DECISION-SUFFICIENT EARLY STOPPING FOR PROCESS REWARD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Checking an LLM-generated solution often requires a process reward model (PRM) to score every reasoning step. We study when this checking can stop early while returning exactly the decision of full verification. We formalize *decision sufficiency*: a prefix is sufficient when every completion allowed by the verifier contract induces the same terminal decision. By bounding the terminal scores reachable from each prefix, we obtain an exact monitor that stops only when the entire reachable range lies within one decision region. Under the stated score-range model, tight closed-form envelopes support minimum, maximum, sum, product, and finite-horizon mean aggregation. The monitor requires no additional training and, with exact bounds, stops at the earliest sufficient prefix under the fixed observation order and continuation model. On ProcessBench, the finite-horizon mean monitor saves 32.52% of verifier input tokens with unchanged decisions, whereas a monotone-threshold shortcut saves none. Across the main Qwen and Llama public-benchmark evaluations, the monitors save 14–33% of verifier input. On a separate 1,000-trace PRMBench timing subset under matched incremental execution semantics, they reduce Qwen and Llama mean verification time by 26.01% and 39.13%, respectively, with identical full incremental decisions and shared-prefix scores.

1 INTRODUCTION

Given a question and an already generated step-by-step solution, a verifier must decide whether to accept the solution, reject it, or abstain from judgment. A process reward model (PRM) supports this task by assigning a score to each reasoning step; the verifier combines these scores and applies a fixed decision rule (Lightman et al., 2024; Zheng et al., 2025; Song et al., 2025). Full verification scores every step. Our task is to *score fewer steps while returning exactly the same final decision*. The computation being shortened is the checking of a completed solution; generation has already finished.

Figure 1a illustrates the input, computation, and output. Consider a ten-step solution whose scores lie in $[0, 1]$, accepted if their final mean is at least 0.5. After eight scores summing to 2, even the largest possible remaining scores give a final mean of only 0.4. The verifier can reject immediately and omit the last two scoring operations. This example captures the intended saving: stop when the remaining work can no longer affect the required output.

Research in this area follows three broad directions. Process supervision and specialized reward models improve the quality of step judgments; process-verification benchmarks measure their ability to identify reasoning errors (Lightman et al., 2024; Zheng et al., 2025; Song et al., 2025). Adaptive inference and dynamic verification allocate computation using intermediate predictions or calibrated risk criteria (Schwartz et al., 2020; Zhong et al., 2025; Wang et al., 2026). Certificates and runtime verification study when partial observations already determine an output (Buhrman & de Wolf, 2002; Kupferman & Vardi, 2001; Bauer et al., 2011). These directions motivate a concrete question for PRMs: how can a fixed verifier avoid unnecessary scoring while preserving its decision on each individual solution? The key question is therefore not whether an intermediate prediction is confident, but whether any completion allowed by the verifier contract can still change the verifier's terminal decision.

The first difficulty is determining whether unseen scores can reverse the decision, because a plausible intermediate judgment need not equal the full judgment. In Figure 1b, four scores of 0.25 suggest

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.

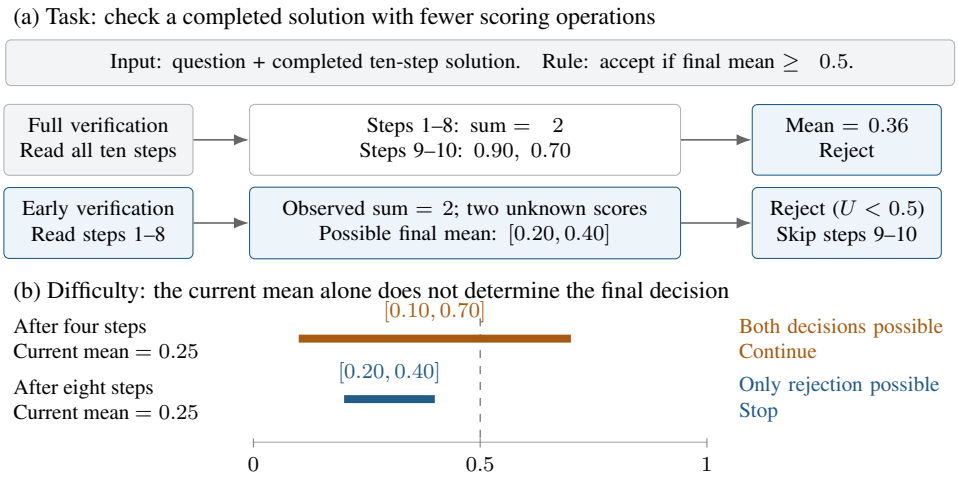

Figure 1: Task and challenge, using an illustrative mean-based verifier. (a) The first eight scores are six 0.25 scores followed by 0.50 and 0; the monitor first certifies rejection at step eight and skips two evaluations. (b) The same running mean can require different actions: only the range of *possible final means* reveals whether unseen scores can change the decision. The dashed line is the acceptance threshold.

rejection, but six later scores of 1 would raise the final mean to 0.7. We define this difficulty as *decision ambiguity*: at least two final decisions remain possible given the observed scores and the allowed remaining scores. Its impact on the task is direct: stopping during ambiguity can change the full verifier's output, violating the required equality.

The second difficulty is resolving that ambiguity without scoring the entire solution. Enumerating possible remaining score sequences is impractical for real-valued scores, and the running aggregate need not move in one direction. Figure 1b shows the same current mean after four and eight steps, although only the latter determines rejection. The computational difficulty is therefore to *bound the remaining scores' effect on the final aggregate*. Without an efficient bound, a verifier must either continue unnecessarily or use a stopping rule that cannot establish decision preservation.

To address decision ambiguity, we formulate a *decision-sufficiency test*: stop only when every allowed completion of the unseen scores yields the same final decision. To make this test efficient, we derive *reachable-score envelopes* from the observed aggregate, score ranges, and remaining step count. Our *exact prefix monitor* updates these envelopes after each step and stops when the entire reachable range falls in one decision region. It requires no additional training and, with exact bounds, identifies the earliest sufficient prefix under the fixed observation order and continuation model. Building on the established all-completions principle, our contributions are:

- We formalize decision sufficiency for process verification under a fixed verifier contract and characterize exact prefix stopping by whether the reachable terminal-score set lies entirely within one decision region, covering both binary and abstaining decisions.
- We derive tight closed-form reachable-score envelopes for minimum, maximum, bounded sum, product, and finite-horizon mean aggregation, yielding $O(1)$-overhead exact prefix monitors; with exact bounds, the first sufficient prefix is the earliest exact stop available under the same fixed observation order and continuation model.
- We show that reachable-state reasoning enables exact stopping beyond absorbing monotone rules: for finite-horizon mean aggregation on ProcessBench, the exact monitor saves 32.52% of verifier input where a monotone-threshold shortcut saves none. Across three PRMs and two public benchmarks, all evaluated structural settings preserve the corresponding full-verifier decision while removing substantial verification work.
- We show that this certified verification redundancy can translate into realized computation savings: on a disjoint 1,000-trace PRMBench timing evaluation under matched incremental

execution semantics, Qwen and Llama preserve all full incremental decisions and shared-prefix scores while reducing mean verification time by 26.01% and 39.13%, respectively.

