CERTIFYING WHEN TO STOP: EXACT EARLY STOPPING FOR PRM-BASED REASONING SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Selecting among generated reasoning traces with a process reward model (PRM) usually requires scoring every trace, even when the remaining scores cannot change the selected answer. We introduce exact sequential certification, a training-free procedure that avoids such redundant verification without changing the output of the full, verify-all selector. The method reveals PRM scores incrementally and maintains lower and upper bounds on the final PRM-weighted vote of each answer group. It stops once one group's lower bound strictly exceeds every competitor's upper bound; if this never occurs, it scores all traces and applies the original selector. The returned answer is therefore identical to verify-all for any reveal order and positive batch size. Across seven mathematics benchmarks and $k \in \{4, 8, 16, 32\}$, all fully evaluated policies produce zero answer mismatches on the same $6{,}628$ item–configuration cells. With leader-first ordering and batch size one, the method reduces PRM calls by 54.8%, estimated PRM tokens by 54.6%, and estimated end-to-end tokens by 29.4% under the audited aggregate accounting. At $k{=}32$, it averages 2.38–18.26 PRM calls instead of 32 and saves 22.9–52.6% end-to-end tokens across benchmarks. A matched audit of three additional 7B generator pools likewise finds zero mismatches over $4{,}788$ generator–item–budget cells and 19.4–52.7% end-to-end savings at $k{=}32$. Exact unanimity, including the corresponding branch of Sensitivity-Aware Verification Allocation (SAVA), is the zero-call special case of the certificate.

1 INTRODUCTION

Test-time scaling often generates multiple reasoning traces and uses a learned verifier to select among them. Repeated sampling, best-of-N selection, and longer deliberation can improve accuracy by spending more inference compute (Wang et al., 2023; Snell et al., 2025; Muennighoff et al., 2025; DeepSeek-AI, 2025). Adaptive sampling methods reduce generation cost by stopping or routing queries according to agreement, confidence, or estimated difficulty (Aggarwal et al., 2023; Li et al., 2024; Wan et al., 2025; Huang et al., 2026; Aghazadeh et al., 2025). A separate line of work uses learned verifiers to rank completed traces or guide search (Lightman et al., 2023; Chen et al., 2025; Zhong et al., 2026). Once a fixed pool of k traces has been generated, however, standard PRM selection still scores every trace before aggregating scores by final answer. In our evaluated pools, this verification accounts for roughly half of the end-to-end token budget.

We ask a narrower question: *when can a system stop scoring a fixed trace pool while being certain that full verification would return the same answer?* This differs from predicting that further verification is unlikely to matter. Our goal is a deterministic certificate: stop only after the observed scores prove that every possible completion of the unseen, bounded scores has the same winner. Figure 1 illustrates this idea through a matchmaking analogy.

We model each PRM evaluation as an expensive access to a bounded trace contribution. After any subset of traces has been scored, the observed weight of an answer group is a lower bound on its final weight. Adding the maximum possible contribution of its unscored traces gives a corresponding upper bound. If one group's lower bound strictly exceeds every rival's upper bound, no unseen score can change the winner. Otherwise, the method reveals another batch and repeats, falling back to verify-all only when necessary. This view extends the exact zero-call unanimity shortcut to non-unanimous pools and every intermediate verification budget.

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.

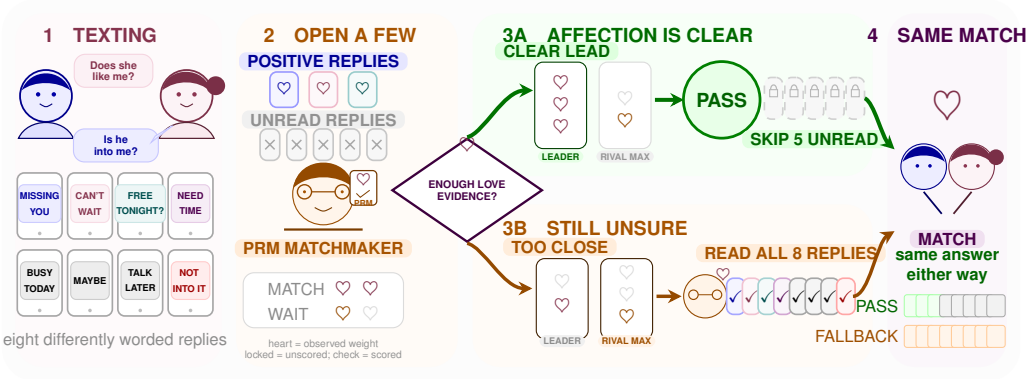


Figure 1: Motivation and method through a matchmaking analogy. Eight candidate phones range from affectionate ("MISSING YOU", "CAN'T WAIT") through hesitant and neutral replies to an explicit rejection, but collapse into only a few final-answer groups. The PRM matchmaker opens leader-group conversations first. Filled hearts depict observed weight (a lower bound), while pale hearts depict the most an unseen reply could still add (an upper bound). If the observed leader already exceeds every rival's possible maximum, the green PASS route safely leaves the remaining phones locked. Otherwise, the orange route reads more replies and eventually falls back to verify-all. Both routes reach exactly the same match; only the number of opened conversations changes.

We evaluate the certificate on the fixed sample pools and PRM-weighted aggregation previously used by Sensitivity-Aware Verification Allocation (SAVA), a count-only gate that either returns plurality without PRM calls or runs verify-all. All sequential variants reproduce verify-all on every applicable benchmark–k configuration. The most token-efficient evaluated policy scores the unweighted plurality group first and checks the certificate after every trace. It reduces PRM calls by 54.8% and estimated end-to-end tokens by 29.4% under the audited aggregate accounting. Savings are not limited to easy or unanimous items: on AIME24 with $k{=}32$, the exact policy saves 38.5% end-to-end tokens even though the empirical SAVA gate sends every item to full verification. A matched-pool audit across three additional generators confirms answer-level exactness and substantial savings (Appendix F).

Conceptually, this reframes verifier efficiency from a prediction problem to a certification problem. PRM computation is removed only when the item-level output of the fully evaluated selector is provably invariant to all unseen scores.

Contributions.

- **PRM verification as bounded-access certification.** We treat each PRM evaluation as an expensive access to a bounded trace weight and derive an anytime certificate for answer-group weighted plurality. Every early return is guaranteed to match the verify-all answer item by item.
- **An anytime generalization of exact unanimity.** The algorithm extends the zero-call unanimity case to arbitrary intermediate verification budgets. It explicitly handles invalid traces, deterministic ties, reveal order, and batched serving, requires no calibrated stopping threshold, and retains verify-all as an exact fallback.
- **A reproducible cost and exactness audit.** All fully evaluated policies have zero mismatches on the same $6{,}628$ item–configuration cells. Batch-one certification reduces PRM calls by 54.8%, estimated PRM tokens by 54.6%, and estimated end-to-end tokens by 29.4%. A three-generator extension adds $4{,}788$ matched cells, also with zero mismatches, and released provenance records.

Position relative to prior efficiency methods. Recent methods reduce test-time compute at different stages. Adaptive generation changes how many traces are produced, search methods change which paths are explored, and decoding methods reduce per-token cost. We instead begin with a fixed candidate pool and treat PRM scores as selectively accessible quantities. The resulting

certificate eliminates verifier calls while preserving the verify-all decision exactly. Because these approaches optimize different resources, their reported reductions are not directly comparable. Appendix Figure 10 and Table 8 provide a stage-aware overview and source-specific quantitative context.

2 SETUP

Pool and benchmarks. We use fixed candidate pools generated by Qwen2.5-7B-Instruct (Yang et al., 2024). The seven benchmarks are MATH500 (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), AIME24, AMC23, Minerva Math (Lewkowycz et al., 2022), OlympiadBench (He et al., 2024), and College Math 500. Generation uses temperature 0.7, top-$p = 0.95$, a 2048-token cap, and n=32 samples per item. We evaluate nested prefixes of size $k \in \{4, 8, 16, 32\}$. Each benchmark has a fixed development/test split, and all reported performance and cost results use only test items.

The verifier. Qwen2.5-Math-PRM-7B (Yang et al., 2024) scores each completion. One PRM call denotes scoring one complete trace, regardless of how several traces are batched by a serving system. The reference selector groups traces by normalized final answer and applies weighted plurality: it sums the transformed scores s_i^β within each group and returns the largest sum. The PRM summary and exponent β are selected on development data separately for each benchmark–k pair. Token accounting uses recorded generation lengths and a step-weighted estimate of PRM tokens calibrated to the available PRM metadata (Appendix C). For a method m, end-to-end cost is the fixed generation cost of all k traces plus the estimated PRM-token cost of exactly the traces scored by m. We report PRM calls, PRM tokens, and end-to-end tokens separately.

PRM score distribution and β. The PRM scores are strongly concentrated near the endpoints. On MATH500, for example, $p_{25} = 0.43$ and $p_{50} = 0.976$. Development selection chooses the grid maximum $\beta = 8$ for 38% of configurations. At this exponent, a score of 0.99 contributes approximately 0.92, whereas $0.1^8 = 10^{-8}$; the transformation therefore suppresses low-scoring traces while retaining high-scoring ones.

