PATH-CONSISTENCY WITH PREFIX ENHANCEMENT FOR EFFICIENT INFERENCE IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-consistency improves reasoning by aggregating multiple sampled solutions, but generating each trajectory from scratch incurs substantial inference cost. We propose path-consistency, an efficient test-time scaling method that reduces this cost through prefix enhancement, a controlled process that progressively extends a shared reasoning prefix across sampling windows and validates each extension with subsequent samples. Answer agreement guides candidate extensions, while trial and recovery determine whether to commit them or roll back to an earlier prefix. All valid answers from completed windows contribute to the final vote. We derive an exact finite-horizon decomposition of the gains and losses in voting accuracy induced by prefix reuse. We also establish a voting error bound for adaptive sampling with finitely many competing answers under explicit conditional-margin assumptions. Our evaluation targets three models of different scales, Llama-3.1-8B-Instruct, Qwen3-30B-A3B, and DeepSeek-V4-Flash, across five reasoning benchmarks. Across diverse models and reasoning benchmarks, path-consistency reduces generated tokens by 25.8% to 42.0%, averaging 34.4% across the long-chain settings, while largely maintaining accuracy. Combining path-consistency with early-stopping methods yields an additional reduction of approximately 20%. These results position path-consistency as a competitive reasoning-sharing method that also integrates flexibly with early-stopping methods for additional savings.

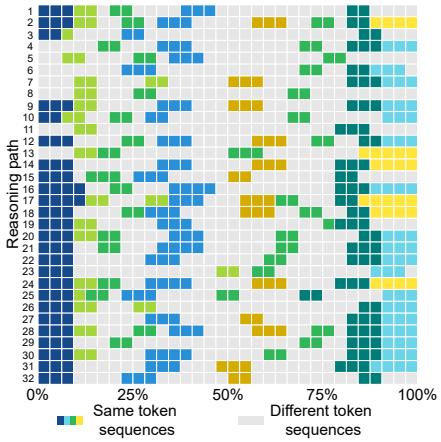
1 INTRODUCTION

Self-consistency (SC) (Wang et al., 2023) samples several reasoning trajectories and aggregates their answers. Its cost grows with the amount of generated reasoning. Even when early samples have found a useful intermediate calculation, later trajectories ordinarily start again from the original question. This suggests reusing part of an earlier solution while continuing to sample diverse suffixes. The potential saving comes from generating fewer new tokens per trajectory. Figure 1 illustrates the observed repetition and contrasts independent sampling with shared-prefix reuse.

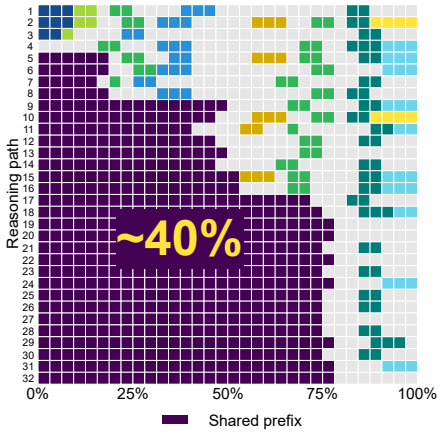
We propose path-consistency (PC), an efficient test-time scaling method for token-efficient reasoning through prefix enhancement, a controlled process that progressively extends and validates a shared reasoning prefix across sampling windows. After a window finishes, answer agreement determines whether to propose a prefix extension. Path-consistency selects a continuous prefix from a trajectory supporting the modal answer and uses it to prompt the next window. Subsequent windows test whether to commit the proposal or recover an earlier prefix. Through this propose–trial–rollback process, the reusable prefix can grow progressively while remaining responsive to later evidence. Later trajectories therefore reuse part of the reasoning already generated and sample new continuations, while all valid answers contribute to the final vote.

Path-consistency has three practical advantages. First, it reduces the amount of new reasoning generated per trajectory while retaining multiple sampled continuations and answer aggregation. Second, it requires no training, fine-tuning, or architectural changes, implementing prefix enhancement entirely at inference time through answer agreement, prefix selection, trial, and recovery. Third, prefix reuse is compatible with different autoregressive language models and can be combined with early-stopping methods for additional token savings. These properties make path-consistency a flexible component for efficient multi-path reasoning.

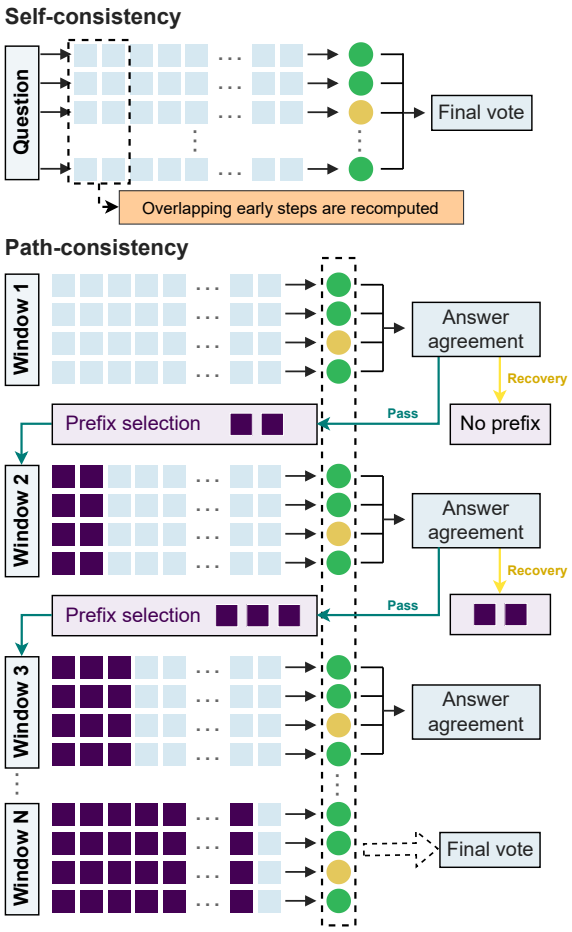
Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping 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.



(a) SC: independently sampled paths



(b) PC: shared-prefix reuse



(c) Overview of SC and PC

Figure 1: (a,b) Self-consistency and path-consistency on the same GSM8K (Cobbe et al., 2021) question using Llama-3.1-8B-Instruct (Grattafiori et al., 2024). Each matrix shows 32 reasoning paths over 32 relative token intervals. Matching colors mark repeated token sequences, purple marks shared prefixes, and gray marks other content. The ∼40% denotes average token savings on GSM8K. (c) Self-consistency samples independent paths, whereas path-consistency progressively reuses prefixes across windows. Circles represent answers, with colors indicating answer categories.

We evaluate path-consistency using three models of different scales, Llama-3.1-8B-Instruct (Grattafiori et al., 2024), DeepSeek-V4-Flash (DeepSeek-AI et al., 2026), and Qwen3-30B-A3B (Yang et al., 2025), across five reasoning benchmarks (Cobbe et al., 2021; Lightman et al., 2023; Zhang & Math-AI, 2024; Zhang & Team, 2025; Rein et al., 2023). Across diverse models and reasoning benchmarks, path-consistency reduces generated tokens by 25.8% to 42.0% while largely maintaining accuracy. Comparisons with reasoning-sharing methods assess path-consistency's accuracy and token efficiency, and combinations with early-stopping methods demonstrate additional savings. Finally, ablation studies examine the roles of source selection and recovery, as well as how the answer agreement threshold governs prefix reuse and informs operating-point selection.

2 Background and Related Work

Self-consistency samples multiple reasoning trajectories and aggregates their answers by voting. Its generation cost can be reduced through early stopping or by sharing previously generated reasoning.

Early-stopping methods. These methods decide when to stop sampling or generation. adaptive-consistency (Aggarwal et al., 2023) uses cumulative answer agreement, whereas early-stopping self-consistency (Li et al., 2024) uses agreement within a sampling window to stop further sampling. DeepConf (Fu et al., 2025) uses model-internal confidence to filter traces during or after generation, although only online termination avoids generating further tokens. Path-consistency instead controls the starting prefix of later trajectories and can complement early-stopping methods while retaining their stopping and aggregation rules.

Reasoning-sharing methods. These methods reuse information from sampled trajectories to guide subsequent reasoning. Progressive-Hint Prompting (Zheng et al., 2024) reuses previous answers as hints. PoLR (Jindal et al., 2026) clusters short prefixes and expands trajectories in the dominant cluster. CPT (Wang et al., 2026) pools and deduplicates intermediate information for sharing across branches. Path-consistency belongs to this family but shares one continuous reasoning prefix within each sampling window. Answer agreement gates its extension, prefix selection chooses its source, and trial and recovery determine whether to retain or retract it.

3 PATH-CONSISTENCY

Path-consistency implements prefix enhancement by progressively extending a continuous shared prefix across sampling windows and validating each extension with subsequent samples. The mechanism consists of three components: ❶ answer agreement, ❷ prefix selection, ❸ trial and recovery, as illustrated in Figure 2 and described below, where recovery is implemented by prefix rollback. The default main setting uses a trajectory budget $B = 32$, window size $m = 4$, and prefix levels $\mathcal{R} = \{0, 0.2, 0.4, 0.6, 0.8\}$, where each $r \in \mathcal{R}$ specifies the fraction of the reasoning trajectory retained as the prefix. All trajectories within a window use the same prefix. Path-consistency updates its prefix and agreement statistics only after all trajectories in that window have finished. Self-consistency and path-consistency use the same maximum budget and final majority-vote rule.