2 Related Work

2.1 Process reward modeling and evaluation

Process reward models score intermediate reasoning steps for solution selection and error detection. Uesato et al. (2022) compared step-level and outcome supervision using learned reward models to reduce reasoning errors. Lightman et al. (2024) trained a verifier on human step judgments and released PRM800K. Wang et al. (2024) proposed Math-Shepherd, which learns step rewards from automatically constructed labels for reranking and reinforcement learning. Luo et al. (2024) proposed OmegaPRM, which combines tree search and binary error localization to collect process labels. Yuan et al. (2025) derived implicit process rewards from an outcome model parameterized by policy-to-reference log-likelihood ratios, using response-level labels.

Zhang et al. (2025) developed consensus filtering of Monte Carlo estimates and model judgments to train PRMs on Qwen2.5-Math (Yang et al., 2024). Pala et al. (2025) proposed PathFinder-PRM, which combines mathematical and consistency error predictions to estimate step correctness. Pisano & Navigli (2026) used formal planning problems to generate labeled reasoning steps. For evaluation, Zheng et al. (2025) introduced ProcessBench, whose expert annotations test first-error localization and recognition of correct solutions. Song et al. (2025) introduced PRMBench to evaluate step judgments across simplicity, soundness, and sensitivity. These studies supply training signals, scoring architectures, and benchmarks for step-level assessment.

2.2 Adaptive computation and verification

Adaptive methods use intermediate signals to vary computation or prediction coverage. Teerapittayanon et al. (2016) proposed BranchyNet, which exits through intermediate classifiers when predictions are sufficiently confident. Schwartz et al. (2020) used calibrated layer-wise confidence to select BERT inference depth. Xin et al. (2020) proposed DeeBERT, which uses intermediate prediction entropy to skip later layers. Schuster et al. (2022) proposed CALM, which selects depth per token and calibrates exits to control expected sequence-level degradation. Geifman & El-Yaniv (2017) developed selective classification that abstains to meet a risk target with high probability.

For reasoning, Snell et al. (2025) allocated test-time computation by problem difficulty through verifier-guided search and response revision. Cheshmi et al. (2025) proposed partial reward modeling to reject candidates from unfinished steps during generation. Zhong et al. (2025) proposed Dyve, which switches between immediate confirmation and extended step analysis. Ni et al. (2026) proposed ReProbe, which estimates step credibility using a small transformer over frozen model states. Wang et al. (2026) proposed Conformal Thinking, which calibrates stopping thresholds to control error risk under a reasoning budget. These methods adjust network depth, candidate generation, verification effort, or prediction coverage using learned signals or statistical criteria.

2.3 Certificates and verification from partial observations

Certificate-based evaluation determines an output when every completion of partial observations gives the same result; runtime verification applies related reasoning to execution prefixes. Buhrman & de Wolf (2002) surveyed certificates that fix a Boolean function's output and their relation to query complexity. Deshpande et al. (2016) used stochastic submodular set cover to approximate minimum expected-cost Boolean evaluation under a product distribution. Fagin et al. (2003) proposed threshold algorithms that stop top-k retrieval when monotone score bounds exclude unseen competitors. Kupferman & Vardi (2001) checked safety properties by finding finite bad prefixes that establish violations regardless of subsequent behavior. Bauer et al. (2011) constructed three-valued monitors that return true or false at the earliest decisive prefix and remain inconclusive otherwise. These frameworks connect partial observations to determined outputs under their respective query models, aggregation assumptions, or temporal semantics.

3 METHOD

The core verification loop is shown in Figure 2. The *prefix-scoring module* uses the existing PRM to evaluate the next step and update the aggregate state, collecting the evidence observed so far. The *reachable-bound module* combines that state with the remaining step count and score ranges to compute the smallest and largest possible final scores. The *decision-sufficiency module* applies the fixed decision rule to this interval: if all values imply one judgment, it returns that judgment and skips the suffix; otherwise, it requests another score. Only the latter two modules are added to the verifier.

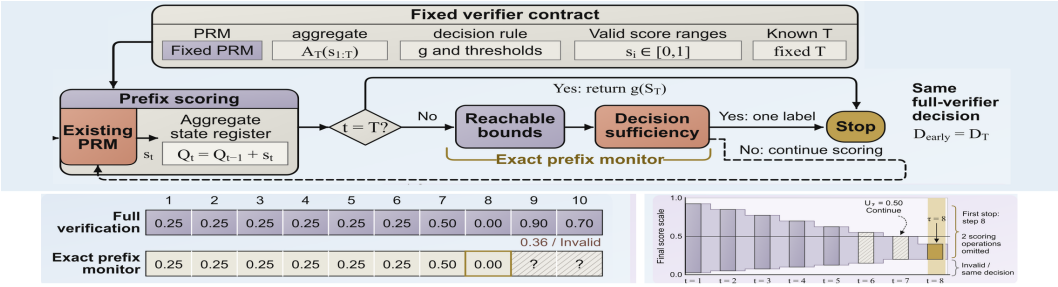

Figure 2: Core verification loop and an illustrative mean-rule execution. The exact monitor bounds all final scores reachable from each prefix: it continues at $t = 7$, stops at $t = 8$ when only INVALID remains, skips two PRM evaluations, and returns the full-verifier decision.

3.1 PROBLEM FORMULATION

The input is a question and a completed reasoning trace with T known scoring boundaries. At boundary t, the PRM produces score s_t. Full verification combines all scores into $S_T = A_T(s_{1:T})$ and returns

$$D_T = g(S_T) \in \mathcal{D}, \tag{1}$$

where the fixed rule g can return VALID, INVALID, or ABSTAIN. Our task is to stop at $\tau \leq T$ and return $D_{\text{early}} = D_T$ with less scoring work. Early and full execution use the same scoring order and prefix computation, so shared prefixes produce identical observations. This equality preserves the full verifier's decisions, including its errors and abstentions.

Let $H_t = s_{1:t}$ be the observed score prefix. A *completion* is a possible sequence of unseen scores for the existing solution, not additional generated text. Write $\mathcal{Z}_{t+1:T}(H_t)$ for the allowed completions under the fixed aggregation, decision rule, step count, and score constraints. The *reachable final-score set* is

$$\mathcal{R}_t(H_t) = \{A_T(s_{1:t}, u_{t+1:T}) : u_{t+1:T} \in \mathcal{Z}_{t+1:T}(H_t)\}. \tag{2}$$

It contains every possible final score consistent with the prefix. For our closed-form bounds, all combinations of unseen scores within their deterministic intervals, such as $[0, 1]$, are allowed. The monitor stores only the aggregate state needed to update these bounds, such as the observed sum for a mean.

3.2 DECISION SUFFICIENCY: WHEN CAN VERIFICATION STOP?

This module determines whether the observed steps already fix the final judgment, resolving the ambiguity caused by unseen scores. It takes the reachable final-score set as input and asks whether the fixed rule assigns the same label to every value in that set.

Definition 1 (Decision sufficiency). *A prefix H_t is decision-sufficient if every allowed completion gives the same final decision:*

$$g(\mathcal{R}_t(H_t)) = \{d\} \quad \textit{for some } d \in \mathcal{D}. \tag{3}$$

The monitor can then stop and return d.

If two terminal decisions remain reachable, no single early decision can agree with every allowed completion. If only one remains, additional scores cannot change it. Decision sufficiency is therefore necessary and sufficient for exact stopping.

