ACTIVATION STEERING IN NEURAL THEOREM PROVERS

Shashank Kirtania
Microsoft, India
t-skirtania@microsoft.com

ABSTRACT

Large Language Models (LLMs) have shown promise in proving formal theorems using proof assistants like Lean. However, current state of the art language models struggle to predict next step in proofs leading practitioners to use different sampling techniques to improve LLMs capabilities. We observe that the LLM is capable of predicting the correct tactic; however, it faces challenges in ranking it appropriately within the set of candidate tactics, affecting the overall selection process. To overcome this hurdle we use activation steering to guide LLMs responses to improve the generations at the time of inference. Our results suggest that activation steering offers a promising lightweight alternative to specialized fine-tuning for enhancing theorem proving capabilities in LLMs, particularly valuable in resource-constrained environments.

1 INTRODUCTION

Interactive proof assistants such as Lean de Moura et al. (2015), Isabelle Wenzel et al. (2008), and Coq Barras et al. (1999) enable the formal verification of mathematical proofs and software by leveraging specialized programming languages Avigad (2023); Ringer et al. (2019). Neural theorem proving, which integrates neural language models with interactive proof assistants, has emerged as a promising approach to automating formal reasoning First et al. (2023); Polu & Sutskever (2020b); Polu et al. (2022); Yang et al. (2023b); Welleck (2023). This integration is mutually beneficial: proof assistants enforce formal correctness, while language models assist in proof construction by predicting and suggesting logical steps. A central challenge in this setting is tactic prediction—determining the appropriate next step at each proof state.

In this work, we investigate activation steering Panickssery et al. (2024); Turner et al. (2024); Lucchetti & Guha (2024) as a technique to enhance tactic prediction in Llemma Azerbayev et al. (2024a) and InternLM2 Ying et al. (2024a). These language models are designed for theorem proving by training and fine-tuning on mathematical data. Activation steering is an inference-time model editing method that modifies a model's internal representations to guide its behavior toward desired outputs. We propose its application in refining tactic selection, aiming to improve both the accuracy and interpretability of proof automation. By systematically influencing LLMs' reasoning process, our approach enables structured interventions that enhance model-driven theorem proving, leading to more reliable and controllable predictions.

We present an approach for steering tactic selection from a pair of prompts (p_1, p_2) that contain a LEAN state s to generate the next step (or tactic) t. However, the LLM successfully predicts t for p_1 but mispredicts for p_2. In each pair p_2 is natural data and p_1 is synthetically generated using p_2. We systematically add the attributes and a high level of structure the proof should follow. These additional attributes guide the model to a specific and more grounded chain of thought to follow while predicting the tactics. These abstractions over the proof help the LLM to do critical decision making while predicting the next step.

2 Related Work

2.1 Formal Theorem Proving

Formal theorem proving encodes theorems and proofs in a machine-verifiable format, ensuring correctness through rigid logical rules. A key component of this field is Interactive Theorem Proving (ITP), where humans collaborate with *proof assistants* such as Isabelle Wenzel et al. (2008), Lean de Moura et al. (2015), and Coq Barras et al. (1999) to formally verify proofs. These assistants allow users to express theorems in higher-order logic and construct verifiable proofs.

In Lean de Moura et al. (2015), proofs are built using *tactics*, which either solve a goal or decompose it into sub-goals.

2.2 Proofstep Generation with Large Language Models

Generating intermediate proof steps is a fundamental challenge in theorem proving, particularly in tactic-based automated theorem provers (ATPs). Early neural approaches (Whalen, 2016; Huang et al., 2019; Bansal et al., 2019; Paliwal et al., 2020; Sanchez-Stern et al., 2020) framed proofstep generation as a classification task, employing models like TreeLSTM and RNN to predict tactics and their arguments. ASTactic (Yang & Deng, 2019) later introduced a grammar-constrained decoder for structured tactic generation.