3.1 ANSWER AGREEMENT GATE

Let $\mathcal{V}$ contain the valid answers admitted by previous successful windows, let $n = |\mathcal{V}|$, and let $v_\star$ be the count of its modal answer $a_\star$. As illustrated in Figure 2(a), we use the modal-vs-rest agreement score

$$c(n, v_\star) = 1 - I_{1/2}(v_\star + 1, n - v_\star + 1), \tag{1}$$

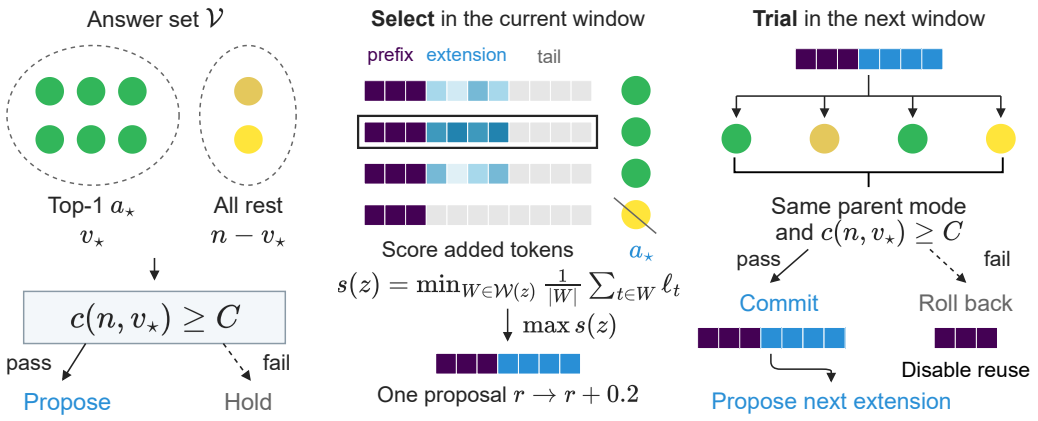
where I_x is the regularized incomplete beta function (Aggarwal et al., 2023). For a fixed answer category with independent Bernoulli observations and a uniform prior, this is the posterior probability that its mass exceeds $1/2$ (Gelman et al., 1995). In path-consistency the category is selected from the data, successive windows use different prefixes, and unsuccessful windows are excluded from $\mathcal{V}$. Thus Equation 1 is an operational agreement score, not a calibrated posterior over correctness or an anytime error bound.

A nonempty answer set with $c \geq C$ permits a proposal. We use $C = 0.8$ in the main experiments as the default operating point. At $C = 1$, the Beta score remains below the threshold for any finite sample size in exact arithmetic, recovering self-consistency. A setting of $C = 0$ removes the confidence barrier but still requires a valid modal answer and a feasible prefix.

3.2 PREFIX SELECTION

Once answer agreement permits an extension, path-consistency selects a source trajectory from the previous completed, accepted window. Eligible trajectories must support the modal answer $a_\star$. For a current prefix level $r < 0.8$, each trajectory proposes a candidate prefix z ending at $\lfloor (r + 0.2)L \rfloor$, where L counts all reasoning tokens in that trajectory, including the inherited prefix but excluding the final answer. Thus each proposal targets the next 20% level while preserving the existing prefix. The endpoint may be moved backward by up to 32 tokens to avoid explicit answer markers and unbalanced braces or code fences, with punctuation boundaries preferred. Only candidates that extend the current prefix are retained.

As illustrated in Figure 2(b), we use local confidence selection (LCS) to rank these candidates using only their newly added tokens. Let ℓ_t be the log probability assigned to token t when it was sampled,



(a) Answer agreement **(b) Prefix selection** **(c) Trial and recovery**

Figure 2: (a) Answer agreement determines whether to propose a prefix extension. (b) Local confidence selection chooses an extension from eligible trajectories. (c) Trial and recovery validate the proposal, enabling further extension on success or returning to an earlier prefix on failure. Purple denotes inherited prefixes and blue denotes new extensions. Circles represent answers, with colors indicating answer categories.

and let $\mathcal{W}(z)$ contain the sliding token windows over the added segment of candidate z. We define

$$s(z) = \min_{W \in \mathcal{W}(z)} \frac{1}{|W|} \sum_{t \in W} \ell_t, \tag{2}$$

where W is a token window and $|W|$ is its number of scored tokens. The score is the lowest mean log probability across these windows, so choosing the largest $s(z)$ favors candidates whose least confident segment has the highest score. Scoring excludes the inherited prefix, which is shared by all candidates. We use windows of 128 tokens with stride 64 and include the final window. Segments shorter than 128 tokens are scored as a whole.

3.3 Trial, Commitment, and Rollback

As illustrated in Figure 2(c), a proposed prefix is used for the next sampling window as a trial. After that window finishes, we compute the agreement score in Equation 1 using $\mathcal{V}$ together with the trial's valid answers. The trial succeeds if the modal answer remains the same as when the prefix was proposed and the score is at least C. We then commit the prefix, add the trial's valid answers to $\mathcal{V}$, and reset the consecutive-failure counter. Otherwise we discard the trial and return to the last committed prefix. A failed window sampled from an already committed prefix pops one level. Two consecutive failures clear the stack and disable further prefixing for that question. A failed window is excluded from future gate and source selection, but its valid answers remain in the final vote and its entire cost remains charged. Earlier admitted answers are not erased when the stack is shortened.

The same agreement score also gates the next prefix extension. Since a successful trial has already met the threshold, no additional agreement calculation is needed after commitment. If sampling budget remains and the prefix level is below 0.8, the accepted window supplies candidate extensions at the next level using the selection procedure above. Thus one window can both validate the preceding proposal and supply the next extension.

4 Voting Accuracy under Prefix Reuse

Prefix reuse reduces repeated generation but also changes the answer distributions and the dependence between sampled trajectories. We study how these changes affect final voting accuracy relative to self-consistency, and under what conditions voting remains reliable with adaptive sampling. First, we derive an exact decomposition of the accuracy difference into the expected contributions of

individual windows. Second, we bound voting error when the correct answer has a sufficient accumulated conditional advantage over every competing answer. Together, these results characterize the benefits and risks of prefix reuse without assuming that answer agreement implies correctness. All probabilities concern a fixed question with correct answer $\hat{a}$, and multiple incorrect answers are allowed. The analysis uses standard telescoping arguments and concentration inequalities for bounded martingale differences (Hoeffding, 1963; Azuma, 1967), with proofs in Appendix A.

4.1 Decomposing the effect of prefix reuse on accuracy

Let H_j be the complete history before window $j \in \{0, \dots, J-1\}$, including traces and controller state. A kernel $K_j^P(\cdot \mid h)$ generates the next window under path-consistency. Define $K_j^S(\cdot \mid h)$ to generate an unprefixed self-consistency window from any history h, including histories created by path-consistency. Let $V_j^S(h) \in [0,1]$ be the probability of final voting correctness if all remaining windows use these self-consistency kernels. The terminal value V_J^S is the correctness indicator under a fixed tie rule. This construction retains all earlier votes when switching to self-consistency.

Theorem 1 (Exact intervention decomposition)**.** *For a fixed finite budget and common initial state, define*

$$d_j(h) = \int V_{j+1}^S(h,u)\,[K_j^P(du \mid h) - K_j^S(du \mid h)]. \tag{3}$$

Then the difference in final accuracy is exactly

$$A_P - A_S = \sum_{j=0}^{J-1} \mathbb{E}_{H_j \sim P}[d_j(H_j)]. \tag{4}$$

In particular, if every expected increment is nonnegative, $A_P \geq A_S$. In general,

$$|A_P - A_S| \leq \min\left\{1, \sum_{j=0}^{J-1} \mathbb{E}_P\, \mathrm{TV}(K_j^P(\cdot \mid H_j), K_j^S(\cdot \mid H_j))\right\}. \tag{5}$$

Writing $d_j^+ = \max(d_j, 0)$ and $d_j^- = \max(-d_j, 0)$ separates the accuracy change into total expected benefit minus total expected harm. This identity includes selection, correlation, failed trials, and rollback through the history and kernels. It makes no small-perturbation approximation. The sufficient condition is deliberately explicit: agreement does not establish the sign of d_j, since V^S depends on correctness and remaining votes. A root window has identical conditional kernels and zero increment, even after earlier prefixes changed its history. Rollback can eliminate further prefix-induced increments but cannot undo earlier votes.

4.2 Bounding voting error under adaptive sampling

Consider a fixed finite answer alphabet $\mathcal{A}$ containing $\hat{a}$. Invalid outputs may be included as a separate symbol. Expose the B sampled answers sequentially, using a filtration $\mathcal{F}_i$ that includes the information controlling their generation. For competitor $a \neq \hat{a}$, set

$$X_i(a) = \mathbf{1}\{A_i = \hat{a}\} - \mathbf{1}\{A_i = a\}, \qquad \mu_i(a) = \mathbb{E}[X_i(a) \mid \mathcal{F}_{i-1}].$$

Theorem 2 (Plurality under conditional margins)**.** *If deterministic constants $\Gamma_a > 0$ satisfy $\sum_{i=1}^B \mu_i(a) \geq \Gamma_a$ almost surely for every incorrect answer, then*

$$\mathbb{P}(\widehat{A} \neq \hat{a}) \leq \min\left\{1, \sum_{a \neq \hat{a}} \exp\left(-\frac{\Gamma_a^2}{2B}\right)\right\}. \tag{6}$$

Ties are treated as failures, so the bound also covers any deterministic tie rule. If invalid answers are discarded, bounding a loss to the invalid symbol is conservative.

