proof structures
- Avoid duplicates due to different derivations
- Limit enumeration via some measure on the structure
- Different measures result in very different lemma sets



Lemma Selection

- Use machine learning
- Model input: Problem (Conjecture + Axioms), Lemma
- Model output: Utility score $u \in [0, 1]$
- Manual and automated (GNN) input features
- Utility based on inference step reduction

Method overview



Lemma Application

- Can be added as axioms
  - Suitable for any prover
- Can have a special treatment
  - Lemmas as macros
  - Replace inner lemma search/enumeration

Iterative Improvement

- Start from a set of problems
- Search from proofs
- Learn from proof attempts
- Fit a model
- Start search again, using the learned model

Learning Requirements, Considered Provers

- Lemma generation requires proof structure enumeration (SGCD)
- We require provers that emit proofs as D-terms (SGCD, Prover9, CMProver, CCS)
- Any prover can be used for evaluation

	SGCD	Prover9	CMProver	leanCoP	CCS-Vanilla	Vampire	E
Goal-driven	●/–	–	●	●	●	○	○
CM-CT	○	–	●	●	–	–	–
Proof Structure Enumeration	●	–	●	○	●	–	–
Resolution / Superposition	–	●	–	–	–	●	●
Output proof as D-term	●	●	●	–	●	–	–
Input lemmas that replace search	●	–	–	–	●	–	–

Learning from Successful Proof Attempts

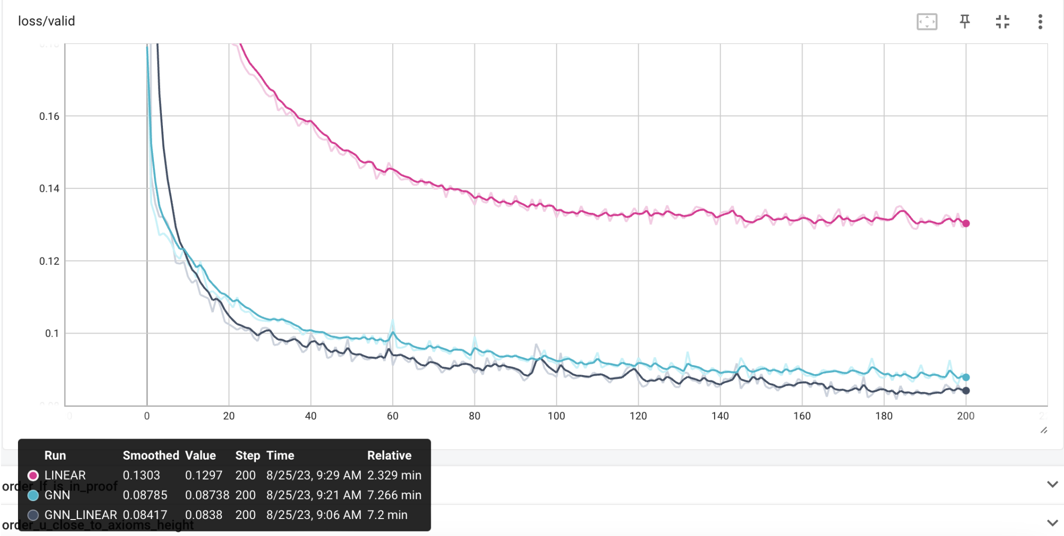
- Utility measure calculation requires a prover that can produce a proof tree structure
- Given a proof, any substructure can be considered as a lemma that we can learn from
- Lots of training signal from a single proof, if the proof is long
- Different proofs of the same problem can be used

Learning from Failed Proof Attempts

- Any proof attempt constructs a sequence of incomplete proof structures
- Most of these have complete substructures
- These are proof terms of formulas proven as a byproduct of proof search
- We can use any such substructures as a proof to learn from
- Similar to Hindsight Experience Replay [Andrychowicz et al., 2017]
  - Pretend that we wanted to prove what we accidentally proved
- Provides huge amounts of training data from failed proofs
  - ˜100K samples with 10 sec timeout