Table 1: Tight exact reachable final-score bounds for $s_i \in [0,1]$ and $1 \le t < T$. Here $Q_t = \sum_{i=1}^{t} s_i$ and $P_t = \prod_{i=1}^{t} s_i$. At $t = T$, every interval becomes the singleton $[S_T, S_T]$.

Aggregation	Prefix state	L_t	U_t
Minimum	$m_t = \min_{i \le t} s_i$	0	m_t
Maximum	$M_t = \max_{i \le t} s_i$	M_t	1
Sum	Q_t	Q_t	$Q_t + (T-t)$
Product	P_t	0	P_t
Mean	Q_t	Q_t/T	$[Q_t + (T-t)]/T$

For a scalar aggregate whose reachable values form an interval $[L_t, U_t]$, the test reduces to comparing two bounds with the decision threshold.

Theorem 1 (Exact interval stopping test). *Suppose $\mathcal{R}_t = [L_t, U_t]$ and $g(s) = \mathbf{1}\{s \ge \gamma\}$. An exact early decision is available if and only if*

$$U_t < \gamma \quad \textit{or} \quad L_t \ge \gamma. \tag{4}$$

The first condition returns 0 *and the second returns* 1.

Proof. Each condition places every reachable value on the same side of the threshold. If neither holds, the exact interval contains a value below γ and a value at least γ, so both decisions remain possible. □

The same test applies to a verifier that can abstain. For correctness scores, let $\gamma_{\mathrm{I}} < \gamma_{\mathrm{V}}$ and define INVALID for $s \le \gamma_{\mathrm{I}}$, VALID for $s \ge \gamma_{\mathrm{V}}$, and ABSTAIN between the thresholds. The stopping conditions are

$$U_t \le \gamma_{\mathrm{I}} \quad \Rightarrow \quad \textsc{Invalid}, \tag{5}$$

$$L_t \ge \gamma_{\mathrm{V}} \quad \Rightarrow \quad \textsc{Valid}, \tag{6}$$

$$L_t > \gamma_{\mathrm{I}} \text{ and } U_t < \gamma_{\mathrm{V}} \quad \Rightarrow \quad \textsc{Abstain}. \tag{7}$$

Each condition means that the entire interval lies in one decision region. Error scores use the corresponding reversed decision rule. If an interval only encloses the reachable set, these tests still preserve the full decision; exact intervals additionally identify every sufficient prefix. The module therefore ensures that an early judgment preserves the full verifier's acceptance, rejection, or abstention for every allowed continuation.

3.3 REACHABLE SCORE BOUNDS: HOW CAN THE TEST BE COMPUTED?

This module makes the sufficiency test computationally practical by replacing a search over unseen score sequences with two bound calculations. It updates L_t and U_t from the observed aggregate and the possible contribution of the remaining steps. Table 1 gives tight exact intervals under the score-range model defined above. Their endpoints are attainable by assigning the unseen scores their lower or upper bounds, and continuity gives all intermediate aggregate values.

For a minimum-based verifier, an observed score at or below the rejection threshold ensures that the final minimum will also be at or below it. A maximum gives the analogous above-threshold stopping condition. For a sum, both its current value and the largest possible remaining contribution matter. More generally, if $0 \le s_i \le u_i$, the final sum lies in $[Q_t, Q_t + \sum_{i>t} u_i]$. For a product, additional factors in $[0,1]$ can only decrease its value. Known future lower bounds $s_i \ge \ell_i$ tighten its reachable interval to $[P_t \prod_{i>t} \ell_i, P_t]$. These bounds expose decisions that the remaining values cannot reverse.

For a finite-horizon mean, the running value Q_t/t can move in either direction. The final mean uses the fixed denominator T, so its possible values are

$$\bar{s}_T \in \left[\frac{Q_t}{T}, \frac{Q_t + (T-t)}{T} \right]. \tag{8}$$

The lower bound assigns zero to every remaining score; the upper bound assigns one. For acceptance threshold γ, the monitor accepts once $Q_t \ge T\gamma$ and rejects once $Q_t + (T-t) < T\gamma$. These are

bounds on the final sum expressed as bounds on the final mean. They allow exact early stopping even though crossing a threshold with the current mean does not by itself fix the decision.

As scores arrive, the lower bound cannot decrease and the upper bound cannot increase. In Figure 1, $Q_8 = 2$ gives $[L_8, U_8] = [0.2, 0.4]$, certifying rejection at threshold 0.5. Thus, the bound module can reveal exact stopping opportunities even when the running mean itself gives no safe stopping signal; Section 4.3 measures the resulting savings.

3.4 The algorithm and its earliest-stop guarantee

Algorithm 1 turns the bound and sufficiency modules into the sequential procedure shown in Figure 2. After each PRM score, it updates the aggregate state and reachable interval. If every reachable score has the same label, the monitor returns that label; otherwise it acquires the next score. At $t = T$, it returns the fixed verifier's full decision.

Algorithm 1 Decision-sufficient prefix monitor under a fixed verifier contract

Require: Completed T-step trace; fixed PRM, aggregation A, rule g, and score bounds
Ensure: Return the corresponding full-verifier decision D_T
1: Initialize the aggregate state
2: **for** $t = 1, \ldots, T$ **do**
3: Score step t with the PRM and update the aggregate state
4: **if** $t = T$ **then return** $g(S_T)$
5: Compute the reachable interval $[L_t, U_t]$
6: **if** $g(s) = d$ for all $s \in [L_t, U_t]$ **then return** d
7: **end for**

The next result shows that, when the interval is exact, this procedure stops at the earliest boundary permitted by the same observation order and completion model.

Proposition 1 (Earliest exact stop under a fixed observation order). *Fix the verifier, allowed completions, and observation order, with stopping allowed at boundaries $1, \ldots, T$. Let τ^* be the first decision-sufficient boundary. For any stopping rule τ using the same prefix information and returning a correct decision for every allowed completion, $\tau^* \leq \tau$ on every score sequence. Algorithm 1 attains τ^* when its intervals are exact.*

Proof. At a boundary earlier than τ^*, at least two final decisions are possible. Stopping there would fail on one of the corresponding completions. At τ^*, all completions agree, and the exact interval test returns that decision. □

The result specifies the earliest stopping point available from the observed scores under the chosen constraints. The algorithm uses $O(1)$ aggregate state and $O(1)$ work per boundary for the rules in Table 1. For varying per-step bounds, remaining bound sums or products can be precomputed or maintained incrementally. This control loop therefore removes every suffix that can be certified unnecessary using the same prefix information and exact bounds, with constant monitoring work at each step.

3.5 Applying the monitor to a PRM

The prefix-scoring interface connects the stopping rule to an existing PRM without changing its observations. Our main Qwen verifier aggregates correctness by minimum: INVALID at scores at most 0.5, VALID at scores at least 0.7, and ABSTAIN otherwise. Its nonfinal stopping condition is $m_t \leq 0.5$. The Llama verifier applies its threshold to sigmoid-transformed $[0, 1]$ scores, which are also used in the bounds. Identical shared-prefix computation ensures that skipped scoring preserves the specified final judgment.

The guarantee concerns the final threshold decision. Producing every step score, a complete error diagnosis, or an unthresholded ranking requires different output information. The bounds assume valid score ranges; for sum and product, floating-point calculations must enclose the reachable values at threshold comparisons. The earliest-stop guarantee is relative to the fixed observation order and completion model. These conditions make explicit what the monitor preserves when it omits a suffix.