The bound accommodates adaptive, non-identically distributed samples and requires an accumulated conditional advantage over each incorrect answer, rather than a correct-answer probability above $1/2$. It therefore allows errors to be distributed among multiple answers. This condition is not

established by observed marginal accuracy or a high Beta score, and an early incorrect prefix can reverse subsequent conditional advantages. The bound is informative only for sufficiently large margins and may be vacuous at $B = 32$. Applying it to open-ended answers also requires justification of the finite-support assumption.

Together, the two results distinguish accuracy preservation from voting reliability. Theorem 1 shows that prefix reuse preserves accuracy relative to self-consistency when its total expected benefit offsets its total expected harm. Theorem 2 bounds the absolute voting error under sufficient conditional advantages, even when sampling adapts to earlier outcomes. Whether prefix selection and recovery achieve a favorable accuracy–cost trade-off is assessed empirically in Section 5.

5 Evaluation

5.1 Experimental setup

Models. We use Llama-3.1-8B-Instruct (Grattafiori et al., 2024), DeepSeek-V4-Flash (DeepSeek-AI et al., 2026), and Qwen3-30B-A3B (Yang et al., 2025). DeepSeek and Qwen operate with thinking enabled. Llama and Qwen are served through vLLM (Kwon et al., 2023) on RTX 4090. DeepSeek is accessed through its official API.

Datasets. We evaluate Path-consistency on five widely used reasoning benchmarks covering mathematical and scientific reasoning, GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2023), AIME 2024 (Zhang & Math-AI, 2024), AIME 2025 (Zhang & Team, 2025), and GPQA-Diamond (GPQA-D) (Rein et al., 2023). We report voting accuracy and mean generated tokens per question.

Hyperparameters. Self-consistency and path-consistency use a trajectory budget of $B = 32$ and a sampling window of $m = 4$. Path-consistency uses the default agreement threshold $C = 0.8$ and the fixed prefix levels $\mathcal{R} = \{0, 0.2, 0.4, 0.6, 0.8\}$. For Llama, we use temperature 0.6, top-$p = 0.9$, and a 2,048-token generation limit. Qwen uses temperature 0.6, top-$p = 0.95$, top-$k = 20$, and a 32,768-token limit. DeepSeek uses high reasoning effort and a 32,768-token limit.

5.2 Progressive prefix reuse and token savings

We use Llama-3.1-8B-Instruct on GSM8K to examine how path-consistency progressively extends a shared prefix and reduces the cost of later sampling windows. Path-consistency reduces mean generated tokens from 8,316 to 5,039 and inclusive generation time from 54.4 to 33.8 seconds per question, with final accuracy of 91.74% versus 92.12% for self-consistency.

Table 1: Detailed statistics of progressive prefix reuse across nominal prefix levels, showing the distribution of sampling windows, shared-prefix lengths, newly generated tokens, relative token savings, and generation-time reductions on Llama-3.1-8B-Instruct/GSM8K.

Level	Windows	Prefix	Tokens	Save (%)	Time (%)
0 (0%)	2,483	0.0	1252.6	0.0	0.0
1 (20%)	1,292	39.4	859.3	−14.3	−13.4
2 (40%)	1,201	87.2	641.7	−34.7	−35.8
3 (60%)	1,182	132.7	445.8	−54.4	−55.6
4 (80%)	4,394	177.5	256.7	−73.3	−73.6

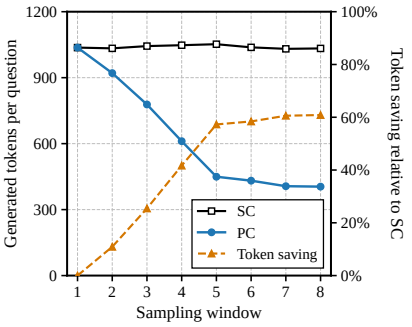


Figure 3: Tokens and savings by window.

Table 1 relates the extent of prefix reuse to the cost of subsequent generation. Windows are grouped by the prefix level used, with level 0 including returns to the empty prefix. Prefix length counts one shared prefix, while generated tokens sum the four continuations per question. Across levels 1 to 4, mean prefix length increases from 39.4 to 177.5 tokens and generated tokens decrease from 859.3 to 256.7 per window. Relative to matched self-consistency windows, token savings rise from 14.3% to 73.3%, accompanied by generation-time reductions of 13.4% to 73.6%. Thus, higher prefix levels are associated with less remaining generation and larger measured savings.

Table 2: Progressive prefix reuse on a GSM8K example with Llama-3.1-8B-Instruct. Black text is the reused prefix and blue text is newly generated. One recorded trajectory is shown per window.

Question: Geb is 10 less than half the age of Haley. If Haley is 26 years old, how old is Geb?

Window 1, prefix level 0%
To find Geb's age, we need to follow these steps: 1. First, we find half of Haley's age. Haley is 26 years old, so half of her age would be 26 / 2 = 13. 2. Next, we need to subtract 10 from this half age to find Geb's age. So, 13 - 10 = 3. Therefore, Geb is 3 years old. (0 prefix tokens; 94 generated tokens)

Window 2, prefix level 20%
To find Geb's age, we need to follow these steps: 1. First, we need to find half of Haley's age. Haley is 26 years old, so half of 26 is 26 / 2 = 13. 2. Since Geb is 10 less than half the age of Haley, we need to subtract 10 from the half of Haley's age. So, Geb's age is 13 - 10. 3. Now, let's calculate 13 - 10 = 3. Therefore, Geb is 3 years old. (17 prefix tokens; 104 generated tokens)

Window 3, prefix level 40%
To find Geb's age, we need to follow these steps: 1. First, we need to find half of Haley's age. Haley is 26 years old, so half of her age would be 26 / 2 = 13. 2. Now that we have half of Haley's age, we need to subtract 10 from this number to find Geb's age. 3. So, Geb's age would be 13 - 10 = 3. Therefore, Geb is 3 years old. (34 prefix tokens; 75 generated tokens)

Window 4, prefix level 60%
To find Geb's age, we need to follow these steps: 1. First, we need to find half of Haley's age. Haley is 26 years old, so half of her age would be 26 / 2 = 13. 2. Since Geb is 10 less than half the age of Haley, we need to subtract 10 from the half of Haley's age. So, Geb's age would be 13 - 10 = 3. Therefore, Geb is 3 years old. (52 prefix tokens; 56 generated tokens)

Window 5, prefix level 80%
To find Geb's age, we need to follow these steps: 1. First, we need to find half of Haley's age. Haley is 26 years old, so half of her age would be 26 / 2 = 13. 2. Since Geb is 10 less than half the age of Haley, we need to subtract 10 from the half of Haley's age. So, 13 - 10 = 3. Therefore, Geb is 3 years old. (79 prefix tokens; 26 generated tokens)

Figure 3 shows how these savings develop over successive sampling windows. Both methods start from the same unprefixed window. As shared prefixes extend, mean generated tokens under path-consistency decline from 1,036.8 in window 1 to 404.5 in window 8. The corresponding token saving reaches 60.9% in the final window. By that point, 89.5% of questions use a prefix, and mean prefix length across all questions reaches 158.6 tokens. Unlike the level-based groups in Table 1, this window-level view includes questions at different prefix levels and those without a prefix.

Table 2 illustrates the same progression through actual generated text. In this example, the shared prefix grows from 0 to 79 tokens over five windows. By the fourth window, it already contains the calculation of half of Haley's age, so later generation can continue from that intermediate result. By the fifth window, only the remaining subtraction and final answer need to be generated, using 26 tokens compared with 94 in the initial displayed trajectory. The expanding black prefix and shorter blue continuation make the source of the savings explicit. Together, the aggregate results and the example show how reusing intermediate reasoning reduces the generation required by later trajectories.

5.3 Main results

Table 3 evaluates DeepSeek-V4-Flash and Qwen3-30B-A3B, both generating long reasoning chains with thinking enabled. Across the ten reported settings, path-consistency reduces generated tokens by 25.8% to 42.0%, averaging 34.4% with equal weight assigned to each setting, while largely maintaining voting accuracy.

DeepSeek matches self-consistency's accuracy on nearly all benchmarks with an average token saving of 29.8%. Qwen matches self-consistency on MATH-500, both AIME benchmarks, and GPQA-D, with a 0.15 percentage-point decrease on GSM8K. Thus, the efficiency benefit appears across both models rather than being confined to a single setting.

Table 3: Accuracy and token cost of long-chain reasoning with thinking enabled.

Model	Dataset	Accuracy (%)			Tokens (# per problem)		
		Pass@1	SC	PC	SC	PC	Save (%)
DeepSeek-V4-Flash	GSM8K	96.74	96.82	96.82	12,514	8,230	−34.2
	MATH-500	86.60	88.00	87.60	41,166	26,968	−34.5
	AIME 2024	93.33	96.67	96.67	288,230	213,162	−26.0
	AIME 2025	86.67	100.00	100.00	351,230	260,523	−25.8
	GPQA-D	83.33	89.90	89.90	309,325	221,526	−28.4
Qwen3-30B-A3B	GSM8K	96.59	96.74	96.59	57,859	33,564	−42.0
	MATH-500	92.80	93.40	93.40	156,956	92,106	−41.3
	AIME 2024	73.33	83.33	83.33	443,797	261,149	−41.2
	AIME 2025	73.33	83.33	83.33	559,278	360,577	−35.5
	GPQA-D	63.64	64.14	64.14	232,994	151,886	−34.8

Both models also retain gains over Pass@1 on the AIME benchmarks, showing that reduced generation cost need not eliminate the benefit of multiple sampled paths. Together with Section 5.2, these results support prefix reuse for both shorter reasoning traces and long thinking chains as a way to reduce token cost while retaining the benefits of answer aggregation.