Recent advances leverage large language models (LLMs) for proof generation, casting tactic prediction as an extitauto-regressive sequence modeling problem. GPT-f (Polu & Sutskever, 2020a) pioneered this approach by training transformers only with decoders to generate structured proof steps. Baldur (First et al., 2023) extended this by producing entire proofs, while POETRY (Wang et al., 2024) adopted a recursive decomposition strategy. Other works (Szegedy et al., 2021; Tworkowski et al., 2022; Welleck et al., 2022; Jiang et al., 2022; Yang et al., 2023a) integrate the selection of premises with tactic prediction, employing retrieval-augmented methods and constrained decoding to improve the coherence of the proof.

Lean-Star Lin et al. (2024) introduced Self-Taught Reasoning, incorporating Chain-of-Thought Wei et al. (2023) reasoning before each tactic to generate synthetic data for fine-tuning LLMs via self-play Chen et al. (2024). Our work leverages these randomly sampled data points from *Lean-STaR-base* as natural data.

2.3 Mechanistic Interpretability

Previous work has focused on localizing and editing factual associations within transformers (Meng et al., 2022) and probing hidden representations for high-level knowledge (Li et al., 2024b; Dong et al., 2023). Such studies perform *implicit evaluations* of model ability, complementing explicit benchmarks (Dong et al., 2023). A key technique in mechanistic interpretability is activation patching (Vig et al., 2020; Variengien & Winsor, 2023), which modifies model activations to influence outputs. This research has suggested the existence of task vectors (Hendel et al., 2023; Ilharco et al., 2022)—representations encoding abstract task information. Activation steering has been employed to mitigate model deceitfulness and sycophancy (Rimsky et al., 2023; Li et al., 2024b), further supporting the presence of task vectors. Steering is based on the linear representation hypothesis (Park et al., 2023), which posits that concepts exist as directions in the embedding space of the model.

For theorem proving, mechanistic interpretability provides insights into how LLMs represent logical structures and reasoning processes. By dissecting these representations, we can identify failure cases, refine tactic prediction, and enhance proof generation. We hypothesize that effective steering transforms activations to align the model's reasoning trajectory with a more structured and verifiable direction.

2.4 Model Steering

Activation-based interventions can directly influence the language model output during inference Dathathri et al. (2019); Subramani et al. (2022). Recent studies demonstrate that activation steering enhances truthfulness , mitigates sycophancy, and improves instruction-following Stolfo et al. (2024), as well as type prediction in code Lucchetti & Guha (2024).

Building on these ideas, we investigate steering vectors in the context of theorem proving. Following prior work Burns et al. (2024); Turner et al. (2024); Arditi et al. (2024); van der Weij et al. (2024), we compute steering vectors based on input pairs differing by a specific feature—here, the presence or absence of synthetic metadata. Unlike previous studies that focused on broad linguistic properties such as sentiment and style, our approach seeks to refine the logical inference pathways of LLMs.

