



Figure 7: Comparison between problems solved with and without steering.

A APPENDIX

In this section, we present a series of representative proof trajectories illustrating both the capabilities and failure modes of language models, and demonstrate how activation steering ameliorates key deficiencies. These examples drawn from Lean and Coq benchmarks reveal not only the quantitative improvements afforded by steering vectors, but also their qualitative impact on proof structure and robustness.

A.1 MINIF2F PROBLEMS SOLVED

We observe that the incorporation of steering vectors results in the generation of a substantial number of new proofs that are not derivable by the base language model alone. We show a detailed plot for all the problems in **miniF2F** in Figure 7.

This improvement in proof discovery is accompanied by a corresponding increase in failure cases. We hypothesize that this phenomenon arises due to biases introduced by the data used to construct the steering vectors. Specifically, the induced vector may encode inductive priors that align well with certain classes of theorems while misaligning with others, leading to erroneous proof trajectories. This trade-off highlights the importance of characterizing and formalizing the inductive biases embedded in steering vectors, particularly when generalizing across diverse proof distributions.

A.2 EXAMPLES OF PROOFS

A.2.1 HOW DOES STEERING HELP IN THE PROOF?

In the example below, we demonstrate how injecting a steering vector introduces the natural language information at each step that helps the LLM to write a proof more robustly. In the example below, we observe the use of high-level overview of the next step which helps in grounding the generation. We also compare it with the unsteered counterpart of the same problem and discuss the key differences in proof writing.

Example: *Mathd Algebra 156* in **miniF2F** benchmark.

```
import Mathlib
import Aesop

set_option maxHeartbeats 0

open BigOperators Real Nat Topology Rat

theorem mathd_algebra_156 (x y : ℝ) (f g : ℝ → ℝ) (h₀ : ∀ t, f t
  ⇔ = t ^ 4)
  (h₁ : ∀ t, g t = 5 * t ^ 2 - 6) (h₂ : f x = g x) (h₃ : f y = g
  ⇔ y) (h₄ : x ^ 2 < y ^ 2) :
```

```

810      y ^ 2 - x ^ 2 = 1 := by
811
812      -- Apply function definitions to rewrite f x and g x in terms of
813      ↪ powers of x
814      have eq1 : x^4 = 5 * x^2 - 6 := by
815        rw [h0, h1] at h2
816        linarith
817
818      -- Apply function definitions to rewrite f y and g y in terms of
819      ↪ powers of y
820      have eq2 : y^4 = 5 * y^2 - 6 := by
821        rw [h0, h1] at h3
822        linarith
823
824      -- Rearranged form: x^4 - 5 x^2 + 6 = 0
825      -- Use known factorization of the quartic as product of
826      ↪ quadratics
827      have h5 : (x^2 - 2) * (x^2 - 3) = 0 := by
828        have h1 : x^4 - 5 * x^2 + 6 = 0 := by linarith
829        have h2 : x^4 - 5 * x^2 + 6 = (x^2 - 2) * (x^2 - 3) := by
830          ring
831          nlinarith
832
833      -- Apply same reasoning for y using identical polynomial
834      ↪ structure
835      have h6 : (y^2 - 2) * (y^2 - 3) = 0 := by
836        have h1 : y^4 - 5 * y^2 + 6 = 0 := by linarith
837        have h2 : y^4 - 5 * y^2 + 6 = (y^2 - 2) * (y^2 - 3) := by
838          ring
839          nlinarith
840
841      -- Conclude that x^2 must be either 2 or 3 based on zero product
842      ↪ property
843      have hx2 : x^2 = 2 ∨ x^2 = 3 := by
844        cases (mul_eq_zero.mp h5) with
845        | inl h =>
846          left
847          nlinarith
848        | inr h =>
849          right
850          nlinarith
851
852      -- Similarly, conclude that y^2 must be either 2 or 3
853      have hy2 : y^2 = 2 ∨ y^2 = 3 := by
854        cases (mul_eq_zero.mp h6) with
855        | inl h =>
856          left
857          nlinarith
858        | inr h =>
859          right
860          nlinarith
861
862      -- Perform case analysis on possible values of x^2 and y^2
863      cases hx2 with
864      | inl hx2l =>
865        -- Case: x^2 = 2
866        cases hy2 with
867        | inl hy2l =>
868          -- Case: y^2 = 2, contradicts h4 : x^2 < y^2