Answer and score conventions. Let $\mathcal{A}$ denote the valid normalized answer groups eligible to win. Each unparseable completion receives its own bookkeeping bucket, matching the reference pipeline, but is non-voting and excluded from $\mathcal{A}$. Such a trace cannot affect an eligible group's weight or bounds. It nevertheless remains in the fixed pool and reveal order and consumes one PRM call if scored. For the cached evaluation arrays, PRM NaNs are median-imputed within the full question (or set to 0.5 if all scores are missing), and scores are clipped to $[10^{-6}, 1]$ before either policy is run. This preprocessing defines the fixed verify-all target used in the audit; an online deployment must instead make its scoring oracle return a bounded score or a predeclared fallback that does not depend on unseen scores. Eligible-answer ties are broken alphabetically. Correctness is read from the pool's stored sample labels rather than re-grading outputs.

Dev-test split. The existing aggregation pipeline selects the PRM summary and exponent β on development data for each benchmark–k pair. These choices define the verify-all target and are held fixed on the test set. Sequential certification introduces no learned threshold or fitted parameter; batch size and reveal order are operational choices evaluated uniformly.

3 EXACT SEQUENTIAL CERTIFICATION

We first specify the reference decision and then derive bounds that remain valid when only a subset of PRM scores is known. For an item with traces $i = 1, \ldots, k$, let $a_i \in \mathcal{A} \cup \{\perp\}$ be trace i's prepared answer label, where $\perp$ denotes a non-voting invalid extraction, and let $s_i \in [10^{-6}, 1]$ be its prepared PRM score. The selected exponent satisfies $\beta > 0$. If $\mathcal{A}$ is empty, both methods return the reference pipeline's no-valid-answer outcome; henceforth assume $\mathcal{A} \neq \varnothing$. The verify-all selector returns

$$a^* = \arg\max_{a \in \mathcal{A}} W(a), \qquad W(a) = \sum_{i: a_i = a} s_i^\beta, \tag{1}$$

with deterministic alphabetical tie breaking. Importantly, the answer labels $a_{1:k}$—and therefore the complete answer partition—are known before any PRM call; only the scores $s_{1:k}$ are initially hidden.

Sequential certification reveals scores according to a chosen order. After the subset $S \subseteq \{1, \dots, k\}$ has been scored, define

$$L_a(S) = \sum_{i \in S : a_i = a} s_i^\beta, \qquad U_a(S) = L_a(S) + \big|\{i \notin S : a_i = a\}\big|. \tag{2}$$

Here, $L_a(S)$ is the weight already observed for group a. Each of its unscored traces can contribute at most one, so $U_a(S)$ adds one unit for each such trace. Because $0 < s_i^\beta \leq 1$, the final weight satisfies $L_a(S) \leq W(a) \leq U_a(S)$ for every $a \in \mathcal{A}$. We use the standard convention that the maximum over an empty challenger set is $-\infty$.

General bounded-weight form. The argument does not depend on the power transform. Suppose trace i contributes a nonnegative weight $g_i(s_i)$ with a known finite upper bound B_i. Then

$$L_a(S) = \sum_{i \in S : a_i = a} g_i(s_i), \qquad U_a(S) = L_a(S) + \sum_{i \notin S : a_i = a} B_i \tag{3}$$

remains valid. Equation 2 is the special case $g_i(s) = s^\beta$ and $B_i = 1$. Thus trace-specific bounds can accommodate known clipping ranges or heterogeneous verifier scales without changing the certificate.

Proposition 1 (Exact stopping certificate)**.** *If an answer $w \in \mathcal{A}$ satisfies*

$$L_w(S) > \max_{a \in \mathcal{A} \setminus \{w\}} U_a(S), \tag{4}$$

then w is the unique verify-all winner, regardless of the unseen PRM scores.

Proof. For every challenger $a \in \mathcal{A} \setminus \{w\}$, Equation 2 gives $W(w) \geq L_w(S) > U_a(S) \geq W(a)$. Hence $W(w) > W(a)$ for every challenger. □

Lemma 1 (Anytime equivalence)**.** *For any scoring order and positive batch size, Algorithm 1 returns the same answer as verify-all.*

Proof. An early return satisfies Proposition 1. Otherwise the algorithm reveals all k scores and evaluates Equation 1 with the reference tie rule. These two cases exhaust every execution. □

The strict inequality in Equation 4 deliberately leaves equality uncertified. At equality, unseen scores may produce a final tie whose winner depends on the alphabetical rule. A tie-aware certificate can be slightly tighter (Appendix A), but the uniform strict condition is simpler and independent of group storage or traversal order.

Incremental state. For each eligible answer group, the implementation stores its observed weight L_a and its remaining unscored count r_a. Initially $L_a = 0$ and $r_a = |\{i : a_i = a\}|$, so $U_a = r_a$. Scoring a voting trace i adds s_i^β to L_{a_i} and decrements r_{a_i}; the corresponding upper bound is then $U_{a_i} = L_{a_i} + r_{a_i}$. A trace labeled $\perp$ changes no group state but still consumes a PRM call if revealed. Initialization and construction of the order cost $O(k)$, a batch update costs $O(b)$, and a direct certificate check costs $O(A)$ for $A = |\mathcal{A}|$. The group state uses $O(A)$ memory, or $O(k + A)$ including the explicit reveal order. No auxiliary model is required.

Ordering and batching. The default order first scores traces supporting the unweighted plurality leader, with alphabetical tie breaking between groups and stable input order within each group. This strategy rapidly increases the most plausible winner's lower bound; after that group is exhausted, it traverses the remaining groups alphabetically and places invalid traces last. Reveal order affects only stopping time. Any fixed order—or any adaptive order based only on labels and already revealed scores—remains exact because the certificate itself uses valid bounds for all unseen contributions. We evaluate batch sizes $b \in \{1, 2, 4, 8\}$ and a deterministic random-order baseline. Batch size one checks the certificate after every PRM call and maximizes token savings; larger batches reduce the number of sequential synchronization rounds.

Algorithm 1 Exact sequential certification

Require: labels $a_{1:k} \in \mathcal{A} \cup \{\perp\}$, eligible groups $\mathcal{A}$, PRM oracle, exponent $\beta > 0$, batch size b

if $\mathcal{A} = \varnothing$ **then**
  return the reference no-valid-answer outcome
end if
if $|\mathcal{A}| = 1$ **then**
  return the sole eligible answer with zero PRM calls
end if
Choose the plurality group using alphabetical tie breaking
Order its traces first; traverse remaining groups in fixed alphabetical order, preserving input order within each group; place $\perp$ traces last
$S \leftarrow \varnothing$
while $|S| < k$ **do**
  Score the next $\min(b, k - |S|)$ traces and add them to S
  Update $L_a(S)$ and $U_a(S)$ for every affected group
  if $L_w(S) > \max_{a \in \mathcal{A} \setminus \{w\}} U_a(S)$ for some w **then**
    return w
  end if
end while
return the alphabetical-tie verify-all argmax

Parallel service. The certificate is checked between batches, not inside a PRM forward pass. Thus b is a systems choice rather than a statistical parameter. A server can use b=1 when token cost dominates, a larger fixed batch when GPU occupancy dominates, or an increasing schedule that begins cautiously and widens only when bounds remain loose. Every such schedule targets the same verify-all answer by the anytime-equivalence lemma.

Unanimity as a special case. If there is at most one eligible answer group, the result is known before any score is revealed. An empty set uses the reference no-valid-answer outcome; a singleton returns its sole eligible answer, even if additional traces are invalid and non-voting. Literal unanimity is the most common singleton case. Consequently, SAVA's exact $R = 0$ unanimity branch is contained in the first branches of Algorithm 1. More generally, the certificate may stop after any $R \in \{1, \ldots, k\}$ calls, including on contested items for which answer counts alone cannot justify a safe decision.

4 SAVA AS AN APPROXIMATE BASELINE

SAVA uses answer counts alone to choose between returning plurality with zero PRM calls and running verify-all. Its strict-unanimity branch is exact, whereas the broader near-unanimity rule selects a vote statistic and threshold on development data. SAVA is therefore a useful zero-or-all efficiency baseline, but its calibrated zero-call decisions may differ from verify-all on test items. Sequential certification instead reveals PRM scores until it proves the verify-all winner, requires no stopping-threshold calibration, and can stop at any intermediate budget.

5 EXACTNESS AND COST RESULTS

We evaluate seven benchmarks at four trace budgets, yielding 28 configurations and $6{,}628$ test item–configuration cells. Each policy with full configuration coverage (leader-first $b \in \{1, 2, 4\}$ and random-order $b = 1$) has exactness rate 1.0 and zero answer mismatches on all $6{,}628$ cells; $b = 8$ likewise has zero mismatches on its $4{,}971$ applicable cells with $k \geq 8$. Consequently, all sequential policies have exactly the same item-level predictions and aggregate accuracy as verify-all. The remaining comparisons therefore concern cost.