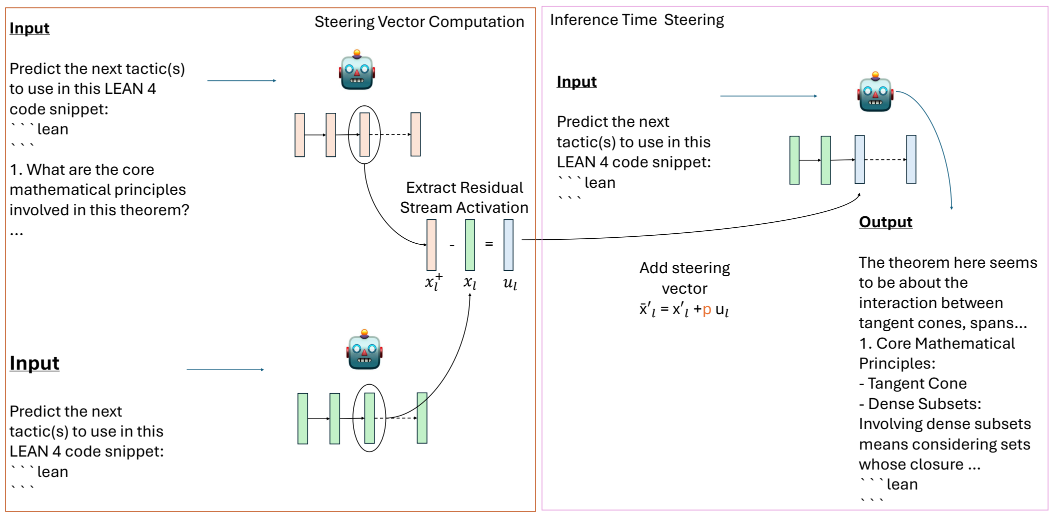

Figure 1: Steering Vectors are computed as difference of activations of p_1 and p_2

For theorem proving, activation steering provides a mechanism to guide the model towards structured reasoning, improving its ability to generate valid proof steps. By leveraging task vectors, we aim to shift model activations to align with correct logical deductions, thereby enhancing both proof coherence and model interpretability. This intervention is particularly valuable in interactive theorem proving, where fine-grained control over reasoning steps can lead to more reliable and verifiable proofs.

3 METHODOLOGY

3.1 CONSTRUCTING STEERING DATASET

Model Choice We build steering datasets for 7B parameter Llemma Azerbayev et al. (2024b) and 7B parameter InternLM2 Ying et al. (2024b). These models are trained to generate formal theorems and code, which is important for the tactic prediction task.

Source Dataset We constructed steering pairs from a randomly sampled subset S from *Lean-STaR* data. The subset consists of approximately ten thousand unique proof stages and tactics. We treat this subset as natural prompt data p_2 and generate a new set p_1 by adding reasoning steps in p_2 representing a lean stage l. This generates a set of pairs (p_1, p_2) which we then prompt InternLM2 to predict the next tactic t_1 and t_2 respectively. We then take pairs where $t_1 \neq t_2$ creating a subset s of prompt pairs. We then validate tactics $t_1, t_2 \in s$ with Lean Prover to generate s'.

This process generates a quadruple $\{p_1, p_2, t_1, t_2\}$ where t_1 and t_2 are valid tactics for a lean stage l. We assume that t_1 is a more optimal tactic for l.

Generating p_1 from p_2 In Fig. 2, we illustrate the process of improving theorem-proving LLMs using *StepBackReasoning* Zheng et al. (2024). Initially, the model is given a prompt asking it to predict the next tactic in a Lean 4 proof. Without additional reasoning guidance, the model produces an incorrect output, such as rfl. which does not align with the required proof strategy. To address this, we introduce a step-back reasoning mechanism. First, we extract the proof state from the given

Lean prompt, as shown in the *StepBackAbstraction* module. This abstraction step helps identify the core mathematical principles underlying the theorem. Using GPT-4, we generate step-back reasoning prompts, which explicitly highlight key mathematical structures, such as exponentiation in a division monoid and properties like associativity and commutativity. These enriched insights are then incorporated into the original theorem-proving prompt, providing the model with a more structured understanding of the problem. As a result, when the theorem-proving LLM is queried again, it produces the correct output $rwpow_m ul$ which correctly applies the relevant exponentiation rule. This process demonstrates how structured reasoning about the problem enhances logical coherence and guides LLMs toward more accurate proof generation in Lean 4.

