SPLICING HINTS INTO REASONING TRACES: LEARNING FROM DEMONSTRATOR FAILURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards improves mathematical reasoning when a final answer can be checked, but a correct answer does not certify the derivation behind it, and proof problems offer no answer to check. Replacing the check with a reasoning model as an in-loop verifier faces a fundamental obstacle: the policy must already produce occasional correct attempts for any reward to exist, which it may never do at olympiad level. We instead move verification and repair out of the training loop and into offline data generation with SPLICE, a workflow in which a second model, given only a reference solution, edits a reasoning model's traces as they are generated: it verifies progress, truncates a failed trace at its last verified step, and splices in an adaptively strengthened hint until progress resumes. The resulting dataset includes repaired examples whose sampled unaided trajectory failed, with failed continuations removed and progress repeatedly checked against the reference solution. A model finetuned on SPLICE traces outperforms the same model distilled from the reasoning model's unedited traces on AIME and, under a grading protocol validated against human grades, on olympiad proofs. The trained model's own traces show a corresponding behavioral signature: at difficult points it announces a branch in the style of the spliced hints and takes riskier steps, with predictive entropy rising there and just after, a pattern absent from the model trained on unedited traces.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) improves mathematical reasoning when final answers can be checked automatically (DeepSeek-AI et al., 2026; Shao et al., 2024; Team et al., 2025; Lambert et al., 2025). Proof problems are different: the solution is the argument itself, and a correct final answer need not certify correct reasoning (Mahdavi et al., 2025; Petrov et al., 2025). A natural alternative is a strong reasoning model acting as a verifier (Zhang et al., 2025b; Bai et al., 2022; Lee et al., 2024; Yuan et al., 2024). With a reference solution, such verifiers can agree closely with human graders (Luong et al., 2025; Ma et al., 2025), but two obstacles remain. First, reinforcement learning needs occasional correct attempts to observe reward, which may be absent on olympiad-level problems (Qu et al., 2026; Zhang et al., 2026; Guo et al., 2026). Second, querying a verifier on every rollout compounds the dominant cost of large-scale RL (Sheng et al., 2025).

Test-time scaffolds decompose reasoning into generation, verification, and revision (Madaan et al., 2023; Shinn et al., 2023; Yao et al., 2023). They can reach very high performance when the underlying model is strong (Huang & Yang, 2025; Wei), but they depend on the model finding and fixing its own errors, which weaker models cannot do without external feedback (Huang et al., 2024). Existing proof-training recipes likewise generally begin with a model that already writes proofs and train on its traces (Hamid et al., 2026; LM-Provers et al., 2026; DeepSeek-AI, 2025). We instead supply verification and repair externally, only during data generation.

Our target is the single-attempt proof generation of an instruction-tuned **student** not post-trained for extended reasoning. A standard baseline finetunes it on traces from a larger **demonstrator** (Yang et al., 2025), either directly as distillation (Hinton et al., 2015; Ho et al., 2023; Magister et al., 2023; Hsieh et al., 2023) or after rejection sampling (Zelikman et al., 2022; Dong et al., 2023; Gulcehre

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

et al., 2023; Singh et al., 2024; Yuan et al., 2023). Both learn only from what the demonstrator can produce unaided.

We propose **SPLICE**, which repairs a demonstrator's failures as they occur (Figure 1). A third model, the **tutor**, receives a reference solution, checks whether the trace has advanced, and writes a hint only when needed; it is never asked to solve the problem. Because checking against a reference can be substantially easier than solving (Luong et al., 2025; Ma et al., 2025), the tutor can contribute competence without supplying a full reasoning trace and is absent at test time. SPLICE truncates failed continuations at the last constructive step, adaptively strengthens a hint until progress is restored in the demonstrator's own style, and resumes generation. A verifier applied after the attempt could only have scored the failed trace; acting during generation, the tutor changes what the demonstrator can reach. Thus the dataset can contain completed solutions built from sampled unaided trajectories that failed, while preserving the demonstrator's successful decisions. A failure here is a trajectory that stalled, not a problem the demonstrator could never solve.

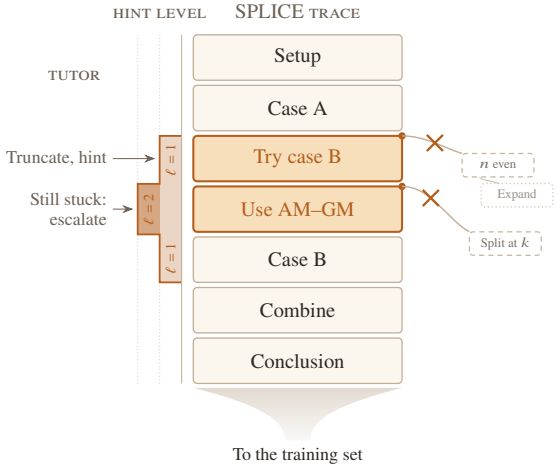


Figure 1: One SPLICE trace. When the tutor finds no newly verified step, the failed continuation is discarded, an adaptively strengthened hint is rewritten in the trace's style and appended to the last constructive prefix, and generation resumes. Hint level rises after stalls and falls after progress; only the surviving trace is retained.

The intervention is deliberately asymmetric. Successful reasoning remains untouched; an incorrect or stalled continuation is discarded rather than revised in place; and hint strength rises only after repeated stalls and falls when progress resumes. This makes the resulting target a single coherent solution rather than a tutor–student dialogue. It also gives each trace a generation record—where progress stopped, how much help was required, and whether the demonstrator recovered—that can later be used to diagnose harmful training examples.