4 EXPERIMENTS

4.1 EVALUATION PROTOCOL

We evaluate 978 internal mathematical traces, all 3,400 ProcessBench traces, and all 6,216 PRMBench examples. ProcessBench spans GSM8K, MATH, OlympiadBench, and Omni-MATH (Cobbe et al., 2021; Hendrycks et al., 2021; He et al., 2024; Gao et al., 2025; Zheng et al., 2025); PRMBench covers nine error categories (Song et al., 2025). The verifiers are Qwen2.5-Math-PRM-7B (Zhang et al., 2025), PathFinder-PRM-7B (Pala et al., 2025), and Llama-3.1-8B-PRM800k-r (Pisano & Navigli, 2026). Qwen applies thresholds $0.5/0.7$ to the minimum correctness score, Llama applies 0.5 to the minimum sigmoid score, and PathFinder applies $0.50/0.75$ to the maximum native error score.

Rules use separate internal development and calibration sets of 300 and 400 traces. Public comparisons pair early and full decisions on stored scores; mechanism and predictive-exit analyses reuse ProcessBench scores. Four ProcessBench labels were seen after the Qwen rule was fixed, and a few PRMBench labels before its protocol was fixed; all PRMBench rules and inputs preceded score acquisition. We therefore report fixed-rule transfer rather than blind benchmark evaluation.

Mismatch counts early/full decision disagreements, and early-stop rate counts traces stopped before the final boundary. Token saving is $1 - \sum_i C_i^{\text{early}} / \sum_i C_i^{\text{full}}$ for processed verifier input C_i; PathFinder charges repeated prefixes, whereas other token results use one incremental stream. Coverage is the non-abstained fraction and selective risk its error rate. We use Clopper–Pearson (CP) risk bounds (Clopper & Pearson, 1934) and paired bootstrap cost intervals (Efron, 1979).

4.2 EXACT DECISION PRESERVATION ACROSS DATASETS AND VERIFIERS

Table 2 compares each fixed verifier with full verification; public rows replay complete scores, while 534 internal suffixes are covered by the minimum-score certificate.

Table 2: Overall structural transfer: decision preservation and verifier-input savings.

Dataset	Verifier	n	Mismatch	Early stop	Token saving
Internal	Qwen	978	0[‡]	54.60%	18.51%
ProcessBench	Qwen	3,400	0	54.97%	23.77%
ProcessBench	PathFinder	3,400	0	60.35%	40.38%[†]
PRMBench	Qwen	6,216	0	56.71%	14.46%
PRMBench	Llama	6,216	0	76.53%	32.81%

[†]PathFinder uses repeated-prefix input accounting. [‡]For 534 internal early-invalid cases, omitted suffixes were not acquired; decision equivalence follows from the minimum-score certificate. Public-benchmark rows permit paired replay against stored full traces.

All five verifier–dataset settings have zero mismatch. Public-benchmark savings span 14.46% for Qwen on PRMBench to 40.38% for PathFinder under repeated-prefix accounting; Llama removes 32.81% on PRMBench. Decision-redundant verification is therefore not confined to a single dataset, checkpoint, or backbone family.

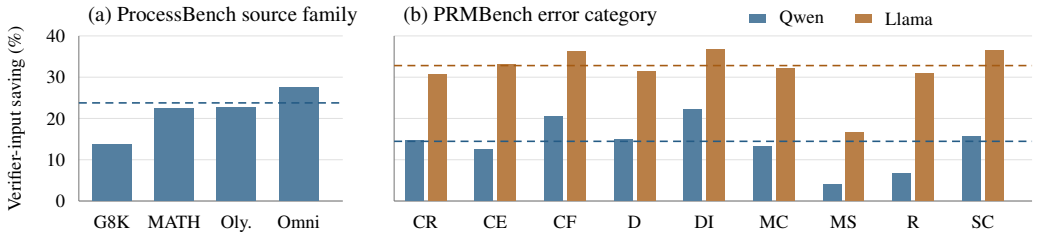

Figure 3: Cross-stratum input savings. Dashed lines are overall results; every stratum preserves the full-verifier decision. CR/CE/CF/D/DI/MC/MS/R/SC denote circular reasoning, confidence error, counterfactual, deception, domain inconsistency, missing condition, multiple solutions, redundancy, and step contradiction.

4.3 WHY REACHABLE BOUNDS MATTER BEYOND MONOTONE STOPPING

We compare mechanisms on the same 3,400 ProcessBench score sequences. A monotone shortcut cannot stop a running mean, whereas Equation 8 bounds its final value using the horizon. Thresholds for the other aggregators were selected on 400 internal calibration traces and define separate full-verifier rules.

Table 3: Aggregator and mechanism analysis on 3,400 ProcessBench traces.

Aggregation and stopping rule	Early stop	Token saving	Mismatch
Minimum, exact bounds	69.24%	33.71%	0
Nonnegative sum, exact bounds	67.32%	31.58%	0
Product, exact bounds	60.82%	26.69%	0
Mean, monotone-threshold shortcut	0.00%	0.00%	0
Mean, exact bounds	69.44%	32.52%	0

For the same mean rule, exact bounds add 69.44 percentage points (pp) of early stopping and 32.52 pp of saving over the shortcut, with zero mismatch. Minimum, sum, and product bounds each remove over one quarter of full input. Reachability therefore works even when the running mean can rise or fall.

4.4 EXACT STRUCTURAL STOPPING VS. PREDICTIVE EXITS

On the same ProcessBench/Qwen scores, Table 4 compares the all-completions certificate with fixed-prefix and calibrated predictive exits, which do not require agreement over every suffix.

Table 4: Exact versus predictive stopping on ProcessBench/Qwen; CP is a one-sided 95% risk bound.

Method	Saving	Mismatch	Coverage	Risk / CP upper
Fixed full	0.00%	0	92.85%	8.81 / 9.68%
Exact prefix monitor	23.77%	0	92.85%	8.81 / 9.68%
Fixed prefix 25%	58.01%	1,475	93.03%	42.52 / 43.99%
Fixed prefix 50%	37.13%	756	91.44%	23.58 / 24.86%
Fixed prefix 75%	13.02%	202	92.06%	12.33 / 13.34%
Calibrated entropy	33.00%	320	99.21%	11.41 / 12.35%
Calibrated class probability	37.07%	429	99.38%	13.20 / 14.13%

Against the fixed 75% exit, the exact monitor saves 10.75 pp more and avoids 202 disagreements. Calibrated entropy adds 9.23 pp but changes 320 decisions and raises risk by 2.60 pp; class probability changes 429 decisions. Only exact stopping preserves all full-verifier outcomes; Table 5 gives its matched-kernel runtime realization.

Table 5: Matched-kernel timing on 1,000 PRMBench traces; brackets give paired-bootstrap 95% intervals.

Verifier	Mismatch	Score $\Delta_{\max}$	Token reduction	Time reduction [95% CI]
Qwen	0/1,000	0	19.76%	26.01% [23.86, 28.23]
Llama	0/1,000	0	32.05%	39.13% [36.90, 41.37]

4.5 ROBUSTNESS ACROSS DATA STRATA AND CONTRACT INFORMATION

Figure 3 rules out one dominant family or error category: savings span 13.66–27.57% across ProcessBench families, 4.05–22.16% for Qwen categories, and 16.60–36.73% for Llama categories. Every stratum has zero mismatch.

