Modeling Tactics as Operators: Effect-Grounded Representations for Lean Theorem Proving

Elisaveta Samoylov **Soroush Vosoughi**
Department of Computer Science, Dartmouth College
{elisaveta.v.samoylov.26@dartmouth.edu, Soroush.Vosoughi}@dartmouth.edu

Abstract

Interactive theorem provers (ITPs) such as Lean expose proof construction as a sequence of tactics applied to proof states. Existing machine learning approaches typically treat tactics either as surface tokens or as labels conditioned on the current state, eliding their operator-like semantics. This paper introduces a representation learning framework in which tactics are characterized by the changes they induce on proof states. Using a stepwise Lean proof corpus, we construct *delta contexts*—token-level additions/removals and typed structural edits—and train simple distributional models (Δ-SGNS and CBOW-Δ) to learn tactic embeddings grounded in these state transitions. Experiments on tactic retrieval and operator-style analogy tests show that Δ-supervision yields more interpretable and generalizable embeddings than surface-only baselines. Our findings suggest that capturing the semantics of tactics requires modeling their state-transformational effects, rather than relying on distributional co-occurrence alone.

1 Introduction & Related Work

Interactive theorem provers (ITPs) such as Lean expose proof construction as a sequence of *tactics* that transform a proof *state*, the local context and current goal(s), until all goals are discharged. Learning to select or synthesize tactics has therefore become a central problem in neural theorem proving, with notable progress in HOL4, HOL Light, and Coq through systems such as TacticToe, HOList, and CoqGym (Gauthier et al., 2021; Bansal et al., 2019; Yang and Deng, 2019). In parallel, LeanDojo has catalyzed research around Lean by providing an open, reproducible environment, datasets, and retrieval-augmented provers (Yang et al., 2023).

Yet, most existing approaches model tactics either as labels to be predicted from the *current* state or as tokens in tactic sequences, rather than as *operators that cause state change*. This elides the essential semantics of tactics, *"what they do to the proof state?,"* and can yield brittle representations tied to surface forms, local names, or co-occurrence statistics.

Problem formulation. We propose a representation-learning view in which a tactic, t, is characterized by its *effect on the state*. Given triples $(s_{\text{before}}, t, s_{\text{after}})$ from stepwise Lean proofs, we learn embeddings that make $E(s_{\text{after}})$ predictable from $E(s_{\text{before}})$ and an operator-like embedding e_t. Concretely, we use simple CBOW/skip-gram objectives in which the *delta* between `state_before` and `state_after`, i.e., token-level additions/removals and typed edits such as goal/hypothesis count changes or relation normalizations, serves as the context of t. This Δ-supervision biases the model to encode a tactic by the *modification it induces* (e.g., the tactic `have` introduces new hypotheses, the tactic `nlinarith` often discharges inequality goals, etc) rather than by memorizing surface patterns.

Proof of concept. This paper is a proof-of-concept showing that surface-level distributional patterns that often suffice in natural-language modeling are inadequate for theorem proving. Capturing tactic semantics demands tailored, effect-grounded representations that model the proof-state transition induced by each tactic.

We instantiate this idea on the **Lean Workbook** corpus (Ying et al., 2024; InternLM, 2024), which contains tens of thousands of contest-level math problems formalized in Lean 4 with natural-language statements, formal statements, and stepwise proof states. Our models are deliberately minimal: (i) Δ-*SGNS* (center = tactic; contexts = positive Δ-tokens and typed edits) and (ii) *CBOW-*Δ (contexts $\rightarrow$ tactic), trained in the spirit of word2vec (Mikolov et al., 2013a,b). We evaluate with (a) $\Delta \rightarrow$ *tactic retrieval* (does a bag of edits identify the tactic?) and (b) an *operator analogy* test adapted from translation embeddings (Bordes et al., 2013): $\cos\big(E(s_{\text{before}}) + e_t,\ E(s_{\text{after}})\big)$.

Despite their simplicity, these models learn meaningful, interpretable tactic embeddings that reflect operator-like behavior.