We use Qwen3-4B-Instruct-2507 as student, Qwen3-30B-A3B-Thinking as demonstrator (Yang et al., 2025), and Gemini 2.5 Pro as tutor (Comanici et al., 2025), with cheaper family members for lighter stages (Appendix B). The controlled baseline finetunes the same student on untutored traces from the same demonstrator (Ho et al., 2023; Magister et al., 2023; Muennighoff et al., 2025; Ye et al., 2025). Under the human-validated grading protocol of Mahdavi et al. (2025), filtered SPLICE training raises the mean proof score from 0.97 to 1.15, an 18% gain, improving on 33 of 45 problems and worsening on 12, and doubles the number of attempts graded 6/7 or above; AIME accuracy rises from 60.6% to 64.4%. The trained student's traces also show a distinctive learned behavior: at difficult points it emits self-hint-like branch cues and predictive entropy rises in and especially after those passages.

Relation to prior work. SPLICE differs from outcome-level RLVR and rejection sampling, which learn from successful completed rollouts: when an unaided trajectory stalls, SPLICE edits the trajectory itself and supplies information needed to resume reasoning (Qu et al., 2026; Zhang et al., 2026; Guo et al., 2026). It is closest to hint-based methods such as POPE, ADHint, and HiLL, but those inject guidance into the learner's RL rollouts, whereas SPLICE spends privileged information once, during offline demonstrator trace construction, and the student receives no hinted prompt at test time (Qu et al., 2026; Zhang et al., 2025a; Xia et al., 2026). Finally, unlike ordinary or solution-conditioned teacher distillation, the reference-conditioned tutor does not author the surrounding solution: it identifies a stalled prefix and intervenes locally, while the demonstrator remains the

source of the reasoning trajectory (Kaur et al., 2026; Zhang, 2026). Appendix A gives the fuller comparison.

Our contributions are: **(i)** an offline trace-editing workflow that turns demonstrator failures into supervised training signal; **(ii)** a controlled comparison against untutored distillation, together with a rejection-sampled-success baseline; **(iii)** an analysis showing that SPLICE-trained students learn where to branch and explore; and **(iv)** a generation-record-based taxonomy of failure modes whose removal produces distinct, diagnostically consistent changes in evaluation behavior.

2 SPLICE: EDITING REASONING TRACES DURING GENERATION

SPLICE generates supervised reasoning traces by controlling the demonstrator's decoding rather than collecting its finished traces: generation can be stopped, the trace cut back to any prefix, and decoding resumed from there. We take efficient trace distillation as the baseline (Muennighoff et al., 2025); the tutor, holding a reference solution, watches the trace as the demonstrator writes it; where progress stalls, the trace is cut back to its last verified step, a hint is spliced in, and the demonstrator resumes from there, until the problem is solved. The trace that survives is one SPLICE trace.

2.1 BUILDING BLOCKS

We write $\mathcal{D}$ for the demonstrator, $\mathcal{T}$ for the tutor, and $\mathcal{M}$ for the student. In a loop, the **demonstrator** $\mathcal{D}$ writes the reasoning trace, chunk by chunk. We assume only that its reasoning traces are accessible and that generation can be resumed from a prefix we supply, which open-weight models provide. The **tutor** $\mathcal{T}$ is the model queried during generation; we assume only that it can compare a written solution against a reference solution. The **student** $\mathcal{M}$ is the instruction-tuned model finetuned on the finished dataset; it takes no part in generation. An instance is a pair (p, r) of a problem p and a human-written reference solution r. The demonstrator's output splits into the reasoning trace $c = (c_1, \ldots, c_n)$, whose steps c_i are the paragraphs, separated by blank lines, and the final solution a, the text following the reasoning block. Generation proceeds in iterations $t = 1, \ldots, B$; iteration t produces one chunk, a single call of at most L tokens.

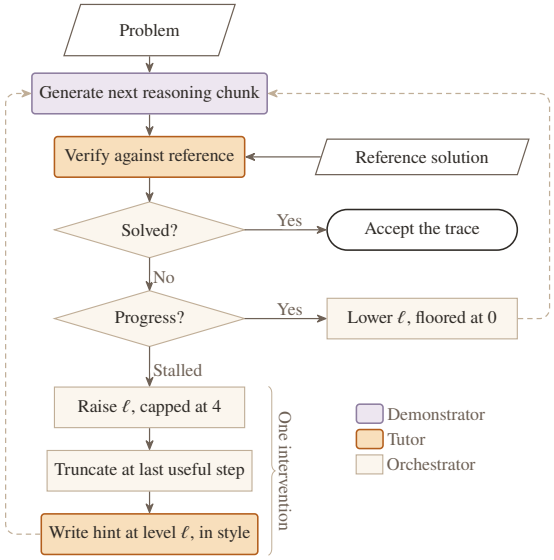


Figure 2: One iteration of the SPLICE loop. Colors mark which model performs each step; the bracketed steps are one intervention.