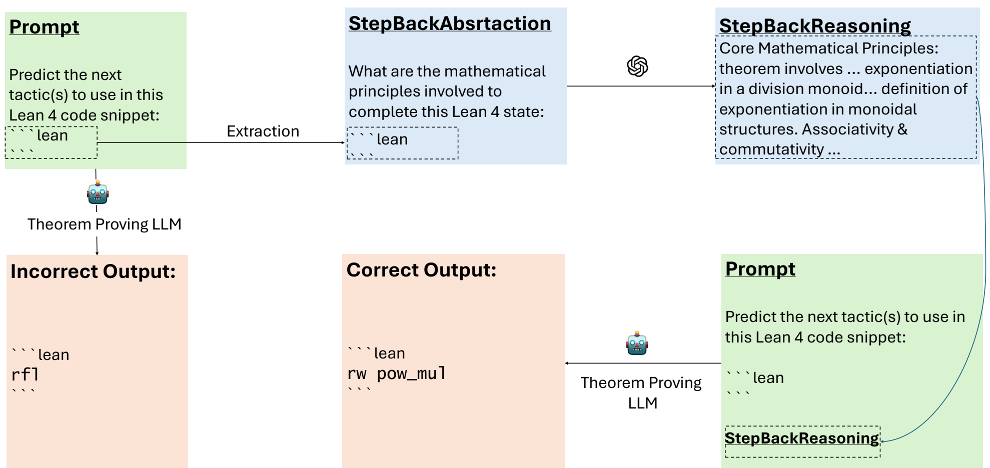

Figure 2: Steering Vectors are computed as difference of activations of p_1 and p_2

3.2 Constructing Steering Vectors

Given dataset of steering pairs and tactics $\{p_1, p_2, t_1, t_2\} \in s'$ and a model M, we apply a forward pass to every $M(p_1)$,$M(p_2)$ to collect values of the *residual stream* vector on queries at the last token of the input layer $\ell \in \{1, ..., L\}$. We isolate the internal representation corresponding to the reasoning by computing the difference in residual stream vectors $v_{1,\ell}$ and $v_{2,\ell}$. More formally, we compute the vector u_ℓ representing the direction of the steering at layer ℓ:

$$\mathbf{u}_\ell = \frac{\mathbf{v}_\ell}{\|\mathbf{v}_\ell\|}, \quad \text{where} \quad \mathbf{v}_\ell = \frac{1}{N}\sum_i^N (\mathbf{p}_{1,\ell,i} - \mathbf{p}_{2,\ell,i})$$

Averaging our different proof states to capture activation values most closely associated with the structured reasoning step independent of the query. The calculation of the direction of the steering is carried out using the representations in the last token of the input, which effectively encapsulates the behavior of the model not only for the next token prediction task, but also for the entire generation following Todd et al. (2024); Scalena et al. (2024). After identifying steering direction, we compute steering vector by re-scailing unit vector u_ℓ by a coefficient c. We use a systematic scaling approach where the value of c is selected to ensure that residual stream activations are assigned to their mean value on inputs that contain the structured reasoning steps. In particular, we compute a new example with residual stream values p' at a given token.

$$c = \bar{z} - p'_\ell u_\ell, \quad \text{where} \quad \bar{z} = \frac{1}{N}\sum_i^N p_{1,i,\ell} u_\ell.$$

The steering vector cu_ℓ is then added to the corresponding residual stream layer and the forward pass is resumed with the updated residual stream value $\tilde{x}'_\ell = x'_\ell + cu_\ell$. This procedure is carried out

Model	Decoding	N	K	S	MiniF2F
GPT-3.5 (FEW-SHOT)	SAMPLING	50	1	1	2.8%
GPT-4 (FEW-SHOT)	SAMPLING	50	1	1	11.9%
LLEMMA-7B	SEARCH	50	1	32	26.2%
INTERNLM2-7B	SEARCH	50	1	32	30.3%
INTERNLM2-7B (SFT)	SEARCH	50	1	32	30.7%
LEAN-COT (INTERNLM2-7B)	SAMPLING	50	32	1	27.0%
LEAN-COT (INTERNLM2-7B)	SEARCH	50	1	32	25.4%
LEAN-STAR (INTERNLM2-7B)	SAMPLING	50	32	1	29.1%
LEAN-STAR (INTERNLM2-7B)	SEARCH	50	1	32	26.2%
CORPA (WITH GPT-4)	CUSTOMIZED	-	60	1	29.9%
OURS (LLEMMA-7B)	SAMPLING	50	32	1	28.1%
OURS (LLEMMA-7B)	SEARCH	50	1	32	26.3%
OURS (INTERNLM2-7B)	SAMPLING	50	32	1	**32.4%**
OURS (INTERNLM2-7B)	SEARCH	50	1	32	**26.8%**

