COCONUT TREE: ADAPTIVE BRANCHING FOR CONTINUOUS LATENT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous latent reasoning replaces verbalized intermediate steps with hidden-state computation. This latent representation allows *implicit* representation of multiple reasoning chains in a single branch. However, we show that a single branch is a lossless implicit beam only when its state remains sufficient for every required future: when one state conflates beams requiring different futures, it cannot serve as a lossless implicit beam. To address this, we introduce COCONUT TREE, a framework that determines when latent futures must remain separate and allocates reasoning width per problem. Across COCONUT, CODI and COLAR releases, the proposed approach consistently improves upon fixed-width and compute-matched baselines by several percentage points on average. To allocate reasoning width, COCONUT TREE uses collision probabilities derived from the model's output-token distribution. A tight universal bound yields a stopping rule that exactly optimizes a penalised local coverage surrogate, whose only hyperparameter is a maximum branching width $B_{\max}$, without training or calibration. Predicted collision marginals track realized next-path gains, and the adaptive allocation consistently improves over the compute-matched fixed-width frontier. Live inference measurements on COCONUT show that adaptive branching reduces wall-clock time by 13.6–60.0% and estimated dense-model FLOPs by 50.3–84.1% relative to using the fixed branching width (at $B = 8$). The framework requires only separately continuable latent states and output-token probabilities, keeping the approach generic. Thus COCONUT TREE explains both when continuous latent reasoning should branch and how much width branching justifies.

1 INTRODUCTION

Chain-of-thought reasoning expresses intermediate steps as language (Wei et al., 2022). Latent reasoning instead performs intermediate computation in continuous hidden states, avoiding the need to verbalize each step (Hao et al., 2025). With latent reasoning, a single continuous state *can* represent and propagate multiple reasoning paths without explicitly instantiating a branch for each (Gozeten et al., 2026). Nevertheless, recent methods demonstrate the benefits from exploring multiple latent trajectories at test time (You et al., 2026; Xu et al., 2025; Wang et al., 2026), but typically use the same ad hoc *reasoning width* (number of branches) for every problem.

To address this, we introduce COCONUT TREE. We characterize *when this implicit representation fails*, and derive a principled rule for *dynamically determining the width* (the number of independent continuations) per problem. At inference, COCONUT TREE uses output-token probabilities as a proxy for unresolved futures and derives collision-based estimates of the marginal gain in target coverage from another path. It expands only while this predicted gain exceeds a stopping threshold derived from the maximum permitted width $B_{\max}$, dynamically allocating the width without requiring any additional calibration (Fig. 1).

Our contributions address three questions:

- **Separation Necessity.** When does a single state fail as an implicit beam? We give a general conflation criterion and characterize exactly when branch-specific computation has strict value. At equal compute, separate states improve pass@K over a shared-state control in all 24 evaluated conditions across COCONUT, CODI, and COLAR (Tables 1 and 3).

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.

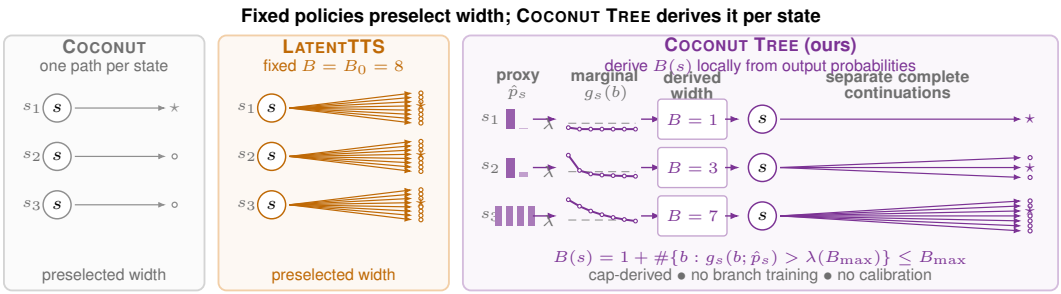


Figure 1: **Latent reasoning methods.** COCONUT uses one path, LATENTTTS a fixed width, and COCONUT TREE (Ours) adapts paths based on predicted marginal coverage gain.

- **Principled Allocation.** How much should the model branch? Under a self-consistent model of target coverage, we derive a collision-based stopping rule that exactly optimizes a penalized local objective. A tight distribution-free bound yields the smallest universal stopping threshold $\lambda(B_{\max})$ that guarantees the width cap $B_{\max}$, leaving the cap as the allocation rule's only hyperparameter.
- **Adaptive Efficiency.** Does per-problem allocation improve coverage over fixed-width policies at matched average compute? Sweeping $B_{\max}$ traces a monotone coverage–compute curve without model- or dataset-specific threshold calibration. Across model backends, the proposed rule yields consistent coverage gains over the compute-matched fixed-width frontier on average across tasks. At $B_{\max} = 8$, adaptive inference reduces wall-clock time by 13.6–60.0% and estimated dense-model FLOPs by 50.3–84.1% relative to fixed width $B = 8$.