```

```

864         exfalse
865         nlinarith
866     | inr hy2r =>
867         -- Case:  $y^2 = 3$ , compute difference
868         have h :  $y^2 - x^2 = 1$  := by
869             rw [hy2r, hx2l]
870             linarith
871         linarith
872 | inr hx2r =>
873     -- Case:  $x^2 = 3$ 
874     cases hy2 with
875     | inl hy2l =>
876         -- Case:  $y^2 = 2$ , contradicts  $x^2 < y^2$ 
877         exfalse
878         nlinarith
879     | inr hy2r =>
880         -- Case:  $y^2 = 3$ , contradicts  $x^2 < y^2$ 
881         exfalse
882         nlinarith

```

Comparison with Unsteered Proof The unsteered model demonstrates a breakdown in aligning its reasoning with the algebraic form of the goal. Specifically, it bypasses the canonical application of the zero-product property via `muleq_zero`, which is essential for decomposing polynomial equalities into tractable cases. Instead, it resorts to a general-purpose disjunction strategy using `eq_or_ne`, which introduces branches without constructive grounding and fails to anchor the analysis in the underlying factorization. This leads to brittle reasoning that lacks semantic traceability to the original assumptions. In contrast, the steered model adheres to a proof structure that mirrors standard mathematical practice: it identifies the factorable structure of the quartic polynomials, applies `muleq_zero` to extract meaningful cases, and maintains the logical continuity required to propagate constraints across steps. The comparison reveals how steering enhances the model’s sensitivity to algebraic invariants and encourages the use of idiomatic tactics, leading to more reliable and interpretable proofs.

Diff of proofs

```

895 diff --git a/mathd_algebra_156_internlm2-5_w_steering.lean b/
896     ↪ mathd_algebra_156_internlm2-5_wo_steering.lean
897 index b79f717..7f48d26 100644
898 --- a/mathd_algebra_156_internlm2-5_w_steering.lean
899 +++ b/mathd_algebra_156_internlm2-5_wo_steering.lean
900 @@ -3,0 @@ import Aesop
901 -
902 @@ -5,0 @@ set_option maxHeartbeats 0
903 -
904 @@ -11,56 +9,47 @@ theorem mathd_algebra_156 (x y : ℝ) (f g : ℝ →
905     ↪ ℝ) (h₀ : ∀ t, f t = t
906 -
907 - have eq1 :  $x^4 = 5 * x^2 - 6$  := by
908 -   rw [h₀, _1h] at _2h
909 -   linarith
910 -
911 - have eq2 :  $y^4 = 5 * y^2 - 6$  := by
912 -   rw [h₀, _1h] at _3h
913 -   linarith
914 -
915 - have h5 :  $(x^2 - 2) * (x^2 - 3) = 0$  := by
916 -   have h1 :  $x^4 - 5 * x^2 + 6 = 0$  := by linarith
917 -   have h2 :  $x^4 - 5 * x^2 + 6 = (x^2 - 2) * (x^2 - 3)$  := by
918 -     ring
919 -     nlinarith

```

```

918 -
919 - have h6 : (y^2 - 2) * (y^2 - 3) = 0 := by
920 -   have h1 : y^4 - 5 * y^2 + 6 = 0 := by linarith
921 -   have h2 : y^4 - 5 * y^2 + 6 = (y^2 - 2) * (y^2 - 3) := by
922 -     ring
923 -     nlinarith
924 -
925 - have hx2 : x^2 = 2 ∨ x^2 = 3 := by
926 -   cases (mul_eq_zero.mp h5) with
927 -   | inl h =>
928 -     left
929 -     nlinarith
930 -   | inr h =>
931 -     right
932 -     nlinarith
933 -
934 - have hy2 : y^2 = 2 ∨ y^2 = 3 := by
935 -   cases (mul_eq_zero.mp h6) with
936 -   | inl h =>
937 -     left
938 -     nlinarith
939 -   | inr h =>
940 -     right
941 -     nlinarith
942 -
943 - cases hx2 with
944 - | inl hx2l =>
945 -   cases hy2 with
946 -   | inl hy2l =>
947 -     exfalse
948 -     nlinarith
949 -   | inr hy2r =>
950 -     have h : y^2 - x^2 = 1 := by
951 -       rw [hy2r, hx2l]
952 -       linarith
953 -     linarith
954 -   | inr hx2r =>
955 -     cases hy2 with
956 -     | inl hy2l =>
957 -       exfalse
958 -       nlinarith
959 -     | inr hy2r =>
960 -       have hx : x ^ 4 = 5 * x ^ 2 - 6 := by rw [h0, ← 1h, 2h]
961 -       have hy : y ^ 4 = 5 * y ^ 2 - 6 := by rw [h0, ← 1h, 3h]
962 -       +
963 -       have eq_x : x ^ 4 - 5 * x ^ 2 + 6 = 0 := by linarith [hx]
964 -       have eq_y : y ^ 4 - 5 * y ^ 2 + 6 = 0 := by linarith [hy]
965 -       +
966 -       have factor_x : (x ^ 2 - 2) * (x ^ 2 - 3) = 0 := by
967 -         calc (x ^ 2 - 2) * (x ^ 2 - 3)
968 -         = x ^ 4 - 3 * x ^ 2 - 2 * x ^ 2 + 6 := by ring
969 -         = x ^ 4 - 5 * x ^ 2 + 6 := by ring
970 -         = 0 := eq_x
971 -       +
972 -       have factor_y : (y ^ 2 - 2) * (y ^ 2 - 3) = 0 := by
973 -         calc (y ^ 2 - 2) * (y ^ 2 - 3)
974 -         = y ^ 4 - 3 * y ^ 2 - 2 * y ^ 2 + 6 := by ring
975 -         = y ^ 4 - 5 * y ^ 2 + 6 := by ring
976 -         = 0 := eq_y