Low-accuracy benchmarks can also certify early. SAVA makes no savings on AIME24 and OlympiadBench at k=32, because its dev-selected threshold sends every item to full verification. The exact certificate still saves 38.5% and 22.9% end-to-end tokens, respectively: it can stop af-

Table 1: Complete result ledger. "Leader" is batch-one leader-first certification and "random" uses the same exact certificate with a deterministic random reveal order. "Calls," "PRM save," and "leader e2e" report leader-first batch one. Accuracy is the verify-all accuracy and is reproduced by both exact policies. All savings are percentages relative to verify-all; SAVA is an approximate zero-or-all baseline. Bold marks the primary leader-first operating point.

Benchmark	k	v-all acc.	calls ↓	PRM save ↑	**leader e2e ↑**	random e2e ↑	SAVA e2e ↑
MATH500	4	82.10	1.23	69.3	**37.5**	37.9	0.0
MATH500	8	83.66	2.46	69.3	**37.5**	38.0	29.9
MATH500	16	83.66	4.47	72.0	**38.9**	38.8	25.9
MATH500	32	85.21	8.76	72.6	**39.3**	38.9	30.6
GSM8K	4	94.65	0.37	90.6	**51.3**	51.4	0.0
GSM8K	8	95.57	0.72	91.0	**51.5**	51.5	0.0
GSM8K	16	95.87	1.48	90.7	**51.5**	51.6	43.8
GSM8K	32	95.72	2.38	92.6	**52.6**	52.5	41.4
AIME24	4	18.75	3.69	7.8	**4.3**	4.3	0.0
AIME24	8	12.50	7.19	10.2	**5.6**	5.6	0.0
AIME24	16	18.75	14.62	8.6	**4.7**	4.9	0.0
AIME24	32	18.75	9.56	70.1	**38.5**	22.5	0.0
AMC23	4	53.33	2.27	43.3	**22.2**	20.5	17.1
AMC23	8	60.00	5.27	34.2	**17.5**	20.5	0.0
AMC23	16	66.67	9.67	39.6	**20.6**	21.7	20.8
AMC23	32	66.67	13.60	57.5	**30.2**	29.2	31.5
Minerva	4	33.85	2.42	39.6	**21.7**	20.1	16.4
Minerva	8	36.15	4.82	39.7	**21.8**	20.2	24.0
Minerva	16	34.62	9.02	43.7	**23.9**	21.6	16.5
Minerva	32	34.62	17.01	46.9	**25.7**	22.8	38.4
Olympiad	4	41.11	2.90	27.4	**14.6**	16.9	12.6
Olympiad	8	42.57	4.67	41.7	**22.2**	19.7	9.0
Olympiad	16	45.19	8.29	48.2	**25.6**	22.6	16.0
Olympiad	32	45.77	18.26	42.9	**22.9**	21.2	0.0
College	4	43.80	1.29	67.8	**35.3**	32.4	27.1
College	8	44.21	2.13	73.4	**38.3**	34.3	34.9
College	16	43.80	5.19	67.5	**35.3**	32.8	34.8
College	32	44.21	7.51	76.5	**40.0**	35.4	45.1

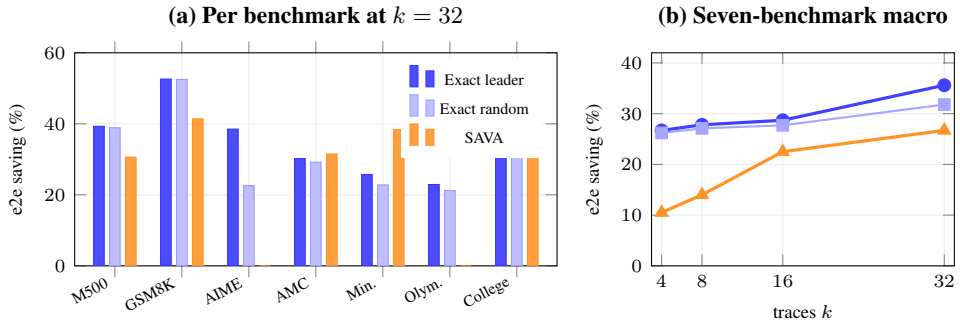


Figure 2: End-to-end token reduction under the evaluation accounting. **(a)** At k=32, exact leader-first certification saves tokens on all seven benchmarks; random ordering is also exact, whereas SAVA is an approximate zero-or-all gate. **(b)** Leader-first has the largest macro-average saving at every k. Both exact policies have zero answer mismatches with verify-all.

ter observing scores even when answer counts alone are insufficient. SAVA is slightly cheaper on AMC23, Minerva Math, and College Math, but those savings come from an empirical zero-call decision rather than a guarantee of matching verify-all.

Table 1 and Figure 2 show consistently large savings on GSM8K and MATH500. AIME24 becomes substantially cheaper only at k=32, where the observed scores separate answer groups that remain contested by counts alone. Savings need not be monotone in k: each budget defines a different candidate prefix and may use a different development-selected PRM summary and exponent.

Aggregate accounting. The primary validator first averages absolute costs within each configuration, then averages these configuration-level costs, and finally forms the reduction ratio. Under

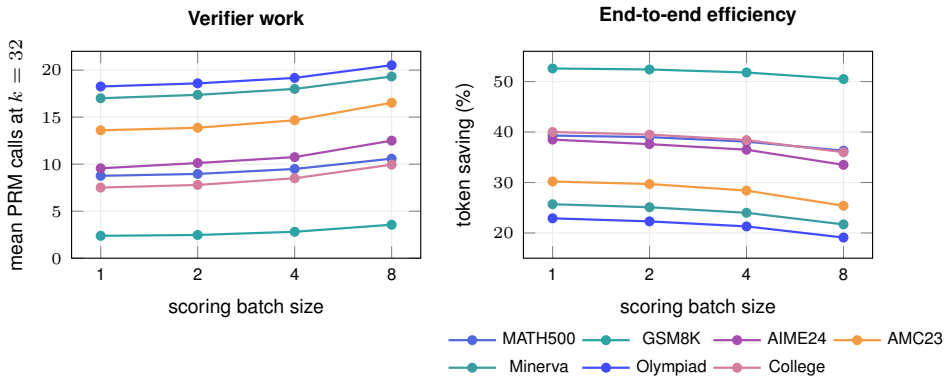


Figure 3: Batch-size trade-off at k=32. Larger batches increase the mean number of revealed PRM scores (left) and therefore reduce end-to-end savings (right). Every curve remains exactly equal to verify-all at the answer level; batch size changes only how often the certificate is checked.

Table 2: Batch-size sensitivity at k=32. Each entry is "end-to-end saving (%) / mean PRM calls." Larger batches reduce synchronization rounds but may reveal extra scores before the next certificate check. The bold header marks the primary token-minimizing setting.

Benchmark	$b = 1$	$b = 2$	$b = 4$	$b = 8$
MATH500	**39.3 / 8.76**	39.0 / 8.96	38.1 / 9.49	36.3 / 10.58
GSM8K	**52.6 / 2.38**	52.4 / 2.47	51.8 / 2.81	50.5 / 3.56
AIME24	**38.5 / 9.56**	37.6 / 10.12	36.5 / 10.75	33.5 / 12.50
AMC23	**30.2 / 13.60**	29.7 / 13.87	28.4 / 14.67	25.4 / 16.53
Minerva Math	**25.7 / 17.01**	25.1 / 17.37	24.0 / 18.00	21.7 / 19.32
OlympiadBench	**22.9 / 18.26**	22.3 / 18.59	21.3 / 19.17	19.1 / 20.52
College Math	**40.0 / 7.51**	39.5 / 7.79	38.4 / 8.50	36.0 / 9.95

this audited aggregate, batch one reduces PRM calls by 54.8%, estimated PRM tokens by 54.6%, and estimated end-to-end tokens by 29.4%. Averaging the 28 cell-wise relative reductions instead gives 54.8%, 54.8%, and 29.7%. Pooling item–configuration costs gives 71.3%, 64.5%, and 35.0% because larger benchmarks and larger k receive more weight. We use the validator's 29.4% value in the abstract and expose all three conventions here rather than treating them as interchangeable.

Ordering. Under the unweighted mean of the 28 cell-wise reductions, random-order certification saves 28.2% end-to-end tokens. Leader-first improves this to 29.7% overall and has a larger advantage at k=32 (35.6% versus 31.8%). The heuristic is not uniformly better in every small cell, so we do not claim pointwise dominance.

Batch size. Checking after every score gives the greatest saving. Macro-average end-to-end savings are 29.7%, 29.1%, 27.2%, and 25.1% for batch sizes $1, 2, 4, 8$, respectively (batch eight is evaluated only when $k \geq 8$). Larger batches may nevertheless be preferable when PRM calls are served in parallel and synchronization latency dominates token cost.