5.4 Comparisons and composition

Table 4 compares path-consistency with reasoning-sharing methods and evaluates its combination with early-stopping methods on Llama-3.1-8B-Instruct and GSM8K. Self-consistency with a reduced budget of $B = 20$ provides an additional comparison at a similar token cost. For path-consistency and self-consistency, we reuse the same initial root window to provide a partially aligned comparison, whereas the other baselines are evaluated with independent inference runs. Consequently, small differences in accuracy may also reflect sampling variability and should not be overinterpreted.

Path-consistency reduces generated tokens by 39.4% relative to self-consistency while largely maintaining accuracy. Compared with simply reducing the trajectory budget to 20, it achieves higher accuracy with fewer tokens, supporting prefix reuse as an alternative to discarding sampling opportunities. Among the reasoning-sharing methods, path-consistency uses 14.1% fewer tokens than PoLR, with closely comparable accuracy. CPT attains slightly higher accuracy in this run but generates slightly more tokens than self-consistency.

Prefix reuse also complements early stopping. Combined with adaptive-consistency, it reduces mean tokens from 4,035 to 3,003, an additional 25.6% saving, while accuracy changes only slightly. Combined with early-stopping self-consistency, it reduces mean tokens from 4,299 to 3,501, an additional 18.6% saving, again with only a small change in accuracy. Thus, path-consistency reduces generation cost both as a standalone reasoning-sharing method and in combination with early-stopping methods, providing further savings of 18.6% to 25.6% over the latter in this evaluation.

Table 4: Method comparisons and early-stopping combinations on Llama-3.1-8B-Instruct/GSM8K.

Method	Accuracy (%)	Δ (pp)	Tokens (#)	Save (%)
Self-consistency	92.12	-	8,316	-
Self-consistency ($B = 20$)	91.28	−0.84	5,214	−37.3
PoLR (Jindal et al., 2026) ($L_p = 128$)	92.19	+0.07	5,863	−29.5
CPT (Wang et al., 2026) (32 workers)	92.80	+0.68	8,411	+1.1
Path-consistency	91.74	−0.38	5,039	−39.4
Adaptive-consistency (Aggarwal et al., 2023)	92.04	−0.08	4,035	−51.4
+ Path-consistency	92.27	+0.15	3,003	−63.9
Early-stopping self-consistency (Li et al., 2024)	92.49	+0.37	4,299	−48.3
+ Path-consistency	92.27	+0.15	3,501	−57.9

5.5 Ablation studies

Contributions of prefix selection and recovery. Table 5 examines local confidence selection and recovery on DeepSeek-V4-Flash/MATH-500 by replacing the selector with uniform selection, disabling recovery, or combining both changes. All configurations use the same questions and shared self-consistency roots. The remaining settings are fixed.

Table 5: Effects of prefix selection and recovery on DeepSeek-V4-Flash/MATH-500.

Method	Accuracy (%)	Δ (pp)	Tokens (# per problem)	Save (%)
Self-consistency	88.00	-	41,166	-
Path-consistency (full)	87.60	−0.40	26,968	−34.5
w/ uniform selection	87.20	−0.80	25,680	−37.6
w/o recovery	86.60	−1.40	24,757	−39.9
w/ uniform selection, w/o recovery	86.40	−1.60	25,405	−38.3

The full configuration achieves the highest accuracy among the path-consistency variants at 87.6%, while saving 34.5% of generated tokens. Uniform selection lowers accuracy to 87.2%, and disabling recovery lowers it to 86.6% and 86.4% under LCS and uniform selection, respectively. The variants save more tokens, but at the cost of lower accuracy. These results support retaining both components when accuracy preservation is prioritized, with recovery showing the larger observed contribution in this setting.

Sensitivity to the agreement threshold. Table 6 examines the trade-off between accuracy and token savings across agreement thresholds on DeepSeek-V4-Flash/GSM8K, with $C = 1$ corresponding to self-consistency. The default $C = 0.8$ preserves its 96.82% accuracy while reducing generated tokens by 34.2%. Lowering C to 0.7 and 0 increases savings to 40.8% and 43.5%, respectively, with small accuracy decreases of 0.08 and 0.23 percentage points.

Table 6: Effect of the agreement threshold on DeepSeek-V4-Flash/GSM8K.

Threshold (C)	Accuracy (%)	Δ (pp)	Tokens (# per problem)	Save (%)
1.0 (Self-consistency)	96.82	-	12,514	-
0.8	96.82	0.00	8,230	−34.2
0.7	96.74	−0.08	7,399	−40.8
0.0	96.59	−0.23	7,070	−43.5

A lower threshold permits prefix reuse under weaker answer agreement, whereas a higher threshold requires stronger agreement and makes reuse more conservative. The threshold therefore offers a direct way to express user priorities. Lower values favor more efficient generation through greater token savings, while higher values favor accuracy preservation.

6 Conclusion

Path-consistency reduces repeated reasoning through prefix enhancement, in which answer agreement, prefix selection, and trial and recovery progressively extend and validate a shared reasoning prefix. Our analysis characterizes how prefix reuse affects voting accuracy and bounds voting error under explicit conditional-margin assumptions. Experiments with three models across five reasoning benchmarks show substantial token savings while largely maintaining accuracy, with further savings when combined with early-stopping methods. These results support prefix reuse as a practical way to make multi-path reasoning more efficient.

Limitations. Shared prefixes may propagate undetected errors, so answer agreement does not guarantee correctness. Path-consistency aims to preserve accuracy rather than ensure it. Parallelism is limited by the window size, since successive windows depend on earlier outcomes. Local confidence selection also requires output token log probabilities. APIs that do not expose them require a fallback without confidence scores, such as uniform random selection among eligible prefixes.

Reproducibility Statement

To support reproduction of our results, Section 5.1 specifies the evaluated models and benchmarks, decoding settings, trajectory budget, sampling window size, agreement threshold, and prefix levels. Appendix C details the controller transitions, prompting setup, deterministic preprocessing such as GPQA option shuffling, and ablation settings. Token savings and generation-time measurements use the accounting conventions in Appendix C.3, including the costs of unprefixed and failed windows. Appendices A and B provide proofs and additional theoretical analysis. The supplementary materials include runnable code for the API experiments, pinned software dependencies, and instructions for setting up the Python environment and running the implementation and evaluation of path-consistency.

AI Use Statement

In this work, we used generative AI tools, primarily ChatGPT, for language polishing and rewriting, formatting and consistency checks, and organizing experimental data and results. We have not used generative AI tools for generating research ideas, conducting experiments, or making scientific conclusions, and the remaining disclosure tasks are not applicable to this work. Additionally, generative AI tools were used to improve the clarity and presentation of the manuscript. We have reviewed all AI-assisted work. All AI-assisted text was manually reviewed and revised by the authors, and the organized experimental data and reported results were verified against the original experimental records. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

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*, pp. 12375–12396, 2023.

Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367, 1967.

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, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue

Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, Dongjie Ji, Erhang Li, Fang Wei, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanting Chen, Guoai Cao, Guolai Meng, Guowei Li, Han Yu, Han Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoling Zhang, Haoming Luo, Haoran Wei, Haotian Yuan, Haowei Zhang, Haowen Luo, Haoyu Chen, Haozhe Ji, Hengqing Zhang, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J Yang, JQ Zhu, Jia Luo, Jia Song, Jia Yu, Jialiang Huang, Jialu Cai, Jian Liang, Jiangting Zhou, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jieyu Yang, Jin Chen, Jin Yan, Jingchang Chen, Jingli Zhou, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jingzi Zhou, Jinhua Zhu, Jiping Yu, Joseph Sun, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junmin Zheng, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Leyi Xia, Li Zhang, Liang Zhao, Lihua Guo, Lingxiao Luo, Linwang Ma, Linyan Zhu, Litong Wang, Liyu Cai, Liyue Zhang, Longhao Chen, MS Di, MY Xu, Max Mei, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Mingxu Zhou, Minmin Han, Ning Wang, Panpan Huang, Panpan Wang, Peixin Cong, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qiwei Jiang, Rui Tian, Ruifan Xu, Ruijie Lu, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqian Chen, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, Ruyi Chen, SH Liu, Shanghao Lu, Shangmian Sun, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoqing Wu, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Shuying Yu, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin, Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu, Tun Wang, W Zhang, WL Xiao, Wangding Zeng, Wei An, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Yang, Wenlve Huang, Wenqing Hou, Wentao Zhang, Wenting Ma, Xi Gao, Xiang He, Xiangwen Wang, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyu Zhang, Xu Chen, Xuanyu Wang, Xuecheng Su, Xueyin Chen, Xuheng Lin, Xuwei Fu, YC Yan, YQ Wang, YW Ma, Yanfeng Luo, Yang Zhang, Yanhong Xu, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Xu, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Shao, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yijia Wu, Yiliang Xiong, Yiling Ma, Ying He, Ying Tang, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, YuKun Li, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuanhao Li, Yuduan Wang, Yuehan Yang, Yuer Xu, Yuhan Wu, Yuhao Meng, Yuheng Zou, Yukun Zha, Yunfan Xiong, Yupeng Chen, Yuping Lin, Yuqian Cao, Yuqian Wang, Yushun Zhang, Yuting Yan, Yutong Lin, Yuxian Gu, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhen Huang, ZF Wu, Zehao Wang, Zehua Zhao, Zehui Ren, Zekai Zhang, Zhangli Sha, Zhe Fu, Zhe Ju, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zheren Gao, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhixuan Chen, Zhiyu Wu, Zhizhou Ren, Zhongyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihua Qu, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Ziyi Wan, Zizheng Pan, and Zongqing Yao. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence, 2025. URL https://arxiv.org/abs/2508.15260.

Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*. chapman and hall/cRc, 1995.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella

Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

Ishan Jindal, Sai Prashanth Akuthota, Jayant Taneja, and Sachin Dev Sharma. The path of least resistance: Guiding llm reasoning trajectories with prefix consensus, 2026. URL https://arxiv.org/abs/2601.21494.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

David A Levin and Yuval Peres. *Markov chains and mixing times*. American Mathematical Society, 2026.

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 *The Twelfth International Conference on Learning Representations*, 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, 2023. URL https://arxiv.org/abs/2305.20050.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 975–984, 2020.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, 2021.

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, 2023. URL https://arxiv.org/abs/2311.12022.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, 2015.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, July 2023.

Xinglin Wang, Hao Lin, Shaoxiong Feng, Peiwen Yuan, Yiwei Li, Jiayi Shi, Yueqi Zhang, Chuyi Tan, Ji Zhang, Boyuan Pan, Yao Hu, and Kan Li. Share more, search less: Collaborative parallel thinking for efficient test-time scaling, 2026. URL https://arxiv.org/abs/2605.27030.

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 *The Eleventh International Conference on Learning Representations*, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (aime) 2025, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. Progressive-hint prompting improves reasoning in large language models, 2024. URL https://arxiv.org/abs/2304.09797.

The appendix provides proofs and additional analysis in Appendices A and B, evaluation details in Appendix C, and sentence-level prefix experiments in Appendix D.

A Proofs and finite-budget analysis

A.1 Proof of Theorem 1

Define the self-consistency continuation value at any history by backward recursion.

$$V_j^S(h) = \int V_{j+1}^S(h,u) K_j^S(du \mid h), \qquad V_J^S(h) = \mathbf{1}\{\text{final vote in } h \text{ is correct}\}.$$

The notation (h, u) includes the deterministic bookkeeping after the next window. Under path-consistency, conditional expectation gives

$$\mathbb{E}_P[V_{j+1}^S(H_{j+1}) - V_j^S(H_j) \mid H_j = h] = d_j(h).$$

Taking expectations and summing over j telescopes. The initial value is A_S and the expected terminal value is A_P, proving Equation 4. This argument needs only bounded measurable values and a finite horizon. It does not require independent windows, a binary answer distribution, or a correct selected prefix. Since $0 \le V_{j+1}^S \le 1$, the standard variational characterization of total variation distance (Levin & Peres, 2026)

$$\mathrm{TV}(P,Q) = \sup_{0 \le f \le 1} |\mathbb{E}_P f - \mathbb{E}_Q f|$$

implies $|d_j(h)| \le \mathrm{TV}(K_j^P, K_j^S)$. The triangle inequality and the fact that two probabilities differ by at most one prove Equation 5.

Scope for stopping controllers. The displayed result compares fixed-budget self-consistency and path-consistency. A stopping controller can be represented by an absorbing terminal state and padding to a common finite maximum horizon. Both the reference value and the reference transition kernel must then include that parent's stopping rule. This extension does not allow replacing a confidence-weighted parent's final vote by a majority vote while claiming to compare the same parent.

A.2 Proof of Theorem 2

Fix a competitor a. The differences $D_i = X_i(a) - \mu_i(a)$ are martingale differences. Conditional on $\mathcal{F}_{i-1}$, their values lie in an interval of width at most two, since $X_i(a) \in [-1, 1]$. The conditional bounded-variable exponential inequality therefore gives

$$\mathbb{E}[\exp(-\lambda D_i) \mid \mathcal{F}_{i-1}] \le \exp(\lambda^2/2), \quad \lambda > 0.$$

This inequality follows by the convex upper bound on the exponential between the two interval endpoints and maximizing over a zero-mean two-point distribution. Iterating conditional expectations yields $\mathbb{E}\exp(-\lambda \sum_i D_i) \le \exp(B\lambda^2/2)$. If the correct answer fails to outvote a, then $\sum_i X_i(a) \le 0$. By the almost-sure lower bound on accumulated conditional means, this implies $\sum_i D_i \le -\Gamma_a$. Markov's inequality now bounds its probability by

$$\exp(-\lambda\Gamma_a + B\lambda^2/2).$$

Taking $\lambda = \Gamma_a/B$ gives $\exp(-\Gamma_a^2/(2B))$. If the final vote is incorrect or tied, at least one incorrect category has count no smaller than the correct category. Union-bounding these events proves the result. Voting failure is covered by the union of competitor events.

What the assumption rules out. An unconditional gap $P(A_i = \hat{a}) > P(A_i = a)$ is insufficient. Conditioning on an early misleading prefix may reverse the gap, violating the almost-sure lower bound. Similarly, the observed mode need not equal $\hat{a}$. The Beta gate operates on the mode, whereas the theorem refers to the externally defined correct answer. An empirical evaluation of the assumption needs correctness information and conditional resampling beyond the current trajectories.

A.3 Exact binary illustration without Taylor expansion

For comparison with a single-promotion analysis, suppose n root answers are independent Bernoulli with correctness probability p_0, and the remaining $r = B - n$ answers are conditionally independent with correctness probability q_k after observing k correct root answers. Fix a tie-as-failure convention. Define

$$Q_r(p, t) = \sum_{j=\max(0,t)}^{r} \binom{r}{j} p^j (1-p)^{r-j},$$

with $Q_r(p, t) = 0$ for $t > r$ and $Q_r(p, t) = 1$ for $t \le 0$. Then

$$A_{\mathrm{PC}} = \sum_{k=0}^{n} \binom{n}{k} p_0^k (1-p_0)^{n-k} Q_r(q_k, \lfloor B/2 \rfloor + 1 - k), \tag{7}$$

$$A_{\mathrm{SC}} = Q_B(p_0, \lfloor B/2 \rfloor + 1). \tag{8}$$

Both parts contribute to the final vote. A gate that does not promote sets $q_k = p_0$ for that history. The formula is exact for this idealized model, including its tie convention; it is not a model of all levels of path-consistency.

For a concrete failure construction take $B = 32, n = 4, p_0 = 0.6, C = 0.8$. If the four root answers are all wrong, their agreement score is $1 - 2^{-5} = 0.96875$. Let the selected wrong prefix force all subsequent answers to be wrong ($q_0 = 0$), while all other histories retain $q_k = 0.6$. The accuracy loss relative to self-consistency is exactly

$$(0.4)^4 Q_{28}(0.6, 17) > 0.$$

Numerically, self-consistency accuracy in this construction is 0.835203095 and the intervention accuracy is 0.821096895, a loss of 1.41 percentage points. Thus even a model whose unprefixed correctness probability exceeds one half can be harmed by a gate-passing prefix. The construction does not assume symmetric beneficial and harmful effects. That symmetry is an additional modeling assumption, not an inference-time property. More generally, a consensus-only controller has no access to the sign of a prefix's correctness effect.

A.4 Shared-prefix dependence

Let a random shared prefix Z induce correctness probability p_Z, and let Y_1, Y_2 be correctness indicators of two suffixes that are conditionally independent given Z. Then $\mathbb{E}[Y_1 Y_2 \mid Z] = p_Z^2$ and $\mathbb{E}Y_i = \mathbb{E}p_Z$, so

$$\mathrm{Cov}(Y_1, Y_2) = \mathrm{Var}(p_Z). \tag{9}$$

For a window of size m with exchangeable correctness indicators, marginal correctness probability $\bar{p}$, and pairwise correlation ρ, summing variances and covariances gives

$$\mathrm{Var}(\bar{Y}) = \frac{\bar{p}(1-\bar{p})}{m}[1 + (m-1)\rho].$$

For $0 < \bar{p} < 1$ and a positive variance inflation factor, this corresponds to $m/[1 + (m-1)\rho]$ independent Bernoulli observations with the same variance. This conditional-mixture illustration is not an estimate of the effective sample size of a full path-consistency run. Neither that value nor a plug-in correction is used to calibrate the actual path-consistency gate.

B Additional analysis of prefix reuse

The following results clarify when prefix reuse can affect the final vote and how shared prefixes change the relationship between trajectory accuracy and voting accuracy.

B.1 When can later windows still change the vote?

Let c_a count all valid votes for answer a already generated, including votes from rejected windows. Let ℓ be a unique current leader and let r be an upper bound on the number of answers still to be generated. Include an unseen competitor with count zero when defining $c_{(2)} = \max_{a \neq \ell} c_a$. Each remaining trajectory supplies at most one unit vote, and earlier votes cannot be removed or reweighted.

Proposition 3 (Irreversible plurality). *If $c_\ell - c_{(2)} > r$, then ℓ is the unique final plurality winner for every possible continuation, including adaptive and dependent continuations. If $c_\ell - c_{(2)} < r$, a continuation assigning every remaining vote to a strongest competitor can overturn the leader. At equality a competitor can tie, so the outcome depends on the fixed tie rule.*

Proof. For any competitor, the largest possible increase in its count relative to the leader is r. Thus its final deficit is at least $c_\ell - c_a - r$. A strictly positive minimum guarantees the leader remains unique. Conversely, giving all r votes to a strongest competitor realizes that maximum increase, proving the remaining statements. Invalid outputs can only reduce the number of added votes and cannot break the sufficient condition. □

The statement concerns the realized vote, not correctness. Once a wrong leader is locked, a rollback cannot repair the final answer under this aggregation rule. Once a correct leader is locked, later prefixes cannot change final correctness either. For a fixed $B = 32, m = 4$, one may test the sufficient condition after window j using $r = 32 - 4j$. This is a diagnostic of the final vote, not a stopping rule used in the experiments. The condition says only that later votes cannot alter the outcome; it does not undo any effect of earlier prefixes on the current counts.