Table 1: Pass rates on the `miniF2F`-test dataset with Lean. This table shows the pass rates of previous works and our work. S is the number of tactics attempted at each expanded node (assumed to be 1 in sampling), and K is the total number of search or sampling attempts per problem.

at a single layer across all token positions, motivated by previous findings that show models tend to deviate from instructions as they generate more tokens Stolfo et al. (2024); Li et al. (2024a)

4 Results and Analysis

Setup We evaluated the technique using *Best First Search*. It is one of the most popular methods to evaluate the theorem-proving ability of a language model Polu & Sutskever (2020b); Yang et al. (2023b); Azerbayev et al. (2023); Lin et al. (2024). For a given language model M, we keep all unexpanded states s_i; each time we expand the best state s_i and use the language model to sample S net tactics $a_{i,1\ldots S}$ for the current state s_i. Following standard practice Polu & Sutskever (2020b); Yang et al. (2023b); Welleck & Saha (2023); Lin et al. (2024) we assume the state with maximum negative log-probabilities is the "best" state. Specifically, we select state s_i with maximum $\sum_{j=0}^{i-1} -\log p(a_j, s_j)$, where $(s_0, a_0), ..., (s_{i-1}, a_{i-1})$ is proof trajectory before state s_i and $\log p(a_j, s_j)$ is the average log probability of each generated token. We expand upto N states and we get successful proofs search when we reach any proof state with no goals.

Dataset We evaluated our technique on *MiniF2F* benchmark Zheng et al. (2022). Which consists of 244 theorems in lean 4. We use the same evaluation setting as previous works Yang et al. (2023b); Welleck & Saha (2023); Ying et al. (2024a).

Sampling We evaluate two different decoding strategies: *sampling* and *search*. *Sampling* involves drawing multiple proof steps stochastically based on the model's output distribution, promoting diversity in the generated proofs. In contrast, *search* incorporates structured exploration techniques to improve proof discovery. Specifically, we consider two widely used search methods: *beam search* and *best-first search*.

Beam Search. Beam search maintains a fixed number k of proof trajectories at each step, selecting the top-ranked candidates based on the model's confidence scores. By preserving multiple plausible proof paths instead of greedily committing to the highest-confidence step, beam search mitigates early pruning errors and allows exploration of alternative reasoning chains. However, the trade-off between beam width and computational cost remains a key consideration: while larger beams enhance robustness, they also introduce significant overhead.

Best-First Search. Best-first search prioritizes proof states according to a heuristic function, typically based on model confidence or learned value estimates. Unlike depth-first or breadth-first strategies, best-first search expands the most promising proof state first, dynamically adjusting the exploration process. In our experiments, we observe that combining *best-first search with sampling* yields notable improvements. We hypothesize that this effect arises because traditional reranking,

Model	Decoding	Random	Steering
LLEMMA-7B	SAMPLING	22.7%	28.1%
	SEARCH	19.2%	26.3%
INTERNLM-7B	SAMPLING	21.4%	32.4%
	SEARCH	18.9%	26.8%

Table 2: Pass rates with randomized vectors and steering vectors.

despite boosting the likelihood of high-reward tactics, may suffer from premature convergence to suboptimal proof paths. In contrast, best-first search, when coupled with sampling, allows for imperfect scoring, thereby encouraging broader exploration and improved intermediate state discovery.