2 RELATED WORK

Latent reasoning architectures COCONUT feeds its final hidden state back as its next input embedding (Hao et al., 2025); CODI distills explicit reasoning into continuous states (Shen et al., 2025); and COLAR learns dynamically compressed probabilistic latent reasoning (Tan et al., 2025). Other work adapts latent depth, updates blocks of latent tokens in parallel, or regularizes latent-state semantics (Ning et al., 2025; Wu et al., 2025; Wei et al., 2026). LaTER combines latent exploration with verification, while Latent Thought Flow models stochastic variable-length trajectories (Li et al., 2026; Zou et al., 2026). **Latent test-time scaling** In terms of scaling test-time compute, LATENTTTS introduces Monte Carlo dropout and additive Gaussian noise, then applies latent best-of-N and fixed-width beam search to improve latent reasoning (You et al., 2026). SoftCoT++ creates diverse continuous thoughts (Xu et al., 2025). GTS learns a context-dependent Gaussian thought sampler while freezing the latent-reasoning backbone (Wang et al., 2026). However, none determine width adaptively. **Adaptive branching** Adaptive branch width has primarily been studied outside latent reasoning. Tree of Thoughts searches over explicit reasoning units (Yao et al., 2023); EDEN adapts token-level width; and SEMIE adaptively branches over semantic continuations (Evans et al., 2026a;b).

In contrast, COCONUT TREE derives a per-problem *latent* reasoning width from Rényi-2 collision probabilities, determining when separate continuation has value and how much width it justifies.

3 PROBLEM SETUP

Let x be a prompt and $s_0 \in \mathcal{S}$ its initial latent state. A state may contain a recurrent hidden vector, latent tokens, masks, and a branch-specific KV cache. A latent reasoner advances as $s_{\ell+1} = F_\theta(s_\ell; \xi_\ell)$, where ξ_ℓ denotes optional stochasticity. Language is generated after a stopping time τ: $y \sim p_\theta(y \mid x, s_\tau)$. This interface covers recurrent latent reasoning and block-based variants. A completed path runs its remaining latent transitions and terminal prediction procedure. In our experiments, the latter returns an answer by selecting the highest-scoring candidate or by decoding until a model-defined termination token or fixed length limit.

Branch width as a conditional-compute decision Let $B_{\max}$ be the sole resource parameter, $\mathcal{B} = \{1, \ldots, B_{\max}\}$ the integer widths, and $\mathcal{T}_b(s) = (a_1, \ldots, a_b)$ the task-normalized predictions

from b completed paths. For a task with target Y, their realized target coverage is:

$$G_b(\mathcal{T}_b(s), Y) = \mathbf{1}\{\exists i \le b : a_i = Y\}.$$

An adaptive policy maps the current state to a state-dependent width $B(s) \in \mathcal{B}$.

COCONUT fixes $B(s) = 1$, LATENTTTS fixes $B(s) = B_0 > 1$, and COCONUT TREE chooses $B(s)$ per problem (Fig. 1). We ask whether one state can preserve and update every explicit-beam future and, if not, whether a statistic available before expansion can allocate width without observing Y.

Complete paths and the shared-state control At state s, for any width $B \le B_{\max}$, a proposal operator draws B candidate successor states and their caches: $\mathcal{P}(s; B) = \{(\tilde{s}_i, \tilde{c}_i)\}_{i=1}^{B}$.

The *independent* policy applies all remaining latent transitions to each pair $(\tilde{s}_i, \tilde{c}_i)$ separately. The *shared* control constructs B copies of one proposal and applies the same number of stochastic transitions. The controls therefore match width, horizon, answer-scoring work, and downstream random-number state; they differ only in whether the initial state and cache are distinct.

Writing $z_i = \mathbf{1}\{a_i = Y\}$ for the success of path i, nested coverage for $K \le B$ is

$$\text{pass@}\, K = G_K(\mathcal{T}_K(s), Y) = \max_{1 \le i \le K} z_i. \tag{1}$$

4 WHEN AND HOW TO BRANCH IN LATENT SPACE

We first address **Separation Necessity** by asking when one latent state is equivalent to an explicit beam, then characterize its failure and show when separate future computation has strict value. We then turn to **Principled Allocation** and derive a single parameter allocation rule.