There is also a useful consequence for Theorem 1. On a locked history, V^S is constant over all admissible next windows, so $d_j(h) = 0$. Accordingly,

$$|A_P - A_S| \le \min\left\{1, \sum_j \mathbb{E}_P\left[\mathbf{1}\{H_j \text{ is not locked}\}\ \mathrm{TV}(K_j^P(\cdot \mid H_j), K_j^S(\cdot \mid H_j))\right]\right\}.$$

This refinement localizes possible final-vote effects to histories where future votes can still matter. It does not assert that unlocked histories are recoverable under the model's actual support, nor that their increments are favorable.

B.2 Higher marginal correctness can yield a worse majority

For a binary illustration, condition on a history with n earlier votes, k of them correct, and $r = B - n$ remaining votes. A random shared prefix Z induces a correctness probability p_Z; conditional on Z and the history, the remaining answers are independent Bernoulli variables. With ties counted as failures, let $t = \lfloor B/2 \rfloor + 1 - k$. Using the binomial tail from Equation 7, the conditional final success probability is

$$A_{\mathrm{shared}}(h) = \mathbb{E}[Q_r(p_Z, t) \mid h]. \tag{10}$$

If an independent prefix were instead drawn for each remaining answer under this stylized model, the probability would be $Q_r(\bar{p}, t)$, where $\bar{p} = \mathbb{E}[p_Z \mid h]$. These two expressions generally differ even at identical marginal correctness. Independent-prefix resampling here is an analytical comparator, not an evaluated path-consistency variant.

Proposition 4 (Curvature of the voting response). *For $1 \le t \le r$ and $0 < p < 1$,*

$$\frac{dQ_r(p,t)}{dp} = r\binom{r-1}{t-1}p^{t-1}(1-p)^{r-t},$$

and the sign of its second derivative is the sign of $(t-1) - (r-1)p$. Thus Q_r is concave above $p = (t-1)/(r-1)$ and convex below it when $r > 1$. If the support of p_Z lies wholly in the concave region, sharing cannot improve the voting probability relative to independent resampling at the same mean; in the convex region the inequality reverses.

Proof. Differentiate the finite binomial sum. Adjacent terms cancel, leaving the stated first derivative. Its logarithmic derivative is $(t-1)/p - (r-t)/(1-p)$, which has the stated sign. Jensen's inequality on either curvature region gives the comparison. For $r = 1$ the response is linear. For $t \le 0$ or $t > r$ it is constant. When the support crosses the inflection point, these local curvature statements imply no universal ordering. □

For a finite-budget counterexample retaining the root votes, take four independent root answers with correctness probability 0.6 and 28 later answers sharing a latent prefix. Independently of the root, let

$p_Z = 1$ with probability 0.65 and $p_Z = 0$ otherwise. The final vote is correct exactly when $p_Z = 1$, hence its accuracy is 0.65. In contrast, 32 independent answers with correctness probability 0.6 succeed with probability $Q_{32}(0.6, 17) = 0.835203095$. Every later answer in the shared construction has higher marginal correctness (0.65 versus 0.6), and the mean across all 32 is 0.64375, yet its majority is worse. This deliberately extreme mixture is not a fitted path-consistency model or an assertion that these prefixes occur in the experiments. It demonstrates why per-trajectory correctness and agreement alone do not identify the accuracy of a shared-prefix vote.

B.3 Agreement thresholds under a fixed answer distribution

A fixed-distribution calculation helps distinguish evidence for answer concentration from evidence for correctness. Suppose n answers are sampled independently from one distribution with correct-answer probability q. For $1/2 < C < 1$, define

$$k_C(n) = \min\{k \in \{0, \dots, n\} : c(n, k) \geq C\},$$

taking $k_C(n) = n + 1$ when the set is empty. The score in Equation 1 is increasing in k, and $k_C(n) > n/2$. Hence the correct answer passes the gate exactly when its count reaches $k_C(n)$, with probability

$$\mathbb{P}(\text{correct answer passes}) = Q_n(q, k_C(n)).$$

This expression concerns the source answer, not the semantic correctness of every token in its prefix.

An incorrect answer can pass only if the total number of incorrect answers is at least $k_C(n)$. Therefore

$$\mathbb{P}(\text{an incorrect answer passes}) \leq Q_n(1 - q, k_C(n)).$$

Equality holds when all incorrect probability mass is concentrated on one alternative. This identifies a worst-case error distribution for a single fixed-distribution check. Raising C increases the required count and reduces both correct and incorrect gate-passing probabilities at that check. In particular, a majority of correct answers can still fail a stringent gate, so the probability of failing to pass cannot be bounded solely by the probability of lacking a correct majority.

Successive checks in path-consistency reuse observations and change the sampling distribution. Their error probabilities therefore cannot be multiplied as if the checks were independent. More generally, if the conditional probability of an incorrect gate-passing decision at check j, given its preceding history, is bounded by ε_j, the probability of any such decision is at most $\sum_j \varepsilon_j$ by a union bound and conditional expectation. The gate itself does not establish these conditional bounds. The adaptive analysis in Section 4 avoids treating repeated agreement as a correctness guarantee.

C Method and evaluation details

C.1 Window transitions and answer aggregation

Table 7 summarizes the controller in Section 3. The admitted set $\mathcal{V}$ controls prefix proposals and validation, whereas the final vote uses all valid completed answers. Answers from previously accepted windows remain in $\mathcal{V}$ after rollback.

Failed windows provide neither admitted answers nor new prefix candidates. A failed candidate is not permanently excluded from all later source pools. When combined with an early-stopping method, sampling ends at a completed window according to that method's stopping rule, and its aggregation rule is retained.

C.2 Datasets and prompting

We use the GSM8K test split, MATH-500, AIME 2024, AIME 2025, and GPQA-Diamond. GPQA option order is shuffled deterministically with seed 2027.

The mathematical reasoning instruction is *Please reason step by step, and put your final answer within* `\boxed{}`. GPQA presents four labeled options and requests the correct option letter in the same format. Llama uses its chat template. DeepSeek and Qwen use thinking mode, with the decoding settings in Section 5.1. These interfaces differ, so comparisons are made within each model and dataset.

Table 7: Prefix updates at window boundaries. Every valid completed answer remains eligible for final aggregation.

State	Observation	Update
Root	Valid answers available	Add answers to $\mathcal{V}$; propose an extension if the gate passes and a candidate exists.
Trial	Same modal answer and $c \geq C$	Commit the prefix, admit answers, clear the failure count, and consider the next extension.
Trial	Modal answer changes or $c < C$	Discard the proposal, return to the committed prefix, and increment the failure count.
Committed	Same modal answer and $c \geq C$	Retain the prefix, admit answers, clear the failure count, and consider the next extension.
Committed	Modal answer changes or $c < C$	Roll back one level and increment the failure count.
Any prefix	Two consecutive failures	Clear the prefix and disable further reuse for this question.

C.3 Token and time accounting

At an empty prefix, path-consistency reuses the corresponding self-consistency window. Its original token count and generation time are charged in full. Windows with nonempty prefixes generate new continuations, and failed windows are also counted.

Let T_{ij} and G_{ij} be generated tokens and generation time for window j of question i. Over n questions, the means are

$$\bar{T} = \frac{1}{n}\sum_{i,j} T_{ij}, \qquad \bar{G} = \frac{1}{n}\sum_{i,j} G_{ij}.$$

Token savings are $1 - \bar{T}_{\mathrm{PC}}/\bar{T}_{\mathrm{SC}}$, and a generation speed ratio is $\bar{G}_{\mathrm{SC}}/\bar{G}_{\mathrm{PC}}$. Tables with negative Save entries instead display the signed relative token change. A nominal prefix fraction is not a measured saving because continuation lengths can change. These measurements do not include all input-processing and orchestration costs or establish peak-memory and API-billing savings.

C.4 Ablation settings

The prefix-selection and recovery ablation fixes the 500 DeepSeek/MATH-500 questions, self-consistency root windows, source eligibility, prefix-length grid, and decoding settings. Uniform selection samples from the same eligible candidates. Disabling recovery retains the active prefix after failure and disables the two-failure reset, while failed-window answers remain excluded from $\mathcal{V}$. All valid answers still enter the final vote. This removes the recovery mechanism as a whole rather than isolating a single rollback operation.

Agreement threshold. Table 6 compares $C \in \{0, 0.7, 0.8, 1\}$ on DeepSeek/GSM8K. At $C = 0$, the gate imposes no minimum agreement score, but a valid modal answer and feasible prefix are still required. At $C = 1$, prefix reuse is disabled. Lower values favor token savings, while higher values make reuse more conservative and favor accuracy preservation. The default remains $C = 0.8$; this sensitivity analysis does not identify a universally optimal threshold.

Prefix length at a fixed history. A supplementary DeepSeek/MATH-500 study fixes the first-window history and the source chosen by local confidence selection, then varies the retained fraction over $\{0, 0.2, 0.4, 0.6, 0.8\}$. It covers 466 questions with feasible prefixes at every level and eight repetitions of a four-answer window per level. Of the remaining questions, 19 do not pass the initial gate and five lack a feasible endpoint. The final-vote diagnostic combines the intervened window with the same four root answers and 24 unprefixed answers. It therefore isolates one window under a fixed continuation, rather than estimating the full continuation expectation in Equation 4. This analysis does not change the fixed prefix-length grid used at inference time.