4.1 Main Results

Our main results are reported in Table 1. Steering the model's activation significantly improves performance over the base model. Notably, we observe that steering markedly increases pass rates when using Best First Search with sampling. We hypothesize that this improvement occurs because reranking may be too narrowly focused—potentially getting trapped in local optima—even though it boosts log probabilities for tactics that follow the highest reward path. In contrast, combining sampling with Best First Search allows for imperfect scoring, which in turn enables the exploration of nodes that lead to better intermediate states.

4.2 Ablations

Random Steering Vectors
A potential validity concern with any intervention involving activation patching is that the observed improvements might not stem from genuine performance enhancements but rather from activating fallback mechanisms McGrath et al. (2023); Lucchetti & Guha (2024). For instance, patching could merely introduce noise into the embedding space, inadvertently triggering alternative pathways that lead to the desired outcome. This phenomenon complicates the interpretability of both patches and steering vectors. To examine this, we conduct an experiment using a randomly generated steering vector (denoted as "Random" in Table 2). Our findings show that even random steering achieves a nonzero accuracy, albeit significantly lower than that of our computed steering vectors. We hypothesize that this residual accuracy arises due to backup circuits. Nevertheless, the substantially higher performance of our computed steering vectors suggests that our approach induces meaningful transformations toward the correct target.

5 Conclusion

We investigate activation steering for tactic prediction by making language models adhere to structured reasoning approaches in theorem proving. We find that by constructing steering pairs using synthetic metadata and natural proof states, we can construct effective steering vectors that improve tactic selection. Our experiments show that steering vectors enhance model performance beyond random interventions and generalize well across different theorem-proving strategies. The effectiveness of our steering approach demonstrates the existence of underlying reasoning pathways that can be systematically influenced within language models. Activation steering proves to be a powerful technique for improving model performance on formal reasoning tasks where fine-tuning may be impractical or resource-intensive. As language models continue to evolve in their theorem-proving capabilities, activation steering may serve as a lightweight alternative to specialized fine-tuning approaches. This could be particularly valuable for interactive theorem proving environments where computational resources are limited. Our reasoning-based steering vectors, for example, could provide an efficient way to enhance proof assistants, particularly useful for applications like automated tactic suggestion. In future work, we aim to study the underlying mechanisms in language models responsible for structured mathematical reasoning. We further wish to explore how reasoning-focused steering vectors may generalize to open-ended theorem proving tasks and investigate their potential for improving proof search strategies.

6 FUTURE WORKS

Our work opens several promising research directions for improving theorem proving with activation steering. From a theoretical perspective, we aim to investigate the geometric properties of steering vectors in the context of formal reasoning, studying how different model layers represent logical structures and how steering vectors interact with these representations. Understanding these properties could lead to more efficient steering methods and deeper insights into how LLMs encode mathematical concepts. On the practical side, several extensions could enhance theorem proving systems, including the development of adaptive steering mechanisms that dynamically adjust based on proof state complexity, the investigation of steering vector composition for handling compound mathematical concepts, and the integration of steering with existing proof search heuristics to improve exploration efficiency. Scalability remains a challenge, and future work should explore techniques for reducing the computational overhead of steering during inference, methods for distilling steering vectors while maintaining their effectiveness, and approaches for generalizing steering vectors across different mathematical domains. Our findings suggest that activation steering could become a powerful tool for enhancing LLM-based theorem provers, particularly in resource-constrained environments where fine-tuning is impractical. We believe that exploring these directions will lead to more robust and efficient theorem proving systems.

REFERENCES

FMCAD '96: Proceedings of the First International Conference on Formal Methods in Computer-Aided Design, Berlin, Heidelberg, 1996. Springer-Verlag. ISBN 3540619372.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction, 2024. URL `https://arxiv.org/abs/2406.11717`.