To weaken the known-length contract, we replace T by a valid bound $U \geq T$, yielding the mean interval $[Q_t/U, (Q_t + U - t)/U]$. Table 6 replays this rule on ProcessBench; $U = 47$ is the observed maximum and only a retrospective sensitivity setting.

Table 6: Horizon-information sensitivity for the mean verifier on ProcessBench ($n = 3{,}400$).

Length information	Early stops	Early-stop rate	Token saving	Mismatch
Exact T	2,361	69.44%	32.52%	0
$U = \lceil 1.10T \rceil$	2,319	68.21%	31.40%	0
$U = \lceil 1.25T \rceil$	2,296	67.53%	30.57%	0
$U = \lceil 1.50T \rceil$	2,236	65.76%	29.16%	0
$U = 47$ (observed maximum)	1,197	35.21%	11.50%	0

The $\lceil 1.5T \rceil$ bound retains 29.16% saving, only 3.36 pp below exact length; even $U = 47$ retains 11.50%. All settings have zero mismatch: sound but looser verifier contracts reduce efficiency conservatively rather than weakening decision preservation.

4.6 Matched-kernel runtime realization

Fresh timed inference fixes tokenizer, precision, kernel, cache, and scoring schedule within each early/full pair. A separate 200-trace subset selects among intervals $\{1, 2, 4, 8, 16\}$ by the largest zero-mismatch reduction. Evaluation uses 1,000 disjoint PRMBench traces, 20 warm-up pairs, alternating order, and CUDA synchronization.

Table 5 shows zero decision and shared-prefix score differences, with time reduced by 26.01% for Qwen and 39.13% for Llama. Figure 4 shows their paired intervals and positive reductions in every length quartile, ruling out a gain driven only by the longest traces.

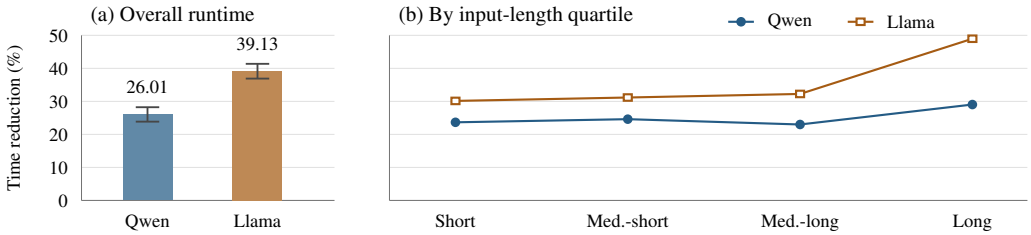

Figure 4: Runtime robustness: (a) paired-bootstrap 95% intervals; (b) input-length quartiles.

Table 7: Timing distribution and mean scoring calls over 1,000 traces (seconds; F/E = full/early).

Verifier	Mean F/E	Median F/E	IQR F/E	Calls F/E
Qwen	0.3304/0.2445	0.2758/0.1977	0.1989–0.3909/0.1322–0.2951	12.556/9.294
Llama	0.3737/0.2275	0.3134/0.1974	0.2231–0.4409/0.1239–0.2932	12.556/7.689

Mean/median time falls by 85.9/78.1 ms for Qwen and 146.2/116.0 ms for Llama; scoring calls fall by 3.262 and 4.867 (Table 7). These gains cover completed-trace verification under incremental execution; one-pass inference can change scores and decisions.

5 Conclusion

We study when verification of a completed reasoning trace can stop without changing a process reward model's terminal decision. Decision sufficiency turns this requirement into a reachable-state test, and tight closed-form envelopes make the test practical for several common aggregation rules, including finite-horizon means for which simple monotone stopping yields no saving. Across three PRMs, ProcessBench, and PRMBench, the resulting monitors preserve the corresponding full-verifier decisions while removing substantial verification work; under matched incremental execution, they reduce Qwen and Llama mean verification time by 26.01% and 39.13%. The guarantee is deliberately verifier-relative: it does not make the verifier more accurate, but certifies when its remaining computation has become irrelevant to the decision it would ultimately return.

REFERENCES

Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology*, 20(4):14:1–14:64, 2011. doi: 10.1145/2000799.2000800. URL https://doi.org/10.1145/2000799.2000800.

Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi: 10.1016/S0304-3975(01)00144-X. URL https://doi.org/10.1016/S0304-3975(01)00144-X.

Seyyed Saeid Cheshmi, Azal Ahmad Khan, Xinran Wang, Zirui Liu, and Ali Anwar. Accelerating LLM reasoning via early rejection with partial reward modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 10433–10447. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.551. URL https://aclanthology.org/2025.findings-emnlp.551/.

C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934. doi: 10.1093/biomet/26.4.404. URL https://doi.org/10.1093/biomet/26.4.404.

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. URL https://arxiv.org/abs/2110.14168.

Amol Deshpande, Lisa Hellerstein, and Devorah Kletenik. Approximation algorithms for stochastic submodular set cover with applications to Boolean function evaluation and Min-Knapsack. *ACM Transactions on Algorithms*, 12(3):42:1–42:28, 2016. doi: 10.1145/2876506. URL https://dl.acm.org/doi/10.1145/2876506.

B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. doi: 10.1214/aos/1176344552. URL https://doi.org/10.1214/aos/1176344552.

Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, 66(4):614–656, 2003. doi: 10.1016/S0022-0000(03)00026-6. URL https://doi.org/10.1016/S0022-0000(03)00026-6.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 100540–100569, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/f9e1e8b56c7e363985ebeb0e9dd1a85c-Paper-Conference.pdf.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html.

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

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 *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf.

Orna Kupferman and Moshe Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001. doi: 10.1023/A:1011254632723. URL https://doi.org/10.1023/A:1011254632723.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. arXiv preprint arXiv:2406.06592, 2024. URL https://arxiv.org/abs/2406.06592.

Jingwei Ni, Ekaterina Fadeeva, Tianyi Wu, Mubashara Akhtar, Jiaheng Zhang, Elliott Ash, Markus Leippold, Timothy Baldwin, See-Kiong Ng, Artem Shelmanov, and Mrinmaya Sachan. ReProbe: Efficient test-time scaling of multi-step reasoning by probing internal states of large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11667–11689. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.536. URL https://aclanthology.org/2026.acl-long.536/.

Tej Deep Pala, Panshul Sharma, Amir Zadeh, Chuan Li, and Soujanya Poria. Error typing for smarter rewards: Improving process reward models with error-aware hierarchical supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 14940–14954. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.807. URL https://aclanthology.org/2025.findings-emnlp.807/.

Raffaele Pisano and Roberto Navigli. Process reward models meet planning: Generating precise and scalable datasets for step-level rewards. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28022–28042. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1292. URL https://aclanthology.org/2026.acl-long.1292/.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17456–17472. Curran Associates, Inc., 2022. doi: 10.52202/068431-1269. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/6fac9e316a4ae75ea244ddcef1982c71-Paper-Conference.pdf.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6640–6651. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.593. URL https://aclanthology.org/2020.acl-main.593/.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b623663fd9b874366f3ce019fdfdd44-Paper-Conference.pdf.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 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)*, pp. 25299–25346. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1230. URL https://aclanthology.org/2025.acl-long.1230/.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. BranchyNet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469. IEEE, 2016. doi: 10.1109/ICPR.2016.7900006. URL https://doi.org/10.1109/ICPR.2016.7900006.