Figure 2 shows one iteration of the SPLICE loop. Each stage is one prompt to one model, and we denote each with a function notation. Generation is the single call GEN, which extends the trace by one chunk; every other stage belongs to the tutor. DECOMPOSE reads r once and returns the ordered substeps S. ACCEPT judges a finished solution a against r and S, returning a verdict together with a list E of errors located in a. ASSESS grades each substep against the solution and finds the last paragraph of the trace that did constructive work: a paragraph is constructive if it derives a correct substep, sets a relevant subgoal, validates a claim or identifies an error, proposes a new angle, takes up an earlier hint, or repairs an earlier mistake, and not if it merely restates what was done, repeats a known error, stalls without proposing an action, or jumps to a conclusion. HINT writes the content of an intervention at the current level, and RESTYLE rewrites that content in the demonstrator's idiom, using patterns Π drawn from the demonstrator's own reasoning and the retained prefix. Two further stages keep generation inside its budget: SHORTEN compresses an over-long trace, and STOPPED reports whether a finished solution is already sitting inside the rea-

Algorithm 1 SPLICE: generating one reasoning trace. The problem p, reference solution r, and decomposition S are shared by all procedures.

Require: problem p, reference solution r; constants B, L, N (Appendix C)
Ensure: an accepted trace (p, c, a), or failure

$S \leftarrow \textsc{Decompose}_{\mathcal{T}}(p, r)$; $c \leftarrow (\,)$; $a \leftarrow \varepsilon$; $\ell \leftarrow 0$; $V \leftarrow 0$
for $t = 1$ **to** B **do**
  $(c, a, \mathit{done}) \leftarrow \textsc{Extend}(c, a)$ ▷ demonstrator writes one chunk
  if not done **then continue**
  $(\mathit{ok}, E) \leftarrow \textsc{Accept}_{\mathcal{T}}(p, r, S, a)$ ▷ N samples, all must agree
  if ok **then return** (p, c, a) ▷ accepted: enters the dataset
  $(c, a, \ell, V) \leftarrow \textsc{Intervene}(c, a, E, \ell, V)$ ▷ tutor cuts and hints
return failure ▷ iteration budget B exhausted

procedure $\textsc{Extend}(c, a)$
  $x \leftarrow \textsc{Gen}_{\mathcal{D}}(p, c, a)$ ▷ at most L tokens
  if the reasoning block is still open **then**
    if $\textsc{Repeats}(x)$ **then**
      cut x at the repeat; close the block
    else if $\textsc{Stopped}_{\mathcal{T}}(p, c + x)$ **then**
      $(c, a) \leftarrow \textsc{SplitAtSolution}(c + x)$
    else
      return $(c{+}\textsc{Paragraphs}(x),\ a,\ \mathit{false})$
  $a \leftarrow a + x$
  if x hit the cap L **then return** (c, a, false)
  return (c, a, true) ▷ a solution exists

procedure $\textsc{Intervene}(c, a, E, \ell, V)$
  $(\sigma, k, u) \leftarrow \textsc{Assess}_{\mathcal{T}}(p, S, c, a, E)$
  $v \leftarrow |\{\, i : \sigma(i) = \mathit{correct} \,\}|$
  if $v > V$ **then** $\ell \leftarrow \max(0,\ \ell - 1)$ ▷ progress
  else $\ell \leftarrow \min(4,\ \ell + 1)$ ▷ stall
  $V \leftarrow \max(V,\ v)$
  $c \leftarrow c_{1:u}$ ▷ truncate
  $h_0 \leftarrow \textsc{Hint}_{\mathcal{T}}(p, r, S, c, a, \sigma, k, \ell, E)$
  $h \leftarrow \textsc{Restyle}_{\mathcal{T}}(p, h_0, c, \Pi)$
  return $(c + h,\ \varepsilon,\ \ell,\ V)$ ▷ resume reasoning

soning block. Only GEN lacks access to r, and only ACCEPT receives a without c. A deterministic orchestrator runs the loop, applies truncations, and stores state; it makes no semantic judgments.

Three quantities drive the loop, all read off ASSESS. The status map σ_t assigns each substep s_i of S a status in $\{\mathit{correct}, \mathit{incorrect}, \mathit{missing}\}$ according to a. Let $v_t = |\{\, i : \sigma_t(i) = \mathit{correct} \,\}|$ be the number of substeps marked *correct* at iteration t, and $V_t = \max_{s \le t} v_s$ the best count reached so far. Iteration t is a **stall** if it does not raise the best count, that is, $v_t \le V_{t-1}$; otherwise it is **progress**. The **pointer** k_t is the number of consecutive substeps marked *correct* from the start, the largest k with $\sigma_t(i) = \mathit{correct}$ for all $i \le k$. The **cut index** u_t is the last paragraph of c that ASSESS marks constructive.

The **hint level** $\ell_t \in \{0, 1, 2, 3, 4\}$, initialised to 0, controls how much a hint may reveal, and the levels form a ladder: level 0 names the next mathematical goal in one line, with no method; 1 adds the specific claim or constraint to establish next; 2 gives a concrete method for that claim, which identity or inequality to apply and what to check while applying it; 3 outlines the whole corrective argument with its intermediate claims; and 4 carries out the single step the demonstrator is stuck on, and nothing beyond it.

2.2 THE SPLICE LOOP

Algorithm 1 alternates generation with assessment. EXTEND handles unfinished reasoning: REPEATS cuts repeated passages, STOPPED detects a prematurely closed reasoning block, SPLITATSOLUTION separates an embedded solution, and PARAGRAPHS forms steps. If c exceeds $L_{\max}$, SHORTEN compresses it. These are budget controls rather than the core intervention; Section 3.2 measures the harm caused by shortening.

Once a solution exists, the main loop decides whether to keep the trace: ACCEPT is sampled N times and every attempt must agree, since one judgment from a strong model is not reliable enough to gate a training example; the grading protocol of Section 4.1 repeats samples for the same reason. Because ACCEPT sees a without c, a trace cannot be accepted on the strength of reasoning its written solution does not carry. A trace accepted at $t = 1$ is one the demonstrator solved unaided, with no hint spliced in; we flag these traces, since they form the Untutored-Correct control of Section 4.4.