Jeremy Avigad. Mathematics and the formal turn. *Bulletin (New Series) of the American Mathematical Society, Received by the editors October 2, 2023.*, 2023. URL `https://www.andrew.cmu.edu/user/avigad/Papers/formal_turn.pdf`.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.06786*, 2023.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics, 2024a. URL `https://arxiv.org/abs/2310.10631`.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An Open Language Model for Mathematics. In *Proceedings of the International Conference on Learning Representations*, 2024b.

Kshitij Bansal, Sarah M. Loos, Markus Norman Rabe, Christian Szegedy, and Stewart Wilcox. HOList: An Environment for Machine Learning of Higher Order Logic Theorem Proving. In *Proceedings of the International Conference on Machine Learning*, 2019.

Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Yann Coscoy, David Delahaye, Daniel de Rauglaudre, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo Herbelin, et al. The Coq Proof Assistant Reference Manual. *INRIA*, 1999.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision, 2024. URL `https://arxiv.org/abs/2212.03827`.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models, 2024. URL `https://arxiv.org/abs/2401.01335`.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *CoRR*, abs/1912.02164, 2019. URL `http://arxiv.org/abs/1912.02164`.

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, pp. 378–388. Springer, 2015.

Xiangjue Dong, Yibo Wang, Philip S Yu, and James Caverlee. Probing explicit and implicit gender bias through llm conditional text generation. *arXiv preprint arXiv:2311.00306*, 2023.

Emily First, Markus Rabe, Talia Ringer, and Yuriy Brun. Baldur: Whole-Proof Generation and Repair with Large Language Models. In *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. *arXiv preprint arXiv:2310.15916*, 2023.

Daniel Huang, Prafulla Dhariwal, Dawn Song, and Ilya Sutskever. GamePad: A Learning Environment for Theorem Proving. In *Proceedings of the International Conference on Learning Representations*, 2019.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Albert Q Jiang, Wenda Li, Szymon Tworkowski, Konrad Czechowski, Tomasz Odrzygóźdź, Piotr Miłoś, Yuhuai Wu, and Mateja Jamnik. Thor: Wielding Hammers to Integrate Language Models and Automated Theorem Provers. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2022.

Kenneth Li, Tianle Liu, Naomi Bashkansky, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Measuring and controlling instruction (in)stability in language model dialogs. In *First Conference on Language Modeling*, 2024a. URL `https://openreview.net/forum?id=60a1SAtH4e`.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36, 2024b.

Haohan Lin, Zhiqing Sun, Yiming Yang, and Sean Welleck. Lean-STaR: Learning to Interleave Thinking and Proving. *arXiv preprint arXiv:2407.10040*, 2024.

Francesca Lucchetti and Arjun Guha. Understanding how codellms (mis)predict types with activation steering, 2024. URL `https://arxiv.org/abs/2404.01903`.

Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. The hydra effect: Emergent self-repair in language model computations. *arXiv preprint arXiv:2307.15771*, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and Editing Factual Associations in GPT, October 2022. URL `http://arxiv.org/abs/2202.05262`. arXiv:2202.05262 [cs].

Aditya Paliwal, Sarah Loos, Markus Rabe, Kshitij Bansal, and Christian Szegedy. Graph Representations for Higher-Order Logic and Theorem Proving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition, 2024. URL `https://arxiv.org/abs/2312.06681`.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Stanislas Polu and Ilya Sutskever. Generative Language Modeling for Automated Theorem Proving. *arXiv preprint arXiv:2009.03393*, 2020a.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving, 2020b.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning, 2022.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2023.

Talia Ringer, Karl Palmskog, Ilya Sergey, Milos Gligoric, and Zachary Tatlock. QED at large: A survey of engineering of formally verified software. *Foundations and Trends in Programming Languages*, 2019. ISSN 23251131. doi: 10.1561/2500000045.

Alex Sanchez-Stern, Yousef Alhessi, Lawrence Saul, and Sorin Lerner. Generating Correctness Proofs with Neural Networks. In *Proceedings of the ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, 2020.