D INITIAL EXPLORATION OF SENTENCE-LEVEL PREFIX REUSE

Before developing the current method, we first explored a sentence-level form of path-consistency for short-chain reasoning. The idea was to reuse complete early reasoning steps rather than generate every trajectory from scratch. This initial approach achieved substantial token savings with generally comparable accuracy on several arithmetic, commonsense, and symbolic tasks, motivating the controlled prefix reuse developed in Section 3. We summarize these earlier experiments and the insights they provided below.

The sentence-level approach samples 20 trajectories and randomly selects from eligible prefix candidates ending at sentence boundaries. Its default schedule allows three extension levels, with updates after every five trajectories. Unlike the current method, it does not use one shared prefix per window, the token-fraction grid, or the same trial-and-recovery controller. The results therefore characterize this initial approach and are not pooled with the main experiments.

The Llama-3-8B experiments use temperature 0.6, top-$p = 0.9$, few-shot reasoning prompts, and an RTX 3090. Here a prefix level counts retained reasoning steps rather than a fraction of trajectory tokens. Comparisons retain their own self-consistency baselines and timing conventions. Reported speedup percentages denote the increase in speed relative to the corresponding baseline, not the percentage reduction in elapsed time.

D.1 ARITHMETIC, COMMONSENSE, AND SYMBOLIC REASONING

Table 8 reports our initial results across ten short-chain reasoning tasks (Cobbe et al., 2021; Patel et al., 2021; Miao et al., 2020; Roy & Roth, 2015; Geva et al., 2021; Suzgun et al., 2023). Sentence-level reuse reduces token generation across all tasks and thresholds, with savings ranging from 4.7% to 36.6%. Arithmetic accuracy is generally close to or above the self-consistency baseline. At $C = 0.8$, GSM8K and Tracking Shuffled Objects improve by 3.7 and 5.2 percentage points, respectively, while Ruin Names and Logical Deduction decline. These results established the practical potential of prefix reuse while showing that its accuracy effects depend on the task.

Table 8: Sentence-level prefix reuse with Llama-3-8B. Each PC entry gives accuracy (%) / token saving (%) relative to its SC baseline. These results use the supplementary 20-trajectory setting.

Dataset	SC accuracy (%)	$C = 0$	$C = 0.7$	$C = 0.8$	$C = 0.9$
Arithmetic reasoning					
GSM8K	64.1	66.6 / 24.0	67.1 / 16.1	67.8 / 16.2	66.6 / 9.9
SVAMP	79.3	78.8 / 36.6	78.5 / 29.4	79.2 / 29.7	79.9 / 22.3
ASDiv	81.0	80.1 / 32.7	81.2 / 27.2	80.0 / 27.1	80.6 / 22.6
MultiArith	96.7	97.8 / 34.8	97.2 / 33.3	98.9 / 33.1	98.3 / 27.2
Commonsense reasoning					
StrategyQA	71.4	70.8 / 27.8	70.6 / 24.0	71.3 / 22.1	71.1 / 16.0
Ruin Names	76.0	71.2 / 9.8	74.4 / 7.2	72.0 / 6.6	71.6 / 4.7
Salient Translation	54.8	58.4 / 25.8	57.2 / 18.3	56.4 / 16.4	55.2 / 10.6
Symbolic reasoning					
Boolean Expressions	88.4	90.0 / 21.3	89.6 / 19.8	88.8 / 20.3	89.6 / 18.3
Tracking Shuffled Objects	54.0	58.0 / 16.7	56.0 / 12.4	59.2 / 10.8	58.4 / 6.8
Logical Deduction	78.8	76.8 / 26.1	78.8 / 20.5	77.2 / 24.8	76.0 / 17.5

Lower thresholds generally permit larger savings, but the trend need not be monotonic after the generated history changes. Logical Deduction, for example, saves more tokens at $C = 0.8$ than at $C = 0.7$. The smaller improvements on Ruin Names also show that the benefit depends on the task, rather than only on the trajectory budget.

D.2 PREFIX GROWTH AND EXTENSION FREQUENCY

Table 9 shows how savings grow with sentence-level prefixes on GSM8K. At $C = 0.8$, token savings rise from 9.8% at level 1 to 33.3% at level 3. This observation motivated progressive prefix extension

in the current method, where prefix length is expressed as a fraction of reasoning tokens rather than a number of sentences.

Table 9: GSM8K efficiency by sentence-prefix level. The left group reports speedup (%); the right group reports token saving (%), each relative to the corresponding unprefixed baseline.

	Speedup (%)				Save (%)			
C	Overall	Level 1	Level 2	Level 3	Overall	Level 1	Level 2	Level 3
0	28.9	18.8	42.6	78.9	24.0	16.9	32.7	47.4
0.7	18.0	10.0	25.3	46.5	16.1	9.4	21.3	33.8
0.8	17.4	9.8	25.0	43.9	16.2	9.8	21.8	33.3
0.9	10.2	4.2	13.3	27.0	9.9	4.2	12.7	23.0

With the same 20-trajectory budget, allowing four rather than three extension levels also changes the update interval from five trajectories to four. Table 10 shows increased savings under the more frequent schedule. Accuracy is unchanged at $C = 0.8$ and lower at $C = 0.7$. Because depth and update interval change together, this comparison does not isolate the effect of window size.

Table 10: Extension depth and update interval on Llama-3-8B/GSM8K with sentence-level prefixes. Speedup and Save are relative to SC.

C	Max. level	Window size	Accuracy (%)	Speedup (%)	Save (%)
0.7	3	5	67.1	18.0	16.1
0.7	4	4	66.3	20.5	20.0
0.8	3	5	67.8	17.4	16.2
0.8	4	4	67.8	18.0	17.8

D.3 Method comparisons across model families

Table 11 compares sentence-level prefix reuse with early-stopping methods. On Llama-3-8B, path-consistency uses fewer tokens than either early-stopping baseline on all three tasks, with accuracy between or close to the comparison methods. The DeepSeek-V3 and Llama-3.2-1B-Instruct results show the same cost advantage, but the smaller model loses some accuracy on GSM8K and StrategyQA. Together, these experiments suggested that sentence-level reuse was useful across model families, while its accuracy remained constrained by the quality of the underlying reasoning.

Table 11: Supplementary method comparisons. Entries are accuracy (%) / mean tokens per question. Each model block is a separate evaluation. AC and ESC denote adaptive-consistency and early-stopping self-consistency. The Llama-3-8B baseline here differs from that in Table 8.

Model	Method	GSM8K	SVAMP	StrategyQA
Llama-3-8B (Grattafiori et al., 2024)	SC	68.4 / 3,860	80.4 / 2,321	71.7 / 1,805
	AC	67.2 / 2,248	79.7 / 935	71.2 / 706
	ESC	67.2 / 2,710	79.2 / 1,154	70.8 / 782
	PC	67.8 / 1,630	79.2 / 816	71.3 / 700
DeepSeek-V3 (DeepSeek-AI et al., 2024)	AC	91.6 / 828	94.7 / 363	84.5 / 326
	ESC	91.6 / 868	94.7 / 358	84.2 / 315
	PC	91.8 / 786	94.7 / 263	84.5 / 230
Llama-3.2-1B-Instruct	AC	48.1 / 2,748	61.1 / 800	63.9 / 1,401
	ESC	47.9 / 2,324	61.9 / 924	63.7 / 1,136
	PC	47.3 / 1,430	62.0 / 711	63.1 / 799

Table 12 provides another threshold comparison on Mistral-7B. Removing the agreement barrier gives the largest token saving but lower accuracy, while higher thresholds retain accuracy closer to the self-consistency baseline with smaller savings.

Table 12: Sentence-level prefix reuse on Mistral-7B/GSM8K. Speedup and Save are relative to SC.

Method	C	Accuracy (%)	Speedup (%)	Save (%)
SC	1.0	59.7	0.0	0.0
PC	0.0	58.5	22.0	20.3
PC	0.8	60.3	11.1	11.4
PC	0.9	60.4	6.2	6.6

D.4 Token allocation and answer agreement

Our initial token analysis in Figure 4 compared generation on trajectories with correct and incorrect final answers. Sentence-level reuse reduces generated text in both groups, supporting the idea that savings come from shortening later continuations rather than only avoiding incorrect answers. Final-answer labels do not identify which individual reasoning tokens are unnecessary. Figure 1 complements this analysis by directly visualizing repeated token sequences.