If the trace is rejected, INTERVENE carries out the intervention, which is where the method differs from resampling. The hint level moves first: it falls on progress and rises on a stall, so the amount

revealed tracks what this demonstrator has needed on this attempt rather than what the problem is presumed to require. Because ℓ is capped at 4, the pipeline can fail: a problem the demonstrator cannot advance on even when handed the step is never completed. At each intervention the trace is cut back to $c_{1:u}$, discarding everything after the last paragraph ASSESS marked constructive. The discarded text never reaches the dataset, and the demonstrator is never asked to revise it: the hint is appended to the constructive prefix and generation resumes from there.

The hint targets s_{k+1}, the first substep not marked *correct*. HINT fixes its mathematical content at level ℓ, while RESTYLE separately rewrites it in the demonstrator's idiom so the supervised target does not alternate visibly between speakers. Each trace stores hint levels, interventions, inserted text, and statuses σ_t, enabling the analyses in Sections 3.2 and 4.4.

Two design choices are central. First, progress is measured against the best substep count reached so far rather than the immediately preceding iteration, because truncation, stochastic tutor judgments, and a demonstrator that ignores the hint and drifts off course all make the raw count nonmonotone. Second, failed continuations are removed before a hint is inserted. Each SPLICE trace therefore contains the constructive prefix, the intervention, and the continuation that follows, never the demonstrator's known error and its subsequent correction. This is what distinguishes trace editing from ordinary critique-and-revision data.

2.3 MAKING THE DEMONSTRATOR FOLLOW A HINT

A reasoning model does not take instruction from inside its own trace. Text inserted into a chain of thought is often ignored, even when it is the correct solution (Cuesta-Ramirez et al., 2025), and an insertion can instead make the model abandon its reasoning and close the block early (Tang et al., 2025; Wu et al., 2025). We saw both in our early runs: resumed from a prefix that ends in a hint, the demonstrator would either close the reasoning block at once, or open the next paragraph with a pivot ("Wait", "But", "Alternatively", "Actually") that discarded the hint and returned to its own line. Three measures address this. Restyling (Section 2.2) makes the hint read as the demonstrator's own thought. Every hint opens by refusing closure: a statement that the reasoning is not finished, one verification to perform first, and two to five explicit questions drawn from the errors E and the next substep s_{k+1}, which restyling keeps as questions (a level-4 hint replaces the questions with an instruction to check the supplied derivation and integrate it). For the first 200 tokens after an insertion, we suppress two kinds of tokens at decoding: the end-of-reasoning token, so the block cannot close, and a fixed list of pivot words (Appendix C), so the demonstrator has to engage with the hint before it can reject it. These measures raise the rate at which a hint is taken up but do not make it certain; the problems on which every hint is ignored are the ones the loop fails on.

2.4 PROBLEM SET

We ran the loop on 2,500 problems: AIME problems from 1983 to 2023, and proof problems from the IMO (1959 to 2024), the IMO Shortlist (1966 to 2023) and Longlists, Shortlist-based team selection tests, Baltic Way, Tuymaada, USAMO, and the Balkan MO Shortlist, none later than 2024, with reference solutions from the corresponding Art of Problem Solving (AoPS) threads. The pool is disjoint from the evaluation exams of Section 4.1: it contains no AIME problem after 2023, no Canadian olympiad problem, no Iranian olympiad problem after 2021, and no IMO problem after 2024. It accepted traces for 2,130 of them, 778 AIME and 1,352 proof problems; on the rest the budget ran out at the top hint level with the demonstrator ignoring or contradicting every hint, so the dataset is selected on completion by construction.

3 THE SPLICE TRACES

3.1 YIELD AND COVERAGE

Of 2,130 accepted traces, 768 are unaided and 1,362 contain at least one spliced hint. Hinted traces are markedly less probable under the demonstrator's base model than untutored traces: median per-token negative log-likelihood is 0.25 nats for untutored traces, 0.29 for unaided SPLICE traces, and 0.38 for hinted traces; fork density (tokens per thousand with predictive entropy above 2 nats)

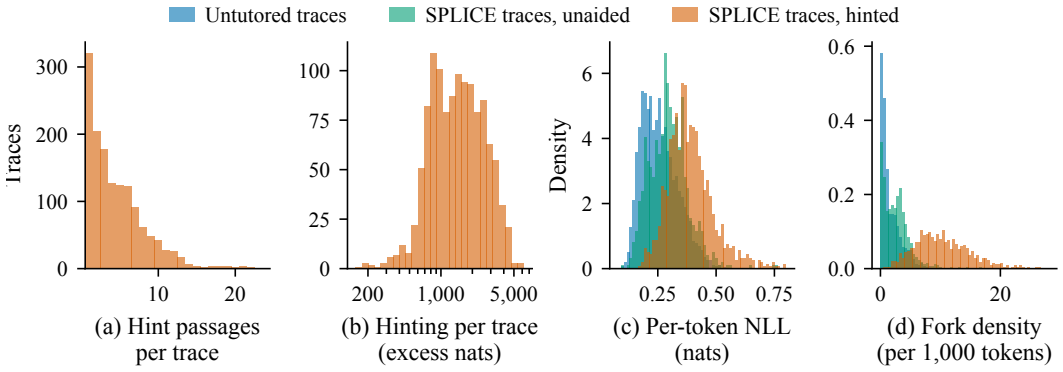


Figure 3: SPLICE trace statistics: (a) hint passages per hinted trace; (b) amount of hinting per trace, measured by excess hint-token negative log-likelihood; (c) per-token negative log-likelihood and (d) fork density under the demonstrator's base model for untutored, unaided SPLICE, and hinted SPLICE traces.

is 1.0, 2.4, and 9.9, respectively (Figure 3). Hinting is long-tailed, and its upper tail contains the excessive-hinting failure mode analyzed below.