6 Termination Behavior and Deployment

Where does stopping occur? Pooling test items across benchmarks within each k, the batch-one method stops at zero calls on 59.6%, 53.3%, 47.2%, and 43.0% of items for $k = 4, 8, 16, 32$, respectively. These items have at most one eligible answer group; literal unanimity is the most common case. Only 27.2%, 18.8%, 14.5%, and 10.6%, respectively, reach full verification. At $k = 32$ the median item needs only two PRM calls, and the 75th percentile needs fourteen. Intermediate certification accounts for 46.4% of all items, or 81.4% of items outside the zero-call set, so the saving is not explained by the zero-call shortcut alone.

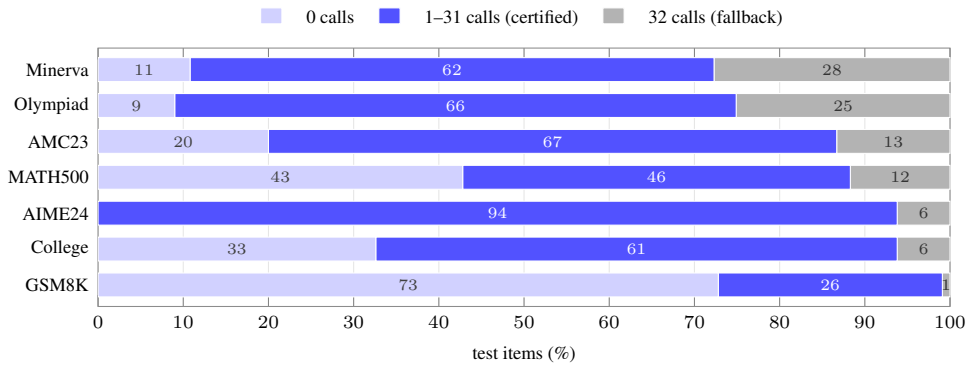


Figure 4: Stopping regimes for leader-first batch-one certification at k=32. The highlighted middle segment is the additional regime enabled by sequential certification beyond the zero-call shortcut. Even on AIME24, OlympiadBench, and Minerva Math, most items certify before the 32-call verify-all fallback. Labels are percentages rounded to the nearest integer.

Figure 4 shows that stopping behavior differs sharply across benchmarks. GSM8K is dominated by zero- and low-call cases: 72.8% of items use the zero-call branch, and only 0.9% require all 32 scores. AIME24 has no zero-call items, yet 93.8% certify before full verification. Minerva Math and OlympiadBench have the longest tails: 27.7% and 25.1% of their items reach verify-all, respectively, while most of the remainder stop at an intermediate budget. Thus vote concentration and separation in observed PRM weight provide two distinct sources of savings.

Computation and memory. Answer grouping and initialization cost $O(k)$ time and $O(k + A)$ memory for A distinct answer groups. Revealing a batch updates only its groups; checking the certificate requires a maximum over the A upper bounds. Even a direct $O(A)$ check after every sample is negligible relative to a PRM forward pass. No model state, learned controller, or auxiliary network is required.

Serving considerations. Batch one is optimal for token count but induces the most synchronization points. A practical server may score a small initial leader batch, check the certificate, and grow later batches when bounds remain loose. Such scheduling changes efficiency but not correctness. Multiple items can also be interleaved: as soon as one item certifies, its unused PRM slots can be reassigned to uncertified items in the next service batch.

7 Evaluation Robustness and Caveats

The evaluation preserves each test item's original row index in the full pool when retrieving PRM scores and step counts. It also reproduces the reference pipeline's NaN imputation, clipping, invalid-answer handling, correctness labels, power transform, and alphabetical tie rule. As an external alignment check, each k=32 verify-all accuracy matches the independently generated aggregation result in Table 1 within 5×10^{-4}.

Exactness is checked at the answer level, not inferred from equal aggregate accuracy. Across the applicable item–configuration records, every leader-first batch size and the random-order variant has zero winner mismatches. A separate randomized synthetic test over $20{,}000$ score/label instances also found no certificate violation. Appendix A gives the proof and implementation invariants; Table 1 reports every one of the 28 configurations, and Appendix C details token accounting.

We additionally test three 7B generator pools on matched subsets of four benchmarks. Across $4{,}788$ generator–item–budget cells and $478{,}800$ reveal-order trials, batch-one certification again has zero answer-ID mismatches, including under both pre-specified GPQA aggregations. At k=32, it saves 38.8–96.3% of PRM tokens and 19.4–52.7% of combined generation-plus-verification tokens. Table 6 reports every matched generator–benchmark cell; Appendix F documents the frozen configurations and outcome-independent removal of one overlength GPQA item.

A reduced fixed-subset diagnostic with the mechanism-distinct generative PRM RLHFlow/Llama3.1-8B-PRM-Mistral-Data likewise has zero winner mismatches over 768 item–k cells. With 64 items per benchmark and untuned $\beta = 1$, leader-first uses $4.755/32$ calls at k=32 (an 85.14% saving), versus $7.932/32$ (75.21%) for the preregistered random order; Appendix F.1 gives the deliberately limited audit.

8 Related Work

Scaling and adaptive generation. Adaptive generation stops or allocates samples from answer stability, confidence, or difficulty (Aggarwal et al., 2023; Li et al., 2024; Wan et al., 2025; Huang et al., 2026; Aghazadeh et al., 2025). Most closely, MARS probes partially generated traces and combines learned answer-switch probabilities with an adversarial bound to recover the eventual self-consistency vote with high probability (Chen et al., 2026b). Our method begins after all traces and answer labels are fixed and gives deterministic agreement with a specified PRM-weighted selector.

Verifier compute and search. Process reward models support best-of-N and search-based selection (Lightman et al., 2023; Shao et al., 2024). Generative verifiers recast reward modeling as next-token prediction and add a separate verification-compute axis (Zhang et al., 2025); fixed-budget studies then compare spending compute on solutions versus repeated generative verification (Singhi et al., 2025). VG-Search and Solve-Detect-Verify vary verifier use during search (Chen et al., 2025; Zhong et al., 2026), whereas our target is one fixed weighted-plurality selector. Faster decoding such as STAND (Song et al., 2025) is complementary. Classical threshold algorithms study aggregation under partial access (Fagin et al., 2003). We instantiate this principle for a different computational object: answer-group weighted-plurality selection in which every score access is a PRM inference. The resulting contract includes the answer partition, invalid traces, deterministic ties, and an exact verify-all fallback.

9 Limitations and Discussion

Answer counts guide the reveal order, whereas observed PRM weights determine whether stopping is safe. This distinction explains the savings on AIME24 and OlympiadBench even when count-only SAVA verifies every item. The guarantee is relative to a fully specified verify-all pipeline; it does not imply that the verifier's answer is correct. It also requires a known finite upper bound on each unseen trace contribution. Empirically, the main study is concentrated on one PRM family and mathematical tasks. The transfer audit adds three generators but uses smaller subsets and one exploratory non-mathematical task, while the second-PRM audit covers only three benchmarks with 64 fixed items each and untuned $\beta = 1$. These experiments do not establish universal transfer across verifiers, generators, or domains. Finally, token reductions are not latency measurements: batch-one scoring introduces synchronization, and realized speedups depend on batching, hardware utilization, and serving load. Evaluating additional verifiers and asynchronous schedules remains future work.

10 Conclusion

We formulate PRM verification as bounded-access certification for answer-group weighted plurality. Exact sequential certification removes score accesses once valid bounds prove that no unseen contribution can change the verify-all winner. The method is training-free, supports arbitrary reveal orders and batch sizes, and retains the original selector as an exact fallback. Leader-first batch-one scoring reduces PRM calls by 54.8% and estimated end-to-end tokens by 29.4% with zero answer mismatches; at k=32, benchmark-level end-to-end savings range from 22.9% to 52.6%. Audits on three additional generators likewise find zero mismatches and 19.4–52.7% savings. Thus verification is eliminated through an item-level certificate, rather than a calibrated prediction that further scores are unnecessary.

Reproducibility Statement

The appendix specifies the proof, evaluation pipeline, per-configuration settings, cost accounting, exactness checks, and limitations. The supplementary material provides the evaluation code, result files, validation logs, and checksums needed to reproduce the reported tables and figures.

AI Use Statement

Generative AI assistants were used for language editing, LaTeX debugging, figure and table presentation, exploratory data analysis, and suggestions on experimental code structure. The authors inspected the resulting code and artifacts and independently checked the experimental design, derivations, citations, scientific claims, and reported numerical values. The authors remain responsible for the manuscript and its results.

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Ehsan Aghazadeh, Ahmad Ghasemi, Hedyeh Beyhaghi, and Hossein Pishro-Nik. CGES: Confidence-guided early stopping for efficient and accurate self-consistency. *arXiv preprint arXiv:2511.02603*, 2025.

Hao Mark Chen, Guanxi Lu, Yasuyuki Okoshi, Zhiwen Mo, Masato Motomura, and Hongxiang Fan. Rethinking optimal verification granularity for compute-efficient test-time scaling. In *Advances in Neural Information Processing Systems*, 2025.