Motivation and expected benefits. Two gaps motivate Δ-supervision: (i) Generalization. State-only classifiers and sequence models must implicitly infer a tactic's effect as their representations often entangle names and pretty-print idiosyncrasies. By centering supervision on structural edits (e.g., added hypotheses, reduced goal lists, relation flips), Δ-aware representations should transfer across theorems and libraries, complementing retrieval-based systems such as LeanDojo (Yang et al., 2023). (ii) Interpretability and modularity. Operator-style embeddings provide concrete semantics. In other words, they predict how a state should move in embedding space under t, enabling tactic-family clustering, edit-level probing, and straightforward integration into search controllers that prefer tactics moving states *toward* a solved region.

Novelty relative to prior work. Prior systems typically (a) learn tactic selection from the current state, (b) predict the next tactic from tactic sequences, or (c) fine-tune LLMs end-to-end to emit Lean scripts (Gauthier et al., 2021; Bansal et al., 2019; Yang and Deng, 2019; Yang et al., 2023). In contrast, we elevate the *state transition* to the supervisory signal: the label for a tactic is its *observed effect*. This yields compact, interpretable embeddings that are immediately useful for tactic ranking and are natural starting points for stronger operator models.

By reframing tactics as operators on proof states and demonstrating that even bag-of-edits supervision suffices to obtain useful embeddings, this work offers a lightweight, interpretable bridge between ITP state representations and neural models, complementary to environment-heavy frameworks and retrieval-augmented provers (Yang et al., 2023).

2 Methodology

2.1 A brief primer on Lean and proof states

Lean is an ITP based on dependent type theory. A proof proceeds by repeatedly applying *tactics* that transform a *proof state*, which consists of (i) a *local context* of named hypotheses and (ii) a (multi)set of *goals* to be proved. In Lean's pretty printer, the turnstile symbol $\vdash$ separates the context from the current goal.

Examples of Lean code can be found in Appendix B.

2.2 Dataset and supervision triples

We use a stepwise Lean proof corpus (Lean Workbook), which provides, for each proof step, a triple $(s_{\text{before}},\ t,\ s_{\text{after}})$, where t is the head tactic applied to transform the pretty-printed proof state s_{before} into s_{after}. Each record also includes a `proof_id` (the theorem/proof identifier). The lines are sorted such that their order corresponds to their position within the proof. We operate on a 25.2k-step slice for the experiments reported in this paper.

Our central idea is to treat the tactic t not merely as a label but as an *operator* whose semantics are expressed by the change from s_{before} to s_{after}. To that end, we introduce Δ*-contexts* which track the various changes in the state of the proof.

In the following sections, we outline the preprocessing steps, the construction of the Δ-contexts, and the construction of the two distributional models (Δ-SGNS and CBOW-Δ).

2.3 Preprocessing and canonicalization

Lean states are textual pretty-prints. To make edits comparable across problems, we process the Lean code in following steps in order to produce the delta contexts :

- **Tokenization.** We split on Lean tokens and mathematical operators (including unicode) such as $\vdash$, $\leq$, $\geq$, ^, *, /, +, -, :, =, (,).
- **Alpha-renaming.** In order to prevent name leakage, We anonymize local variable and hypothesis names (e.g., `a, b, c, h, ha, hb, ...`) to placeholders ($\_x1, \_x2, \_h1, \_h2, \ldots$).
- **Formatting normalization.** We normalize whitespace and numeric formats.
- **Goal/context split.** We detect the first $\vdash$ and treat the tokens to its left as context and to its right as goal; if no goal remains, we append a marker `NO_GOALS`.
- **Multi-goal states.** If a state contains multiple goals, we use the union of their tokens. Set-level pooling keeps the representation permutation-invariant.

2.4 Delta construction: tokens and typed edits

The first Δ-construction is token-level and considers the change in token-count in the context before and after the tactic. For each pair $(s_{\text{before}}, s_{\text{after}})$, we compute *token*-level deltas for each token w:

$$\Delta(w) = \text{count}_{s_{\text{after}}}(w) - \text{count}_{s_{\text{before}}}(w). \quad (1)$$

We keep only the *positive* token deltas and emit a context symbol `TOK_`w for each net added token w, repeated $|\Delta(w)|$ times. To inject weak structure, we add *typed* edit indicators extracted by simple regex rules:

- **Hypothesis count change:** Δ`_ADD_HYP` / Δ`_REM_HYP` based on the changes in counts of `name : type` hypothesis pairs in the context.
- **Goal change:** Δ`_GOAL_SOLVED` if $\vdash$ disappears; otherwise Δ`_GOALS_+k/-k` for detected changes.
- **Relation flips:** The Δ`_REL_GE_TO_LE`, Δ`_REL_LE_TO_GE` when the goal changes direction.
- **Operator counts:** Δ`_ADD/REM_SYM_`σ for $\sigma \in \{$`^, *, /, +, -, =,` $\leq$`,` $\geq\}$.