4.1 WHEN WIDTH ONE IS A LOSSLESS IMPLICIT BEAM

Represent an explicit latent beam as a discrete probability distribution over latent states:

$$\mu = \sum_{k=1}^{B} \pi_k \delta_{h_k}, \qquad \pi_k \ge 0, \quad \sum_k \pi_k = 1, \tag{2}$$

where δ_h denotes the Dirac measure concentrated at h. For a *deterministic* continuation C[1], applying C to each beam state yields the pushforward distribution $C_{\#}\mu = \sum_{k=1}^{B} \pi_k \delta_{C(h_k)}$.

Definition 1 (Lossless implicit latent beam)**.** *Let Φ map an explicit beam to a single latent state, let $\Psi : \mathcal{S} \to \mathcal{S}$ be the width-one transition, and let O be the required continuation outcome. The mapping Φ is lossless for the considered beam class if there exists an outcome map R such that, for every beam μ,*

$$\Psi(\Phi(\mu)) = \Phi(C_{\#}\mu), \qquad R(\Phi(\mu)) = O(\mu). \tag{3}$$

The first identity is dynamic closure and the second is outcome sufficiency. Together they make width-one and explicit propagation equivalent at every horizon.

The continuation outcome may be an answer distribution, a set of reachable answers, or the best terminal value among branches, so the definition includes pass@K (from Eq. (1)). This equivalence is possible in principle. With sufficient dimension, Φ could concatenate all weighted states and Ψ could apply C blockwise. Such a state is width one in layout but has not compressed away the beam's information.

Remark 1 (A same-size state cannot represent every arbitrary beam)**.** *Suppose $B > 1$, each h_k ranges over a d-dimensional open set, and Φ is a continuous map into one d-dimensional state. Ordered beams with fixed weights contain an open subset of $\mathbb{R}^{Bd}$, so invariance of dimension rules out an injective $\Phi : \mathbb{R}^{Bd} \to \mathbb{R}^{d}$. Exact same-size representation remains possible for a lower-dimensional task family or when all beams that Φ conflates have the same continuation outcome.*

[1]The definition is stated for deterministic C and Ψ for clarity. For stochastic transitions, branchwise continuation induces a Markov kernel over beams, and dynamic closure requires equality in distribution between the width-one update and the single-state representation of the explicitly continued beam.

4.2 FAILURE THROUGH REPRESENTATIONAL CONFLATION

A beam-to-state map *conflates* two beams when it maps them to the same latent state. Such conflation is harmful when the beams require different continuation outcomes.

Proposition 1 (Future-relevant conflation criterion)**.** *For horizon t, if two beams μ and ν satisfy*

$$\Phi(\mu) = \Phi(\nu) \quad \textit{but} \quad O(C^t_{\#}\mu) \neq O(C^t_{\#}\nu). \tag{4}$$

then no width-one procedure whose future computation depends only on $\Phi(\cdot)$ can be correct for both. This also holds for a randomized transition with the same conditional noise law for both beams at their shared state.

Proof. Both beams give the algorithm the same state. A deterministic update therefore produces the same future state and outcome, contradicting Eq. (4). With identical conditional noise laws, their output laws also remain identical. □

Width one can therefore be sufficient only if its state preserves every distinction that can change a required future; otherwise the reasoner must retain separate states or an equivalently informative distributional representation.

Illustrative special case: mean compression. The conflation criterion is representation-agnostic. To make one possible failure concrete, consider compressing a beam to its weighted mean, $\Phi_{\text{mean}}(\mu) = \sum_k \pi_k h_k$. Dynamic closure of this mean representation would require averaging before continuation to match continuing each beam state and averaging afterward: $C(\sum_k \pi_k h_k) = \sum_k \pi_k C(h_k)$. For continuous C, this equality holds for every finite beam iff C is affine. The two-state case requires C to preserve every convex combination, which characterizes continuous affine maps; affine maps satisfy the equality directly. Thus every non-affine continuation admits some beam for which averaging before continuation differs from averaging afterward.

For example, let $C(h) = h^2$. The beams $\mu_0 = \delta_0$ and $\mu_\pm = \frac{1}{2}\delta_{-1} + \frac{1}{2}\delta_1$ both have mean zero. After branchwise continuation, however, their means are zero and one, respectively, as shown in Fig. 2.