Kejia Chen, Jiawen Zhang, Yihong Wu, Kewei Gao, Jian Lou, Zunlei Feng, Mingli Song, and Ruoxi Jia. Confidence-aware alignment makes reasoning LLMs more reliable. *arXiv preprint arXiv:2605.07353*, 2026a.

Wenbo Chen, Puheng Li, Mengyang Liu, Weijie Su, and Tianpei Xie. Mars: Margin-adversarial risk-controlled stopping for parallel llm test-time scaling. *arXiv preprint arXiv:2606.12935*, 2026b.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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

Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 102–113, 2003.

Austin Feng, Marius Alonso, and Ambroise Odonnat. Optimal self-consistency for efficient reasoning with large language models. *arXiv preprint arXiv:2511.12309*, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024. URL https://arxiv.org/abs/2402.14008.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. DeepMath-103K: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks Track*, 2021. arXiv:2103.03874.

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *arXiv preprint arXiv:2504.07086*, 2025.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. CaTS: Calibrated test-time scaling for efficient LLM inference. In *International Conference on Learning Representations (ICLR)*, 2026.

Junseok Kim, Nakyeong Yang, Kyungmin Min, and Kyomin Jung. Reliability-aware adaptive self-consistency for efficient sampling in LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21575–21590, 2026.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022. URL https://arxiv.org/abs/2206.14858.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *International Conference on Learning Representations (ICLR)*, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. General-reasoner: Advancing LLM reasoning across all domains. *arXiv preprint arXiv:2505.14652*, 2025.

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. *arXiv preprint arXiv:2501.19393*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

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. *arXiv preprint arXiv:2402.03300*, 2024.

Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. When to solve, when to verify: Compute-optimal problem solving and generative verification for LLM reasoning. *arXiv preprint arXiv:2504.01005*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters for reasoning. In *International Conference on Learning Representations*, 2025.

Woomin Song, Saket Dingliwal, Sai Muralidhar Jayanthi, Bhavana Ganesh, Jinwoo Shin, Aram Galstyan, and Sravan Babu Bodapati. Accelerated test-time scaling with model-free speculative sampling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In *Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2025. arXiv:2408.17017.

Hong Wang, Zhezheng Hao, Jian Luo, Chenxing Wei, Yao Shu, Lei Liu, Qiang Lin, Hande Dong, and Jiawei Chen. Scheduling your LLM reinforcement learning with reasoning trees. *arXiv preprint arXiv:2510.24832*, 2026.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2203.11171.

Wei Xiong, Hanze Dong, Chenlu Ye, Han Wang, Nan Zhong, Heng Jiang, Nanqing Zhao, Xuefeng Ji, Chao Yan, Tao Jiang, Zhen Zhang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for RLHF under KL-constraint, 2024. URL https://huggingface.co/RLHFlow/Llama3.1-8B-PRM-Mistral-Data. Official model card and scoring implementation.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *International Conference on Learning Representations*, 2025.

Jianyuan Zhong, Zeju Li, Zhijian Xu, Xiangyu Wen, Kezhi Li, and Qiang Xu. Solve-detect-verify: Inference-time scaling with flexible generative verifier. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 47377–47401, 2026.

APPENDIX

This appendix separates the mathematical guarantee from the empirical cost analysis. We first prove the certificate and document the implementation audit (Appendices A–B). We then define the token accounting and analyze all 28 benchmark–k cells in the main-paper ledger (Appendices C–D). Appendix E resolves when certification occurs and how reveal order and batch size affect cost. The remaining sections report cross-generator transfer, a reduced second-PRM diagnostic, external context, reproducibility details, and the exact scope of the guarantee.

A FORMAL GUARANTEE AND EXACTNESS AUDIT

Let $\mathcal{A}$ be the eligible valid-answer groups defined in the main text. For $a \in \mathcal{A}$, let $I_a = \{i : a_i = a\}$, and let S denote the traces already scored. Non-voting invalid buckets lie outside $\mathcal{A}$. The implementation maintains

$$L_a(S) = \sum_{i \in S \cap I_a} s_i^\beta, \qquad U_a(S) = L_a(S) + |I_a \setminus S|. \tag{5}$$

Because prepared scores lie in $[10^{-6}, 1]$ and $\beta > 0$, every unseen contribution s_i^β lies in $(0, 1]$. Therefore $L_a(S) \le W(a) \le U_a(S)$ for every eligible group.

If an implementation declares a score floor s_{lo} in advance, it may soundly tighten the lower bound to

$$L_a(S) = \sum_{i \in S \cap I_a} s_i^\beta + |I_a \setminus S| s_{\mathrm{lo}}^\beta.$$

The reported implementation could use $s_{\mathrm{lo}} = 10^{-6}$. An exhaustive comparison over $4{,}608$ records found no difference between this tight form and Equation 5 in winner, calls, or stopping branch, so we retain the simpler bound in the core presentation and reported policies.

Sound early stopping. Suppose the algorithm returns w before all traces are scored. Its stopping condition gives $L_w(S) > U_a(S)$ for every $a \neq w$. Therefore

$$W(w) \ge L_w(S) > U_a(S) \ge W(a),$$

so w is the unique verify-all winner for every possible assignment of the unseen scores. The argument uses only the bounds, not the scoring order or batch size.

Exact fallback. If no certificate fires, then $S = \{1, \dots, k\}$ and $L_a(S) = U_a(S) = W(a)$ for every eligible answer group. The implementation applies the same alphabetical tie rule as the reference verify-all pipeline. Early certification and exact fallback cover every execution, proving answer-level equivalence.

Tie-aware refinement. The reported rule uses a strict inequality against every challenger. A slightly tighter certificate could exploit the known tie priority: strict inequality is needed against groups that precede w alphabetically, whereas non-strict inequality suffices against groups that follow it. No reported result uses this refinement; the uniform strict rule is simpler and conservative.

General bounded weights. For a nonnegative trace weight $g_i(s_i)$ with known upper bound B_i, replace Equation 5 by

$$L_a(S) = \sum_{i \in S \cap I_a} g_i(s_i), \qquad U_a(S) = L_a(S) + \sum_{i \in I_a \setminus S} B_i.$$

The proof is unchanged. The experiments use $g_i(s) = s^\beta$ and $B_i = 1$.

Table 3: Answer-level exactness checks. Batch size eight is applicable only for $k \geq 8$.

Policy	configurations	item–configuration cells	mismatches
Leader first, $b = 1$	28	6,628	0
Leader first, $b = 2$	28	6,628	0
Leader first, $b = 4$	28	6,628	0
Leader first, $b = 8$	21	4,971	0
Deterministic random order, $b = 1$	28	6,628	0

Table 3 reports the item-level audit for every evaluated policy. The exactness comparison is performed on the returned answer ID, not inferred from equal aggregate accuracy. A separate randomized synthetic test over 20,000 score/label instances also produced no certificate violation.

B Reference Evaluation Pipeline

Pool and PRM alignment. PRM arrays follow the row order of the complete generated pool. Filtering the pool to the test split before indexing would compress row numbers and attach scores to the wrong questions. Our evaluation instead retains every record's original pool index, filters only the evaluation iterator, and uses the original index to retrieve PRM scores and step counts.

Answer groups. The generated pool already contains extracted answers and per-sample correctness labels. Valid answers are normalized by the reference aggregation code. Each invalid extraction receives its own non-voting bookkeeping bucket rather than being merged into an artificial answer group. These buckets are excluded from $\mathcal{A}$, cannot win, and do not change any eligible group's bounds. They nevertheless remain positions in the complete trace pool and reveal order, and an invalid trace consumes one PRM call if scored. Answer keys are sorted once to make every tie deterministic.

Score preparation. For the cached evaluation arrays, missing PRM scores are replaced by the median of the question's available scores; if every score is missing, all are set to 0.5. Scores are then clipped to $[10^{-6}, 1]$ before either policy is run. This offline preparation defines the audited verify-all target; it is not an online query to unrevealed scores. Verify-all uses the power weight s_i^{β}, with the PRM summary and β inherited from the existing development-set selection for that benchmark–k pair. Sequential certification adds no threshold or fitted parameter.

Correctness and external checks. Correctness is read from the stored pool labels rather than re-grading generated text. At k=32, verify-all reproduces all seven independently generated aggregation accuracies within 5×10^{-4}. The measured values are 85.21, 95.72, 18.75, 66.67, 34.62, 45.77, and 44.21% for MATH500, GSM8K, AIME24, AMC23, Minerva Math, OlympiadBench, and College Math.

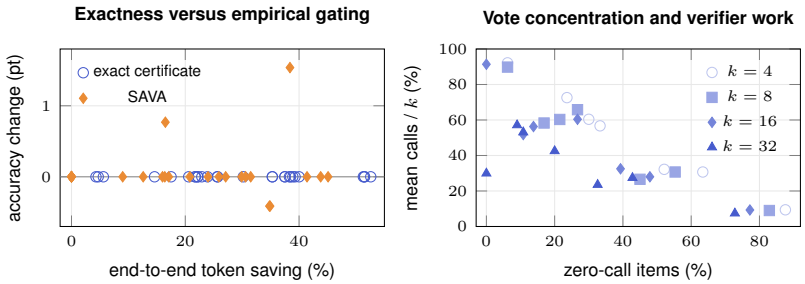


Figure 5: Two diagnostic views over all 28 configurations. **Left:** hollow blue markers are exact sequential certification and therefore lie at zero accuracy change; orange markers are SAVA and may differ from verify-all. **Right:** zero-call rate versus normalized verifier work; marker shape encodes k and marker area increases with the full-verification rate.