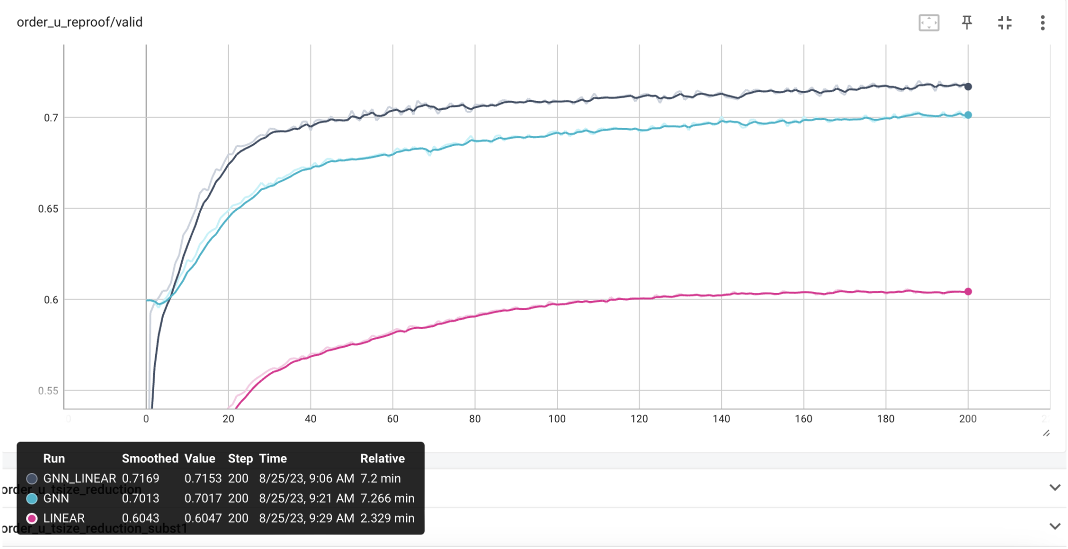
Model Fitting: Linear Model vs Graph Neural Network

Validation Loss



Model Fitting: Linear Model vs Graph Neural Network

Ability to predict correct order



Problemwise learning from failed attempts

- Prover: SGCD (provecd_sgcd_s1.pl)
- Time limit: 10 sec
- Total problems: 312

Train a separate model for each problem

- Prover: SGCD (provecd_sgcd_s1.pl)
- Time limit: 10 sec
- Total problems: 411

Train a single model for all problems.

Learn from	Iteration					Total
	0	1	2	3	4	
success	199	203	206	216	205	222 (+23)
failure	199	211	219	209	205	229 (+30)
both	199	212	207	**223**	200	**230 (+31)**

Learning both from failed and successful proof attempts

- Prover: portfolio of diverse SGCD strategies (f_sgcd_tsize)
- Time limit: 10 sec
- Total problems: 411

Learn from	Iteration					Total
	0	1	2	3	4	
both	236	257	246	249	244	263 (+27)

Learning to Identify Useful Lemmas from Failure

1. Learning to Identify Useful Lemmas
2. Learning from Successful as well as Failed Proof Attempts
3. Experiments
4. **Learning Subtree/Unit Lemmas**
5. Conclusion

Conclusion

- Lemmas are helpful to find a proof
- Generate, filter, apply lemmas
- A lot of signal can be extracted from failed proof attempts that is useful for learning
- Lemma generation brings a bit of resolution into non-resolution based provers
- Blurs the distinction between forward and backward reasoning

References I

[Andrychowicz et al., 2017] Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. (2017).
Hindsight experience replay.
In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[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.

[Łukasiewicz, 1948] Łukasiewicz, J. (1948).
The shortest axiom of the implicational calculus of propositions.
In *Proc. of the Royal Irish Academy*, volume 52, Sect. A, No. 3, pages 25–33.
Republished in [Łukasiewicz, 1970], p. 295–305.

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

References II

[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.

[Rawson et al., 2023] Rawson, M., Wernhard, C., Zombori, Z., and Bibel, W. (2023).
Lemmas: Generation, selection, application.
CoRR, abs/2303.05854.
Submitted, preprint: `https://arxiv.org/abs/2303.05854`.

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

[Wernhard, 2022a] Wernhard, C. (2022a).
CD Tools – Condensed detachment and structure generating theorem proving (system description).
`https://arxiv.org/abs/2207.08453`.

[Wernhard, 2022b] Wernhard, C. (2022b).
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.
Preprint: `https://arxiv.org/abs/2209.12592`.

References III

[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, 2023] Wernhard, C. and Bibel, W. (2023).
Investigations into proof structures.
Preprint, `http://cs.christophwernhard.com/papers/investigations/`.

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