3.2 Failure modes of the pipeline

The generation record exposes three systematic defects among the 2,130 accepted traces: compression artifacts from SHORTEN (98 traces), leakage of the reasoning block into the final solution (18), and excessive hinting at the maximum level without verified progress (238). The first two corrupt the generated target; the third records failure to progress and can teach that behavior. We therefore predicted that removing the first two would improve the top of the score distribution, while removing excessive hinting would primarily improve termination; Section 4.3 tests these predictions. The same cut also drops 10 traces whose write-up answers a different problem from the one posed.

4 Results

4.1 Datasets and evaluation protocol

We finetune the same student, Qwen3-4B-Instruct-2507, with one recipe on five datasets, so that the training data are the only experimental variable. **Untutored** is the distillation baseline: the demonstrator's 2,500 traces, generated with no tutor. **Untutored-Correct** is the rejection-sampling baseline: the 768 of those traces that solved the problem unaided; containing no hint, it is also the negative control for the self-hint analysis of Section 4.4. **SPLICE-NF** is all 2,130 accepted SPLICE traces; **SPLICE** removes the generation-failure traces (2,009); **SPLICE-HC** also removes excessive hinting (1,790). The first two against SPLICE ask whether repairing the demonstrator's failures outperforms distilling or discarding them; the three SPLICE datasets isolate each filter. We evaluate each trained student single-attempt, with no scaffold, retries or tutor, on two tracks. The AIME track is the 90 problems of AIME 2024–2026, eight samples each, scored by the boxed answer. The proof track is 45 problems from eight Canadian, Iranian and IMO exams of 2024–2026, 32 samples each, graded on the final write-up and never on the reasoning trace.

We grade with the protocol of Mahdavi et al. (2025): each sample is matched to a cluster of the reference solutions and scored from 0 to 7 against a rubric authored for that cluster. Scores aggregate three judges, one over repeated draws; the equal-weighted aggregate agrees with human grades at Pearson $r = 0.80$ on 90 human-graded proofs. Appendix G gives the judges, the weighting, the validation and the full scoring rule; thresholds below apply to the aggregate score. Beyond the mean score, we report two threshold metrics for proofs: existence at T, the share of the 45 problems with at least one of the 32 attempts scoring $\geq T$, and the number of attempts scoring $\geq T$ among the 1,440. Existence asks whether a problem is within the student's reach at all; the count asks how

Table 1: Student performance. AIME accuracy (%), eight samples per problem, 30 per exam; proof grading over 1,440 attempts per student: aggregate mean score, existence, the share of the 45 problems with a sample scoring $\geq T$, and the number of attempts scoring $\geq T$. Best in bold, second in blue.

	AIME accuracy (%)				Proofs				
						Existence, T (%)		Attempts, T (n)	
	2024	2025	2026	All	Mean	≥ 5	≥ 6	≥ 5	≥ 6
Untutored	63.3	60.0	58.3	60.6	0.97	15.6	11.1	41	20
Untutored-Correct	65.8	57.1	56.7	59.9	1.04	13.3	11.1	48	29
SPLICE-NF	65.0	59.6	60.0	61.5	**1.21**	20.0	11.1	62	25
SPLICE	**67.9**	**64.6**	60.8	**64.4**	1.15	**24.4**	13.3	**70**	41
SPLICE-HC	65.0	**64.6**	**62.9**	64.2	1.14	20.0	**17.8**	69	**45**

often it gets there. Intervals are 95% percentile bootstrap intervals over problems (45 for proofs, 90 for AIME), paired across students, with 10,000 resamples; best-of-k intervals also resample the 32 solutions within each problem.

Force-closing. Students finetuned on a few thousand traces sometimes fall into repetition and never emit the end-of-reasoning marker. On AIME this is harmless: we score the last boxed answer wherever it appears, so a looping attempt is scored like any other and is usually wrong. On proofs it leaves nothing to grade, since only the solution written after the reasoning block is graded, never the trace, and dropping such attempts would favor whichever student terminates most often. We therefore cut repetitions at their onset (or otherwise at the last complete paragraph), append a fixed closure, and allow at most 5,000 additional tokens.

4.2 Performance of the trained students

Main performance. On AIME (Table 1), filtered SPLICE improves overall accuracy from 60.6% to 64.4% (64.2% for SPLICE-HC); SPLICE-NF and Untutored-Correct remain near baseline. On proofs, every SPLICE variant raises the mean score, SPLICE by $+0.17$ and SPLICE-NF by $+0.24$, but only the filtered datasets improve its top: attempts scoring ≥ 6 double, from 20 to 41 and 45, and SPLICE-HC has existence at 6 of 17.8% against the baseline's 11.1%.[1]

Where the gain sits. Table 2 shows that the best-of-k gap grows with sampling: relative to Untutored, SPLICE gains $+0.21$ at $k = 4$ and $+0.31$ at $k = 32$, spread over 27 of the 45 problems. The gain is clear of zero through $k = 8$ ($+0.20$, CI $+0.02$ to $+0.39$) and no longer resolved at $k = 16$ (CI -0.03 to $+0.44$), since the best of many samples is an extreme statistic; SPLICE-NF's gain holds at every k. At thresholds 6 and above, filtering is decisive: SPLICE-NF sits within two points of the baseline curve while the filtered datasets lead it by three to six. Appendix E gives the full score distribution.