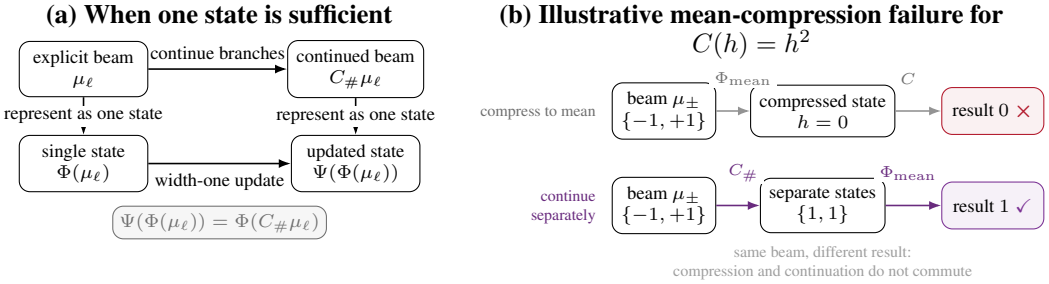


Figure 2: **Width-one implicit beams. (a)** A one-state representation is lossless when representation and continuation commute while preserving outcomes. **(b)** In an illustrative special case, averaging before nonlinear continuation loses a distinction preserved by separate continuation.

This example neither assumes that latent reasoners literally average branches nor models our shared-state control. It shows one concrete way the premise of Prop. 1 can arise when the resulting discrepancy affects the required outcome. Our shared-state control tests the broader limitation empirically without assuming mean compression, while collision marginals determine width.

4.3 WHEN SEPARATE COMPUTATION HAS STRICT VALUE

Even if one state preserves several modes, a single future computation may not serve them all. Let mode r_k have probability π_k, let u denote a subsequent computation, and let $Q(r_k, u) \in \mathbb{R}$ score the terminal outcome obtained by applying u to mode r_k, with larger scores preferred. We compare two regimes. In the *shared* regime, every mode must use the same computation u, and V_{shared} is the best expected value attainable under this restriction. In the *expanded* regime, the modes remain separate, so each can use its own optimal computation; V_{expanded} is the resulting expected value:

$$V_{\text{shared}} = \max_u \sum_k \pi_k Q(r_k, u). \quad (5) \qquad V_{\text{expanded}} = \sum_k \pi_k \max_u Q(r_k, u). \quad (6)$$

Their difference, $\Delta_{\text{mode}} = V_{\text{expanded}} - V_{\text{shared}}$, is the value of mode-specific future computation.

Proposition 2 (Value of mode-specific continuation)**.** *Assume the maxima in Eqs.* (5) *and* (6) *are attained. Then* $\Delta_{\text{mode}} \geq 0$*. Moreover,* $\Delta_{\text{mode}} = 0$ *if and only if there exists one computation* $u^\star$ *that maximizes* $Q(r_k, u)$ *for every positive-mass mode. Thus, mode-specific continuation has strictly greater value exactly when no such common optimizer exists.*

Proof. For any shared computation u,

$$V_{\text{expanded}} - \sum_k \pi_k Q(r_k, u) = \sum_k \pi_k \Big[\max_v Q(r_k, v) - Q(r_k, u)\Big] \geq 0.$$

Because this inequality holds for every u, maximizing its shared-computation term over u gives $\Delta_{\text{mode}} \geq 0$. If a common optimizer exists, equality follows. Conversely, if equality holds, let $u^\star$ attain V_{shared}. The nonnegative sum above must then be zero, forcing every bracketed term to vanish for each positive-mass mode. Therefore, $u^\star$ optimizes every such mode. □

This proposition isolates the computational limitation of a shared continuation. A single state may preserve multiple modes yet remain suboptimal if those modes require different future computations. Multiple modes alone therefore do not justify branching: separation matters only when width one conflates a future-relevant distinction (Prop. 1) or retained modes require incompatible optimal continuations (Prop. 2).

4.4 Principled Allocation: From future modes to adaptive width

The preceding results establish when separation can matter, but not how much width it justifies. We therefore model the unresolved futures at state s as a distribution over future-equivalence modes and ask how likely B independent paths are to cover the target mode. This turns the qualitative necessity of separation into a local allocation objective. Unlike representational conflation, which is a failure of the beam-to-state representation, the collision probability below measures redundancy among the future modes that remain exposed.

At state s, write $p_s = p(\cdot \mid s)$, with $p_{s,i} = \Pr(M = i \mid s)$. Its order-two Rényi effective support is

$$H_2(M \mid s) = -\log \sum_i p_{s,i}^2, \qquad D_2(p_s) = \exp H_2(M \mid s) = \frac{1}{\sum_i p_{s,i}^2}. \quad (7)$$

Here $\sum_i p_{s,i}^2$ is the probability that two independent continuations share a mode, making $D_2(p_s)$ the effective number of modes with substantial probability mass.

We use self-consistency as a calibration model: conditional on s, the same distribution represents uncertainty about the target mode and the modes reached by sampled paths.