Figure 5 separates answer-level exactness from the empirical relationship between vote concentration and verifier work. Most SAVA cells also happen to have zero aggregate accuracy change, but this is an empirical observation rather than an answer-level guarantee. Its nonzero cells range from -0.41 points on College Math to $+1.54$ points on Minerva Math. Exact certification stays on the zero line for every configuration. The right panel shows the expected negative relationship between unanimity and PRM work, while AIME24 k=32 demonstrates that zero-call rate is not the whole story: despite no unanimous items, most questions certify after only a small fraction of the 32 scores.

B.1 Per-configuration verifier settings

The PRM summary and exponent are properties of the verify-all target inherited from development selection; they are not tuned for sequential certification. Table 4 records them explicitly so each result row can be reconstructed without consulting hidden defaults.

Table 4: Test-set size and verify-all PRM configuration. Each cell is "summary/β" for the corresponding k.

Benchmark	test n	$k=4$	$k=8$	$k=16$	$k=32$
MATH500	257	min/4	min/8	min/8	min/8
GSM8K	654	min/8	min/8	min/8	min/2
AIME24	16	min/4	min/4	min/8	mean/0.5
AMC23	15	min/0.5	min/8	min/8	min/1
Minerva Math	130	min/4	min/8	min/8	min/8
OlympiadBench	343	min/8	min/1	min/1	min/2
College Math	242	last/8	last/8	min/8	min/1

C Token Accounting

Generation is fixed across methods. For one question,

$$C_{\text{gen}} = \sum_{i=1}^{k} t_i^{\text{gen}}, \tag{6}$$

where t_i^{gen} is the recorded generation length. Per-trace PRM tokens are estimated from the recorded step counts:

$$\widehat{t}_i^{\text{prm}} = \frac{q_i}{k^{-1}\sum_{j=1}^{k} q_j}\,\bar{t}^{\text{prm}}, \tag{7}$$

where q_i is the trace's step count and $\bar{t}^{\text{prm}}$ is the benchmark-level PRM-token metadata. If the mean step count is zero, every trace is assigned $\bar{t}^{\text{prm}}$. For the set S_m scored by method m,

$$C_m = C_{\text{gen}} + \sum_{i\in S_m} \widehat{t}_i^{\text{prm}}, \qquad \text{save}_m = 1 - \frac{C_m}{C_{\text{verify-all}}}. \tag{8}$$

The official validator averages costs over the 28 benchmark–k cells before taking ratios, yielding 54.8% fewer calls, 54.6% fewer PRM tokens, and 29.4% fewer end-to-end tokens for batch one. Averaging the 28 cell-wise relative reductions instead gives 54.8%, 54.8%, and 29.7%. Pooling all item–configuration costs gives 71.3%, 64.5%, and 35.0%. The abstract uses the validator's aggregate convention.

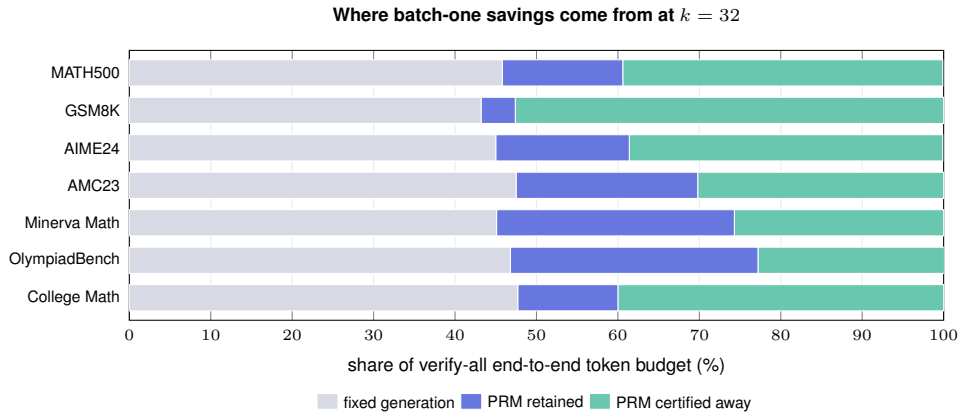


Figure 6: Token-budget decomposition at k=32. Gray is the generation cost shared by every policy, blue is PRM computation still performed by batch-one certification, and green is PRM computation removed by the certificate. The green segment is exactly the end-to-end saving because generation is fixed.

Figure 6 explains why call reduction and end-to-end reduction differ. GSM8K eliminates almost the entire verifier share, but generation still occupies roughly half of the verify-all budget. Minerva Math and OlympiadBench retain more PRM work because their answer groups remain competitive for longer. The certificate never reduces the gray generation segment; it composes with generation-side methods rather than replacing them.

D Ordering and Detailed Regimes

Main-paper Table 1 reports all 28 benchmark–k configurations, including both exact reveal orders and the approximate SAVA baseline. Here we analyze the regimes exposed by that ledger.

Four recurring regimes. The ledger exposes four behaviors that a single aggregate number hides. GSM8K is the easy-certification regime: answer concentration makes almost every remaining PRM call redundant. MATH500 and College Math combine many zero-call items with substantial additional savings on non-unanimous items. Minerva Math and OlympiadBench are long-tail regimes in which a meaningful fraction of questions still requires all 32 scores. AIME24 is the clearest ordering stress test: at k=32, leader-first saves 38.5% end to end, compared with 22.5% for random ordering, even though both return exactly the same answer.

Exactness versus an approximate gate. SAVA sometimes spends less and sometimes more than sequential certification, but its operating point is qualitatively different. Its nonzero accuracy changes on Minerva Math and College Math show that equal aggregate accuracy is not an answer-level certificate. Sequential certification instead treats the verify-all output as fixed and asks how soon bounds prove that it cannot change. The comparison therefore separates two choices: whether the target answer may change, and how efficiently a fixed target can be recovered.

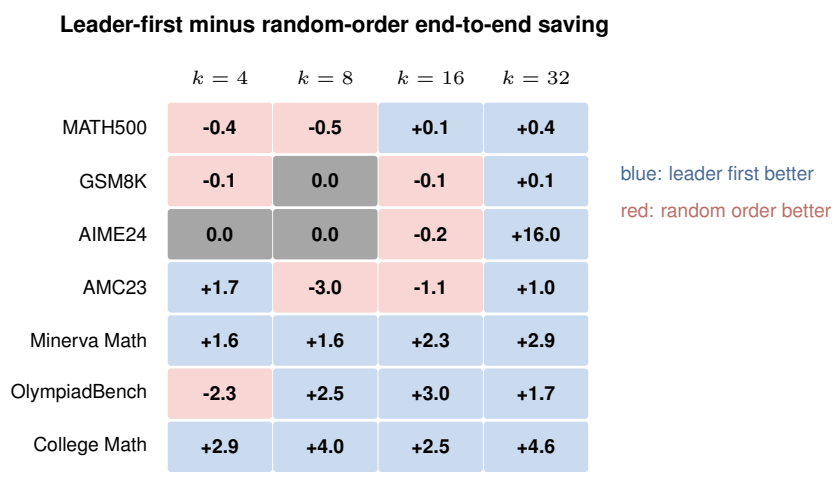


Figure 7: End-to-end saving of leader-first batch one minus random-order batch one. Both methods are exact; the figure isolates scheduling efficiency. Positive cells favor leader-first. The large AIME24 k=32 gain occurs because scoring the plurality group first rapidly builds a certifying lower bound.

Figure 7 isolates the benefit of reveal order while holding the exact certificate fixed. Leader-first is not pointwise optimal. Random ordering is slightly better in some low-k cells, most visibly AMC23 at k=8 and OlympiadBench at k=4. The scheduling heuristic matters most when a dominant answer group exists but its weight must be exposed before the challengers' upper bounds can be ruled out. This yields the 16.0-point advantage on AIME24 at k=32 and consistent gains on College Math.

E Batch Size and Stopping-Time Details

Main-paper Figure 3 visualizes the token–synchronization trade-off induced by delayed certificate checks. The curves are smooth because batch size changes only when the certificate is checked. A certificate that becomes valid after call r is observed at r for $b = 1$ but can be delayed to the end of the enclosing batch for $b > 1$. GSM8K remains inexpensive even at $b = 8$ because unanimity and early separation are common. Minerva Math and OlympiadBench start from higher call counts and lose more absolute verifier work as batches widen. Main-paper Table 2 gives the corresponding numerical values.