Conditioning. The gains are not explained only by reaching a gradeable write-up more often. Conditioned on natural termination, each SPLICE-trained student scores above Untutored; conditioned on attempts already scoring ≥ 4, the filtered datasets retain their advantage. Force-closed SPLICE attempts can also be useful when the cut removes repetition after a solution has effectively been found. Detailed termination and conditioning statistics are in Appendix E. The proof results also separate *breadth* from *depth*: SPLICE-NF often produces reasonable attempts and has the highest mean score, but filtering matters from score 5 upward.

[1] Paired over problems, 95% bootstrap intervals, SPLICE / SPLICE-HC / SPLICE-NF against Untutored: AIME $+3.9$ [$+1.0$, $+6.9$], $+3.6$ [$+0.7$, $+6.7$], $+1.0$ [-2.4, $+4.3$] points; mean proof score $+0.17$ [$+0.09$, $+0.27$], $+0.16$ [$+0.07$, $+0.26$], $+0.24$ [$+0.14$, $+0.35$]; rate of attempts scoring ≥ 5, $+2.0$ [$+0.6$, $+3.8$], $+1.9$ [$+0.3$, $+4.1$], $+1.5$ [$+0.1$, $+3.3$] points. Against Untutored-Correct: AIME $+4.6$, $+4.3$, $+1.7$, the last not clear of zero; mean score $+0.11$ [-0.01, $+0.23$], $+0.10$ [-0.02, $+0.22$], $+0.17$ [$+0.09$, $+0.27$]; rate ≥ 5, $+1.5$, $+1.5$, $+1.0$, all three clear of zero.

Table 2: Expected best-of-k score, and pass@k (%), the probability that at least one of k samples scores $\geq T$; both unbiased estimators over each problem's 32 samples, averaged over problems. pass@32 is the existence columns of Table 1, and pass@1 the share of attempts in Appendix E.

	Best-of-k score				pass@k (%)							
					$T=4$		$T=5$		$T=6$		$T>6$	
k	4	8	16	32	4	16	4	16	4	16	4	16
Untutored	1.69	2.03	2.33	2.60	13.9	20.9	7.4	12.2	4.2	8.7	4.0	8.5
Untutored-Correct	1.82	2.18	2.52	2.85	15.9	25.6	8.3	12.2	6.2	10.8	5.5	10.5
SPLICE-NF	**1.96**	**2.30**	**2.61**	2.87	**19.2**	**28.5**	10.3	16.4	5.5	10.4	5.0	10.0
SPLICE	1.90	2.23	2.56	2.91	16.9	24.2	**11.8**	**18.9**	7.9	12.2	7.4	12.1
SPLICE-HC	1.90	2.25	2.58	**2.94**	17.9	26.2	10.3	16.1	**8.3**	**14.2**	**8.0**	**14.1**

4.3 What filtering changed

The generation record predicts what each filter changes. Removing 121 generation-failure traces (SPLICE-NF $\rightarrow$ SPLICE) costs a little at low scores but improves the top of the proof distribution: at score ≥ 6, SPLICE has 41 attempts on 6 problems against SPLICE-NF's 25 attempts on 5 problems, and among attempts already scoring ≥ 4 the share reaching 6 rises from 19% to 33%. AIME likewise moves from +1.0 to +3.9 points over Untutored. These are the dimensions expected to suffer when the target contains compression artifacts or a leaked reasoning block.

Removing a further 219 traces with maximum-level hinting but no progress (SPLICE $\rightarrow$ SPLICE-HC) has a different effect. Natural termination rises from 820 to 927 of 1,440 attempts and the share of force-closed attempts that had entered a repetition falls from 80% to 70%, while the top of the distribution moves little, 45 attempts scoring ≥ 6 on 8 problems against 41 on 6. Thus the two filters do not behave as generic quality filters: generation-failure filtering improves the top of the score distribution, whereas excessive-hint filtering primarily improves termination. This directional agreement matters because the filters were defined from generation records rather than downstream test labels: the defects identified during trace construction predict the dimensions of student behavior that change after those traces are removed.

Untutored-Correct is the rejection-sampling contrast, and on this problem set it cannot be scaled: the 768 traces are every problem the demonstrator solves unaided. The question is whether the hinted traces add to them, and they do: SPLICE-NF, which contains all 768, gains $+0.17$ on the mean proof score over Untutored-Correct (CI $+0.09$ to $+0.27$), and the filtered datasets gain $+4.6$ and $+4.3$ points on AIME, where Untutored-Correct scores 59.9% against the baseline's 60.6%. What the correct traces alone recover is $+0.07$ on the mean and 29 attempts scoring ≥ 6 against the baseline's 20, on the same five problems, and the student often fails to terminate. SPLICE adds training signal exactly where rejection sampling has none: after an unaided trajectory has ceased to make constructive progress. Across SPLICE variants, more hinting in training is also associated with more self-hinting and repetition at evaluation (Appendix F).

4.4 What the trained student stores

Hint tokens are only 3.5 to 4.6% of the training tokens, but six times less probable under the untrained student than the demonstrator's own text, so they carry a fifth to a quarter of the initial finetuning loss. With no tutor at evaluation, SPLICE and SPLICE-HC emit self-hint passages on 30.5% and 23.1% of proof samples (22.2% and 15.0% on AIME), before any detected repetition; students trained without hints emit none. The proof/AIME emission ratio also tracks the corresponding training-set hint ratio, suggesting that the behavior is context-conditional rather than a generic verbal tic. We test whether these passages mark decision points by scoring evaluation traces under the trained student and its pre-finetuning checkpoint, separating the self-hint, up to three following paragraphs, and other text. Table 3 reports per-token loss and fork density (tokens per thousand with predictive entropy above 2 nats).

Table 3: Per-token NLL and fork density (tokens per thousand with predictive entropy > 2 nats) by trace region. "Following" is up to three paragraphs after a self-hint; text after detected repetition is excluded.

	Region	NLL base	NLL trained	Relative drop	Fork density base → trained
Untutored	All text	0.330	0.121	63%	6.8 → 2.9
SPLICE	Other text	0.399	0.133	67%	7.7 → 3.6
	Self-hint passage	0.258	0.059	77%	10.7 → **12.5**
	Following text	0.521	0.182	65%	17.3 → **35.0**
SPLICE-HC	Other text	0.404	0.136	66%	7.7 → 3.9
	Self-hint passage	0.268	0.064	76%	11.7 → **14.4**
	Following text	0.579	0.203	65%	18.0 → **37.6**

Finetuning's default effect is confidence: in ordinary text, loss and fork density fall. Around self-hints the pattern reverses selectively. Fork density rises inside the passage and roughly doubles immediately after it, whereas the passage itself becomes highly predictable (about 0.06 nats/token). Tutor-written hint text remains much less probable than the student's own self-hints. The pattern is therefore consistent with a stereotyped, student-generated cue that marks an unresolved branch, followed by increased exploration, rather than memorization of tutor wording. This analysis does not identify self-hinting as the cause of the score gain. Pooled self-hinting attempts score half a point higher, but within-problem comparisons remove most of that advantage ($+0.16$ for SPLICE, 95% CI $+0.01$ to $+0.34$; $+0.12$ and $+0.08$ for SPLICE-NF and SPLICE-HC), indicating a composition effect: self-hints occur most often on problems near the student's capability boundary. Detailed comparisons are in Appendix E.

5 Limitations and open problems

The workflow's limits are set by what it does not know. The 2,500 problems were chosen without regard to where the demonstrator's ability ends: some need no help, others stay unsolved at the highest hint level, and every tutor call spent on either is wasted or worse, since the second kind produces the excessive-hinting traces; predicting where a hint can change the outcome would remove that failure at the source. The tutor is stronger than both the student and the demonstrator, and we have not measured how much weaker it could be. Its reasoning traces are not exposed, so distilling from the tutor is not an available baseline; its final solutions carry no reasoning, and finetuning on them is ordinary solution-only training, not a competing recipe for reasoning traces.

The evidence has limits of its own. The entropy and self-hint analyses show that SPLICE training produces a branching behavior; they do not show that the behavior produces the score gain, and the test that would, finetuning with the hint tokens masked from the loss, we have not run. Our academic compute budget limited evaluation to AIME and 45 proof problems; the intervals in Section 4 reflect that, and a larger proof set is needed before differences at a single threshold can be read with confidence.

6 Conclusion

SPLICE moves verification and repair out of reinforcement learning and into offline data generation. By truncating failed continuations and inserting adaptively strengthened, restyled hints, it converts demonstrator failures into supervised traces unavailable to ordinary distillation or rejection sampling.

With student and demonstrator fixed, filtered SPLICE improves olympiad-proof grading and AIME accuracy. Filtering generation failures recovers the top of the proof-score distribution; removing excessive hinting improves termination. At test time, the student also emits self-hint-like branch cues at difficult points and explores more strongly afterward, showing that guidance in training traces can transfer a branching pattern while keeping the tutor out of inference.

AI USE STATEMENT

The SPLICE workflow, its design, the data selection and the experiments are the authors' own. Language models are the object of study: the demonstrator, the tutor and the grading judges are all language models, and the training data are generated by the workflow of Section 2, which is the subject of the paper. Beyond that, generative AI served as a writing and coding assistant: it edited and rephrased text, critiqued drafts as a reviewer would, and helped with analysis scripts and figures. All AI-assisted text and code were reviewed by the authors, all citations were verified by the authors, and the authors take full responsibility for the manuscript and artifacts.

REPRODUCIBILITY STATEMENT

Section 2 specifies the SPLICE loop and its stages, with the models behind each stage in Appendix B, the constants in Appendix C, the stage signatures in Appendix D, the prompt of every stage in Appendix H, and the input–output contract of each stage in Appendix I; Section 2.4 describes the problem sources. Section 4.1 defines the five datasets and the evaluation protocol, including the grading procedure and the treatment of unfinished attempts, with the judge validation in Appendix G and the finetuning recipe and evaluation settings in Appendix J. Code, generation records, and evaluation outputs will be released on acceptance.

REFERENCES

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback, 2022. URL https://arxiv.org/abs/2212.08073.

Gheorghe Comanici et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

Jhouben Cuesta-Ramirez, Samuel Beaussant, and Mehdi Mounsif. Large reasoning models are not thinking straight: on the unreliability of thinking trajectories, 2025. URL https://arxiv.org/abs/2507.00711.

DeepSeek-AI. Deepseekmath-v2: Towards self-verifiable mathematical reasoning, 2025. URL https://arxiv.org/abs/2511.22570.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2026. URL https://arxiv.org/abs/2501.12948.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment, 2023. URL https://arxiv.org/abs/2304.06767.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (rest) for language modeling, 2023. URL https://arxiv.org/abs/2308.08998.

Yiqiu Guo, Xueting Han, Qi Jia, Guangtao Zhai, and Jing Bai. Latentrevise: Learning from zero-hit reasoning, 2026. URL https://arxiv.org/abs/2606.29938.

Jubayer Ibn Hamid, Ifdita Hasan Orney, Michael Y. Li, Omar Shaikh, Yoonho Lee, Dorsa Sadigh, Chelsea Finn, and Noah Goodman. Spiral: Learning to search and aggregate, 2026. URL https://arxiv.org/abs/2606.23595.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14852–14882, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.830. URL https://aclanthology.org/2023.acl-long.830/.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.507. URL https://aclanthology.org/2023.findings-acl.507/.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet, 2024. URL https://arxiv.org/abs/2310.01798.

Yichen Huang and Lin F. Yang. Gemini 2.5 pro capable of winning gold at imo 2025, 2025. URL https://arxiv.org/abs/2507.15855.

Simran Kaur, Narutatsu Ri, Yinghui He, Liam Fowl, and Sanjeev Arora. Rethinking on-policy self-distillation for thinking models, 2026. URL https://arxiv.org/abs/2607.05184.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Ren Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26874–26901. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/lee24t.html.

LM-Provers, Yuxiao Qu, Amrith Setlur, Jasper Dekoninck, Edward Beeching, Jia Li, Ian Wu, Lewis Tunstall, and Aviral Kumar. Qed-nano: Teaching a tiny model to prove hard theorems, 2026. URL https://arxiv.org/abs/2604.04898.

Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Clara Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu H. Trinh, Quoc V. Le, and Junehyuk Jung. Towards robust mathematical reasoning. 2025. URL https://arxiv.org/abs/2511.01846. EMNLP 2025.

Wenjie Ma, Andrei Cojocaru, Neel Kolhe, Bradley Louie, Robin Said Sharif, Haihan Zhang, Vincent Zhuang, Matei Zaharia, and Sewon Min. Reliable fine-grained evaluation of natural language math proofs. 2025. URL https://arxiv.org/abs/2510.13888. ICLR 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1773–1781, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.151. URL https://aclanthology.org/2023.acl-short.151/.

Hamed Mahdavi, Alireza Hashemi, Majid Daliri, Pegah Mohammadipour, Alireza Farhadi, Samira Malek, Yekta Yazdanifard, Amir Khasahmadi, and Vasant G. Honavar. Brains vs. bytes: Evaluating llm proficiency in olympiad mathematics. In *arXiv preprint arXiv:2501.xxxxx*, 2025. URL https://openreview.net/forum?id=V4RIJxt02s.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Ivo Petrov, Jasper Dekoninck, Lyuben Baltadzhiev, Maria Drencheva, Kristian Minchev, Mislav Balunović, Nikola Jovanović, and Martin Vechev. Proof or bluff? evaluating llms on 2025 usa math olympiad. In *ICML 2025 Workshop on AI for Mathematical Reasoning (AI4MATH)*, 2025. URL https://openreview.net/forum?id=3v650rMO5U.

Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. Pope: Learning to reason on hard problems via privileged on-policy exploration, 2026. URL https://arxiv.org/abs/2601.18779.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, 2025. doi: 10.1145/3689031.3696075. URL https://arxiv.org/abs/2409.19256.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, et al. Beyond human data: Scaling self-training for problem-solving with language models, 2024. URL https://arxiv.org/abs/2312.06585.

Siao Tang, Xinyin Ma, Gongfan Fang, and Xinchao Wang. Concisehint: Boosting efficient reasoning via continuous concise hints during generation, 2025. URL https://arxiv.org/abs/2506.18810.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, et al. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL https://arxiv.org/abs/2501.12599.

Alexander Wei. openai-imo-2025-proofs. URL https://github.com/aw31/openai-imo-2025-proofs. Repository.

Tsung-Han Wu, Mihran Miroyan, David M. Chan, Trevor Darrell, Narges Norouzi, and Joseph E. Gonzalez. Are large reasoning models interruptible?, 2025. URL https://arxiv.org/abs/2510.11713.

Yu Xia, Canwen Xu, Zhewei Yao, Julian McAuley, and Yuxiong He. Learning to hint for reinforcement learning, 2026. URL https://arxiv.org/abs/2604.00698.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning, 2025. URL https://arxiv.org/abs/2502.03387.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57905–57923. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/yuan24d.html.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023. URL https://arxiv.org/abs/2308.01825.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning, 2022. URL https://arxiv.org/abs/2203.14465.

Feng Zhang, Zezhong Tan, Xinhong Ma, Ziqiang Dong, Xi Leng, Jianfei Zhao, Xin Sun, Yang Yang, and Guanjun Jiang. Adhint: Adaptive hints with difficulty priors for reinforcement learning, 2025a. URL https://arxiv.org/abs/2512.13095.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction, 2025b. URL https://arxiv.org/abs/2408.15240.

Wenhao Zhang. Beyond absolute imitation: Anchored residual guidance for privileged on-policy distillation, 2026. URL https://arxiv.org/abs/2606.10385.

Wenyuan Zhang, Shuaiyi Nie, Zhengyang Ai, Chengguang Tang, Xinghua Zhang, Yi Liu, Tingwen Liu, and Pinyan Lu. A survey on hint-based rlvr: Overcoming zero-advantage failures with external textual signals. *Preprints*, June 2026. doi: 10.20944/preprints202606.1050.v1. URL https://doi.org/10.20944/preprints202606.1050.v1.