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

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Xi Wang, Anushri Suresh, Alvin Zhang, Rishi More, William Jurayj, Benjamin Van Durme, Mehrdad Farajtabar, Daniel Khashabi, and Eric Nalisnick. Conformal Thinking: Risk control for reasoning on a compute budget. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.03814.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2246–2251. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.204. URL https://aclanthology.org/2020.acl-main.204/.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. arXiv preprint arXiv:2409.12122, 2024. URL https://arxiv.org/abs/2409.12122.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 73511–73525. PMLR, 2025. URL https://proceedings.mlr.press/v267/yuan25c.html.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.547. URL https://aclanthology.org/2025.findings-acl.547/.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.50. URL https://aclanthology.org/2025.acl-long.50/.

Jianyuan Zhong, Zeju Li, Zhijian Xu, Xiangyu Wen, and Qiang Xu. Dyve: Thinking fast and slow for dynamic process verification. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 22320–22333. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1136. URL https://aclanthology.org/2025.emnlp-main.1136/.

APPENDIX

A AI USE DISCLOSURE

Generative AI tools assisted research ideation, theoretical and experimental development, coding, literature discovery, result organization, and writing. The authors are responsible for verifying all claims, results, and citations.

B ADDITIONAL THEORY AND DECISION RULES

This appendix gives supporting characterizations, refinements of the completion model, numerical validation of the bounds, and the selective-decision replay.

B.1 EXACTNESS OF THE STOPPING CONDITION

Proposition 2 (Exactness characterization). *A prefix admits a decision that agrees with every allowed completion if and only if* $|g(\mathcal{R}_t(H_t))| = 1$.

Proof. If the reachable decisions are the singleton $\{d\}$, every completion returns d. If two distinct decisions are reachable, any decision returned now disagrees with at least one of their corresponding completions. □

The requirement concerns every completion allowed by the model of unseen scores. In the interval model, every combination of remaining scores within their specified intervals is allowed. Additional constraints from the verifier can reduce this set. A bound that ignores such constraints remains valid whenever it contains every feasible final score.

B.2 ENCLOSING INTERVALS AND TIGHTER CONSTRAINTS

If $\mathcal{R}_t \subseteq \widehat{\mathcal{R}}_t$ and $g(\widehat{\mathcal{R}}_t) = \{d\}$, then $g(\mathcal{R}_t) = \{d\}$. An enclosing interval therefore suffices for exact stopping. If it crosses a decision threshold, a tighter bound may still determine the decision.

Proposition 3 (Refining the set of allowed scores). *If two valid completion models satisfy* $\mathcal{Z}'_{t+1:T}(H_t) \subseteq \mathcal{Z}_{t+1:T}(H_t)$*, every prefix sufficient under* $\mathcal{Z}$ *is also sufficient under* $\mathcal{Z}'$.

Proof. The corresponding reachable sets satisfy $\mathcal{R}'_t(H_t) \subseteq \mathcal{R}_t(H_t)$. A decision map constant on $\mathcal{R}_t(H_t)$ is constant on its subset. □

Deterministic score constraints can thus increase savings. For a mean, the exact interval in Equation 8 follows because the unseen sum ranges over $[0, T-t]$. Both endpoints are attained by assigning all remaining scores zero or one. For variable score bounds, the remaining minimum and maximum contributions replace these endpoints. The theoretical statements use exact arithmetic; floating-point implementations of sum and product bounds must maintain enclosing intervals at threshold comparisons. The elapsed-time experiments use cumulative minimum aggregation.

B.3 NUMERICAL VALIDATION OF THE REACHABLE BOUNDS

We compare the closed-form reachable intervals with exhaustive enumeration on 878,400 finite-grid cases and with 100,000 randomized reachability checks. These checks evaluate the bound calculations rather than model inference.

The analytic intervals agree with the reference calculations in every checked case, with no observed discrepancy across either set of checks.

This agreement checks the implementation on the enumerated and randomized cases; the mathematical guarantees follow from the stated score-range assumptions and proofs.

B.4 SELECTIVE DECISIONS AND ABSTENTION

Proposition 4 (Selective interval test). *For the correctness-score decision rule with* INVALID *at* $s \leq \gamma_{\mathrm{I}}$, VALID *at* $s \geq \gamma_{\mathrm{V}}$, *and* ABSTAIN *otherwise, an exact interval permits early stopping precisely when it is contained in one of these three decision regions. The three conditions are given in Section 3.*

Proof. Containment gives one decision for every reachable score. If containment fails, an exact interval intersects at least two decision regions. □

We examine whether early stopping preserves all three outputs of the main Qwen rule on the internal evaluation and ProcessBench. The rule uses the cumulative minimum correctness score, with $\gamma_{\mathrm{I}} = 0.5$ and $\gamma_{\mathrm{V}} = 0.7$. ProcessBench permits a paired comparison with stored full traces; the internal evaluation instead omits suffix acquisition after a certified early rejection.

The internal full verifier returns 262 valid, 677 invalid, and 39 abstaining decisions; the monitor certifies 534 early-invalid cases and otherwise reaches the final step. On ProcessBench, the corresponding full counts are 1,104/2,053/243, and 1,869 invalid cases stop early. Both settings have zero mismatch, no early valid or abstaining decisions, and no change to the original abstentions. The 534 internal suffixes were not acquired; their equality follows from the minimum-score certificate.

Before the last step, an unseen score can lower the minimum to zero. Consequently, only an INVALID decision can be certified early under this rule; valid and abstaining cases still require the complete trace. Agreement on the 534 omitted internal suffixes follows from this minimum-score property, rather than from a paired measurement of those suffixes.

C SENSITIVITY TO TRACE-LENGTH INFORMATION

For a completed candidate, T is known before scoring. To study less precise length information, we replay the mean monitor using an upper bound $U \geq T$. For scores in $[0, 1]$ and observed sum Q_t, enclosing the possible final means over all allowed terminal lengths gives

$$\left[\frac{Q_t}{U}, \frac{Q_t + (U - t)}{U}\right]. \tag{9}$$

The lower endpoint decreases and the upper endpoint increases as the allowed length grows. Both therefore take their most conservative values at U.

Table 6 reports a post-hoc sensitivity analysis on the same 3,400 ProcessBench sequences using $U = \lceil cT \rceil$ and the retrospective dataset maximum $U = 47$. A deployed verifier would need a valid bound specified for its inputs. All early decisions in this analysis are INVALID. A looser length bound widens the possible final-mean interval and delays its containment in one decision region, reducing savings without weakening decision preservation.

D EVALUATION PROTOCOL AND ADDITIONAL RESULTS

D.1 MODELS, THRESHOLDS, AND DATA USE

The main Qwen rule aggregates correctness scores by minimum: VALID at a minimum of at least 0.7, INVALID at a minimum of at most 0.5, and ABSTAIN otherwise. Llama marker scores are sigmoid-transformed and use the model-card binary threshold, with INVALID below 0.5. PathFinder uses its native per-step error outputs with an evaluation-specific cumulative maximum: VALID at error at most 0.50, INVALID at error at least 0.75, and ABSTAIN between these values. Its token costs count repeated prefix evaluations. Qwen and Llama represent different backbone families and use different process-supervision pipelines: the evaluated Qwen PRM uses consensus-filtered Monte Carlo and model-judged labels (Zhang et al., 2025), while the evaluated Llama checkpoint uses PRM800K-derived supervision (Pisano & Navigli, 2026).