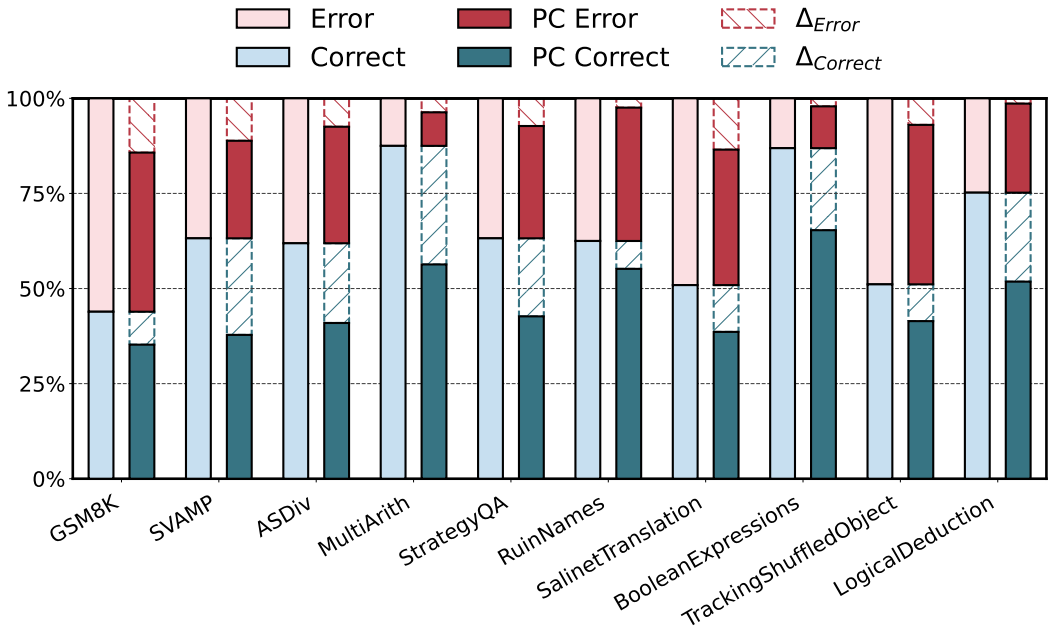


Figure 4: Token allocation in the initial sentence-level experiments. Light bars show SC, dark bars show PC, and hatched areas indicate reductions relative to SC. Correct and Error classify trajectories by their final answers, not individual tokens.

We also examined whether answer agreement could guide prefix reuse. Figure 5 shows that stronger agreement after ten self-consistency trajectories is associated with correct answers across several tasks. Figure 6 compares the final distributions on GSM8K, where sentence-level reuse produces more concentrated agreement. Incorrect answers nevertheless occur at high agreement. These observations motivated agreement-guided control, but not the interpretation of agreement as a correctness certificate.

D.5 Illustrative reasoning trajectories

Table 13 illustrates progressive sentence-prefix reuse. Inherited text is retained while the newly generated suffix becomes shorter. It shows how the initial approach reused complete reasoning steps, complementing the token-fraction example of the current method in Table 2.

A second example asks how much water remains after two girls each receive one sixth of 24 liters and a boy receives 6 liters. The correct answer is 10 liters. Table 14 summarizes the Llama-3.2-1B-Instruct

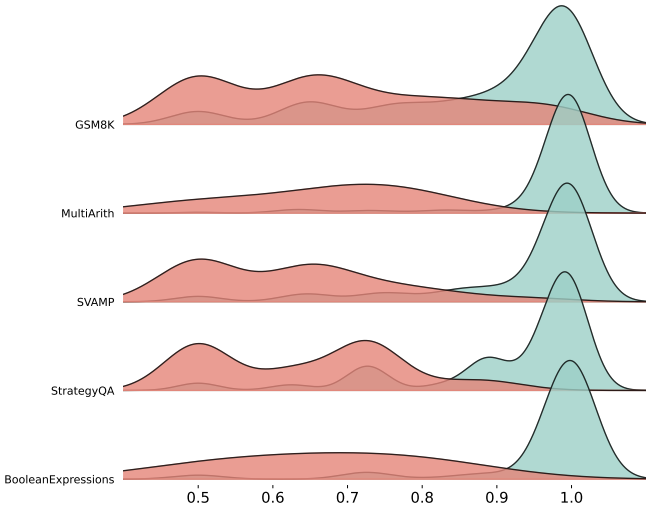
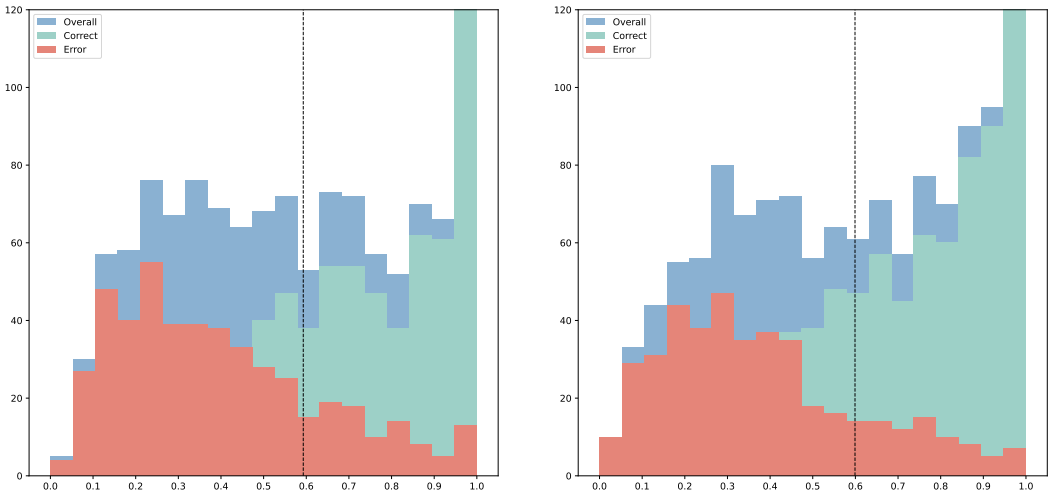


Figure 5: Answer agreement after ten SC trajectories in the initial experiments. Green and red show the distributions for correct and incorrect answers, respectively.



(a) Self-consistency

(b) Path-consistency

Figure 6: Final answer agreement on GSM8K in the sentence-level experiments, with $C = 0.8$ for PC. Blue shows all questions, green correct answers, and red incorrect answers. The panels use the same axis scales.

Table 13: A sentence-level prefix example. Black text is reused and blue text is newly generated.

Question	A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
No prefix	To make a robe, we need 2 bolts of blue fiber and half that much white fiber. We can write this as 2 + 1/2 * 2. Now 1/2 * 2 = 1, so the equation becomes 2 + 1 = 3. The answer is 3.
Level 1	To make a robe, we need 2 bolts of blue fiber and half that much white fiber. 2 * 1/2 = 1 bolt of white fiber. 2 + 1 = 3 bolts in total. The answer is 3.
Level 2	To make a robe, we need 2 bolts of blue fiber and half that much white fiber. 2 * 1/2 = 1 bolt of white fiber. 1 + 2 = 3 bolts in total. The answer is 3.
Level 3	To make a robe, we need 2 bolts of blue fiber and half that much white fiber. 2 * 1/2 = 1 bolt of white fiber. 1 + 2 = 3 bolts in total. The answer is 3.

trajectories. Prefix levels increase and later return to zero; the final vote is correct despite earlier disagreement. The example shows that a return to unprefixed sampling can occur, but does not prove that the reused text was wrong or that recovery caused the correct final answer.

Table 14: Window-level summary of the water-allocation example. Tokens are the total newly generated tokens in each five-trajectory window. The recorded final answer is 10.

Window	Prefix level	Sampled answers	Tokens
1	0	8, 10, 14, 14, 14	383
2	1	2, 14, 14, 10, 16	283
3	2	10, 10, 10, 10, 12	89
4	0	10, 10, 2, 14, 14	380

D.6 Controller overhead and parallelism

The sentence-level Llama-3-8B/GSM8K measurements in Table 15 show that agreement evaluation and prefix selection take little time relative to model generation in that setting. They do not measure the full controller cost of the current method, which also ranks local confidence scores and manages trial and recovery.

Table 15: Controller timing in the supplementary 20-trajectory Llama-3-8B/GSM8K evaluation. Percentages are relative to the reported model-generation time.

Component	Time (s)	Relative time (%)
Model generation	2.2	100
Answer agreement evaluation	0.000096	0.0044
Prefix construction	0.00013	0.0059

Windowed reuse also constrains parallelism. Trajectories within a window can run concurrently, but the next window depends on the preceding answers and prefix decision. Larger windows allow more concurrent generation, whereas smaller windows offer more frequent opportunities to extend a prefix. Token savings therefore need not translate into proportional latency reductions, especially when independent self-consistency trajectories can be served with greater parallelism.

D.7 Prompt structure and scope

The sentence-level experiments use few-shot demonstrations containing a question, reasoning, and final answer. Table 16 summarizes their format without repeating the full demonstration sets. These prompts are distinct from the chat and thinking interfaces used in the main evaluation.

Table 16: Prompt structure for the supplementary sentence-level experiments. Demonstrations supply intermediate reasoning followed by an explicit final answer.

Task family	Demonstration content	Answer format
Arithmetic reasoning	Arithmetic steps leading to a numerical result	The answer is [number].
StrategyQA	Short factual reasoning for a yes/no question	The answer is yes/no.
Ruin Names	Comparison of candidate humorous edits	The answer is [option].
Salient Translation	Comparison of a source sentence and its translation	The answer is [error-type option].
Boolean Expressions	Evaluation using logical operator precedence	The answer is True/False.
Tracking Shuffled Objects	State updates after successive swaps	The answer is [option].
Logical Deduction	Ordering objects from the supplied constraints	The answer is [option].

Implications for the current method. The initial experiments showed that reusing intermediate reasoning can shorten later trajectories across several short-chain tasks. They also exposed the trade-off between more aggressive reuse and accuracy preservation. The current method builds on this idea with one continuous shared prefix per window, a token-fraction length policy, local confidence selection, and explicit trial and recovery. The earlier results motivate this design direction; the experiments in Section 5 evaluate the resulting method.

These experiments do not train or modify the evaluated models. Their computational benefits and failure modes depend on the available reusable reasoning, the agreement threshold, and the generation interface. Reducing inference cost may improve accessibility but can also reduce the cost of harmful uses of the underlying models. Dataset text and model weights remain subject to their providers' terms and are not redistributed with the supplementary results.