The final *delta context* for a step is the multiset

$$\begin{aligned}\mathcal{C} \;&=\; \{\texttt{TOK\_}w : \Delta(w)\!>\!0\}\\ &\cup\; \{\text{typed edit indicators}\}\end{aligned}$$

Appendix A walks through a worked example illustrating the procedures in Sections 2.3 and 2.4.

2.5 Models

We learn three lightweight distributional models using only the delta contexts $\mathcal{C}$ and tactic heads t. Let $e_x \in \mathbb{R}^d$ denote the embedding of token/tactic x.

Δ-SGNS (tactic $\rightarrow$ delta). We train a skip-gram with negative sampling where the *center* word is the tactic head `TACTIC` t, and each $c \in \mathcal{C}$ is a *context* word. The per-pair loss is

$$\begin{aligned}\ell_{\text{SGNS}}(t, c) \;&=\; -\log\sigma(e_t^\top e_c)\\ &\quad - \sum_{c^- \sim P_n} \log\sigma(-e_t^\top e_{c^-}) \quad (2)\end{aligned}$$

with negatives c^- drawn from a unigram distribution raised to the $3/4$ power. This objective encourages the tactic vector e_t to co-occur with the *added* tokens and typed edits it tends to induce.

CBOW-Δ (delta $\rightarrow$ tactic). We train a CBOW model that averages the context embeddings and predicts the tactic:

$$\bar{e}_{\mathcal{C}} \;=\; \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} e_c \quad (3)$$

$$\begin{aligned}\ell_{\text{CBOW}}(\mathcal{C}, t) &= -\log\sigma(\bar{e}_{\mathcal{C}}^\top e_t)\\ &\quad - \sum_{t^- \sim P_n} \log\sigma(-\bar{e}_{\mathcal{C}}^\top e_{t^-}). \quad (4)\end{aligned}$$

with negatives t^- sampled from the tactic frequency distribution.

Sequence-only baseline (no state). For comparison, we train a skip-gram model over *tactic sequences* within each proof (window size w), ignoring states entirely (the standard tactic co-occurrence baseline).

2.6 State embeddings and the operator-analogy test

To probe operator-like behavior without a parametric state encoder, we define a simple bag-of-tokens state embedding using the learned table:

$$v(s) \;=\; \frac{1}{|\text{tok}(s)|}\sum_{w\in\text{tok}(s)} e_{\texttt{TOK\_}w}. \quad (5)$$

We then test the *analogy* by checking whether $v(s_{\text{before}}) + e_t$ aligns with $v(s_{\text{after}})$ in the learned space. This mirrors translation tests in distributional semantics and knowledge-graph embeddings, but here the translation vector is the tactic embedding.

2.7 Training protocol and hyperparameters

We split data by `proof_id` into train/validation/test with ratios $80/10/10$ to avoid leakage across steps of the same proof. To mitigate tactic imbalance (e.g., the tactics `simp` and `have` are very frequent), we cap training steps per tactic at a maximum K (we use $K = 5000$). The validation/test are left untouched. Unless otherwise stated, we use the following settings (chosen for stability and speed):

- **Dimensions:** $d = 256$ for Δ-SGNS and CBOW-Δ; $d = 128$ for the sequence baseline.
- **Negative sampling:** 15 negatives per positive for Δ-SGNS and CBOW-Δ; 10 for the sequence baseline; negatives drawn from unigram$^{3/4}$.

- **Optimization:** learning rate 0.03; gradient clipping at norm 5.0; 15 epochs (Δ-SGNS/CBOW-Δ and sequence baseline).
- **Windows:** sequence baseline window $w = 4$.

3 Experiments and Results

Our goal is to verify the central hypothesis that *a tactic is defined by its effect on the proof state*. To attribute performance to specific sources of signal and to rule out alternative explanations (e.g., sequence co-occurrence or name leakage), we run three ablations that toggle which Δ-features are visible to the model: (1) **Tokens-only** — contexts contain *only* added surface tokens (`TOK_`w). This tests whether surface additions alone suffice to identify tactics. (2) **Typed-only** — contexts contain *only* typed structural edits (Δ`ADD_HYP`, Δ`GOAL_SOLVED`, relation flips, operator-count changes). This tests whether structural signals alone are sufficient. (3) **Full** — *both* tokens and typed edits. This tests complementarity and the necessity of structural cues for best performance.