Proposition 3 (Self-consistent semantic coverage)**.** *Let* $M^\star$ *denote the* target mode*: the future-equivalence mode that produces the correct task outcome. Let* $M_1, \ldots, M_B$ *denote the modes reached by* B *sampled complete paths. Under the self-consistent model, suppose* $M^\star, M_1, \ldots, M_B$ *are independent draws from* p_s*. Then the probability that at least one of the* B *paths reaches* $M^\star$ *is:*

$$C_B(p_s) = \Pr(M^\star \in \{M_1, \ldots, M_B\} \mid s) = \sum_i p_{s,i} \left[1 - (1 - p_{s,i})^B\right]. \quad (8)$$

The marginal coverage gain from adding path $B + 1$ *is*

$$g_s(B) = C_{B+1}(p_s) - C_B(p_s) = \sum_i p_{s,i}^2 (1 - p_{s,i})^B. \quad (9)$$

In particular, width-one coverage equals the collision probability: $C_1(p_s) = \sum_i p_{s,i}^2 = 1/D_2(p_s)$.

Proof. Conditioned on $M^\star = i$, none of the B sampled paths reaches it with probability $(1 - p_{s,i})^B$. Averaging the complement over $M^\star \sim p_s$ gives $C_B(p_s)$; subtracting consecutive widths gives $g_s(B)$. Setting $B = 1$ and using the definition of D_2 gives the final identity. □

For J balanced modes, $D_2(p_s) = J$; using $B = J$ paths gives $C_J(p_s) = 1 - (1 - 1/J)^J \to 1 - e^{-1}$. The self-consistency assumption is also the method's boundary: collision statistics measure mode diversity but need not predict target coverage when model mass is misaligned with task utility.

Because $g_s(B)$ is the predicted benefit of the next path, a natural policy adds paths while this marginal exceeds a stopping threshold. We derive that threshold from a tight distribution-free bound, leaving $B_{\max}$ as the only user-specified parameter:

$$\lambda(B_{\max}) = \frac{B_{\max}^{B_{\max}}}{(B_{\max}+1)^{B_{\max}+1}}, \qquad B(s) = \min \arg\max_{B \in \mathbb{N}_{\geq 1}} \{C_B(p_s) - \lambda(B_{\max})(B-1)\}. \quad (10)$$

Proposition 4 (Optimal local stopping)**.** *For every mode distribution p_s and integer $B \geq 1$,*

$$g_s(B) \leq \frac{B^B}{(B+1)^{B+1}}, \quad (11)$$

and the bound is tight. The exact solution to Eq. (10) *is*

$$B(s) = 1 + \sum_{b=1}^{\infty} \mathbf{1}\{g_s(b) > \lambda(B_{\max})\} \leq B_{\max}. \quad (12)$$

Moreover, no smaller universal stopping threshold guarantees this cap.

Proof. Writing $g_s(B) = \sum_i p_{s,i}[p_{s,i}(1-p_{s,i})^B]$ and using $\sum_i p_{s,i} = 1$ gives

$$g_s(B) \leq \max_{0 \leq x \leq 1} x(1-x)^B = \frac{B^B}{(B+1)^{B+1}}.$$

The maximum occurs at $x = 1/(B+1)$ and is attained by the uniform distribution over $B+1$ modes. Because the bound decreases with B, no marginal at $b \geq B_{\max}$ exceeds $\lambda(B_{\max})$. Conversely, for the uniform distribution over $B_{\max}+1$ modes, $g_s(B_{\max}) = \lambda(B_{\max})$, so any smaller threshold would admit width $B_{\max}+1$. Finally,

$$g_s(B) - g_s(B+1) = \sum_i p_{s,i}^3 (1-p_{s,i})^B \geq 0,$$

so the objective is discrete-concave. Its smallest maximizer therefore includes exactly those successive paths for which $g_s(b) > \lambda(B_{\max})$; the strict inequality breaks ties toward less compute. □

Algorithm 1 COCONUT TREE: adaptive complete-path inference

Require: Prompt x, latent reasoner F_θ, maximum width $B_{\max}$
Ensure: Task-normalized predictions $\mathcal{T}_B = (a_1, \ldots, a_B)$

$s_0 \leftarrow \text{INITIALIZE}(x)$; $(a_1, \mathcal{L}) \leftarrow \text{COMPLETEPATH}(\text{copy}(s_0), \text{det})$
if a finite candidate set is available **then**
  $\hat{p}_{s_0,j} \leftarrow \exp(S_j) / \sum_m \exp(S_m)$ ▷ Scores from $\mathcal{L}$
else
  $t^* \leftarrow \min \arg\max_{t \in \mathcal{I}} D_2(q_{s_0,t})$; $\hat{p}_{s_0} \leftarrow q_{s_0,t^*}$ ▷ Probabilities from $\mathcal{L}$
$\lambda \leftarrow B_{\max}^{B_{\max}} / (B_{\max}+1)^{B_{\max}+1}$; $B \leftarrow 1$
while $B < B_{\max}$ and $\sum_j \hat{p}_{s_0,j}^2 (1 - \hat{p}_{s_0,j})^B > \lambda$ **do**
  $B \leftarrow B + 1$
if $B > 1$ **then**
  $\{(\tilde{s}_i, \tilde{c}_i)\}_{i=2}^{B} \leftarrow \mathcal{P}(s_0; B-1)$ ▷ Fresh stochastic proposals
  for $i = 2, \ldots, B$ **do** ▷ Independent; may be batched
    $(a_i, \cdot) \leftarrow \text{COMPLETEPATH}((\tilde{s}_i, \tilde{c}_i), \text{stoch})$
return $\mathcal{T}_B = (a_1, \ldots, a_B)$

Inference procedure. The proposed approach is detailed in Alg. 1. A deterministic first path provides prediction a_1 and recorded scores or probabilities $\mathcal{L}$. These determine a fixed proxy $\hat{p}_{s_0}$ and total width B, including the first path. The remaining $B-1$ paths use fresh stochastic proposals from s_0, each with its own state and KV cache. Allocation requires neither an additional model pass nor access to Y. Here S_j is candidate j's sequence log-likelihood, and $q_{s_0,t}$ is the full-vocabulary next-token distribution at eligible generated position $t \in \mathcal{I}$ of the first path. Implementation details appear in Appendix A.4. For this deterministic-first protocol, exact local optimality concerns the i.i.d. coverage surrogate; the width cap holds for any proxy distribution.

5 Experimental Validation

We evaluate publicly released CoCONUT checkpoints (124M GPT-2 and 1B Llama-3.2 backbones) on ProsQA, MultiArith, GSM8K, GSM-Hard, and SVAMP using pass@K, then test architectural transfer with released CoDI and CoLaR checkpoints. For separation, independent and shared arms use exactly matched model compute. For allocation, adaptive policies are compared with the *compute-matched fixed-width frontier*: the upper envelope of fixed-width policies $B \in \mathcal{B}$, interpolated at the adaptive policy's mean path count. This is the best input-agnostic fixed-width baseline at matched average compute, so gains reflect separation or allocation rather than additional compute. For allocation comparisons, we use mean completed-path count as the compute proxy; wall-clock time and estimated FLOPs are evaluated separately to avoid implementation-specific details affecting (agnostic) compute comparisons. Full experimental details are given in Appendix A.

Experimental map. Sec. 5.1 evaluates **Separation Necessity**: it tests whether preserving separate latent states reaches correct futures that equal-compute continuation from one shared state misses. Sec. 5.2 evaluates **Adaptive Efficiency**: it tests whether the **Principled Allocation** rule converts that opportunity into better coverage at matched average width. Additional validation appears in Sec. 5.3, with full diagnostics and ablations in Appendix C.

5.1 Separation Necessity

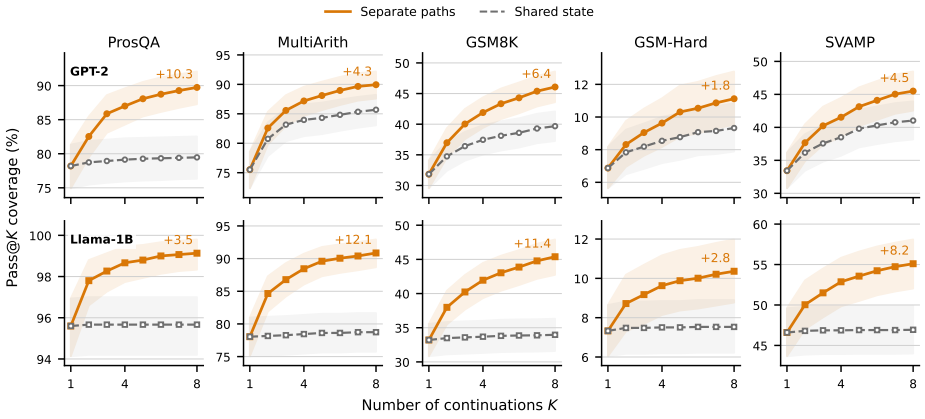


Figure 3: **Separation Necessity—Independent continuation improves coverage.** Pass@K for separate paths (orange) and equal-compute shared-state continuations (gray). Shading gives bootstrap 95% intervals; annotations give the paired gap at $K = 8$.

Table 1: **Separation Necessity.** Pass@8 for width-one, equal-compute shared-state, and separate-state continuation. Δ = Separate − Shared; [brackets] give paired 95% intervals.

	GPT-2 (124M)				Llama-3.2 (1B)			
Dataset	$B = 1$	Shared ($B = 8$)	Separate ($B = 8$)	**Δ [95% CI]**	$B = 1$	Shared ($B = 8$)	Separate ($B = 8$)	**Δ [95% CI]**
ProsQA	79.4%	79.5%	**89.7%**	+10.3 [8.3, 12.3]	97.6%	95.7%	**99.1%**	+3.5 [2.4, 4.7]
MultiArith	80.9%	85.7%	**89.9%**	+4.3 [3.1, 5.5]	84.0%	78.7%	**90.9%**	+12.1 [10.1, 14.1]
GSM8K	34.0%	39.7%	**46.1%**	+6.4 [5.4, 7.5]	36.1%	34.0%	**45.4%**	+11.4 [10.1, 12.8]
GSM-Hard	7.7%	9.3%	**11.1%**	+1.8 [1.3, 2.4]	8.2%	7.5%	**10.4%**	+2.8 [2.2, 3.6]
SVAMP	35.6%	41.0%	**45.5%**	+4.5 [3.5, 5.5]	48.5%	46.9%	**55.1%**	+8.2 [6.9, 9.4]

We test the practical value of preserving distinct proposal states by comparing independent continuations (Seperate) with a shared-state control (Shared) under matched compute. Fig. 3 reports pass@K for $K = 1, \ldots, 8$, and Table 1 summarizes the $K = 8$ endpoint. Together, these results support **Separation Necessity**: independent paths outperform both $B = 1$ and the shared-state control in all evaluated conditions, exceeding the latter by 1.8–12.1 points at $K = 8$ and reaching correct futures missed by continuation from a single shared state. Because these checkpoints were trained natively at width one, this benefit arises despite pressure to compress useful future information into one latent state. It requires no additional training or signals, only test-time scaling using information already present in the model, empirically demonstrating the benefit of independent continuations.

5.2 ADAPTIVE EFFICIENCY

Table 2: **Adaptive Efficiency.** COCONUT TREE at $B_{\max} = 8$ versus its compute-matched fixed-width frontier. Δ = COCONUT TREE − Fixed; mean width gives realized cost.

	GPT-2 (124M)				Llama-3.2 (1B)			
Dataset	Mean Width	COCONUT TREE	Fixed Frontier	Δ [95% CI]	Mean Width	COCONUT TREE	Fixed Frontier	Δ [95% CI]
ProsQA	2.182	**85.80%**	83.13%	+2.67 [1.78, 3.62]	1.118	**98.00%**	97.65%	+0.35 [0.03, 0.76]
MultiArith	2.128	**86.67%**	83.64%	+3.03 [2.05, 3.96]	1.919	**88.45%**	85.50%	+2.95 [2.02, 3.89]
GSM8K	3.289	**41.65%**	40.33%	+1.32 [0.76, 1.88]	3.462	**42.15%**	40.79%	+1.36 [0.93, 1.78]
GSM-Hard	2.174	**9.78%**	8.82%	+0.96 [0.53, 1.42]	2.449	**9.60%**	9.08%	+0.52 [0.24, 0.81]
SVAMP	2.943	**42.03%**	39.91%	+2.12 [1.46, 2.67]	2.689	**52.40%**	51.15%	+1.25 [0.73, 1.72]

Having established the value of separation, the remaining question is whether width can be allocated automatically. The exact local rule for COCONUT TREE in Prop. 4 does so using only $B_{\max}$; setting $B_{\max} = 8$ throughout yields $\lambda(8) = 8^8/9^9 \approx 0.043$ without calibration.

Table 2 compares COCONUT TREE with the compute-matched fixed-width frontier at the same realized mean cost, isolating the benefit of adaptive allocation from that of additional expansions. Sweeping $B_{\max}$ yields monotonically increasing coverage across all ten trajectories (Fig. 4); condition-level results and the increasing mean gain are in Appendix C.1. Together, these results establish **Adaptive Efficiency**: one common rule adapts width per example without a dataset- or model-specific operating point (Fig. 6) and convincingly outperforms the compute-matched frontier in all evaluated conditions.

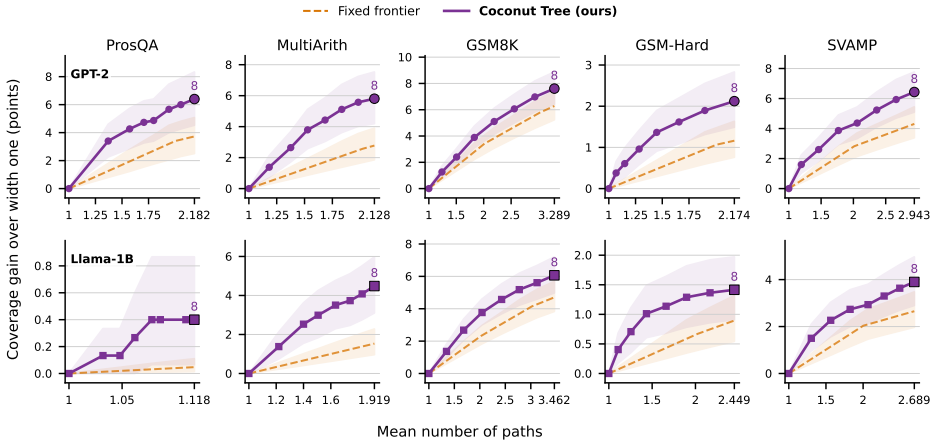


Figure 4: **Adaptive Efficiency—Coverage–width frontiers.** COCONUT TREE (purple) and compute-matched fixed width (orange) for $B_{\max} = 1, \ldots, 8$; shading gives bootstrap 95% intervals.

Gains over the compute-matched frontier span 0.35–3.03 points. The smallest, 0.35 points on ProsQA, is a near-ceiling case: deterministic width-one coverage is already 97.6%, so the rule leaves 92.4% of examples at width one (mean width 1.118) while still improving coverage. Across conditions, width is heterogeneous, with width-one rates spanning 26.4–92.4% (Fig. 6), showing that extra paths are allocated selectively (where justified) rather than uniformly (Fig. 4).

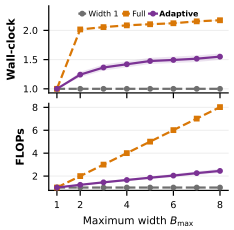


Figure 5: **Resource savings.** Relative average costs; bands give bootstrap 95% intervals.

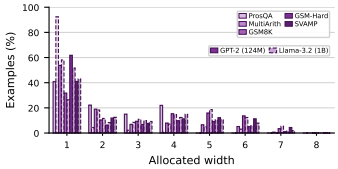


Figure 6: **Adaptive widths.** Per-example allocations by dataset and backbone.

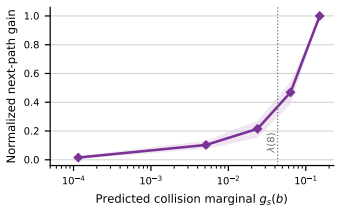


Figure 7: **Score validation.** Normalized next-path gain; the dotted line marks $\lambda(8)$.

5.3 Additional Validation (Expanded in Appendix C)

End-to-end resource savings Fig. 5 compares wall-clock time and FLOPs across $B_{\max}$ for width one, full width $B = B_{\max}$, and adaptive $B(s) \leq B_{\max}$. This cap-matched resource test complements the compute-matched coverage comparison above (Appendix B). Due to the heterogeneous nature and width saving (Fig. 6) of the adaptive rule, the proposed approach (relative to full width) reduces wall-clock time by a dataset-weighted mean of 27.1%, estimated dense-model FLOPs by 69.5%, peak live CUDA allocation by 70.5%, and gross board energy by 31.9%. It is faster in all evaluated conditions, with latency reductions from 13.6% to 60.0%. The score itself adds only 0.36–0.56 ms, and the local stopping calculation averages 3.7 µs, providing a substantial speedup over fixed widths.

Principled Allocation: Branch-score validation The stopping theorem uses $g_s(b)$ as a proxy for the marginal value of expanding from b to $b+1$ paths. For each example and b, Fig. 7 pairs the score before expansion with the realized $\Delta_{s,b} = \text{pass@}(b+1) - \text{pass@}\, b$; Appendix C.2 gives condition-level results. Above-threshold opportunities yield 0.98–5.41 points of marginal coverage per path versus 0.12–0.90 below threshold, and every paired example-cluster bootstrap interval excludes zero. Thus, the collision marginal predicts when expansion is useful, not merely where the output-token distribution is diffuse, supporting its use as a proxy for marginal expansion value.

Architectural generality Separation generalizes across architectures: separate pass@8 exceeds the equal-compute shared-state control in all 24 evaluated conditions, with every paired interval excluding zero (Table 3, left). Allocation (Table 3, right) transfers strongly to CoDI: all ten point estimates are positive and nine intervals exclude zero; