Does CoT-Pass@k Really Check the CoT? A Multilingual Mathematical Audit

Tarık Tuna Taşaltı[1,2] **Burcu Hüdaverdi**[1] **David Semedo**[2]
[1]Dokuz Eylül University, İzmir, Türkiye
[2]NOVA School of Science and Technology, Universidade NOVA de Lisboa, Caparica, Portugal
tasaltitariktuna@gmail.com

Abstract

Pass@k measures whether a model reaches a correct answer under repeated sampling, but never how: a lucky guess counts the same as sound reasoning. CoT-Pass@k was proposed to close that gap, adding an LLM-as-judge that must assess a solution's reasoning chain before it counts. Its value rests entirely on one assumption: that the judge catches flawed reasoning. That assumption has never been tested inside the metric that depends on it, and never outside English, though the metric's claims concern models used in many languages. We report the first audit of that verification step, run under the metric's own protocol on a multilingual suite of five mathematical benchmarks in English, Turkish and Portuguese, two of them natively written. We corrupt correct solutions with deterministic edits that damage the chain and the final answer separately. We observe that all three judges accept corrupted chains almost as often as clean ones. V4-Flash and Qwen3.6 reject a solution sharply only when its final answer is wrong and accept a wrong answer more readily when the chain agrees with it; the metric's own judge accepts most wrong answers as well. Our study shows that chain–answer agreement dominates the two larger judges' verdicts and that all three fail to reliably detect the tested reasoning errors. Consequently the difference Pass@k − CoT-Pass@k averages 19.7 points on an earlier solver generation but only 4.1 on the current one. What little remains depends on the token budgets on both sides and on the generation mode; raising the generation budget moves Pass@64 by more than fifty points while the difference stays at zero. We close with two checks any judged reasoning metric should pass before its numbers are read as evidence about reasoning.

1 Introduction

Pass@k measures whether any of k sampled generations answers a question correctly. It has become the standard instrument for mapping what a model can reach under repeated sampling, and it anchors the current debate on whether reinforcement learning with verifiable rewards extends or merely sharpens a base model's reasoning. But Pass@k never looks at how the answer was reached: a lucky guess and sound reasoning count the same.

CoT-Pass@k (Wen et al., 2026) was proposed to close exactly this hole. An LLM judge assesses each correct solution and must approve the reasoning chain before the solution counts, so the metric promises to separate models that reason from models that guess. The promise rests entirely on one assumption: the judge actually assesses the chain. That assumption is testable; the literature on reasoning-error detection gives ample reason to doubt it: verifiers are weak at exactly the step-level judgments the metric needs (Jacovi et al., 2024; Zheng et al., 2025), step-level error identification is generally harder than solution-level for every method tested (Xia et al., 2025), weaker still on long chains (He et al., 2025), and prone to crediting reasoning that merely looks valid (Wang et al., 2025b).

In this paper, we audit CoT-Pass@k under its own protocol, comprising the original judge prompt (Figure 3), three judgments per solution and the original verification strategies, with controlled error injection: deterministic edits that corrupt a solution's reasoning chain and its final answer separately (Section 4.2). The audit runs on five mathematical benchmarks in three languages, including native Turkish and Portuguese sets (Section 3): the metric is applied to models used far beyond English, and its judge has only ever been measured in it.

The audit returns a clear mechanism, and the mechanism predicts the metric's behaviour. Acceptance barely moves when we corrupt the chain under any judge. Under V4-Flash and Qwen3.6 it collapses when the final answer is wrong and rises again when that wrong answer is carried con-

Proceedings of the 6th Workshop on Multilingual Representation Learning (MRL 2026), pages 422–444
October 28, 2026 ©2026 Association for Computational Linguistics

sistently throughout the chain, so chain–answer agreement dominates the verdicts and the judges fail to reliably detect the tested errors in the steps; the metric's own judge passes most wrong answers as well (Section 5.1). It follows that wherever answers are mostly right, CoT-Pass@k must collapse onto Pass@k, and it does: across two solver generations, the average difference falls from 19.7 points to 4.1 (Section 5.2). The original study's solver was a Qwen2.5 base checkpoint, on which the metric does separate; on the solvers it would be applied to now, what it adds over Pass@k is close to nothing. Switching the generation mode reverses which language the same solvers look stronger in, and raising the token budget moves Pass@64 by more than fifty points while the difference Pass@64 − CoT-Pass@64 stays at zero at every step (Section 5.3).

Our contributions: (i) the first audit of CoT-Pass@k's verification step, under the metric's own protocol, where other error injection tests the judge but not a metric built on it (Sun et al., 2026; Mittal and Arike, 2026); (ii) a controlled, deterministic error-injection design whose conditions separate chain validity from answer correctness, including a consistency-preserving wrong-answer condition; (iii) a multilingual benchmark suite built for this audit[1]: translated AIME 2026 (MAA, 2026) and native sets from the 2026 Turkish olympiad (TÜBİTAK, 2026) and the Portuguese exams of PHEB (Tavares et al., 2026). These carry, to our knowledge, the first CoT-Pass@k results on natively written Turkish and Portuguese benchmarks; (iv) a two-generation comparison of the metric's collapse onto Pass@k, with question-level uncertainty quantified; (v) an analysis showing that what the metric reports depends on the token budgets, the generation mode and judgments stopped at the max-token limit; and (vi) concrete recommendations for reporting and auditing judged reasoning metrics.

2 Related Work

What Pass@k and CoT-Pass@k measure. Pass@k, adopted from program synthesis (Chen et al., 2021), is the evidence base for the debate on whether reinforcement learning with verifiable rewards extends a base model's reasoning or only sharpens its sampling (Yue et al., 2025). Pass@k is a coverage measure that rises with the sample budget (Brown et al., 2024), which is why its critics add a reliability threshold (Liu et al., 2025; Dragoi et al., 2025) or replace it with a posterior success estimate (Hariri et al., 2026); none of these three metrics inspects the chain. CoT-Pass@k (Wen et al., 2026) enters that debate as an instrument: an LLM judge must approve the reasoning chain before a solution counts, so the metric should separate reasoning from guessing. The original study supports its judge by agreement with larger open-weights judges; Section 5.1 shows why agreement is not verification: two judges that both track the final answer agree for the wrong reason.

Turkish and Portuguese benchmarks. Turkish has TurkBench (Toraman et al., 2026) and Cetvel (Er et al., 2026) and Portuguese has MATH-PT (Teixeira et al., 2026) and PHEB (Tavares et al., 2026), but none measures Pass@k, and only PHEB grades written solutions, against the official rubrics of open-ended questions our suite does not use; TurkBench draws on TÜBİTAK olympiads but predates the 2026 exam. MCLM (Son et al., 2025) translates AIME 2024 into 55 languages with GPT-4o, Turkish among them, checks only that the answers and equations survive translation, and scores the final answer alone.

Judge-free evaluation. Where gold step labels exist, reasoning can be scored without a judge (Uesato et al., 2022; Lightman et al., 2024), and ProcessBench (Zheng et al., 2025) and PRMBench (Song et al., 2025) score verifiers against known step errors and find existing process reward models weak at identifying the faulty step. These benchmarks measure judges in isolation; the verdict never leaves the benchmark, so nothing connects a judge's failure there to the numbers a deployed metric reports. Sobhani et al. (2026) extends error injection to multilingual mathematics, Turkish among its languages, though its injected chains are Bangla and English only; Zhao et al. (2026) corrupt chains across languages to measure the solver's reliance on its own trace. In concurrent work, Garcia (2026) shows that the effect in such studies tracks where the answer is stated, not where the computation happens. In all three, a corrupted chain scores a model against a known key, the gold answer or, in MathMist, the injected error; in none does a verdict on a chain become the number a published metric reports.

[1]Code and the benchmark suite: https://github.com/ttasalti/evalhub.

Judge-dependent evaluation. Where the judge itself is the instrument, its failure modes are documented: preference biases, degradation on long chains, credit for reasoning that only looks valid (Zheng et al., 2023; He et al., 2025; Wang et al., 2025b), weaker judges accept wrong answers more readily once a fluent chain is attached (Tu et al., 2026), and a chain need not state the true reason for its answer at all (Turpin et al., 2023). Two recent studies come closest to ours in method. In concurrent work, Sun et al. (2026) build valid-answer, invalid-reasoning solutions and find frontier judges credit up to half of them as flawless, the verdicts tracking the answer. Mittal and Arike (2026) inject errors into PRM800K chains and find judges detect that a chain is wrong far more reliably than where. Both are English-only and stop at the judge. Our preliminary study (Taşaltı et al., 2026) benchmarks both metrics on the three AIME 2026 versions and raises, without testing, the question this audit starts from, whether the judge catches subtle reasoning errors. Neither study, nor any other we know of, audits a published judged metric under the metric's own protocol, follows the judge's verdicts into the number the metric reports, or ties what remains to the token budgets and the generation mode: the three steps this audit takes, in three languages.

3 Multilingual Benchmarks

The suite holds five benchmarks in three typologically distant languages: Germanic English, Romance Portuguese, and Turkic Turkish, the last agglutinative and outside the Indo-European family altogether. The English core is the AIME 2026 competition set (MAA, 2026): thirty problems with integer answers, on which none of our solvers reaches the ceiling. Two translated benchmarks carry the same thirty problems into Portuguese and Turkish: the problems were first machine-translated, then manually audited by native speakers of each language, six for Portuguese and two for Turkish, one of them an author. Two native benchmarks complete the suite: the thirty-two first-stage problems of the 2026 Turkish TÜBİTAK mathematics olympiad (TÜBİTAK, 2026), and 166 problems we draw from PHEB (Tavares et al., 2026), a multi-subject benchmark of Portuguese national exams, taking its multiple-choice mathematics questions with ground-truth answers and converting them to open-ended form; two further graduate students, native speakers of Portuguese and independent of the translation audit, checked that each question kept has a single ground-truth answer and can be posed without its options. All Portuguese material in the suite is European Portuguese (the variety far less represented in training corpora than Brazilian Portuguese): the original exams, the translation by construction and native audit. The pairing is deliberate: each non-English language gets one translated benchmark, which isolates the language while holding the problems fixed, and one native benchmark, which removes any translation artifact.

The suite spans a wide difficulty range: the AIME sets and the Turkish olympiad are competition mathematics, the Portuguese exams are high-school level and nearly saturated by current models. Contamination risk is limited for the competition material. The Turkish olympiad, held on 16 May 2026, postdates the public release of every model we use. AIME 2026, held in February, postdates the release of Qwen2.5 and R1-distill and the stated January 2025 training cutoff of Gemma 4; Qwen3.5, Qwen3.6 and V4-Flash publish no training cutoff and appeared only three to eleven weeks after the first exam, little time for its problems to reach their training data. The Portuguese exams, written between 2006 and 2023, predate every model. AIME problems are mirrored in public corpora within days of release, while we found neither the 2026 Turkish olympiad nor the Portuguese exam questions of PHEB on Hugging Face, and the olympiad nowhere but in the organiser's PDF, the least exposed benchmark of the five.

4 Method

4.1 Solvers, Judges and Metrics

Solvers. We evaluate two model generations and a second current-generation family. The earlier generation is Qwen2.5-7B and Qwen2.5-32B (Qwen Team, 2024), run as base models and, through their Instruct counterparts, in non-thinking mode. The current generation is Qwen3.5-4B and Qwen3.5-9B (Qwen Team, 2026a) in base, non-thinking and thinking configurations, joined by Gemma-4-E2B-it and Gemma-4-E4B-it (Gemma Team, 2026) in non-thinking and thinking mode. Base checkpoints run under their template's default thinking setting, and every solver draws 64 generations per question (16 on the Portuguese exams) at temperature 0.6 and top-p 0.95, with a 16,384-token generation budget unless a section states otherwise (the error-injection study raises it per benchmark, Sec-

tion 4.2). Serving templates and the sample-count choice are given in Appendix A.2.

Judges. Two open-weights judges cover the paper: Qwen3.6-35B-A3B (Qwen Team, 2026b) and Gemma-4-26B-A4B-it, written Qwen3.6 and Gemma4; the error-injection study and the cross-generation replication add DeepSeek V4-Flash (DeepSeek-AI, 2026), a far larger open-weights model from a third family that we run through a commercial API. A fourth judge corroborates the earlier generation and scores the error-injection study as well, DeepSeek-R1-0528-Qwen3-8B (DeepSeek-AI, 2025; Qwen Team, 2025), written R1-distill: it is the verifier of Wen et al. (2026) and the reason our earlier generation is the Qwen2.5 family, so one arm of our comparison reruns the metric on its own solver and its own judge rather than on a reconstruction of them.

Judging protocol. All judges run in thinking mode at the same temperature 0.6 and top-p 0.95, with a 16,384-token judgment budget, and judge every correct solution three times; Appendix A.2 gives the two budget exceptions. A judgment that reaches its own max-token limit without emitting a verdict counts as neither an approval nor a rejection; the majority rule, used throughout the paper, compares approvals against rejections. Solvers are prompted in the language of the benchmark; their chains, like those of the reasoning models studied by Wang et al. (2025a) and Yong et al. (2025), are free to switch into English mid-solution, and on the non-English benchmarks the Qwen3.5 base checkpoints and every thinking-mode solver write nearly every chain in English or in a mixture, the instruction-tuned solvers in non-thinking mode almost none (Figures 11 and 13 in Appendices C.5 and D.4). Verification is therefore standardised: every solution meets the same instrument, the original English verification prompt of Wen et al. (2026), extended by two lines (Appendix A.1) that name the problem's language and direct the judge to score the mathematics regardless of the chain's language, keeping the judging directly comparable to the original study. The other two verification strategies of the original metric are defined and ablated in Appendix B.

Metrics. Pass@k estimates the probability that at least one of k sampled generations reaches the correct final answer. We compute it with the standard unbiased estimator $1 - \binom{n-c}{k} / \binom{n}{k}$, averaged over questions, where c of the n generations of a question are correct (Chen et al., 2021). CoT-Pass@k (Wen et al., 2026) counts a generation as a success only if its final answer is correct *and* the judge approves its reasoning chain: the same estimator with c replaced by the number of generations that are both correct and approved. We write the difference between them out as Pass@k − CoT-Pass@k; at $k = n$ it is the share of questions with a correct solution but no approved one.

4.2 Error Injection

We test whether the judge actually assesses chains with *error injection*: we edit solutions so that the reasoning chain and the final answer are corrupted separately, and we measure what the judge detects.

We use two current-generation solvers, Qwen3.5-4B and Qwen3.5-9B, in thinking mode, with a generation budget of 65,536 tokens for English AIME, 32,768 for the AIME translations and the Turkish olympiad, and 16,384 for the Portuguese exams. The original solutions are correctly answered generations to which all four error types can be applied (Appendix A.2). From every model×benchmark cell we randomly select 72 such solutions, giving 720 original solutions. Each is expanded into five variants, the unedited *clean* control and one copy per error type, for a total of $720 \times 5 = 3{,}600$ variants. The **intermediate numeric error** replaces one number drawn at random from between 40% and 70% of the solution's length, never one that appears in the question or the answer, in the middle of the solution, where Mittal and Arike (2026) confine their injections so that a judge cannot find them by inspecting the first or last step, and leaves the final answer correct. The **truncation error** deletes the last 25% of the reasoning chain and re-appends the boxed final answer on a new line, leaving a visible seam. With these two error types the reasoning chain is corrupted but the final answer stays correct. The **final-answer error** replaces only the final answer and leaves the reasoning chain untouched. The **consistent final-answer error** replaces every occurrence of the correct final-answer value in the reasoning chain with the same wrong number, so the chain stays consistent and supports the wrong final answer. All edits are deterministic string edits (no language model writes the corruptions), and every wrong number is a near miss of the value it replaces: shifted by one or two, two digits transposed, or a single digit changed. The chain edits adapt the early-answering and adding-mistakes perturbations

of Lanham et al. (2023), which truncate a chain or insert one mistaken step into it, to a judge-side test with deterministic edits and no regeneration, and the numeric edits adapt the rule-based numeric perturbations of Singh et al. (2025), tightened here to near misses. The design gives each error type one job. Models reading a chain are known to follow its stated final answer, most strongly at small scale (Garcia, 2026), and judges to confirm the answer rather than check the steps (Sun et al., 2026); a rejection under the first two error types can therefore only come from assessing the chain, and extra acceptance under the consistent error can only come from rewarding that consistency.

Three thinking-mode judges score every variant, V4-Flash, Qwen3.6 and R1-distill, under the three-judgment majority rule of Section 4.1. All comparisons are paired: each judge is evaluated only on original solutions it judged in all five variants, the same 720 for all three judges. Acceptance rates carry 95% Wilson intervals (Wilson, 1927); conditions are compared pairwise with an exact McNemar test (McNemar, 1947) on the same solutions, and the paired difference between two conditions is bounded by a bootstrap over those solutions.

4.3 Cross-Generation Design

To measure whether the difference Pass@k − CoT-Pass@k still opens in current-generation models, we compare two solver generations under one fixed setting: the anchor judge Qwen3.6 with a 16k generation budget, on the four 64-sample benchmarks, with every curve run to $k = 64$, under the majority rule. Throughout the paper *generation* refers to the solvers being measured, never to the judges, which are held fixed across both. Both generations enter as base checkpoints and, through their instruction-tuned counterparts, in non-thinking mode (Section 4.1). The two arms hold different things fixed: in the non-thinking arm the mode is matched exactly, with thinking explicitly disabled on both sides, while in the base arm each family's base checkpoint runs under its template's default thinking setting. The Gemma-4 solvers enter in non-thinking mode only, and Gemma4 joins the anchor judge on this grid; the two further judges of Appendix C cover the earlier generation.

A model enters the comparison only if both judges returned verdicts for at least 30 of its correct solutions in every cell: both modes, all four benchmarks; a model that misses the threshold anywhere is dropped entirely, since a partial grid would open the comparison to selection effects (Appendix C.1 lists the models this removes). The Portuguese exams are excluded (they have no non-thinking runs, and even in the earlier generation the difference there stays between 0 and 3.6 points, at that benchmark's k=16), and thinking mode is excluded because Qwen2.5 has no thinking counterpart; thinking-mode solvers are studied in Section 5.1. Intervals on the difference resample questions within a cell.

4.4 Token-Budget Measures

Both the solver and the judge can run out of tokens, and in both cases the consequence is scored as a failure that the model may not have committed.

On the solver side, a generation still writing at the max-token limit is cut off before it states a boxed final answer and is graded wrong; thinking mode makes this common, because the chain consumes the same budget the answer has to fit in.

On the judge side the same thing happens one level up. A judgment that spends its whole budget thinking emits no verdict, and under the majority rule it cannot contribute an approval: a solution can lose its majority, and be counted against CoT-Pass@k, without any judge having rejected it. We count these stops per judgment and isolate their effect with a counterfactual: recomputing CoT-Pass@k with stopped judgments counted as approvals. The band between the two curves is an upper bound on how much of the difference Pass@k − CoT-Pass@k the judge's token budget, rather than its verdicts, produced, since some of the stopped judgments would have ended in rejection had they finished.

We also run a budget ladder that raises the generation budget itself (Section 5.3).

5 Results

5.1 The Judges Miss the Injected Reasoning Errors

Figure 1 shows acceptance for the five variants under majority-correct. All three judges accept the unedited clean control at high rates, 93% for V4-Flash, 98% for Qwen3.6 and 90% for R1-distill.

The two error types that keep the final answer correct barely move them. Under the intermediate numeric error all three judges stay at their clean rates, 93%, 98% and 92%, although every edited chain now carries a number that the steps around it

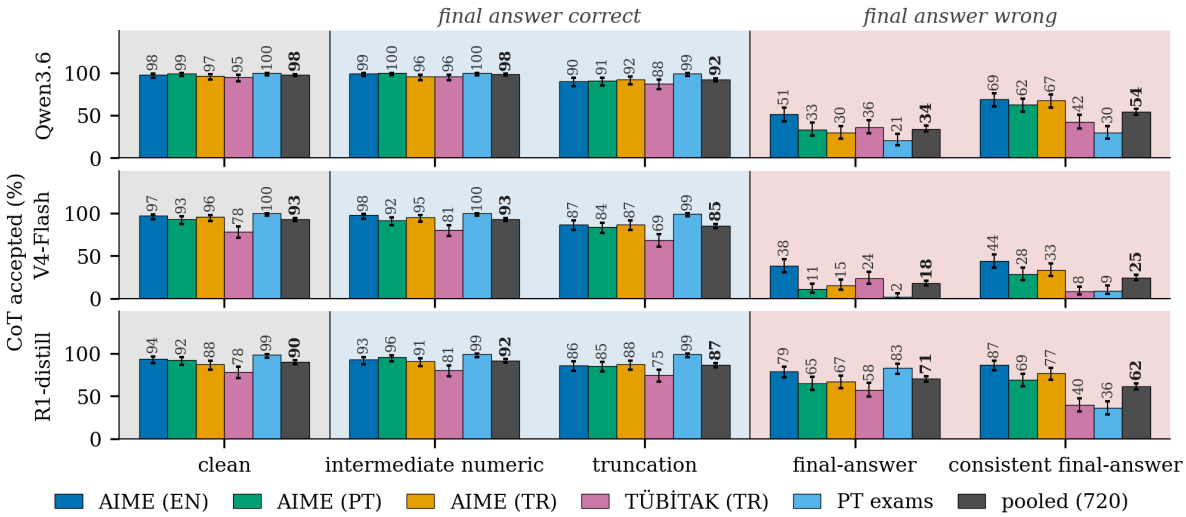

Figure 1: CoT acceptance under injected errors per benchmark and judge (rows), majority-correct strategy of Section 4.1, 95% Wilson intervals; the dark grey bar is the rate pooled over the five benchmarks. Every judge assesses the same 720 original solutions, 144 per benchmark and condition. R1-distill is the judge the metric specifies.

no longer support: a paired McNemar test over the same 720 solutions finds no evidence of a shift from the clean control ($p = 1$, $p = 0.55$ and $p = 0.23$), and the paired difference is bounded within 1.5 points in either direction for V4-Flash and Qwen3.6 and within 5 for R1-distill (95% bootstrap intervals). The truncation error costs 8, 6 and 3 points.

Corrupting the final answer gives the opposite picture. The final-answer error leaves the chain untouched and replaces only the number at the end, and acceptance collapses at once, to 18% for V4-Flash and 34% for Qwen3.6, and only to 71% for R1-distill. V4-Flash and Qwen3.6 can reject, sharply and in agreement with each other, but they did so only when the final answer was wrong. The metric's own judge barely rejects even there, leaving 71% of the wrong answers accepted. Changing a number inside the chain costs no judge a measurable share of its acceptance; changing only the number at the end costs 75 and 64 points for V4-Flash and Qwen3.6 and 20 for R1-distill.

The consistent final-answer error shows why. It corrupts strictly more of the reasoning than the final-answer error, since the wrong value now runs throughout the chain, yet acceptance rises instead of falling: to 25% for V4-Flash and to 54% for Qwen3.6, which accepts the majority of these variants. A judge that verified the mathematics would order the two the other way round. Both conditions are built from the same 720 solutions, so the ordering can be tested pairwise: McNemar's test rejects equality for V4-Flash and Qwen3.6 ($p = 2.4 \times 10^{-3}$ and $p = 2.6 \times 10^{-14}$), with the discordant pairs running the same way for each. R1-distill does not show the rise, accepting 62% of the consistent variants against 71% of the plain ones ($p = 1.7 \times 10^{-4}$; Appendix B). **None of the three judges reliably detects the tested errors in the steps, and for V4-Flash and Qwen3.6 chain–answer agreement dominates the verdicts.**

The verification strategy changes the rates, not the findings (Table 1 in Appendix B), and the size of the edit matters only for the consistent wrong answer; the intermediate error is missed at every size (Figure 4).

The pattern repeats on every benchmark (Figure 1; Appendix B gives the rates per benchmark). Under every judge the intermediate numeric error stays within four points of the clean control on all five, and the translated AIME sets behave like the English original. A wrong final answer costs V4-Flash and Qwen3.6 47 to 98 points everywhere, most on the short Portuguese exams, and R1-distill 15 to 27, and the Turkish olympiad is where the judges doubt correct chains too.

5.2 CoT-Pass@k Collapses onto Pass@k in Current Solvers

Section 5.1 predicts where CoT-Pass@k should collapse onto Pass@k: the judge mostly approves a chain that agrees with its final answer, so the

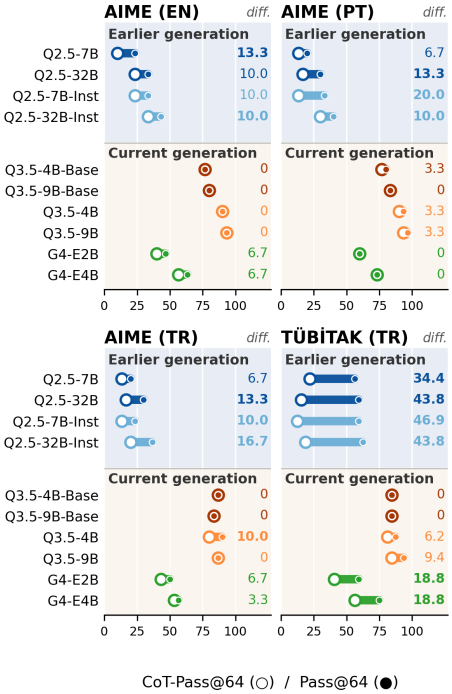

Figure 2: CoT-Pass@64 (open) and Pass@64 (filled) per solver and 64-sample benchmark under the anchor judge Qwen3.6 at 16k; row-end numbers give the difference. Q2.5/Q3.5 = Qwen2.5/Qwen3.5, G4 = Gemma-4-it.

difference Pass@k − CoT-Pass@k can come only from correct answers carrying chains the judge rejects, and should vanish where those are rare. Figure 2 confirms the prediction. Among the earlier-generation solvers (Qwen2.5), the difference Pass@64 − CoT-Pass@64 is large for every model on every benchmark: 6.7–20 points on the AIME benchmarks and 34–47 points on the Turkish olympiad, for both base and non-thinking models.

Among the current-generation solvers (Qwen3.5 and Gemma-4) the same judge and budget find almost nothing. Qwen3.5-9B-Base ends at exactly zero on all four benchmarks, Qwen3.5-4B-Base on three of the four, and no Qwen3.5 model exceeds 10 points anywhere in it; the Gemma-4 models stay at or below 6.7 on the AIME benchmarks and reach 18.8 only on the Turkish olympiad, still well below the smallest earlier-generation value there, 34.4. Averaged over the four benchmarks, the difference is 19.7 points (95% CI $[16.4, 23.0]$) in the earlier generation against 4.1 ($[2.7, 5.5]$) in the current one (per solver, 15.6–22.1 against at most 8.2), and the contrast itself is 15.6 ($[12.0, 19.3]$). The intervals are bootstrap intervals over questions, the sampled unit in the framework of Miller (2024); Figure 7 in Appendix C.2 gives one for every solver. **On current-generation solvers, CoT-Pass@k stays within a few points of Pass@k, and the judge changes almost nothing of what Pass@k already reports.**

Nor does this depend on the anchor judge. Four different judge models, among them the verifier of the original study, reproduce the earlier generation's average difference within two points of each other; Gemma4, the only other judge that covers both generations in full, reproduces the split between them (Tables 2 and 3 in Appendix C.1, per judge and per model). Under both judges that span the generations, the collapse is a property of the metric, not of the judge we chose to anchor it on.

The divergence is not present from the start: at $k = 1$ the generations are indistinguishable, 2.7 points against 2.0, and Figure 6 in Appendix C.2 follows them apart to 19.7 against 4.1. What separates them is what repeated sampling adds: in the earlier generation each new correct answer had a growing chance of carrying a chain the judge would reject; in the current one it does not.

Per generation the verdicts do change. Averaged over the four benchmarks, the anchor judge approves 61.6% of the earlier generation's answer-correct generations and 94.8% of the current generation's, and Gemma4 62.7% and 93.2% (Figure 8 in Appendix C.3). Two things therefore shrink the difference. The judges reject far fewer of the current generation's correct chains, and the rejections that remain rarely reach CoT-Pass@64, because a question keeps its CoT-Pass@64 as long as one of its c correct generations is approved, and the current generation's solved questions typically hold dozens (Appendix C.4). Comparing the generations at the same c leaves the gap in place (Figure 9).

The collapse holds on all four benchmarks, across three languages, and at every k we measure, with the earlier-minus-current contrast carrying a 95% interval that excludes zero on each benchmark separately. Three readings short of a generational change do not survive the data. Higher accuracy alone does not produce it. At an identical Pass@64 of 59.4 on the Turkish olympiad, the two earlier-generation solvers lose 43.8 and 46.9 points where the current one loses 18.8 (Figure 2), and the acceptance rates above separate the generations at every level of accuracy (Figure 9). It is not model size: scaling Qwen2.5 from 7B to 32B leaves the difference as large, 34.4 against 43.8, while at the closest matched scale, Qwen2.5-7B

against the larger Qwen3.5-9B, the difference falls from 15.6 to zero in the base arm and from 22.1 to 3.3 in the non-thinking one. And it is not the mode mix: the mode-matched arm alone, Qwen2.5-Instruct against Qwen3.5 non-thinking, shows the collapse from 10–47 points down to at most 10. The comparison remains observational, between model families rather than a controlled intervention, and isolates none of what produces the change.

5.3 Raising the Budget Moves the Scores, Not the Difference

The residual that Section 5.2 leaves behind is small under the anchor judge. Both sides of the metric spend token budget, the judge to take in the chain and still reason, the solver to finish the chain and still answer, and both move the numbers that get reported (Table 4 in Appendix D.1, at the shared 16k budget).

The judge's side first. The two judges stop at their own max-token limit at very different rates. Judging the Qwen3.5 solutions on the four competition benchmarks, Gemma4 stops on 33 to 54% of its judgments; judging the Gemma-4 solutions on the same benchmarks, on 2 to 14%. Qwen3.6 stays at or below 18% throughout, and most of its stops come after a verdict has already been written, while Gemma4's stops rarely contain one. A stopped judgment cannot contribute an approval, so it can only widen the difference (Section 4.4). And the two judges do report different differences on the same solutions: averaged over these four benchmarks at $k = 64$, 5.7 points under Gemma4 against 1.4 under the anchor. The counterfactual of Section 4.4 measures how much of that is budget: counting Gemma4's stopped judgments as approvals removes 92.7% of its 5.7, an upper bound on the budget's share, against 47.9% in non-thinking mode, where the stops are rarer and the two judges nearly agree to begin with, 5.9 against 5.8 (Figure 12 in Appendix D.2). Under Gemma4, the judge's token budget can account for most of what the metric reports in thinking mode; under Qwen3.6, on the very same solutions, there is almost nothing left to report. The collapse itself shows under either judge (Section 5.2); what remains on top of it can be an artifact of the judge one happens to pick.

The solver's side mirrors it one level down, and the cleanest view changes nothing but the generation mode. On the same thirty problems, enabling thinking raises the share of English generations that stop at the max-token limit from 27.9% to 86.7% (the chains grow, the budget does not) while the Turkish rate rises only to 39.9% (Table 6 in Appendix D.4). Pass@64 inverts with it, English against Turkish: 91.7 to 88.3 in non-thinking mode, 40.0 to 85.0 with thinking on. Turning on the chain of thought that CoT-Pass@k exists to check reverses which of the two languages these models look stronger in. We therefore read language differences throughout as effects of the generation mode and the token budget, not statements about ability in a language; the mode even sets the language the chain is written in (Figure 13 in Appendix D.4).

The benchmarks line up the same way: the Portuguese exams, the easiest of the five and the least bound by the max-token limit on either side (Table 4), carry the smallest difference, 0.60 points at most, at that benchmark's k=16. Where solutions fit the budgets, there is nothing left for the metric to report. The Turkish olympiad marks the one residual the budget reading does not carry: the largest difference the anchor judge reports here, 9.38 points under the Gemma-4-E4B solver, sits where essentially nothing stops at the max-token limit, not one of that solver's generations, and 0.1% of the anchor judge's. Whatever it tracks, it is not the budget; we return to it in Section 6.

So far the budget has been held fixed and everything else varied; moving it turns the reading into an intervention. Table 5 in Appendix D.3 compares the two solvers of Section 4.2 at a low and a high generation budget: the same problems, more room to write. The high budget is 32k, except 64k on English AIME, whose chains still ran into the max-token limit at 32k.

The solver's side of Table 5 responds exactly as Table 4 predicts: the share of generations stopping at the max-token limit collapses, from 84–90% to 9–18% on English AIME, and accuracy climbs with it, Pass@1 from 10.5 and 16.2 to 75.4 and 84.5 there, Pass@64 from 40.0 to 96.7 there and by 3 to 17 points on the other benchmarks. What CoT-Pass@64 adds on top of Pass@64 never moves at all. The ladder's solutions are judged as everywhere else, by the anchor judge at a budget that rises with the solver's, and the difference Pass@64 − CoT-Pass@64 is exactly zero in all sixteen benchmark × solver × budget cells. The anchor judge still rejects some solutions, but no question ever loses all of its correct ones. **CoT-Pass@64 follows Pass@64 point for point up the ladder: whatever the budget produces, the**

metric certifies. The ladder is reported under the anchor judge alone (Appendix D.3).

6 Discussion and Conclusion

What the metric measures. A judgment that approves a corrupted chain, rejects it once its final answer is wrong, and approves it again once that wrong value runs throughout the chain is a judgment in which agreement between the chain and the final answer dominates and the tested errors in the mathematics count for little. The metric's own judge fits only the first clause, since it approves most wrong answers too. That agreement is nearly free for a solver whose answers are usually right, which is why, on current-generation solvers, the metric returns what Pass@k already returns. What it reports is thus a property of the solver–judge pair: the same judge filters the two generations differently, and the same solvers score differently under a judge that stops at the max-token limit.

Two checks. The question we put to CoT-Pass@k is the one work on construct validity puts to any instrument, whether it captures the construct it names (Bean et al., 2025), and our results turn it into two checks that a judged reasoning metric should pass before its numbers are read as evidence about reasoning. **(i) Publish how the judge responds to injected errors.** Give it solutions whose chains are corrupted while their final answers stay correct; if approval does not move, nothing the metric reports separates a sound chain from a corrupted one. The check is run once, by whoever proposes the judge, on several hundred solutions; in our runs the whole error-injection panel took about a sixth of the judge tokens the anchor judge spent on the solutions of Section 5.2 (Appendix A.2). **(ii) Report the metric beside the judge-free metric it wraps, under more than one token budget and generation mode.** The difference between the two is all the judge adds, and publishing it per solver and per benchmark, at more than one token budget, costs a column.

Where the metric still separates. None of this makes chain-level verification a bad idea; it makes an unaudited judge a bad instrument. The metric does separate the earlier generation, and the original study itself observes that the separation it reports shrinks on benchmarks the base model already solves (Wen et al., 2026), but harder problems do not restore its meaning, since the judges do not react to corrupted chains there either. What the metric cannot carry is the debate it entered: on current-generation solvers a conclusion drawn from it inherits the Pass@k evidence it was meant to go beyond, and whether verifiable rewards extend reasoning has to be settled with instruments that measure the chain directly, such as the causal-importance and sufficiency metrics of Yu et al. (2026), who find that verifiable rewards raise accuracy without reliably making the chain causally important or sufficient.

What the languages add. The verifier failure is not English-specific in the tested settings. On the same thirty AIME problems in three languages the judges miss the injected errors, reject wrong answers and approve clean chains at rates at most 6.3 points below the English original's, and sometimes above them (Section 5.1), and averaged over solvers the translations are approved within 3 points of the original or above it (Appendix C.3), so we find no sign that Turkish or Portuguese makes verification harder for these judges. What moves the rates is the native benchmarks, the Turkish olympiad on the correct chains and the Portuguese exams on the wrong answers, and the olympiad also carries the one difference the budget reading does not explain (Section 5.3), for which the Limitations name the candidates. The language a chain is written in follows the solver rather than the benchmark, and its effect on the verdicts is small and inconsistent: within 3 points for the Qwen3.5 base checkpoints in every cell, 13 points in favour of English for one earlier-generation solver on the olympiad, and in favour of mixed chains under V4-Flash and R1-distill there (Figures 10 and 5, Appendix C.5).

Conclusion. An instrument that promises to assess reasoning has to be audited the way any instrument is, against inputs whose correct verdict is known by construction, as BLEU once was against constructed variants that human judges would rank far lower (Callison-Burch et al., 2006). Under that audit CoT-Pass@k's verification step fails to reliably detect the tested errors and, under V4-Flash and Qwen3.6, its verdicts are dominated by chain–answer agreement, what the metric adds over Pass@k has collapsed on current-generation solvers, and what remains of it depends largely on the token budgets on both sides and on judgments stopped at the max-token limit.

Limitations

The injection is deterministic, and deliberately so. String edits keep the correct verdict known by construction: no second model has to be trusted to corrupt a chain, and every variant is reproducible. The cost is realism, and a lenient reader might treat an edited step as a slip rather than broken reasoning. That reading cannot explain the consistent final-answer error, which corrupts strictly more of the chain than the plain one yet is accepted more often by V4-Flash and Qwen3.6; and where our edits are too mild, the bias understates rather than overstates how little the judges assess.

What the edits cover. Truncation leaves a visible seam, so the 3 to 8 points it costs are an upper bound on what a chain that stops short costs these judges. The injected solutions come from two Qwen3.5 thinking-mode solvers, so the judges are audited on one family's style of chain, and on the non-English benchmarks these chains are written in English or in a mixture rather than in the benchmark's language (Appendix C.5), so outside English the judges are audited on non-English problems with English or mixed reasoning. We did not check by hand whether each edited number carries into the steps that follow; an edit in an auxiliary line corrupts less of the mathematics than a broken derivation, and the consistent final-answer condition, whose corruption needs no such check, carries the ordering result.

Scale is bounded on both sides, solver and judge. Our current-generation solvers reach 9B parameters, so the collapse is not directly verified on larger current models; within the earlier generation, where we do scale, size does not produce it, since 7B to 32B leaves the difference as large. On the judge side, the audit rests on three judges larger in total parameters than the one the metric specifies (Qwen3.6, Gemma4 and DeepSeek V4-Flash) and on that original 8B judge itself (Wen et al., 2026), which in our error-injection study misses the chain-level errors as they do but rejects a wrong final answer far less often; it does not reach the flagship tier of any lab. The judges differ in how much they accept, and the weakest one also differs in what it accepts. None of them reacts to the intermediate numeric error. V4-Flash and Qwen3.6 accept the consistent wrong answer more often than the plain one, R1-distill does not, and it accepts 71% of the plain wrong answers outright. We read the original judge's behaviour off its acceptance rates and did not audit the reasoning behind its verdicts, so whether it accepts on the form of a solution rather than on its content remains open.

The residual is observed, not explained. The collapse is least complete on the Turkish olympiad: against earlier-generation differences of 34–47 points there, the current-generation solvers fall far under the anchor judge, the Qwen3.5 solvers to at most 9.4 points and the Gemma-4 solvers to 18.8 (Figure 2), but not to zero. That benchmark is at once natively written, the least publicly exposed of the five, and at olympiad level, and our suite cannot separate these: the Portuguese exams are natively written too, but they are high-school level and nearly saturated, so they cannot isolate the effect of native language. The multilingual reach is three languages in one domain, so the findings are not English-specific in the tested settings, and we claim nothing beyond those settings. The audit is also mathematical throughout, and whether the same behaviour holds in code or in domains with no single verifiable answer, our data cannot say.

Ethics Statement

This work evaluates publicly released language models on published mathematical exams, used for research evaluation only; no personal data and no human subjects are involved. The translations were audited by an author and volunteer native speakers, and the corrupted solutions exist only as test inputs to the judges. The audit concerns a published metric, not its authors: we run it under its own prompt and protocol and report where it still separates models. Generation and judging ran largely on H200 GPUs, with part of the judging through commercial APIs. The examination material belongs to its original sources; the AIME problems are credited to the MAA AMC and used for non-commercial research, as are our translations of them. We used AI assistants for code, figure scripts, literature checks and language editing.

Acknowledgments

This work was supported by the AMALIA project under Measure RE-C05-i08 of the Portuguese national Programa de Recuperação e Resiliência. We also acknowledge the support of the NOVA LINCS project (UID/04516/2025).

References

Andrew M. Bean, Ryan Othniel Kearns, Angelika Romanou, Franziska Sofia Hafner, et al. 2025. Measuring what matters: Construct validity in large language model benchmarks. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS 2025), Datasets and Benchmarks Track*. ArXiv:2511.04703.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 249–256. ACL Anthology E06-1032.

Mark Chen et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

DeepSeek-AI. 2026. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*.

Marius Dragoi, Ioana Pintilie, Florin Gogianu, and Florin Brad. 2025. Beyond pass@k: Breadth-depth metrics for reasoning boundaries. *arXiv preprint arXiv:2510.08325*.

Yakup Abrek Er, Ilker Kesen, Gözde Gül Şahin, and Aykut Erdem. 2026. Cetvel: A unified benchmark for evaluating language understanding, generation and cultural capacity of LLMs for Turkish. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1052–1085, Rabat, Morocco. Association for Computational Linguistics. ArXiv:2508.16431.

Gabriel Garcia. 2026. The last word often wins: A format confound in chain-of-thought corruption studies. *arXiv preprint arXiv:2605.10799*.

Gemma Team. 2026. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*.

Mohsen Hariri, Amirhossein Samandar, Michael Hinczewski, and Vipin Chaudhary. 2026. Don't pass@k: A Bayesian framework for large language model evaluation. In *The Fourteenth International Conference on Learning Representations (ICLR)*. ArXiv:2510.04265.

Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Z.Y. Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, and Bo Zheng. 2025. Can large language models detect errors in long chain-of-thought reasoning? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18468–18489, Vienna, Austria. Association for Computational Linguistics. ArXiv:2502.19361 (DeltaBench).

Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Or Honovich, Michael Tseng, Michael Collins, Roee Aharoni, and Mor Geva. 2024. A chain-of-thought is as strong as its weakest link: A benchmark for verifiers of reasoning chains. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4615–4634, Bangkok, Thailand. Association for Computational Linguistics. ArXiv:2402.00559 (REVEAL).

Amir Hossein Kargaran, Ayyoob Imani, François Yvon, and Hinrich Schütze. 2023. GlotLID: Language identification for low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6155–6218, Singapore. Association for Computational Linguistics.

Amir Hossein Kargaran, François Yvon, and Hinrich Schütze. 2024. MaskLID: Code-switching language identification through iterative masking. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 459–469, Bangkok, Thailand. Association for Computational Linguistics.

Tamera Lanham et al. 2023. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations (ICLR)*. ArXiv:2305.20050 (PRM800K).

Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. 2025. Are your LLMs capable of stable reasoning? In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17594–17632, Vienna, Austria. Association for Computational Linguistics. ArXiv:2412.13147 (G-Pass@k).

MAA. 2026. American invitational mathematics examination (AIME) 2026. `https://maa.org/maa-invitational-competitions/`. Administered by the Mathematical Association of America. AIME I, 5 February 2026; AIME II, 11 February 2026. Accessed 2026-08-11.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Evan Miller. 2024. Adding error bars to evals: A statistical approach to language model evaluations. *arXiv preprint arXiv:2411.00640*.

Avni Mittal and Rauno Arike. 2026. C2-faith: Benchmarking LLM judges for causal and coverage faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2603.05167*. V2, June 2026.

Qwen Team. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Qwen Team. 2026a. Qwen3.5. https://huggingface.co/Qwen/Qwen3.5-4B, https://huggingface.co/Qwen/Qwen3.5-9B. Model cards; released 2 March 2026. Accessed 2026-09-25.

Qwen Team. 2026b. Qwen3.6-35B-A3B. https://huggingface.co/Qwen/Qwen3.6-35B-A3B. Released 16 April 2026. Accessed 2026-09-25.

Joykirat Singh, Akshay Nambi, and Vibhav Vineet. 2025. Exposing the achilles' heel: Evaluating LLMs ability to handle mistakes in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27044–27065, Vienna, Austria. Association for Computational Linguistics.

Mahbub E Sobhani, Md. Faiyaz Abdullah Sayeedi, Tasnim Mohiuddin, Md. Mofijul Islam, and Swakkhar Shatabda. 2026. MathMist: A parallel multilingual benchmark dataset for mathematical problem solving and reasoning. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 2524–2550. Association for Computational Linguistics.

Guijin Son, Jiwoo Hong, Hyunwoo Ko, and James Thorne. 2025. Linguistic generalizability of test-time scaling in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14333–14368, Vienna, Austria. Association for Computational Linguistics. ArXiv:2502.17407 (MCLM).

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. PRMBench: A fine-grained and challenging benchmark for process-level reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25299–25346, Vienna, Austria. Association for Computational Linguistics. ArXiv:2501.03124.

Mingzhong Sun, Teresa Yeo, Armando Solar-Lezama, and Tan Zhi-Xuan. 2026. An enigma of artificial reason: Investigating the production-evaluation gap in large reasoning models. *arXiv preprint arXiv:2606.01462*.

Tarık Tuna Taşaltı, David Semedo, and Burcu Hüdaverdi. 2026. Cross-lingual mathematical reasoning in LLMs: Benchmarking base, non-think, and think modes on multilingual AIME 2026 with pass@k and CoT-pass@k. In *Proceedings Book of the VII. International Applied Statistics Congress (UYIK 2026)*, page 227, Istanbul, Türkiye. Tokat Gaziosmanpaşa University.

Diogo Tavares, Rafael Ferreira, Afonso Simplício, Gonçalo Vinagre, Ana Carolina Condez, Inês Calvo, Inês Vieira, David Semedo, and João Magalhães. 2026. PHEB: An European Portuguese high school-level LLM benchmark. In *Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026)*, pages 4673–4683, Palma, Mallorca, Spain. European Language Resources Association (ELRA).

Tiago Teixeira, Ana Carolina Erthal, Juan Belieni, Beatriz Canaverde, Diego Mesquita, Miguel Faria, Eliezer de Souza da Silva, and André F. T. Martins. 2026. MATH-PT: A math reasoning benchmark for european and brazilian portuguese. In *Proceedings of the 17th International Conference on Computational Processing of Portuguese (PROPOR 2026), Volume 1*, pages 1005–1010, Salvador, Brazil. Association for Computational Linguistics. ArXiv:2604.25926.

Çağrı Toraman et al. 2026. Turkbench: A benchmark for evaluating turkish large language models. In *Proceedings of the Second Workshop Natural Language Processing for Turkic Languages (SIGTURK 2026)*, pages 126–154, Rabat, Morocco. Association for Computational Linguistics.

Minzhu Tu, Shiyu Ni, and Keping Bi. 2026. How long reasoning chains influence LLMs' judgment of answer factuality. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 44957–44972, San Diego, California, United States. Association for Computational Linguistics.

TÜBİTAK. 2026. Ulusal bilim olimpiyatları: Geçmiş sınav soruları. https://bilimolimpiyatlari.tubitak.gov.tr/tr/gecmis-sinav-sorulari. 2026 mathematics olympiad, first stage. Accessed 2026-08-11.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Mingyang Wang, Lukas Lange, Heike Adel, Yunpu Ma, Jannik Strötgen, and Hinrich Schütze. 2025a.

Language mixing in reasoning language models: Patterns, impact, and internal causes. *arXiv preprint arXiv:2505.14815*.

Qian Wang, Zhenheng Tang, Zhanzhi Lou, Nuo Chen, Wenxuan Wang, and Bingsheng He. 2025b. Towards evaluating fake reasoning bias in language models. *arXiv preprint arXiv:2507.13758*. THEATER benchmark.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. 2026. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *The Fourteenth International Conference on Learning Representations (ICLR)*.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6).

Edwin B. Wilson. 1927. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2025. Evaluating mathematical reasoning beyond accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27723–27730. ArXiv:2404.05692.

Zheng-Xin Yong, M. Farid Adilazuarda, Jonibek Mansurov, Ruochen Zhang, Niklas Muennighoff, Carsten Eickhoff, Genta Indra Winata, Julia Kreutzer, Stephen H. Bach, and Alham Fikri Aji. 2025. Crosslingual reasoning through test-time scaling. *arXiv preprint arXiv:2505.05408*.

Qinan Yu, Alexa Tartaglini, Peter Hase, Carlos Guestrin, and Christopher Potts. 2026. Outcome rewards do not guarantee verifiable or causally important reasoning. *arXiv preprint arXiv:2604.22074*.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *Advances in Neural Information Processing Systems*. ArXiv:2504.13837.

Raoyuan Zhao, Yihong Liu, Hinrich Schütze, and Michael A. Hedderich. 2026. A comprehensive evaluation of multilingual chain-of-thought reasoning: Performance, consistency, and faithfulness across languages. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 5223–5247.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1024, Vienna, Austria. Association for Computational Linguistics. ArXiv:2412.06559.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), Datasets and Benchmarks Track*. ArXiv:2306.05685.

A Experimental Setup

A.1 Prompts

Solver prompts. Every solver receives the problem statement followed by one instruction sentence in the language of the benchmark:

> **EN:** Let's think step by step and output the final answer within `\boxed{}`.
> **TR:** Adım adım düşün ve nihai cevabı `\boxed{}` içerisinde ver.
> **PT:** Vamos pensar passo a passo e apresentar a resposta final dentro de `\boxed{}`.

The Turkish line serves the AIME translation and the Turkish olympiad; the Portuguese line serves the AIME translation and the Portuguese exams.

Judge prompt. Figure 3 gives the English judge prompt in full; it reproduces the verification prompt of Wen et al. (2026). The judge fills `{question}` and `{solution}` and must end with `\boxed{yes}` or `\boxed{no}`.

Multilingual variants. The Turkish and Portuguese judge prompts are identical to the English one except for exactly two additions. First, the opening sentence tags the problem language: "a given math problem **(written in Turkish)**" (respectively **Portuguese**). Second, a fourth task is appended:

```
4.  **Language Consideration**:
Ignore whether the solution is
provided in Turkish, English, or
a combination of both (language
switching).  Focus exclusively
on mathematical and logical
correctness, disregarding the
language used in the evaluation.
```

Everything else (the task list, the error taxonomy, the output format, and the verdict convention) is unchanged from Wen et al. (2026), so the multilingual judgments are the same instrument as the original English one up to these two additions.

```
You are an expert in mathematics and logical reasoning.  Your task is to evaluate
the correctness of a solution to a given math problem, with a **strong emphasis on
the reasoning process**, not just the final answer.
Below is the **Problem** and the **Solution (Provided by another AI model)**:
--
**Problem**:
{question}
**Solution (Provided by another AI model)**:
{solution}
--
Please perform the following tasks:
1.  **Analyze the solution step-by-step**, paying close attention to:  -
Computational accuracy - Logical consistency - Conceptual understanding - Whether
the reasoning is valid and complete
2.  **Identify any issues or errors in the reasoning**, even if the final answer is
correct.  Classify them into the following categories (if applicable):  -
**Calculation Error**:  Mistakes in arithmetic, algebraic manipulation, or numerical
computation.  - **Logical Error**:  Invalid reasoning, flawed logic, or incorrect
inference.  - **Conceptual Error**:  Misunderstanding or misuse of mathematical
concepts or definitions.  - **Omission / Incompleteness**:  Missing steps,
incomplete justification, or not addressing all parts of the question.  - **Other**:
Any other type of error that does not fit into the above categories.
3.  **Provide a final judgment** on whether the solution is logically sound and free
of errors in reasoning.
Please format your response as follows:
--
**Issues Identified:**
- [Issue 1]:  [Classification] - [Brief explanation] - [Issue 2]:  [Classification] -
[Brief explanation] - ...
Let's think step by step and output your final judgment within \boxed{}
\boxed{yes} or \boxed{no}
```

Figure 3: The English judge prompt, reproduced from Wen et al. (2026). `{question}` and `{solution}` are filled per judgment; the Turkish and Portuguese variants differ from it by the two additions given in Appendix A.1.

A.2 Configuration

Serving and sampling. All models are served with their family's official chat templates: for the Qwen3.5 instruct models thinking is explicitly disabled or enabled, while base checkpoints run under their template's default thinking setting; sampling parameters are set explicitly for every run. Every solver draws 64 generations per question, and 16 on the Portuguese exams, whose near-saturation makes larger samples uninformative and whose 166 questions keep the total sample count comparable to the other benchmarks. Our pipeline extends an open-source evaluation harness with these judging stages.

Judge budgets. The judgment budget is 16,384 tokens throughout, with two exceptions. V4-Flash judges at 20,480 tokens wherever it appears, and in the error-injection study of Section 4.2 all three judges share that setting; the budget ladder of Section 4.4 raises the anchor judge's budget in step with the solver's.

Judge version. V4-Flash is the April 2026 preview build of DeepSeek-V4-Flash, queried through the DeepSeek API before 31 July 2026, when the same API name moved to re-trained builds; the preview weights remain published.

Judging cost. Pass@k needs only the solver's generations, while CoT-Pass@k also sends every answer-correct generation to the judge three times. On the judge matrix of Section 5.2 the ten solvers produced 78,080 generations with 416 million output tokens, of which 20,059 were answer-correct. Qwen3.6 and Gemma4 judged all of these, 60,177 judgments each, and V4-Flash and R1-distill, which cover the earlier generation only, judged its 2,180 answer-correct generations, 6,540 judgments each, for 133,434 judgments and 797 million judge output tokens in all. A judgment averages 3.5 thousand output tokens under V4-Flash and 7.9 thousand under R1-distill, so judging one correct solution costs 10 to 24 thousand tokens, and in the same cells Qwen3.6 and Gemma4 emit 0.82 and 0.92 tokens for every solver token, which nearly doubles the output of a Pass@k evaluation.

Variant	any	majority	all
V4-Flash			
Clean	96.2	92.9	89.2
Intermediate numeric error	96.1	93.1	90.1
Truncation error	89.3	85.1	78.8
Final-answer error	47.2	18.1	6.4
Consistent final-answer error	49.0	24.6	12.4
Qwen3.6			
Clean	99.4	97.8	81.2
Intermediate numeric error	99.2	98.2	81.0
Truncation error	93.9	92.1	69.2
Final-answer error	67.2	34.3	7.8
Consistent final-answer error	75.6	54.2	27.8
R1-distill			
Clean	96.9	90.1	36.4
Intermediate numeric error	96.0	91.9	42.2
Truncation error	92.1	86.7	67.4
Final-answer error	91.9	70.6	28.3
Consistent final-answer error	80.3	61.8	23.1

Table 1: Acceptance rates (%) under the three verification strategies of the original CoT-Pass@k judge: a variant is accepted when at least one of its three judge generations approves it (any-correct), when more approve than reject (majority-correct), or when all three do (all-correct). The majority-correct column is the one plotted in Figure 1. 720 original solutions per error type for each judge; R1-distill is the judge the metric specifies.

The error-injection study adds 32,400 judgments, 720 solutions in five variants, three judgments each, under three judges. The jobs in our cluster's accounting add up to about 620 H200 GPU-hours of generation and open-weights judging, exploratory runs included; this is a lower bound, since runs outside that record, V4-Flash, which ran through the DeepSeek API, and the Qwen3.6 judgments of two error-injection cells, served through a commercial API when the cluster was unavailable, are not in it.

Selecting solutions for error injection. The original solutions of Section 4.2 are drawn from correctly answered generations that stopped before the max-token limit and are longer than 600 tokens, keeping only solutions to which all four error types can be applied.

B CoT Verification Strategies

The original CoT-Pass@k judge turns the three judgments of a solution into one verdict with a *verification strategy*: *any*-correct accepts a variant if at least one of its three judgments approves it, *majority*-correct if more of them approve than reject, and *all*-correct only if all three approve. Any- and all-correct count a judgment that ends without a verdict as a rejection and majority-correct leaves it out, which matters unevenly. Such judgments are 8.5% of Qwen3.6's, touching one in five of its variants, reach fewer than one in a hundred of V4-Flash's variants, and make up 16.5% of R1-distill's, almost all ending without a boxed verdict rather than at the token limit, which is why R1-distill's all-correct rate falls to 36% on the clean control. Requiring two approvals outright would lower every Qwen3.6 rate by 1.7 to 5.3 points without reordering the five conditions, leave V4-Flash unchanged, and lower R1-distill's rates by 2 to 11 points. R1-distill is also unstable on identical input, splitting 22% of the clean controls that received three verdicts between approval and rejection, against 2% for Qwen3.6 and 7% for V4-Flash.

Table 1 repeats the error injection experiment under all three strategies. No strategy detects the intermediate numeric error, whose acceptance stays within 1 point of the clean control in all six combinations of V4-Flash and Qwen3.6 and, for R1-distill, falls at most 1 point below it, and in every combination of V4-Flash and Qwen3.6 the consistent final-answer error is accepted more often than the plain one, while R1-distill orders the two the other way under all three strategies. What the strategies change is only how far apart they place the two groups. For V4-Flash and Qwen3.6, all-correct separates them furthest, but relative to any-correct it also rejects 7.1 points more of the unedited V4-Flash controls and 18.2 points more of the Qwen3.6 ones, so part of its strictness is disagreement between generations of the same judge rather than error detection.

Size and kind of the edit. Figure 4 splits the same 720 solutions by how the wrong number was formed and by how far it moved. The intermediate numeric error is missed whatever the edit, at 86 to 100% acceptance in every cell of every judge, and its position between 40 and 70% of the chain moves acceptance by at most 3 points. The plain final-answer error is also insensitive to size, within 10 points across the four size bins. Only the consistent final-answer error reacts to the size of the number. A wrong answer within 1% of the correct one is accepted 71% of the time by Qwen3.6, 34% by V4-Flash and 69% by R1-distill, one more than double the correct value 26, 9 and 49%.

Rates per benchmark. Figure 1 gives the majority-correct rates per benchmark. The inter-

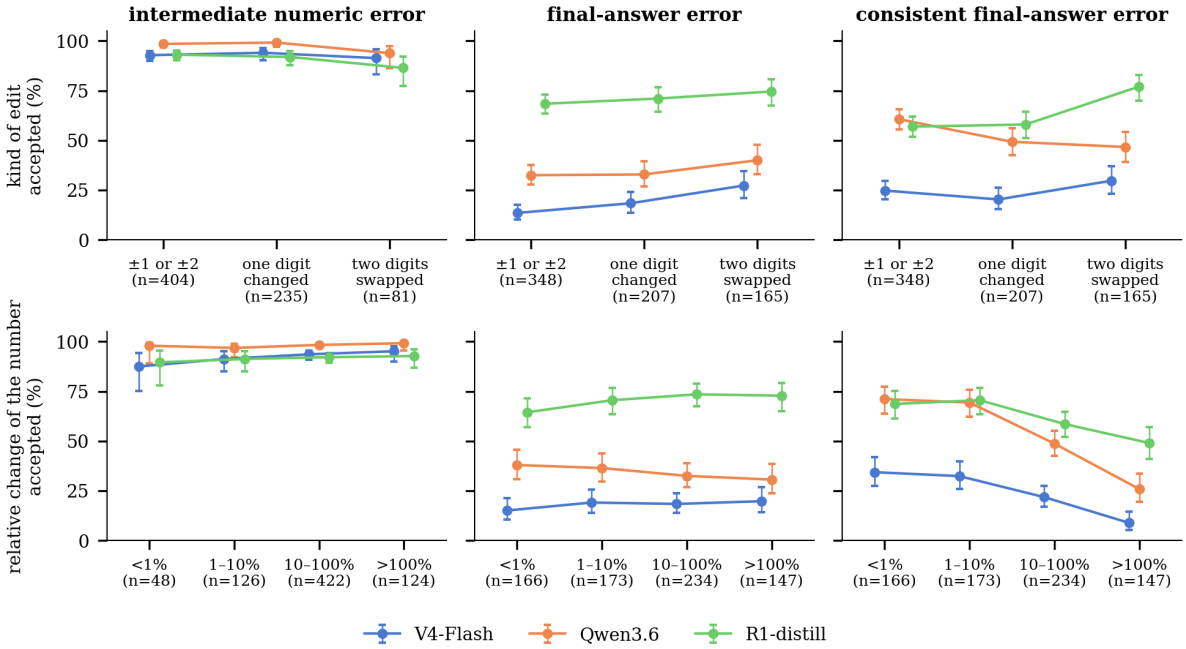

Figure 4: Acceptance (%, majority-correct) by the kind and the size of the injected edit, with 95% Wilson intervals, for the three conditions that change a number (columns). Top, how the wrong number was formed; bottom, its relative change $|\text{new} - \text{old}|/|\text{old}|$. Each row of panels partitions the same 720 solutions of every condition, and the number of solutions in a category is given under its label.

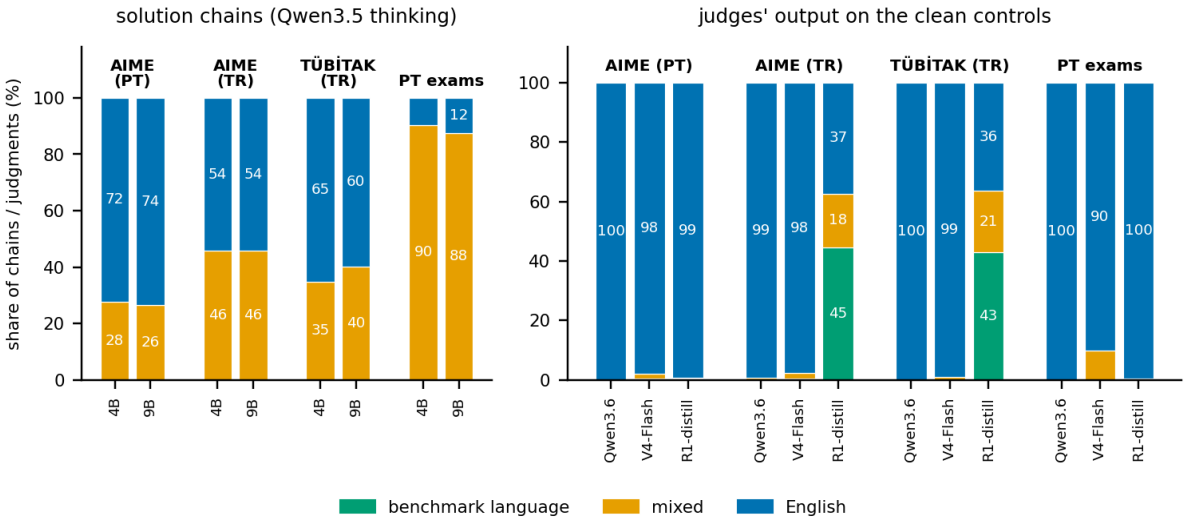

Figure 5: Language of the error-injection study, by benchmark. Left, the 720 original solutions of Section 4.2 (Qwen3.5-4B and 9B, thinking mode), classed by the share of their letters, mathematics removed, in the benchmark's language (at least 80%, *benchmark language*), in English (at least 80%, *English*) or in between (*mixed*); the edits do not change a chain's language, so the five variants share the classification. Right, the judges' output on the clean controls, thinking and visible answer together, classed the same way. On English AIME every chain and every judgment is English, so that benchmark is left out. Under the wrong-answer conditions Qwen3.6's shares stay within 1 point of these, R1-distill's all-English share on the Turkish benchmarks falls from 36 to 37% to 19 to 26%, and V4-Flash's mixed share on the Portuguese exams falls from 10% to under 3%. Appendix C.5 gives the method.

mediate numeric error shifts no benchmark under any judge (exact McNemar on 144 solutions each, smallest $p = 0.30$), and R1-distill's loss of 15 to 27 points under a wrong final answer is significant on every benchmark ($p < 10^{-3}$). The Portuguese exams carry the shortest solutions, a median of 3.7k tokens against 12k to 24k on the other four, and every judge accepts their clean control at 99 to 100%. The Turkish olympiad gives each judge its lowest clean control and is the one benchmark where V4-Flash accepts the consistent error less often than the plain one; R1-distill does so there and on the Portuguese exams as well (exact McNemar $p \leq 1.6 \times 10^{-3}$ for all three reversals). On the three AIME sets, the same problems in three languages, the rates move together.

C Judges, Sampling and Uncertainty

Two tables and six figures support Section 5.2, one group per subsection.

C.1 The Judge Matrix

Table 2 gives the difference Pass@64 − CoT-Pass@64 for every earlier-generation solver under all four judges, and Table 3 for every current-generation solver under the two judges that cover both generations: together the per-model form of the claim that the collapse does not depend on the judge.

The verdict threshold of Section 4.3 removes from the comparison grid the Gemma-4 base checkpoints (too few correct answers to support a rate), Qwen3.5-0.8B (under the threshold on three of the four benchmarks as a base checkpoint and on all four in non-thinking mode) and Qwen3.5-2B (over it only as a base checkpoint).

Solver	Qwen3.6	Gemma4	V4-Flash	R1-distill
Q2.5-7B	15.6	13.9	18.0	15.6
Q2.5-32B	20.5	22.1	23.0	21.3
Q2.5-7B-I	22.1	23.0	23.8	22.1
Q2.5-32B-I	20.5	20.5	21.3	21.3
Average	19.7	19.9	21.5	20.1

Table 2: Pass@64 − CoT-Pass@64 per earlier-generation solver (rows), averaged over the four 64-sample benchmarks (weighted by how many questions each contributes, as in Figures 6 and 7) under all four judges (columns). Judge generation budgets: 16,384 for Qwen3.6, Gemma4 and R1-distill, 20,480 for V4-Flash; every judge runs in thinking mode with three judgments per solution. The last row averages the solvers. Q2.5 = Qwen2.5 (-I Instruct).

Solver	Qwen3.6	Gemma4
Q3.5-4B-B	0.8	1.6
Q3.5-9B-B	0.0	0.8
Q3.5-4B	4.9	6.6
Q3.5-9B	3.3	4.1
G4-E2B	8.2	8.2
G4-E4B	7.4	4.9
Average	4.1	4.4

Table 3: As Table 2, for the current-generation solvers under the two judges that cover them. Q3.5 = Qwen3.5 (-B base), G4 = Gemma-4-it.

C.2 The k Ladder and Question-Level Uncertainty

Figure 6 follows the same difference as the number of sampled generations grows. Figure 7 reports the question-level bootstrap intervals quoted in the text, per solver.

One row of Figure 7 needs a warning. Qwen3.5-9B-Base has a difference of exactly zero on each of the four benchmarks, so every resample of its questions returns zero and its interval collapses to a point. That is a structural limit of resampling an all-zero indicator, not a statement that the value is known without uncertainty, and the same applies to any cell-level interval of the form $[0, 0]$.

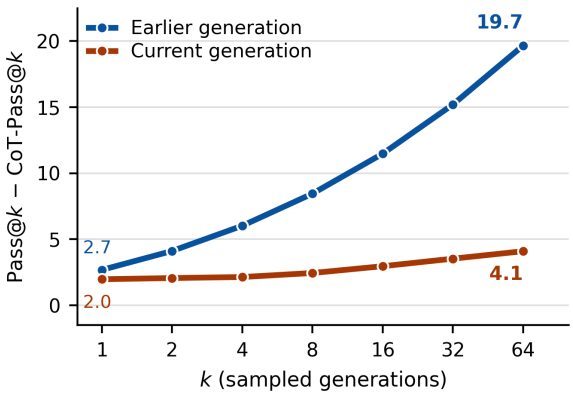

Figure 6: The difference Pass@k − CoT-Pass@k against the number of sampled generations, under the anchor judge Qwen3.6 at 16k. Each solver is a weighted mean over its four benchmarks, weighted by how many questions each contributes; each curve is then the unweighted mean over the solvers of that generation. Endpoints are labelled; Figure 7 gives the per-solver spread at $k = 64$.

C.3 Acceptance of Correct Generations

Section 5.2 reports the collapse as the difference Pass@64 − CoT-Pass@64, and at k=64 that difference can shrink for a reason that has nothing

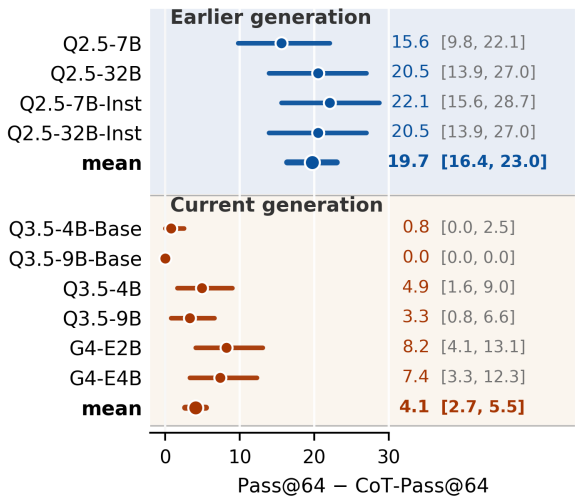

Figure 7: Question-level bootstrap 95% intervals for Pass@64 − CoT-Pass@64 under the anchor judge Qwen3.6 at 16k, in points, with the numeric interval at the right of each row. Each solver is a weighted mean over its four benchmarks, weighted by how many questions each contributes; *mean* is the unweighted mean over the solvers of a generation. 10,000 resamples of questions, percentile intervals; cells are resampled independently. Q2.5/Q3.5 = Qwen2.5/Qwen3.5, G4 = Gemma-4-it.

to do with the judge. A question keeps its CoT-Pass@64 as long as one of its correct generations is approved, so where a question holds many correct generations the judge can reject most of them without moving the metric. Two figures, here and in Appendix C.4, separate that saturation from the verdicts. Figure 8 gives the quantity the difference hides, the acceptance rate of answer-correct generations, per solver and benchmark and under both judges. The two generations never overlap, 57 to 70% against 80 to 99% per solver, and on the native Turkish olympiad the anchor judge approves only 39 to 47% of the earlier generation's correct generations and 75 to 99% of the current generation's. Averaged over the ten solvers the Turkish translation sits within 3 points of the English original under both judges (Wilcoxon signed-rank test (Wilcoxon, 1945), $p = 0.49$), the Portuguese translation runs 6 points above it under the anchor judge ($p = 0.004$) and 3 above under Gemma4 ($p = 0.25$), and the olympiad sits 15 and 14 points below the Turkish translation ($p = 0.002$), so the language of the problem does not lower the rate and the benchmark does.

C.4 Saturation and Conditioning on c

The tick labels of Figure 9 give the number of question and solver pairs in each bin of c. The earlier generation's solved questions sit at $c \leq 8$, where 80 of 116 pairs lose every correct generation, and the current generation's at $c \geq 33$, where 1 of 287 does, so the saturation is real and it favours the current generation. The figure then holds c fixed. At every c the current generation keeps more of its questions and more of its correct generations. At $c = 1$–2 CoT-Pass@64 is 24 against 68 under the anchor judge and the judge accepts 23% against 66% of the correct generations; at $c = 33$–64 it is 83 against 100 and 74% against 96%, so in both generations the judges reject most where a question holds few correct generations. The gap between the generations therefore survives conditioning on c, and what saturation explains is only how the current generation's remaining rejections, 4 to 34% of its correct generations, turn into a difference of zero.

C.5 Language of the Chains

Solvers are prompted in the benchmark's language and their chains may switch into English (Section 4.1). To measure how often, every answer-correct generation of the judge matrix is stripped of its mathematics (display and inline formulas, boxed answers, LaTeX commands, digits and operator symbols) and cut into lines and sentences of at least twenty letters; each segment receives one label from GlotLID (Kargaran et al., 2023), and segments that restate the question are dropped. A chain is *target* when at least 80% of its letters are in the benchmark's language, *English* when at least 80% are English, and *mixed* otherwise. As a second layer, MaskLID (Kargaran et al., 2024), which detects code switching by iterative masking, is run on the first 1,000 words of every earlier-generation chain, of a fifth of the current generation's chains and of every panel chain; the two layers agree on whether a chain is mixed for 93% of the chains they share. Labels outside the three languages cover under 3% of the letters, and a threshold of 70 or 90% changes the *target* share by at most one point.

The language of a chain is a property of the solver (Figures 10 and 11). The Qwen3.5 base checkpoints, which think by default, write 79 to 93% of their chains on the non-English benchmarks in English and the rest mixed, while the instruction-tuned solvers of both generations answer in the

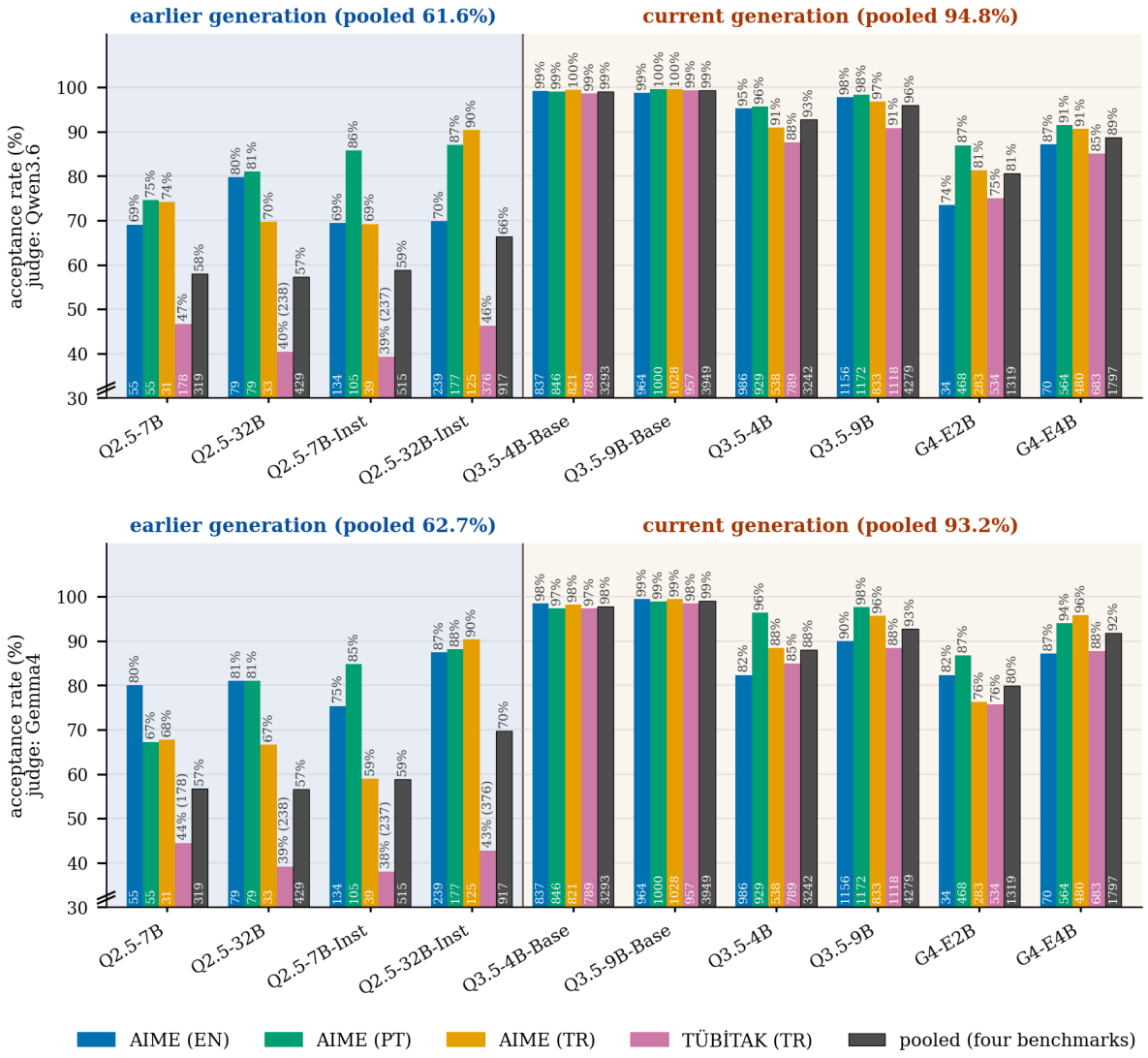

Figure 8: Acceptance rate of answer-correct generations per solver and benchmark, the share of a solver's correct generations whose reasoning chain the judge approved, under Qwen3.6 (top) and Gemma4 (bottom), for the solvers of Section 5.2 at the 16k budget. The number above each bar is that rate in percent; the number at its base is the count of answer-correct generations in the cell, given in parentheses after the rate where the bar is too short. The rejected share is 100 minus the rate. The dark grey bar of each solver gives its rate pooled over the four benchmarks, with the pooled count at its base, and the panel headers give the rate pooled over each generation. The vertical axis starts at 30. Q2.5/Q3.5 = Qwen2.5/Qwen3.5 (-Inst instruct, -Base base), G4 = Gemma-4-it.

benchmark's language at least 97% of the time; only the Qwen2.5 base checkpoints split, and the Portuguese exams show the same division. Because the classes coincide with solvers, acceptance by language cannot be read across solvers. Within a solver the judges accept English and mixed chains alike, within 3 points for every Qwen3.5 base cell, and the one solver with enough chains of both kinds, Qwen2.5-32B on the Turkish olympiad, is accepted 48% of the time in English and 35% in Turkish.

The error-injection panel holds no chain in the benchmark's language (Figure 5). There the verdicts move with the language under two judges on the Turkish benchmarks. On the Turkish olympiad V4-Flash accepts 94% of the mixed clean chains against 69% of the all-English ones and R1-distill 85 against 74%, and R1-distill also accepts the wrong final answer more often in a mixed chain, 72 against 49% there and 73 against 63% on the Turkish AIME set; Qwen3.6 stays within 4 points on every clean control, and on the Portuguese benchmarks every judge's clean controls stay within 7.

The judges themselves reason in English. On a

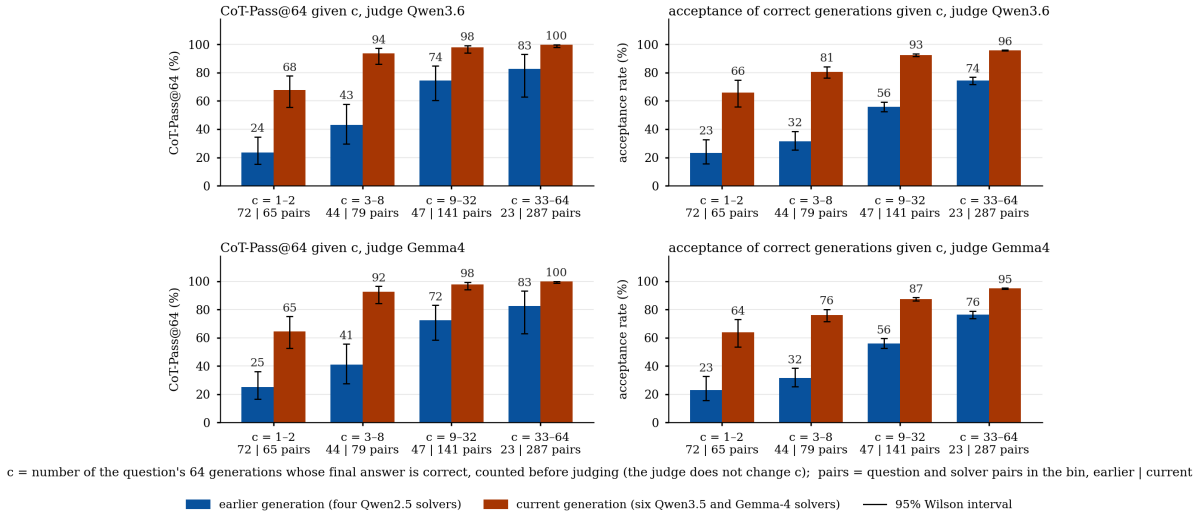

Figure 9: CoT-Pass@64 after conditioning on c, the number of a question's 64 generations whose final answer is correct, counted before judging, for the ten solvers of Section 5.2 at the 16k budget, pooled over the four benchmarks, under both judges (rows), with 95% Wilson intervals. Left, the share of question and solver pairs in the bin in which the judge approved at least one correct generation, which is CoT-Pass@64 for that bin since Pass@64 there is 100 by construction; right, the share of answer-correct generations in the bin that the judge approved, the acceptance rate of Figure 8 at fixed c, so that the two generations can be compared at the same c. The tick labels give the number of question and solver pairs in each bin, earlier | current. Per benchmark, the Turkish olympiad has the lowest value in both panels in every bin of the earlier generation, and Figure 8 gives the acceptance rates per benchmark.

sample of 300 judgments per judge and benchmark from the judge matrix, all four judges write 78 to 100% of their judgments on the Turkish and Portuguese benchmarks entirely in English and none in the benchmark's language, whatever the language of the chain (Figure 11). The error-injection panel gives the same picture with one exception. On the two Turkish benchmarks R1-distill thinks mostly in Turkish in 40 to 47% of its judgments while its visible verdict stays in English, and the judgments it thinks through in Turkish approve more often, 78 against 64% of the clean controls on the Turkish AIME set and 65 against 62% on the olympiad.

D Token Budgets and Generation Mode

D.1 Stops at the Max-Token Limit

Table 4 supports Section 5.3 with the share of solver generations and of judgments that stop at the max-token limit, per benchmark and solver, with the difference each judge reports.

D.2 Judge Budgets and the Difference

Figure 12 decomposes the difference Pass@64 − CoT-Pass@64 that each judge reports, under the counterfactual of Section 4.4: judgments that stop at the max-token limit are counted as approvals instead, and what survives is a lower bound on the

		solver	Gemma4		Qwen3.6	
Benchmark	Solver	stops	stops	diff.	stops	diff.
AIME (EN)	Q3.5-4B	89.6	35.0	3.33	15.7	0.00
	Q3.5-9B	83.9	33.0	3.33	17.5	0.00
	G4-E2B	0.2	2.1	6.67	6.5	3.33
	G4-E4B	1.4	2.7	10.00	6.0	3.33
AIME (PT)	Q3.5-4B	47.7	53.5	13.33	5.2	0.00
	Q3.5-9B	44.0	49.8	0.00	3.6	0.00
	G4-E2B	0.1	9.8	3.33	1.4	3.33
	G4-E4B	0.2	7.5	6.67	1.9	0.00
AIME (TR)	Q3.5-4B	44.2	51.0	0.00	10.0	0.00
	Q3.5-9B	35.6	53.8	10.00	8.3	0.00
	G4-E2B	0.5	7.6	3.33	3.0	3.33
	G4-E4B	0.2	9.8	6.67	3.2	0.00
TÜBİTAK (TR)	Q3.5-4B	44.4	47.6	6.25	3.6	0.00
	Q3.5-9B	35.6	49.1	6.25	3.4	0.00
	G4-E2B	0.8	13.6	3.12	0.2	0.00
	G4-E4B	0.0	10.3	9.38	0.1	9.38
PT exams	Q3.5-4B	3.6	5.0	0.00	1.5	0.00
	Q3.5-9B	1.7	3.5	0.00	1.3	0.00
	G4-E2B	0.1	0.7	0.00	0.2	0.60
	G4-E4B	0.0	0.4	0.00	0.2	0.00

Table 4: Stops at the max-token limit on both sides, in thinking mode at the 16k generation budget: the share of solver generations that stop at the max-token limit (stops, %), and per judge the share of its judgments that do, with the difference Pass@$k_{\max}$ − CoT-Pass@$k_{\max}$ (diff.) under that judge. $k_{\max}$ is 64, or 16 for the Portuguese exams; stop rates come from each generation's recorded finish state. Q3.5 = Qwen3.5, G4 = Gemma-4-it.

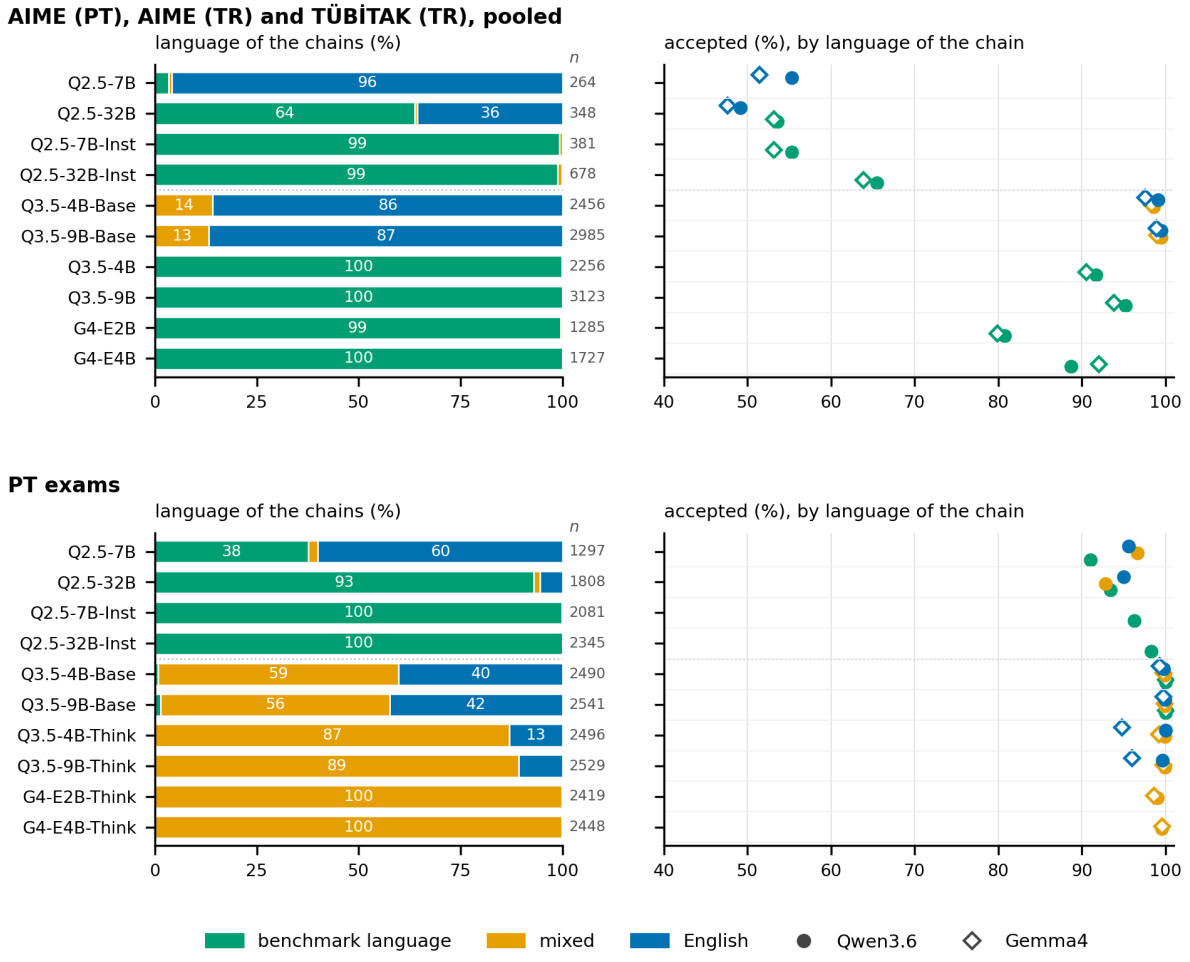

Figure 10: Language of the reasoning chains (left) and acceptance by language (right) for the answer-correct generations of the ten solvers of Section 5.2 at the 16k budget, on the three non-English benchmarks pooled (top) and on the Portuguese exams (bottom), where thinking runs stand in for the non-thinking runs they lack. A chain is in the *benchmark language* when at least 80% of its letters, mathematics removed, are in that language, *English* when at least 80% are English, *mixed* otherwise. Acceptance is the majority verdict of Qwen3.6 (filled) and Gemma4 (open); a class with fewer than 20 chains of a solver is not plotted, and n is the number of chains. The dotted line separates the earlier generation from the current one. Pooling mixes benchmarks with different acceptance rates, so within-solver comparisons are read per benchmark in Appendix C.5. Q2.5/Q3.5 = Qwen2.5/Qwen3.5 (-Inst instruct, -Base base, -Think thinking mode), G4 = Gemma-4-it.

part carried by the judge's verdicts, since a stopped judgment that would have rejected is credited as an approval. The scope is the fully crossed design of Section 5.3: the four current-generation solvers on the four 64-sample benchmarks, in both generation modes, every cell scored by both judges at the 16k budget.

The same counterfactual applied to the earlier generation certifies the other side of the contrast: the four Qwen2.5 solvers on the same four benchmarks, every correct solution judged, at the same 16k solver budget. Crediting every stopped judgment as an approval leaves, of the average difference each judge reports in Table 2, at least 15.2 of the anchor judge's 19.7 points, 12.5 of Gemma4's 19.9, 18.6 of R1-distill's 20.1, and 20.1 of V4-Flash's 21.5. Under the two judges that rarely stop at their own limit the difference barely moves: the earlier generation's side of the generational contrast is carried by verdicts, not by the judges' budgets.

D.3 The Budget Ladder

Table 5 supports Section 5.3: the two thinking-mode solvers of Section 4.2 at their low and raised generation budgets, row by row. The ladder is reported under the anchor judge alone: at the raised budgets Gemma4 stopped at its own max-token limit so often that serving slowed and its runs could

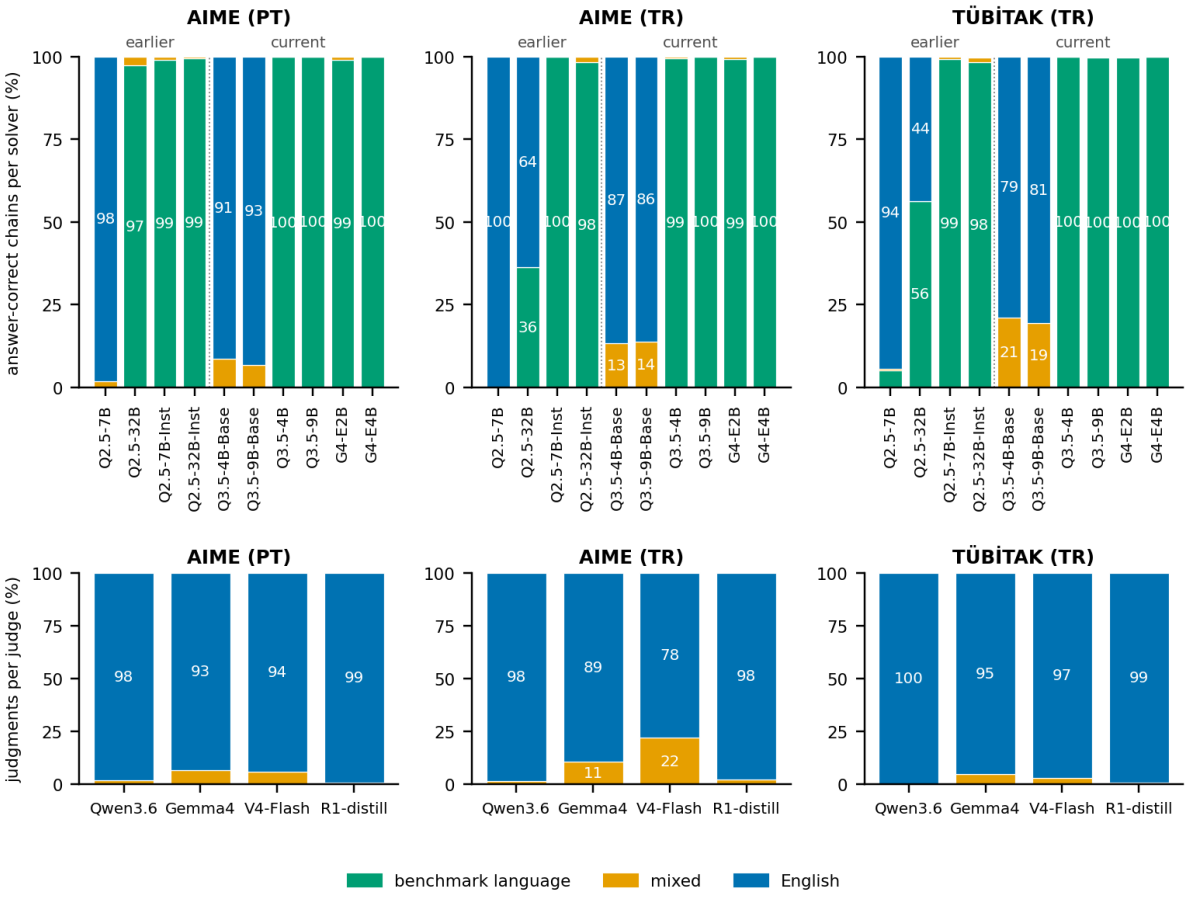

Figure 11: Language of the answer-correct chains and of the judgments in the judge matrix of Section 5.2, per benchmark. Top, the ten solvers at the 16k budget, earlier generation left of the dotted line and current generation right of it (-Inst instruct, -Base base); bottom, a sample of 300 judgments per judge and benchmark. Classes as in Figure 5. English AIME is left out, every chain and judgment there being English. Figure 10 gives the same shares pooled over the three benchmarks together with acceptance by class.

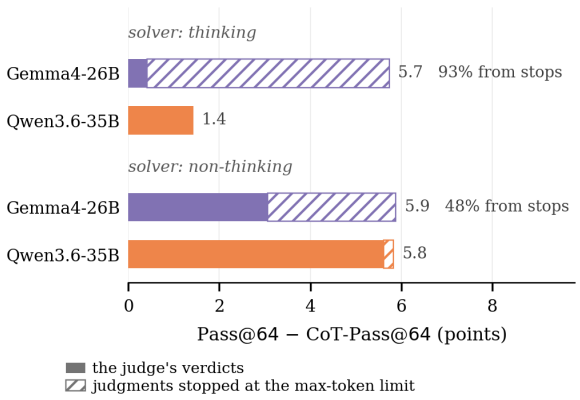

Figure 12: The difference Pass@64 − CoT-Pass@64 per judge and solver mode, split into the part carried by the judge's verdicts (solid) and the part that disappears when judgments stopped at the max-token limit are counted as approvals (hatched). Cell means over the four 64-sample benchmarks and the four current-generation solvers, 16k budget on both sides.

not be completed, the instrument buckling under the budgets it was given.

D.4 Generation Mode and Language

Table 6 supports Section 5.3: the same thirty AIME problems in three languages, solved by the same two solvers in both generation modes. Neither reading of the language difference is more correct than the other: both are conditional on a mode and a budget a paper would ordinarily report in a footnote.

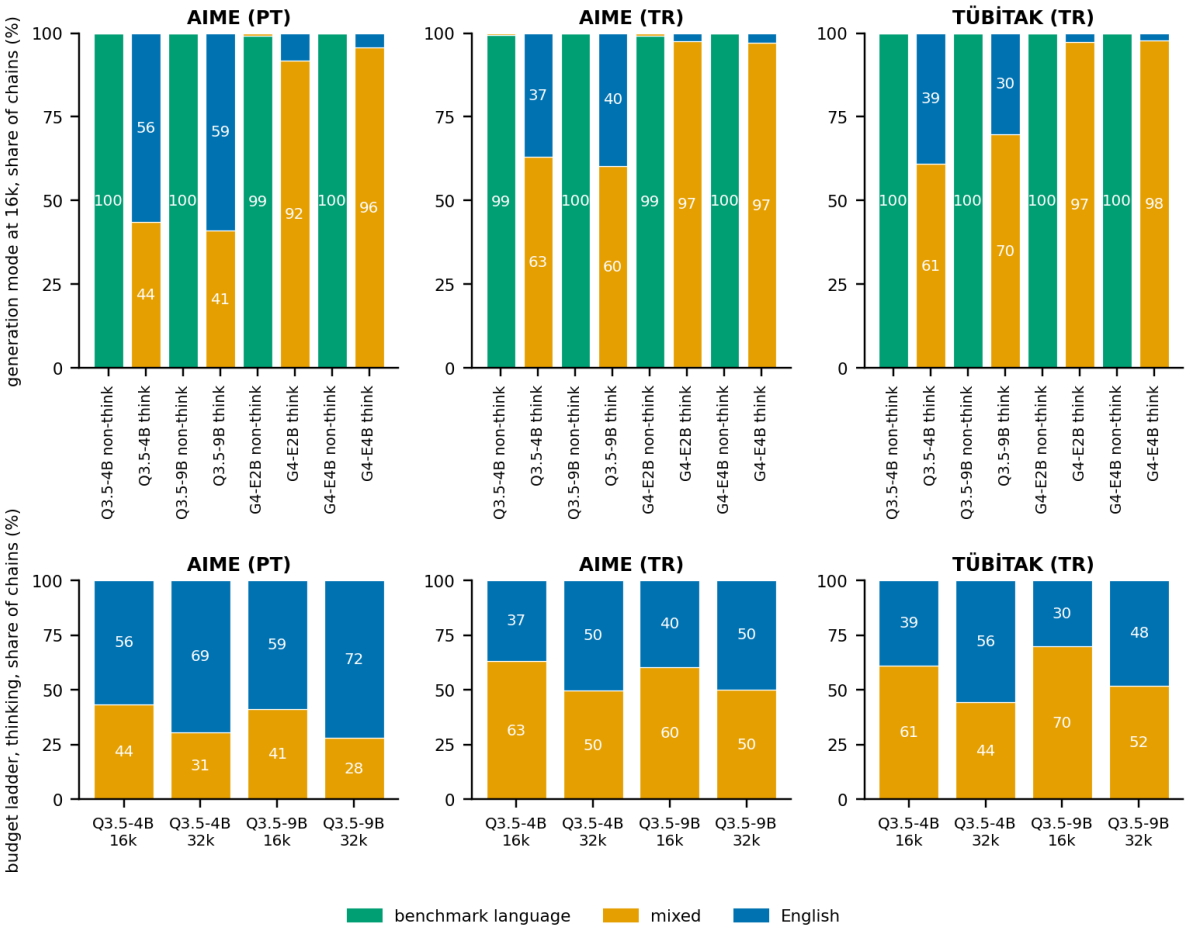

Figure 13: Language of the answer-correct chains under the generation mode and the token budget of Section 5.3, per benchmark. Top, the four current-generation solvers at the 16k budget in non-thinking and in thinking mode; bottom, the two Qwen3.5 thinking-mode solvers at the low and the raised generation budget of the ladder (Table 5). Classes as in Figure 5; English AIME left out. Turning thinking on moves every chain out of the benchmark's language, and raising the budget moves the mixed chains further toward English.

		stops at max tokens (%)		Pass@1 (%)		Pass@64 (%)	
Benchmark	Solver	low	high	low	high	low	high
AIME (EN)	4B	89.6	18.4	10.5	75.4	40.0	96.7
	9B	83.9	8.9	16.2	84.5	40.0	96.7
AIME (PT)	4B	47.7	7.6	51.0	71.0	83.3	93.3
	9B	44.0	3.8	55.3	78.2	80.0	96.7
AIME (TR)	4B	44.2	5.3	51.8	68.5	80.0	93.3
	9B	35.6	2.9	62.6	76.0	90.0	93.3
TÜBİTAK (TR)	4B	44.4	6.0	45.4	65.5	87.5	93.8
	9B	35.6	3.0	55.4	75.4	90.6	93.8

Table 5: The budget ladder: the same solver on the same benchmark at its low and high generation budget. Low is 16,384 tokens everywhere; high is 32,768, except 65,536 on English AIME (the text explains the jump). Share of solver generations stopping at the max-token limit, Pass@1 and Pass@64; thinking mode throughout. 4B/9B = Qwen3.5.

Mode	Lang.	stops at max tokens (%)	Pass@1 (%)	Pass@64 (%)
non-thinking	EN	27.9	55.8	91.7
	PT	2.4	54.7	95.0
	TR	16.6	35.7	88.3
thinking	EN	86.7	13.4	40.0
	PT	45.8	53.2	81.7
	TR	39.9	57.2	85.0

Table 6: The same thirty AIME problems in English and in the Portuguese and Turkish translations: Qwen3.5-{4B,9B} at the 16k generation budget, both modes. Columns give the share of generations that stop at the max-token limit, Pass@1 and Pass@64, averaged over questions and over the two models.