We evaluate two Δ-aware models (Δ-SGNS and CBOW-Δ) and a sequence-only SGNS baseline (no states) in order to separate *state-change* supervision from *tactic co-occurrence*.

3.1 Controls against leakage and distributional artifacts

All ablations share the same proof-level split and test cardinalities, so differences reflect the availability of Δ-features rather than label-set changes or sample-size effects. Local-name anonymization and pretty-print canonicalization further reduce the chance of learning from spurious surface cues (e.g., hypothesis names). Together, these controls support the conclusion that the observed gains derive from modeling *state change*.

3.2 $\Delta \rightarrow$ tactic retrieval

We report test-set Mean Reciprocal Rank (MRR) and Recall@k for both Δ-SGNS (center=tactic, contexts=Δ-tokens) and CBOW-Δ (contexts$\rightarrow$tactic). Table 1 shows the results. CBOW-Δ shows clear complementarity: *Full* outperforms *Tokens-only* and *Typed-only* (MRR 0.085 vs. 0.068/0.037), a $\sim$25% relative gain over Tokens-only, indicating that typed edits add non-redundant information beyond surface additions. *Typed-only* retains non-trivial accuracy (MRR 0.037), implying structural cues are necessary but not sufficient. *Tokens-only* captures surface regularities (rewrites, symbol traces) but misses generalization afforded by typed edits. On Δ-SGNS, *Tokens-only* $>$ *Full* suggests skip-gram with a shared table is sensitive to heterogeneous context types. Simple gating or reweighting of typed cues likely will help when using SGNS.

	Δ-SGNS				CBOW-Δ			
Ablation	MRR	R@1	R@5	R@10	MRR	R@1	R@5	R@10
Full	0.032	0.017	0.047	0.054	**0.085**	**0.058**	**0.116**	**0.144**
Tokens	**0.039**	**0.021**	**0.056**	**0.069**	0.068	0.051	0.085	0.095
Typed	0.009	0.004	0.009	0.012	0.037	0.012	0.053	0.086

Table 1: Test $\Delta \rightarrow$ tactic retrieval. Higher is better.

3.3 Operator behavior

We test whether $v(s_{\text{before}}) + e_t$ aligns with $v(s_{\text{after}})$ in the learned space (cosine and rank among after-states). Table 2 shows these results. Typed edits *improve* alignment when combined with tokens: *Full* surpasses *Tokens-only* (cosine 0.67 vs. 0.62; rank 422.3 vs. 429.2, lower is better). Under *Typed-only*, the analogy metric is not meaningful because state embeddings $v(\cdot)$ aggregate `TOK_`w vectors that are *untrained* in that ablation (cosine ≈ 0, rank ≈ 1 by construction).

Ablation	Avg Cosine	Avg Rank (lower$\downarrow$)	N
Full	0.67	422.3	2590
Tokens-only	0.62	429.2	2590
Typed-only	0.00	1.0 †	2590

Table 2: Operator-analogy for Δ-SGNS: $\text{cosine}\big(v(s_{\text{before}}) + e_t, v(s_{\text{after}})\big)$ and rank of the true after-state among test after-states.

3.4 Sequence-only baseline (control)

A tactic-sequence SGNS achieves next-tactic MRR $= 0.081$ on N=525 bigrams. While this is a *different* prediction target, it serves as a control: co-occurrence alone does not explain $\Delta \rightarrow$ tactic performance, and the best CBOW-Δ MRR is achieved on a strictly harder task that conditions on state edits rather than neighboring tactics.

3.5 Structure of tactic embeddings

Figure 1 visualizes 2D UMAP projections of the 256-dimensional tactic embeddings learned under the *Full* setting for CBOW-Δ (Fig 1a), Δ-SGNS (Fig 1b), and the sequence-only baseline (Fig 1c).

These UMAP projections show that the CBOW-Δ and Δ-SGNS spaces exhibit visually separable clusters, consistent with effect-grounded organization of tactics. By contrast, the sequence-only baseline forms a diffuse, nearly isotropic cloud with no obvious cluster structure under the same projection. These observations support our claim that co-occurrence–only objectives, which can work well in natural-language settings, do not induce discriminative tactic representations in Lean to the same extent as Δ-aware training.

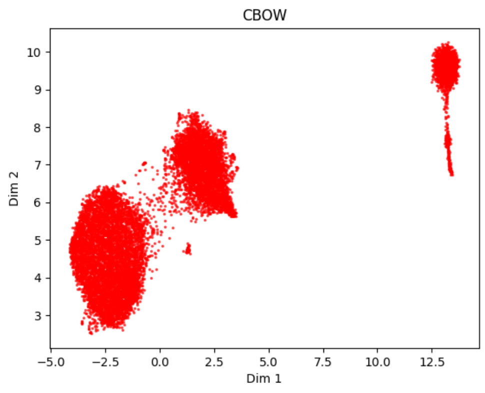
4 Discussion and Conclusion

This paper introduced a representation-learning perspective in which tactics are modeled as operators defined by their effects on proof states. By leveraging Δ-supervision—token-level additions/removals and typed structural edits—we trained lightweight CBOW and skip-gram models that learn embeddings grounded in state transitions. Our experiments demonstrated that these embeddings capture tactic semantics more robustly than surface-only or sequence-based baselines. Typed edits, in particular, provided complementary information that improved generalization and interpretability, enabling operator-style behaviors such as tactic-family clustering and state-translation analogies.

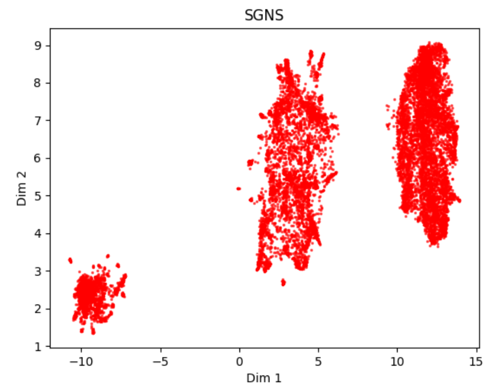
Beyond empirical gains, the central contribution lies in showing that surface-level distributional regularities, often sufficient in natural language modeling, are inadequate for theorem proving. Capturing the semantics of tactics requires effect-grounded representations that model the transformations they induce. While our models are deliberately minimal, they establish a lightweight, interpretable bridge between symbolic proof states and neural methods, complementary to large-scale LLM fine-tuning and retrieval-augmented provers. These findings motivate future work on scaling Δ-supervision to larger datasets, enriching structural edits, and integrating operator embeddings into end-to-end proof search systems.

Limitations

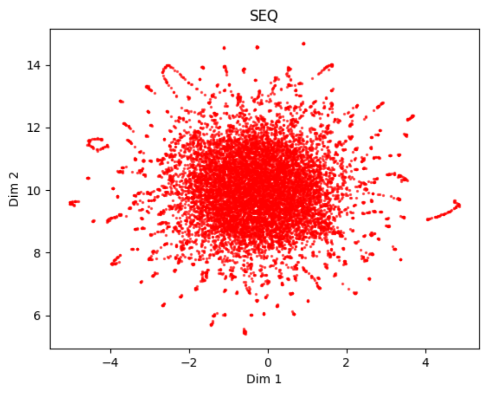
Our study is a proof-of-concept with several limitations. First, we restrict ourselves to a relatively small subset of the Lean Workbook corpus, leaving open whether results scale to larger, more diverse mathematical domains. Second, our models are intentionally minimal and ignore contextual subtleties such as multi-tactic interactions, proof search heuristics, or long-range dependencies. Third, the Δ-contexts rely on simple tokenization and regex-based typed edits, which may omit more nuanced structural information (e.g., type-class resolution or meta-variable instantiation). Finally, we evaluate on proxy tasks such as retrieval and anal-



(a) CBOW



(b) SGNS



(c) SEQ baseline

Figure 1: UMAP visualizations of the 256-dimensional embeddings for CBOW, SGNS, and SEQ (sequence-only baseline) under the Full setting.

ogy, which, while informative, are not substitutes for downstream proof success. Addressing these limitations will require richer state encodings, integration with stronger search strategies, and broader evaluation benchmarks.

Ethical considerations

This work focuses on mathematical theorem proving in Lean and does not involve human subjects, sensitive data, or societal risks.

Acknowledgment

This work was partially supported by student research awards from the Neukom Institute for Computational Science, the Lovelace Research Program, and the Leslie Center at Dartmouth College.

References

Kshitij Bansal, Sarah Loos, Markus Rabe, Christian Szegedy, and Stewart Wilcox. 2019. HOList: An environment for machine learning of higher order logic theorem proving. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 454–463. PMLR.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26.

Thibault Gauthier, Cezary Kaliszyk, Josef Urban, Ramana Kumar, and Michael Norrish. 2021. Tactictoe: Learning to prove with tactics. *Journal of Automated Reasoning*, 65(2):257–286.

InternLM. 2024. Lean workbook (and lean workbook plus) dataset. `https://huggingface.co/datasets/internlm/Lean-Workbook`. Hugging Face dataset card; reports ∼57k Lean Workbook and ∼83k Lean Workbook Plus problems.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26.

Kaiyu Yang and Jia Deng. 2019. Learning to prove theorems via interacting with proof assistants. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6984–6994. PMLR.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. Leandojo: Theorem proving with retrieval-augmented language models. In *NeurIPS 2023 Track on Datasets and Benchmarks*. Paper available at NeurIPS Datasets and Benchmarks.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. 2024. Lean workbook: A large-scale lean problem set formalized from natural language math problems. *arXiv preprint arXiv:2406.03847*.

A Worked Example: Building a Δ-Context

We illustrate Sections 2.3–2.4 by reconstructing the Δ-context for the proof step:

State-Before
a b c : $\mathbb{R}$
ha : $0 < a$ hb : $0 < b$ hc : $0 < c$
habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$
$\vdash \dfrac{a^3}{(1 - a^8)} + \dfrac{b^3}{(1 - b^8)} + \dfrac{c^3}{(1 - c^8)} \geq \dfrac{9}{8}$
Tactic
have h1 := sq_nonneg (a^2 - 1)
State-After
a b c : $\mathbb{R}$
ha : $0 < a$ hb : $0 < b$ hc : $0 < c$
habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$
h1 : $0 \leq (a^2 - 1)^2$
$\vdash \dfrac{a^3}{(1 - a^8)} + \dfrac{b^3}{(1 - b^8)} + \dfrac{c^3}{(1 - c^8)} \geq \dfrac{9}{8}$

Step 1: Tokenization (Sec. 2.3). We split on Lean/Unicode symbols and mathematical operators and on identifiers. The subphrase

$$\texttt{h1} : 0 \leq (a^2 - 1)^2$$

is tokenized as

$$[\,\texttt{h1},\ :,\ 0,\ \leq,\ (,\ \texttt{a},\ \hat{},\ 2,\ \texttt{-},\ 1,\ ),\ \hat{},\ 2\,]$$

Step 2: Alpha-renaming / canonicalization (Sec. 2.3). Local variable and hypothesis names are anonymized to prevent name leakage:

$$\begin{array}{lll} \texttt{a} \mapsto \_x1 & \texttt{b} \mapsto & \_x2 \\ \texttt{c} \mapsto \_x3 & \texttt{h1} \mapsto & \_h6 \\ \texttt{ha}, \texttt{hb}, \texttt{hc}, \texttt{habc}, \texttt{h} \mapsto & \_h1 \ldots \_h5 & \end{array}$$

Thus the added hypothesis pretty-prints canonically as

$$\texttt{\_h6} : 0 \leq (\_x1^2 - 1)^2.$$

Step 3: Formatting normalization (Sec. 2.3). We normalize whitespace and numeric formats; there is no effect on the symbol sequence above.

Step 4: Goal/context split (Sec. 2.3). We split at the first $\vdash$. The new hypothesis belongs to the *context* (left of $\vdash$); the *goal* (right of $\vdash$) is unchanged in this step. There are no multiple-goal peculiarities here (multi-goal pooling is unnecessary).

Step 5: Token-level deltas (Sec. 2.4). Let $\Delta(w) = \text{count}_{\textit{after}}(w) - \text{count}_{\textit{before}}(w)$. The only positive deltas arise from inserting the hypothesis line $\texttt{\_h6} : 0 \leq (\_x1^2 - 1)^2$. The multiset of *added tokens* (with multiplicities) is:

$$\begin{array}{llllll} \texttt{\_h6}^{\times 1} & :^{\times 1} & 0^{\times 1} & \leq^{\times 1} & (^{\times 1} & \texttt{\_x1}^{\times 1} \\ \wedge^{\times 2} & 2^{\times 2} & \texttt{-}^{\times 1} & 1^{\times 1} &)^{\times 1} & \end{array}$$

Step 6: Typed edit indicators (Sec. 2.4). Regex-based structural edits are as follows:

$\Delta_{\texttt{ADD\_HYP}}$	(one hypothesis added)
$\Delta_{\texttt{ADD\_SYM\_-}}$	(a minus sign added)
$\Delta_{\texttt{ADD\_SYM\_}\leq}$	(a $\leq$ added)

There is no Δ_GOAL_SOLVED and no relation flip ($\leq \leftrightarrow \geq$) in this step.

Step 7: Final Δ-context C (Sec. 2.4). The training context is the multiset:

$$\begin{aligned} &\{\texttt{TOK\_\_h6},\ \texttt{TOK\_:},\ \texttt{TOK\_0}, \\ &\quad \texttt{TOK\_}\leq,\ \texttt{TOK\_(},\ \texttt{TOK\_\_x1},\ \texttt{TOK\_\^{}}, \\ &\quad \texttt{TOK\_2},\ \texttt{TOK\_\^{}},\ \texttt{TOK\_2},\ \texttt{TOK\_-}, \\ &\quad \texttt{TOK\_1},\ \texttt{TOK\_)}\,\} \\ &\cup \{\,\Delta_{\texttt{ADD\_HYP}},\ \Delta_{\texttt{ADD\_SYM\_-}}, \\ &\quad \Delta_{\texttt{ADD\_SYM\_}\leq}\,\} \end{aligned}$$

(Repeated TOK_^ and TOK_2 indicate multiplicity.)

B Examples

Table B1 shows sample lean proofs with different number of tactics. Table B2 shows examples of State-Before, Tactic, State-After triples from the Lean Work-book Corpus.

Proof #	Tactics

1	`norm_num [ha, hb, hc, habc, h]`
	`have h1 := sq_nonneg (a^2 - 1)`
	`have h2 := sq_nonneg (b^2 - c^2)`
	`nlinarith`

2	`rintro a b c ⟨h1, h2, h3, h4, h5, h6⟩`
	`nlinarith [sq_nonneg (a - b), sq_nonneg (b - c), sq_nonneg (c - a)]`

3	`push_neg`
	`refine' ⟨1, 1, ⟨by norm_num, by norm_num⟩, by norm_num⟩`

4	`push_neg`
	`refine' ⟨1, 2, 3, by norm_num⟩`

5	`simp [Int.ModEq]`
	`omega`

Table B1: Sample proofs. Each line is one tactic.

State-Before	Tactic	State-After
x y z : ℤ ⊢ (x^2 + 1) * (y^2 + 1) * (z^2 + 1) = (x + y + z)^2 - 2 (x y + y z + z x) + (x y + y z + z x)^2 - 2 x y z (x + y + z) + x^2 y^2 z^2 + 1	ring	no goals
a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ ⊢ $a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8) \geq \frac{9 \cdot 3^{1/4}}{8}$	`norm_num [ha, hb, hc, habc, h]`	a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ ⊢ $\frac{9}{8} \leq a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8)$
a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ ⊢ $\frac{9}{8} \leq a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8)$	have h1 := `sq_nonneg (a^2 - 1)`	a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ h1 : $0 \leq (a^2-1)^2$ ⊢ $\frac{9}{8} \leq a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8)$
a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ h1 : $0 \leq (a^2-1)^2$ ⊢ $\frac{9}{8} \leq a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8)$	have h2 := `sq_nonneg (b^2 - c^2)`	a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ h1 : $0 \leq (a^2-1)^2$ h2 : $0 \leq (b^2-c^2)^2$ ⊢ $\frac{9}{8} \leq a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8)$
a b c : ℝ ha : $0 < a$ hb : $0 < b$ hc : $0 < c$ habc : $a * b * c = 1$ h : $a^4 + b^4 + c^4 = 1$ h1 : $0 \leq (a^2-1)^2$ h2 : $0 \leq (b^2-c^2)^2$ ⊢ $\frac{9}{8} \leq a^3/(1-a^8)+b^3/(1-b^8)+c^3/(1-c^8)$	nlinarith	no goals

Table B2: Sample sentences (before, tactic, after) from the Before-Tactic-After dataset.