Internal development and calibration sets contain 300 and 400 traces, respectively; the separate internal evaluation contains 978. Rules are selected before evaluation, and score records are retained before joining the reference labels. The public evaluations use all 3,400 ProcessBench cases and

all 6,216 PRMBench rows. Five source identifiers are duplicated in PRMBench, so row identities distinguish examples. Both Qwen and Llama score acquisitions completed all 6,216 rows without backend failure.

ProcessBench was reused for the later aggregation, predictive-exit, and length-sensitivity analyses. Four ProcessBench labels had been seen during schema inspection after the main Qwen rule was fixed. Before the PRMBench evaluation protocol was fixed, its public viewer exposed the schema and a small number of labeled examples. The PRMBench rules and input list were fixed before model acquisition and local scoring. We report these evaluations as fixed-rule transfers and disclose the prior exposure here. The separate timing subsets are disjoint from each other but belong to the same benchmark used for the stored-score evaluation.

D.2 Mean aggregation on the calibration traces

We replay the finite-horizon mean monitor on the 400 internal calibration traces and compare its verifier input cost with full verification of the same traces. This uses the existing calibration observations rather than a new evaluation set.

The monitor removes 25.22% of full-verification input under this mean rule.

The reduction shows that bounding the contribution of the remaining scores can eliminate scoring work on the calibration data as well. Because these observations were used in development, this result is a descriptive replay rather than independent confirmation.

D.3 Internal evaluation

We apply the fixed Qwen minimum-score rule to 978 internal evaluation traces, separate from the development and calibration sets. We measure token saving and selective risk, including risk by predicted class. Risk bounds use a one-sided 98.33% Clopper–Pearson (CP) interval, adjusted for three planned checks; suffixes after certified early rejections are not acquired.

The monitor reduces mean verifier input by 80.65 tokens per trace, from 435.78 to 355.13, yielding 18.51% aggregate saving. Coverage and selective risk remain unchanged at 96.01% and 6.18%, respectively (939 covered examples and 58 errors), with an 8.07% risk upper bound. Predicted-valid cases have 9.43 percentage points higher risk than predicted-invalid cases (Table 8). Separately, the mean per-example normalized saving has an 11.51 percentage-point one-sided Hoeffding lower bound.

Table 8: Internal evaluation, including risk by predicted class. Exact stopping leaves these quantities unchanged.

Subset	Population	Covered	Errors	Risk	CP upper
All	978	939	58	6.18%	8.07%
Predicted invalid	677	677	24	3.55%	5.38%
Predicted valid	262	262	34	12.98%	18.07%

The reduction comes from 534 certified early-invalid cases. The remaining 143 invalid, 262 valid, and 39 abstaining cases reach the final step. The minimum-score proof establishes agreement for the omitted suffixes; exact stopping preserves the verifier's class-dependent error rates as well as its decisions.

D.4 ProcessBench results by data family

We group all 3,400 ProcessBench traces by their four source datasets and apply the same Qwen rule in each group. This analysis measures how savings and inherited verifier quality vary across data families. We also evaluate PathFinder on these traces using its maximum-error rule and repeated-prefix input accounting, reporting its aggregate result separately.

For Qwen, Omni-MATH yields 13.91 percentage points more token saving and a 20.95-point higher early-stop rate than GSM8K (Table 9). MATH and OlympiadBench differ by only 0.15 points in

Table 9: Qwen exact prefix monitor on ProcessBench, by data family.

Family	n	Early stop	Saving	Coverage	Risk
GSM8K	400	41.75%	13.66%	97.00%	6.96%
MATH	1,000	51.90%	22.49%	94.70%	8.13%
OlympiadBench	1,000	55.60%	22.64%	91.40%	9.85%
Omni-MATH	1,000	62.70%	27.57%	90.80%	9.25%

saving. PathFinder reduces repeated-prefix input cost by 40.38% relative to its full verifier, with bootstrap interval $[38.55, 42.17]$, and stops early on 60.35% of traces. Its decisions, 99.0% coverage, and 13.52% selective risk are unchanged.

The family-level variation shows that a single fixed stopping rule can remove different amounts of work across inputs. Coverage and risk describe the underlying verifier and are preserved within each setting. PathFinder's reduction uses repeated-prefix accounting, so its percentage is not a direct comparison with Qwen's token saving under a different execution cost model.

D.5 PRMBench results by error category

We partition all 6,216 PRMBench examples into their nine error categories and replay the fixed Qwen and Llama minimum-score rules. For each verifier and category, we measure early-stop frequency, token saving, coverage, and selective risk against the corresponding full-verifier computation.

Table 10: PRMBench results by error category. Early is the fraction stopping before the final step.

Verifier	Stratum	n	Early	Saving	Coverage	Risk
Qwen	Circular reasoning	758	45.38%	14.58%	83.64%	41.17%
	Confidence error	757	72.92%	12.51%	95.90%	7.99%
	Counterfactual	757	77.68%	20.42%	93.92%	11.95%
	Deception	750	63.73%	14.98%	90.80%	21.00%
	Domain inconsistency	757	73.71%	22.16%	89.96%	12.19%
	Missing condition	756	44.31%	13.25%	85.58%	41.27%
	Multiple solutions	165	6.06%	4.05%	94.55%	6.41%
	Redundancy	758	21.90%	6.78%	88.26%	73.24%
	Step contradiction	758	65.17%	15.59%	86.81%	15.81%
	Overall	6,216	56.71%	14.46%	89.49%	26.98%
Llama	Circular reasoning	758	72.30%	30.55%	100%	24.54%
	Confidence error	757	85.07%	33.14%	100%	5.81%
	Counterfactual	757	86.79%	36.20%	100%	10.04%
	Deception	750	76.13%	31.45%	100%	18.80%
	Domain inconsistency	757	87.58%	36.73%	100%	9.38%
	Missing condition	756	68.92%	32.00%	100%	25.53%
	Multiple solutions	165	26.67%	16.60%	100%	40.00%
	Redundancy	758	57.65%	30.94%	100%	39.71%
	Step contradiction	758	88.65%	36.36%	100%	8.05%
	Overall	6,216	76.53%	32.81%	100%	18.32%

Domain inconsistency yields 18.11 percentage points more token saving than multiple solutions for Qwen, and 20.13 points more for Llama (Table 10). Across all categories, Llama saves 18.35 points more input tokens than Qwen relative to each model's own full computation. All decisions remain unchanged, including the overall selective risks of 26.98% for Qwen and 18.32% for Llama.

Early-stop frequency alone does not determine token saving: Qwen's confidence-error and domain-inconsistency categories differ by only 0.79 points in early-stop rate but by 9.65 points in saving. Savings also depend on how much input remains when a decision becomes fixed. The risk columns identify errors inherited from the full verifiers; the models' different coverage levels must also be considered when comparing these risks.