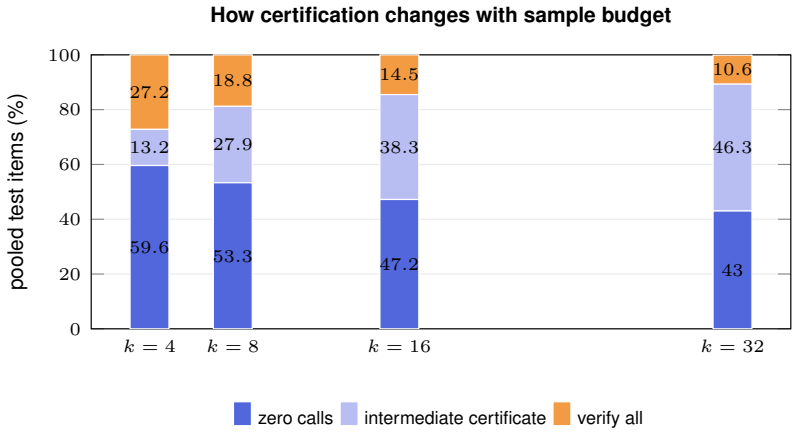


Figure 8: Pooled stopping regimes across all seven benchmarks. Zero-call cases have at most one eligible answer group; literal unanimity is their common special case. Intermediate cases are the additional items made possible by sequential certification. As k grows, the zero-call branch becomes less common but full verification also becomes less common.

Figure 8 shows that the intermediate region grows from 13% at k=4 to 46% at k=32. This is the main distinction from a zero-call shortcut: larger candidate sets more often contain multiple eligible groups, yet they also provide enough scored evidence for one answer group's lower bound to separate from every rival upper bound. Only 11% of pooled k=32 items reach all 32 calls. Table 5 resolves these stopping times by benchmark.

Table 5: Stopping-call distribution for leader-first batch one at k=32. Percentiles are computed over test items.

Benchmark	median	75th pct.	90th pct.	zero calls	all 32
MATH500	2	14	32	42.8%	11.7%
GSM8K	0	2	6	72.8%	0.9%
AIME24	5	7	31	0.0%	6.2%
AMC23	11	22	26	20.0%	13.3%
Minerva Math	17	32	32	10.8%	27.7%
OlympiadBench	22	31	32	9.0%	25.1%
College Math	4	10	26	32.6%	6.2%

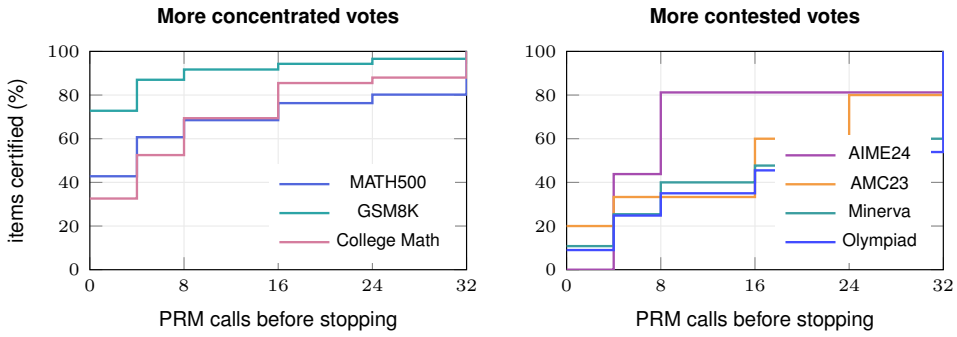


Figure 9: Batch-one stopping CDF at k=32, sampled at six PRM-call budgets. The vertical axis is the fraction of items certified by the stated budget. The left panel contains benchmarks with more concentrated votes; the right panel shows tasks whose answers remain contested for longer.

Figure 9 reveals behavior hidden by mean calls. On GSM8K, 72.8% of items stop at zero and more than 90% stop by eight calls. AIME24 has no zero-call items, but over 80% certify by eight calls at k=32. Minerva Math and OlympiadBench have long tails: roughly one quarter of their items require full verification, while the remainder spread across intermediate budgets. A single global stopping budget would therefore waste verifier calls on easy items and truncate difficult ones; the certificate adapts item by item.

F Cross-Generator Transfer Audit

We test whether the certificate remains useful when the candidate traces come from different reasoning models. The extension uses matched pools from Open-Reasoner-Zero-7B (Hu et al., 2025), DeepMath-Zero-7B (He et al., 2025), and General-Reasoner-7B (Ma et al., 2025). For each generator, we evaluate 100 matched items from MATH500, GSM8K, and OlympiadBench, and 99 from GPQA-Diamond (Rein et al., 2023), with 32 traces per item and $k \in \{4, 8, 16, 32\}$. The MATH500, GSM8K, and OlympiadBench verifier summaries and exponents are copied from the main experiment's frozen development selection. GPQA is exploratory: the primary analysis uses mean score with $\beta = 1$, while a pre-specified secondary analysis uses the main pipeline's development-selected configuration. Both have zero answer mismatches.

Table 6 reports the complete k=32 generator–benchmark matrix.

Table 6: Cross-generator transfer at k=32 with leader-first batch-one certification. "PRM save" counts verifier-input tokens; "e2e save" uses generation plus full-verification tokens. Certified accuracy equals verify-all accuracy in every row, with zero answer-ID mismatches.

Generator	Benchmark	n	v-all acc.	calls / 32 ↓	PRM save ↑	e2e save ↑
ORZ	MATH500	100	81.0	9.64	70.1%	36.6%
ORZ	GSM8K	100	98.0	1.46	95.8%	52.7%
ORZ	OlympiadBench	100	44.0	19.49	39.7%	20.3%
ORZ	GPQA-Diamond	99	32.3	8.82	73.3%	39.2%
DeepMath	MATH500	100	80.0	10.70	66.8%	34.1%
DeepMath	GSM8K	100	97.0	1.23	96.3%	52.7%
DeepMath	OlympiadBench	100	36.0	19.86	38.8%	19.4%
DeepMath	GPQA-Diamond	99	31.3	7.20	78.7%	41.1%
General	MATH500	100	79.0	9.74	69.8%	37.6%
General	GSM8K	100	95.0	2.60	92.0%	51.6%
General	OlympiadBench	100	42.0	18.48	43.4%	23.3%
General	GPQA-Diamond	99	37.4	11.16	65.6%	37.1%
Macro over 12 cells				**10.03**	**69.2%**	**37.1%**

The extension contains 1,197 generator–item records and 4,788 generator–item–budget cells. With 100 deterministic reveal-order permutations per cell, the primary evaluation compares 478,800 returned answer IDs and finds zero mismatches; the secondary GPQA analysis also finds zero. At k=32, the macro averages across the 12 rows are 68.7% fewer PRM calls, 69.2% fewer PRM tokens, and 37.1% fewer end-to-end tokens.

Context-limit audit. The PRM checkpoint has 4,096 native positions and no configured RoPE scaling. An initial audit found one GPQA item for which 71 complete PRM inputs exceeded the 4,096-position limit, reaching up to 4,897 tokens. We discarded the affected prefix scores and applied one outcome-independent eligibility rule uniformly: if any trace for a matched question exceeds the scorer context in any generator, exclude that question from every generator. This removes exactly one shared GPQA item and leaves the same 99 IDs in the same order for all three generators. A rerun with a fail-fast scorer records zero partial truncations, exceptions, or over-context inputs across all 38,304 retained traces. Scores and step counts for every retained row are bit-identical to the audited initial run, showing that the correction changes only the excluded row.

F.1 Second-PRM robustness audit

To check that certification is not specific to the main verifier's scoring mechanism, we ran a reduced, frozen diagnostic with the generative PRM `RLHFlow/Llama3.1-8B-PRM-Mistral-Data` (Xiong et al., 2024), at revision `0a4c93ad67513db6f2a22005c9271e2b71632c4a`. We used the official incremental generative-PRM scoring loop and defined each trace score as the mean of its step rewards. Verify-all aggregates s_i^β with fixed, untuned $\beta = 1$.

The audit fixes 64 test items each from GSM8K, MATH500, and OlympiadBench, 32 traces per item, and prefixes $k \in \{4, 8, 16, 32\}$: 6,144 scored traces and 768 item–k cells in total. Here one PRM call means scoring one complete trace, not an internal step forward. Leader-first uses batch size one; the primary random baseline uses the preregistered seed 20260811. Every exactness comparison uses the returned answer ID.

Table 7: Reduced second-PRM audit. Accuracy is shared by verify-all and both exact policies. "Rnd." uses seed 20260811; mismatch entries are leader-first/random winner mismatches. Savings are relative to k calls.