```

```

972 +
973 + have x_cases : x ^ 2 = 2 ∨ x ^ 2 = 3 := by
974 +   cases' eq_or_ne (x ^ 2 - 2) 0 with h h
975 +   · left; linarith [h]
976 +   · right
977 +     have : x ^ 2 - 3 = 0 := by
978 +       have : (x ^ 2 - 2) ≠ 0 := h
979 +       exact eq_of_mul_eq_zero_right factor_x this
980 +       linarith [this]
981 +
982 + have y_cases : y ^ 2 = 2 ∨ y ^ 2 = 3 := by
983 +   cases' eq_or_ne (y ^ 2 - 2) 0 with h h
984 +   · left; linarith [h]
985 +   · right
986 +     have : y ^ 2 - 3 = 0 := by
987 +       have : (y ^ 2 - 2) ≠ 0 := h
988 +       exact eq_of_mul_eq_zero_right factor_y this
989 +       linarith [this]
990 +
991 + cases' x_cases with hx_2 hx_3
992 +   · cases' y_cases with hy_2 hy_3
993 +     · ex falso
994 +       rw [hx_2, hy_2] at 4h
995 +       exact lt_irrefl 2 4h
996 +     · rw [hy_3, hx_2]
997 +       norm_num
998 +   · cases' y_cases with hy_2 hy_3
999 +     · ex falso
1000 +     · rw [hx_3, hy_2] at 4h
1001 +     · norm_num at 4h
1002 @@ -68 +57,2 @@
1003 - nlinarith
1004 +   rw [hx_3, hy_3] at 4h
1005 +   exact lt_irrefl 3 4h

```

A.2.2 DOES STEERING INTRODUCE ONLY USEFUL NATURAL LANGUAGE INFORMATION?

We analyze the proofs and the natural language incorporation added by steering vectors, and we were surprised to observe that in few cases the natural language instructions were incorrect and the model was able to come up with the correct formal proofs. This shows the capability of choosing the correct step not only depends on the correctness of the thoughts but also at the confidence of the model at the time of generation.

Example of Mathd Algebra 313 where the intermediate step is incorrectly annotated.

The failure arises from the premature application of the simplification tactic `field.simp` immediately after the symbolic rewrite of i as v/z , based on the hypothesis $v = i \cdot z$. At this stage of the proof, neither v nor z have been instantiated with their concrete values. Consequently, the simplification is applied to the expression $i = (i \cdot z)/z$, which reduces trivially to $i = i$ under cancellation. While syntactically valid, this reasoning is circular and fails to advance toward the proof goal, which requires computing an explicit value for i . Furthermore, such simplification is semantically brittle because it assumes invertibility of z (i.e., $z \neq 0$), which is not yet established or unfolded in the proof context. The appropriate use of `field.simp` should follow substitution of the concrete definitions for both v and z , ensuring the resulting expression is a well-defined complex fraction amenable to symbolic manipulation and numeric normalization.