E Additional Predictive Stopping Comparisons

We evaluate predictive stopping rules that supplement the exact monitor with statistical thresholds on the same ProcessBench observations. Selection uses a 10% risk target. Table 11 reports three selected rules and one mixture-based rule for which no feasible configuration was found under the calibration requirements. Extra saving is measured relative to the exact monitor; "Preempted" counts examples where predictive stopping occurs before a later exact stopping point in the stored full trace.

Table 11: Additional predictive stopping rules on ProcessBench. "Extra" is saving beyond the exact monitor; "Preempted" is the fraction of examples where a predictive exit occurs before a later exact stop.

Boundary	Extra saving	Preempted	Mismatch	Risk	CP upper
Fixed CP	9.59 pp	10.38%	326	11.59%	12.54%
Bonferroni	9.16 pp	10.35%	318	11.41%	12.35%
Polynomial	9.23 pp	10.38%	320	11.41%	12.35%
Mixture	No feasible selected configuration				

The selected predictive rules add 9.16–9.59 percentage points of token saving but change 318–326 decisions that the exact monitor preserves. Selective risk rises by 2.60–2.78 points above the exact monitor's 8.81%, and the one-sided CP upper bounds exceed the 10% selection target by 2.35–2.54 points.

These predictive rules obtain additional savings by allowing exits without an exact decision certificate, including exits that preempt a later exact stop. Their evaluation risk bounds do not meet the selection target, so the added savings must be considered together with the changed decisions and increased observed risk.

F Incremental Execution and Timing Details

F.1 Numerical consistency across execution modes

On 200 diagnostic PRMBench traces, we compare one-pass full inference, independent prefix recomputation, and incremental KV-cache inference using identical tokenized inputs. There are 3,195 scored markers per model. We compare both numerical scores and threshold decisions to determine whether execution mode can be changed without changing the reference output.

Table 12: Numerical comparison of one-pass and incremental execution on 200 diagnostic traces, with 3,195 scored markers per model.

Verifier	Median $\lvert\Delta\rvert$	P95 $\lvert\Delta\rvert$	Max $\lvert\Delta\rvert$	Decision mismatch
Qwen	0	0.00781	0.11035	2/200
Llama	0.00246	0.01438	0.04259	2/200

Switching between one-pass and incremental execution changes the final decision on 1% of diagnostic traces for each model (Table 12). Each of the other pairwise comparisons likewise changes two decisions per model, with differing example identities. The disagreements occur near the threshold: one Qwen trace has minimum scores of 0.4961, 0.5078, and 0.5039 under the three modes, while one Llama trace gives 0.4865, 0.4973, and 0.5099.

Numerical differences across execution modes can cross a decision boundary, so execution mode is part of the fixed reference computation. Accordingly, the main timing experiment holds incremental execution and its scoring schedule fixed in each early/full pair. Its 26–39% time reductions are relative to incremental full verification and do not establish equivalence to one-pass inference.

F.2 SELECTION OF THE SCORING INTERVAL

The initial execution-mode comparison uses PRMBench rows 1–200, scoring-interval selection uses rows 201–400, and timing evaluation uses rows 401–1400. We test intervals of $1, 2, 4, 8, 16$ reasoning steps and select the largest observed time reduction among settings with zero early/full decision mismatch. Thus the 1,000 timing-evaluation cases are separate from implementation selection.

Table 13: Time reduction on the 200-trace scoring-interval selection set. Brackets are paired bootstrap 95% intervals.

Interval (steps)	Qwen reduction	Llama reduction
1	7.11% [4.07, 10.57]	22.22% [17.36, 27.09]
2	6.61% [3.80, 9.93]	21.05% [16.14, 25.81]
4	5.93% [3.46, 8.86]	19.16% [14.13, 24.23]
8	3.73% [1.75, 6.11]	15.97% [11.27, 21.01]
16	2.16% [0.83, 3.76]	9.33% [5.65, 13.16]

Every model–interval combination has zero early/full decision mismatch and zero shared-prefix score difference. Scoring at every step gives the largest point estimate for both models, exceeding the reduction at interval 16 by 4.95 percentage points for Qwen and 12.89 points for Llama (Table 13). We therefore select interval 1.

Coarser intervals provide fewer opportunities to stop once the decision is fixed. The overlapping uncertainty intervals do not establish a statistically distinct optimum; selection follows the stated point-estimate rule. These development estimates are separate from the evaluation results in Table 5.

F.3 TIMING PROCEDURE AND DISTRIBUTION

The evaluation uses 20 warm-up pairs followed by 1,000 paired traces, alternating early and full execution order. CUDA synchronization brackets each timed computation, and model, device, precision, attention implementation, tokenizer, and cache settings are fixed across the pair. Full and early execution use the same scoring boundaries.

Table 7 reports the per-trace time distributions and scoring calls. Peak allocated GPU memory is approximately 13.49 GiB for Qwen and 15.32 GiB for Llama. Both means and medians decrease, and fewer scoring calls accompany the reduction under the fixed execution configuration. These measurements characterize verification of completed traces rather than end-to-end generation time.

F.4 TIMING BY INPUT LENGTH

We partition the same 1,000 timing-evaluation traces into quartiles of full-input token length for each model. Table 14 reports each group's token and time reductions relative to its paired full-verification baseline; the ranges use each model's tokenization.

Table 14: Incremental timing evaluation by full-input token quartile. Ranges are inclusive full-input token counts.

Verifier	Quartile (range)	n	Token reduction	Time reduction
Qwen	Short (122–269)	250	14.98%	23.66%
	Medium-short (270–369)	253	16.76%	24.61%
	Medium-long (370–514)	247	16.84%	23.00%
	Long (515–1,709)	250	24.10%	29.04%
Llama	Short (107–276)	251	22.44%	30.14%
	Medium-short (277–390)	249	24.51%	31.17%
	Medium-long (392–546)	250	25.32%	32.25%
	Long (547–1,977)	250	41.45%	48.97%

All length groups obtain positive time reductions. The longest group exceeds the shortest by 5.38 percentage points in time reduction for Qwen and 18.83 points for Llama; the corresponding differences in token reduction are 9.12 and 19.01 points.

The larger reduction in the longest group is descriptive: trace length also correlates with the number and positions of scoring boundaries. This grouping therefore identifies where savings are larger without isolating length as their cause.

G Scope and Research Statements

G.1 Scope

Savings depend on the aggregation rule, its threshold, and the location of decisive scores. Minimum aggregation can reject after one low score. Mean aggregation requires enough evidence that the remaining scores cannot change the decision. Tighter valid constraints on those scores can expose additional stopping opportunities.

The algorithm preserves a specified final threshold decision, including abstention. Applications that require every step score, a complete error diagnosis, or a ranking based on the unthresholded aggregate have different output requirements. The guarantees use valid score ranges and horizon information, and the earliest-stop result assumes a fixed observation order and the stated completion model. Our empirical study uses mathematical PRMs; timing results compare the same incremental execution for early and full verification. Extending the bounds to other decision rules and selecting a different observation order are natural next steps.

G.2 Reproducibility statement

The accompanying anonymous core implementation provides the stopping rules, score-bound calculations, example traces, and correctness tests. It evaluates supplied score traces. The appendices describe model and dataset sources, main decision rules, cost accounting, and timing procedures; model inference is separate from the core package.

G.3 Ethics statement

The study evaluates existing automated verifiers and collects no new human-subject data.