Daniel Scalena, Gabriele Sarti, and Malvina Nissim. Multi-property steering of large language models with dynamic activation composition. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 577–603. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.blackboxnlp-1.34. URL `http://dx.doi.org/10.18653/v1/2024.blackboxnlp-1.34`.

Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. Improving instruction-following in language models through activation steering, 2024. URL `https://arxiv.org/abs/2410.12877`.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. Extracting latent steering vectors from pretrained language models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 566–581, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.48. URL `https://aclanthology.org/2022.findings-acl.48/`.

Christian Szegedy, Markus Rabe, and Henryk Michalewski. Retrieval-Augmented Proof Step Synthesis. In *Proceedings of the Conference on Artificial Intelligence and Theorem Proving*, 2021.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=AwyxtyMwaG`.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering, 2024. URL `https://arxiv.org/abs/2308.10248`.

Szymon Tworkowski, Maciej Mikuła, Tomasz Odrzygóźdź, Konrad Czechowski, Szymon Antoniak, Albert Jiang, Christian Szegedy, Łukasz Kuciński, Piotr Miłoś, and Yuhuai Wu. Formal Premise Selection With Language Models. In *Proceedings of the Conference on Artificial Intelligence and Theorem Proving*, 2022.

Teun van der Weij, Massimo Poesio, and Nandi Schoots. Extending activation steering to broad skills and multiple behaviours, 2024. URL `https://arxiv.org/abs/2403.05767`.

Alexandre Variengien and Eric Winsor. Look before you leap: A universal emergent decomposition of retrieval tasks in language models. *arXiv preprint arXiv:2312.10091*, 2023.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. Causal mediation analysis for interpreting neural nlp: The case of gender bias. *arXiv preprint arXiv:2004.12265*, 2020.

Haiming Wang, Huajian Xin, Zhengying Liu, Wenda Li, Yinya Huang, Jianqiao Lu, Zhicheng Yang, Jing Tang, Jian Yin, Zhenguo Li, et al. Proving Theorems Recursively. *arXiv preprint arXiv:2405.14414*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL `https://arxiv.org/abs/2201.11903`.

Sean Welleck. Neural theorem proving tutorial. `https://github.com/wellecks/ntptutorial`, 2023.

Sean Welleck and Rahul Saha. LLMSTEP: LLM Proofstep Suggestions in Lean. In *International Conference on Neural Information Processing Systems Workshop on MATH-AI*, 2023.

Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. NaturalProver: Grounded Mathematical Proof Generation with Language Models. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2022.

Makarius Wenzel, Lawrence C Paulson, and Tobias Nipkow. The isabelle framework. In *Theorem Proving in Higher Order Logics: 21st International Conference, TPHOLs 2008, Montreal, Canada, August 18-21, 2008. Proceedings 21*, pp. 33–38. Springer, 2008.

Daniel Whalen. Holophrasm: A Neural Automated Theorem Prover for Higher-Order Logic. *arXiv preprint arXiv:1608.02644*, 2016.

Kaiyu Yang and Jia Deng. Learning to Prove Theorems via Interacting with Proof Assistants. In *Proceedings of the International Conference on Machine Learning*, 2019.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem Proving with Retrieval-Augmented Language Models. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2023a.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Neural Information Processing Systems (NeurIPS)*, 2023b.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, Yudong Wang, Zijian Wu, Shuaibin Li, Fengzhe Zhou, Hongwei Liu, Songyang Zhang, Wenwei Zhang, Hang Yan, Xipeng Qiu, Jiayu Wang, Kai Chen, and Dahua Lin. Internlm-math: Open math large language models toward verifiable reasoning, 2024a. URL `https://arxiv.org/abs/2402.06332`.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, et al. InternLM-Math: Open Math Large Language Models Toward Verifiable Reasoning. *arXiv preprint arXiv:2402.06332*, 2024b.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models, 2024. URL `https://arxiv.org/abs/2310.06117`.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=9ZPegFuFTFv`.