Lean Proof for Mathd Algebra 313

```

1023 import Mathlib
1024 import Aesop

```

```

1026
1027 set_option maxHeartbeats 0
1028
1029 open BigOperators Real Nat Topology Rat
1030
1031 theorem mathd_algebra_313 (v i z : ℂ) (h₀ : v = i * z) (h₁ : v = 1
1032   ↪ + Complex.I
1033 ) (h₂ : z = 2 - Complex.I) : i = 1 / 5 + 3 / 5 * Complex.I := by
1034
1035   -- rewrite i in terms of v and z, to isolate i from the product
1036   have h3 : i = v / z := by
1037
1038     -- replace v with i x z from hypothesis h₀
1039     rw [h₀]
1040
1041     -- simplify division by cancelling z terms directly (incorrect
1042     ↪ )
1043     field_simp [Complex.ext_iff, h₂]
1044
1045     -- replace i by the fraction v/z, then rewrite v and z
1046     rw [h3, h₁, h₂]
1047
1048     -- simplify complex division into real and imaginary parts
1049     field_simp [Complex.ext_iff]
1050
1051     -- normalize numbers to explicit fractions
1052     norm_num
1053

```

A.3 TRANSFERABILITY OF STEERING VECTORS

To investigate the transferability of steering vectors across formal systems, we evaluated their impact when applied to a Coq-based proof assistant. The primary objective of this evaluation was to assess whether these steering vectors merely encode syntactic preferences for generating natural language interleaved with formal proof steps, or whether they also capture reasoning strategies that generalize across distinct proof environments.

In this setting, we observed that the InternLM2.5 model, when guided by the learned steering vectors, was able to successfully construct a proof for *Putnam_1988_B1* in Coq. In contrast, the same model failed to complete the corresponding proof in Lean. This discrepancy suggests that the steering vectors encapsulate transferable reasoning features that are not specific to the prover, possibly due to differences in tactic libraries, proof term verbosity, or the model’s learned prior over formal syntax. The result highlights the non-trivial interaction between the encoded reasoning direction and the target proof environment, underscoring the potential of steering vectors as cross-system inductive biases.

Problem Statement for *Putnam 1988 B1*

A composite number is a positive integer that can be expressed as
 \hookrightarrow a product (ab) , where (a) and (b) are (not necessarily
 \hookrightarrow distinct) integers from the set $(\{2, 3, 4, \text{dots}\})$.

Problem: Show that every composite number can be written in the
 \hookrightarrow form

$$xy + xz + yz + 1,$$

where (x, y, z) are positive integers.

Putnam 1988 B1 proof in Coq

```

1080 Require Import Arith Lia Ring.
1081
1082 (* A number is composite if it can be written as a product of two
1083     $\hookrightarrow$  integers greater than 1 *)
1084 Definition composite (n: nat) : Prop :=
1085   exists x y, 1 < x /\ 1 < y /\ n = x * y.
1086
1087 Theorem Putnam_1988_B1 n :
1088   3 < n ->
1089   composite n ->
1090   exists x y z : nat,
1091     0 < x /\ 0 < y /\ 0 < z
1092     /\ n = x*y + y*z + z*x + 1.
1093 Proof.
1094   (* Using the definition of composite, obtain integers a and b
1095       $\hookrightarrow$  such that  $n = a * b$  with  $a, b > 1$  *)
1096   intros _ [a [b [Ha [Hb Hab]]]].
1097
1098   (* The goal is to express n in the form  $x*y + y*z + z*x + 1$ 
1099       $\hookrightarrow$  where x, y, z are positive *)
1100
1101   (* Define x and y as one less than a and b, respectively *)
1102   set (x := a - 1). (* Then  $a = x + 1$  *)
1103   set (y := b - 1). (* Then  $b = y + 1$  *)
1104
1105   (* Set z to 1 in order to satisfy symmetry in the expression *)
1106   set (z := 1).
1107
1108   (* Establish positivity of x, y, z *)
1109   exists x, y, z.
1110
1111   (* Since  $a > 1$ ,  $x = a - 1 > 0$  *)
1112   split; [ lia | ].
1113
1114   (* Since  $b > 1$ ,  $y = b - 1 > 0$  *)
1115   split; [ lia | ].
1116
1117   (* z is defined as 1, so clearly positive *)
1118   split; [ lia | ].
1119
1120   (* Now verify the identity  $n = x*y + y*z + z*x + 1$  *)
1121   unfold x, y, z.
1122
1123   (* Replace n by  $a * b$  as given in the hypothesis *)
1124   rewrite Hab.
1125
1126   (* Simplify the right-hand side of the target expression *)
1127   simpl.
1128
1129   (* Resolve the final arithmetic equivalence *)
1130   lia.
1131 Qed.
1132
1133

```