Lean4trace: Data augmentation for neural theorem proving in Lean

Vasilii Nesterov[1], Yermek Kapushev[2], Mikhail Burtsev[3]

[1]Moscow Institute of Physics and Technology, [2]Yandex, [3]London Institute for Mathematical Sciences

Problem statement and motivation

- PROBLEM: training data for formal theorem proving is **very scarce**.
- SOLUTION: **data augmentation**.

We release **Lean4trace**[a] – tool for data exctration from Lean 4 sources. Its advantages are:

- Deep integration into Lean 4 compiler. Lean4trace works **along** with Lean 4 compiler and has full access to the internal state of the compiler.
- Ability of proof modification on-the-fly. It allows us to augment data by modifying existing proofs.
- Small overhead of RAM in comparison with other tools.

Augmentation 1: Tactic decomposition

- Human-written proofs are often compressed, meaning that we can potentially extract more than one proof state from a single tactic. We take two most frequent tactics in Mathlib: `rw` and `simp` and decompose its complex applications into elemental ones.
- `rw` applies given rules in given order, so the decomposition is simple.
- `simp` applies rules in arbitrary order, so Breadth-First-Search is used.
- By this we augment the dataset with **all intermediate proof states**.

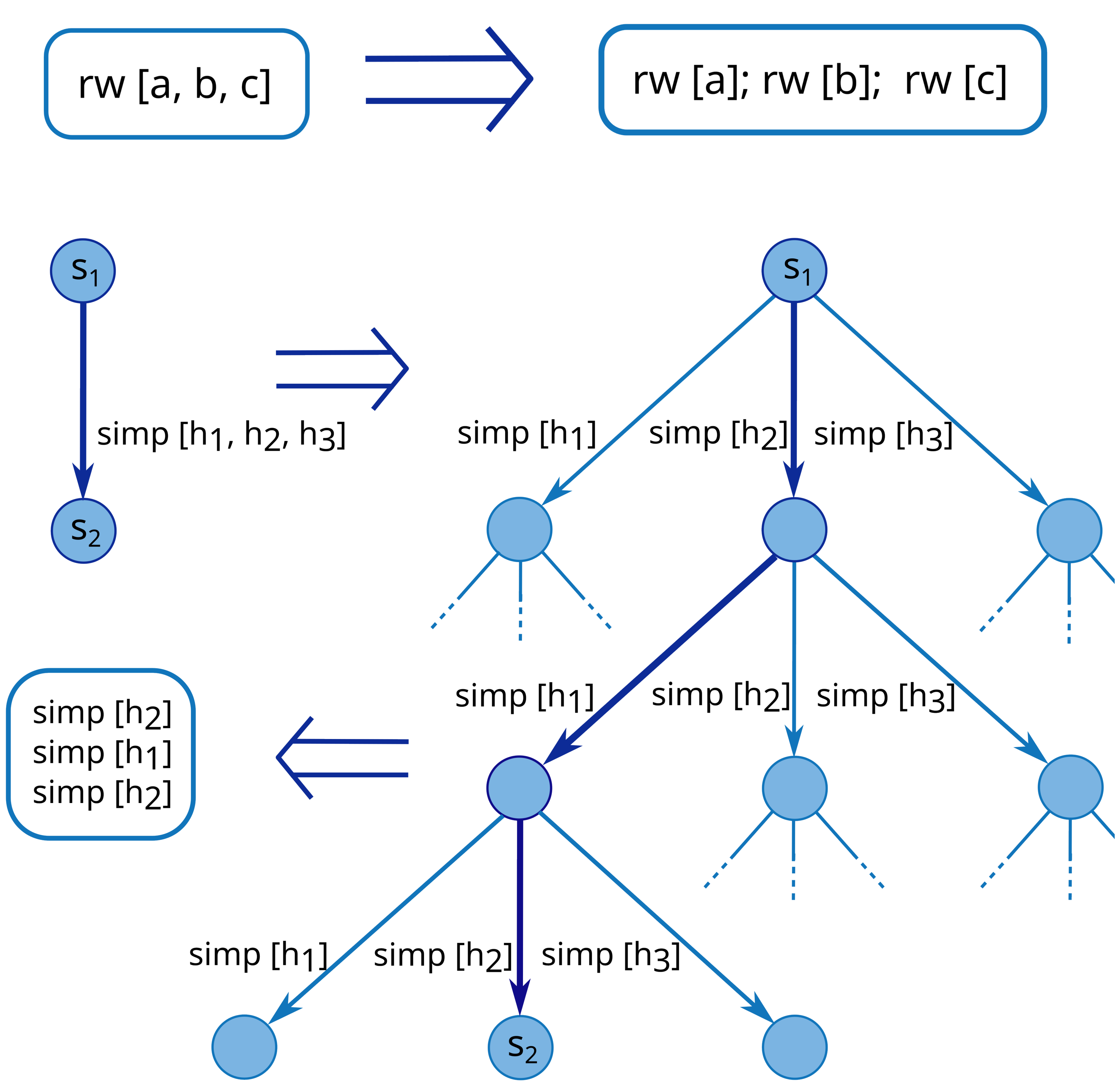

Figure 1:Decomposition of the `rw` and `simp` tactics. In the original proof, the proof state s_2 is obtained by applying `simp [h₁, h₂, h₃]` in state s_1. Using BFS, we find a sequence `simp [h₂]; simp [h₁]; simp [h₂]` which also leads to s_2. In this example, h_2 is used twice, and h_3 can be omitted. Such situations actually occur in Mathlib proofs.

[a]https://github.com/vasnesterov/Lean4trace

Comparison with LeanDojo extraction

Our tool works faster and requires less RAM while extracting more proof states and allowing augmentations.

Dataset	# proof states	Time	RAM, GB
LeanDojo tracing	273k	1 h	48
Canonical (our tracing)	352k	31 min	17
`rw` decomposition	110k	34 min	18
`simp` decomposition	37.7k	11 h	24
Automatic tactics	318k	7 days	10

Table 1:Resources required for tracing.

Augmentation 2: Automatic tactics

- Some tactics in Lean are designed for non-trivial proof automation and require no guidance from the user. We test such tactics against every proof state in augment dataset with **all successful applications** (i.e. when the tactic finish the proof).
- Statistics shows that automatic tactic are used far rarely than can be (see below). In total, automatic tactics can close 23.6% goals.

Automatic tactic	Solved goals, %	Frequency in source, %
aesop	21.8	0.13
simp_all	16.6	<0.01
simp_arith	9.6	<0.01
tauto	8.9	0.08
solve_by_elim	7.8	0.02
norm_num	5.6	0.02
abel	1.5	0.08
omega	1.4	0.01
nlinarith	1.1	0.01

Table 2:Number of goals can be solved by auto tactics.

Results in theorem proving

- We use the same training/evaluation pipeline as with LeanDojo[a], but vary training data.
- Both augmentations improves quality on Mathlib dataset.
- We achieve best known Pass@1 with very small model (only 299M parameters).

Model & training data	Mathlib	MiniF2F
ReProver		
LeanDojo data	48.6	26.5
Canonical	56.3	**35.6**
Canonical + Tactics decomposition	**58.0**	30.0
Canonical + Automatic tactics	57.6	33.6
Thor + expert iteration		35.2
COPRA + GPT-4		30.7
Thor		29.9
Lean Expert Iteration		29.6

Table 3:Pass@1 for theorem proving.

[a]see "LeanDojo: Theorem Proving with Retrieval-Augmented Language Models" paper: https://arxiv.org/abs/2306.15626