Benchmark	k	n	VA/policy acc.	LF calls	LF save	Rnd. calls	Rnd. save	mismatch
GSM8K	4	64	90.6%	0.312	92.19%	0.328	91.80%	0/0
GSM8K	8	64	90.6%	0.656	91.80%	0.703	91.21%	0/0
GSM8K	16	64	90.6%	0.922	94.24%	1.125	92.97%	0/0
GSM8K	32	64	90.6%	1.672	94.78%	1.891	94.09%	0/0
MATH500	4	64	84.4%	0.875	78.12%	1.109	72.27%	0/0
MATH500	8	64	84.4%	1.203	84.96%	1.703	78.71%	0/0
MATH500	16	64	89.1%	1.812	88.67%	3.000	81.25%	0/0
MATH500	32	64	85.9%	3.344	89.55%	5.344	83.30%	0/0
OlympiadBench	4	64	15.6%	2.016	49.61%	2.562	35.94%	0/0
OlympiadBench	8	64	15.6%	3.391	57.62%	5.141	35.74%	0/0
OlympiadBench	16	64	21.9%	5.562	65.23%	8.984	43.85%	0/0
OlympiadBench	32	64	20.3%	9.250	71.09%	16.562	48.24%	0/0
Micro (192 items)	4	192	63.5%	1.068	73.31%	1.333	66.67%	0/0
Micro (192 items)	8	192	63.5%	1.750	78.12%	2.516	68.55%	0/0
Micro (192 items)	16	192	67.2%	2.766	82.71%	4.370	72.69%	0/0
Micro (192 items)	32	192	65.6%	4.755	85.14%	7.932	75.21%	0/0

Leader-first has zero winner mismatches in all 768 cells, and each of the five audited random seeds (20260811, 1, 7, 42, and 12345) separately has zero mismatches in all 768 cells. Under the strict definitions zero-call $\equiv$ calls $= 0$, early $\equiv$ calls $< k$, and full $\equiv$ calls $= k$, leader-first is early on $745/768$ cells, including $352/768$ zero-call cells, and reaches full verification on $23/768$. This is a small fixed-subset robustness diagnostic with untuned aggregation, not a matched replication of the main experiment.

G External Efficiency and Accuracy Context

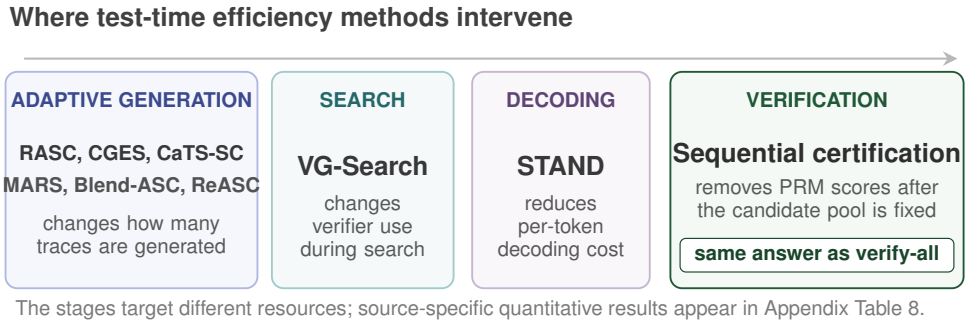


Figure 10: A stage-aware view of recent test-time efficiency methods. Adaptive generation changes the number of traces, search methods change how candidate paths are explored and scored, and decoding methods reduce per-token cost. Sequential certification instead removes PRM scores after the candidate pool is fixed while preserving the verify-all answer. These stages target different resources and do not constitute a common-scale comparison.

Table 8: Full source-aligned comparison underlying Figure 10. Methods target different stages, models, budgets, and cost units, so the reported reductions are not a common-scale leaderboard.

Method	Compute reduced	Reported reduction	Accuracy/decision guarantee
RASC (Wan et al., 2025)	generation	about 70% fewer samples	maintains accuracy
CGES (Aghazadeh et al., 2025)	generation	69.4% fewer calls ($16 \rightarrow 4.9$)	within 0.06 accuracy points
CaTS-SC (Huang et al., 2026)	generation	94.2% fewer samples at one reported operating point	target accuracy
MARS (Chen et al., 2026b)	generation	25–47% fewer self-consistency tokens	high-probability full-vote agreement under stated assumptions
Blend-ASC (Feng et al., 2025)	generation	6.8× fewer samples on average	SC accuracy–cost
VG-Search (Chen et al., 2025)	search / verify	over 52% fewer FLOPs	improved search accuracy
STAND (Song et al., 2025)	decoding	60–65% lower latency	maintained accuracy
ReASC (Kim et al., 2026)	generation	up to 70% lower inference cost	preserved accuracy
Ours	**verification**	54.8% **fewer PRM calls;** 29.4% **fewer e2e tokens**	**exact verify-all answer**

Table 9: Accuracy context on five mathematical benchmarks. The six rows marked "standardized" are the multi-seed LightEval re-evaluation of Hochlehnert et al. (2025); the two 2026 rows are source-reported. Our row uses PRM selection over fixed Qwen2.5-7B-Instruct pools and is not directly comparable because candidate budgets, test subsets, and grading protocols differ. This table is context, not a baseline leaderboard for certification.

System	AIME24 ↑	AMC23 ↑	MATH500 ↑	Minerva ↑	Olympiad ↑
Ours: verify-all, k=32	18.8	66.7	85.2	34.6	45.8
Qwen2.5-7B-Instruct (standardized)	12.3	52.8	77.3	34.9	38.9
Open-Reasoner-Zero-7B (standardized)	19.7	59.5	83.9	31.6	47.6
R1-Distill-Qwen-7B (standardized)	52.3	91.5	94.1	40.1	67.3
Sky-T1-mini (standardized)	51.3	90.0	93.3	41.8	67.7
DisCO-logL (standardized)	51.3	92.2	94.0	44.7	67.6
Skywork-OR1-7B (standardized)	60.7	94.0	95.7	43.0	73.6
Re-Schedule$_{\text{sigmoid}}$ (Wang et al., 2026)	18.2	69.2	81.0	41.5	43.3
CASPO+CaT (Qwen-Math-7B) (Chen et al., 2026a)	26.7	67.5	81.9	40.5	46.9

Table 9 provides accuracy context for the generator and selection pipeline used in our experiments. Because the listed systems differ in sampling budgets, evaluation subsets, and grading protocols, these values do not measure the relative quality of the certification policies studied here.

H Reproducibility Checklist and Limitations

Released evaluation artifacts. The merged result file contains all 28 benchmark–k configurations, with one per-configuration JSON file in the accompanying results directory. The full validator checks configuration coverage, finite token fields, cost monotonicity, answer-level exactness, and the seven external k=32 accuracy values. It reports "VALIDATION PASSED." SHA-256 checksums are provided for the merged JSON, scripts, logs, validator output, and audit report.

Determinism. The dev/test splits, answer sorting, tie rule, leader-first order, and random baseline seed are fixed. Sequential certification itself contains no learned threshold. Re-running the evaluator on the same pools and prepared PRM arrays therefore reproduces the same stopping calls and winners.

Cost and scope. Generation lengths are recorded, but PRM tokens are step-weighted estimates; wall-clock latency also depends on batching and hardware. Experiments cover seven mathematical benchmarks in the main study, plus smaller matched subsets from three additional generators and

exploratory GPQA-Diamond. All experiments use at most 32 traces. The main results remain concentrated on one PRM family; the second generative-PRM audit is limited to three benchmarks, 64 fixed items per benchmark, and untuned $\beta = 1$, so it does not demonstrate comprehensive verifier transfer. The proof requires bounded trace weights and guarantees equivalence to verify-all, not correctness of the verifier. Adaptive batching, tighter bounds, and faster verifier decoding remain compatible extensions.

Table 10 summarizes the precise boundary of this claim.

Table 10: Boundary of the claim. The certificate is deliberately a systems guarantee relative to a specified verify-all selector.

Guaranteed	Required	Not claimed
Same returned answer as verify-all	Known finite upper bound on every unseen trace contribution	That the verify-all answer is correct
Soundness for any score-reveal order and positive batch size	Identical answer grouping, score preparation, aggregation, and tie rule	Universal accuracy or cost transfer to an unseen generator, domain, extractor, or PRM
Zero answer mismatches in every audited evaluation cell	Fixed candidate pool for the compared policies	Wall-clock speedup equal to token reduction

H.1 Deployment checklist

Before applying the certificate to a new verification pipeline, a practitioner should check five items.

1. **Freeze the target selector.** Specify answer normalization, invalid answer handling, the score transform, aggregation rule, and deterministic tie behavior before measuring savings.
2. **Bound every unseen contribution.** Use a known model range, an explicit clipping rule, or trace-specific bounds B_i. An empirical percentile is not a sound upper bound.
3. **Audit indexing before cost.** Confirm that pool rows, PRM arrays, step counts, and train/dev/test membership refer to the same examples. The released validator treats alignment failures as fatal.
4. **Compare returned answer IDs.** Equal aggregate accuracy can conceal offsetting answer changes. Exactness must be tested item by item against verify-all.
5. **Report synchronization separately.** Batch-one minimizes revealed tokens, while larger batches may better match accelerator throughput. Token savings and wall-clock savings answer different deployment questions.

Compatible extensions. The proof permits adaptive reveal orders, nonuniform trace-specific upper bounds, and asynchronous batches, provided the stopping test always uses valid bounds. Tighter score ranges or answer-group-specific bounds can only make certification earlier. Conversely, a verifier with unbounded weights requires a different certificate; silently truncating its range changes the verify-all target and must be reported as a separate method.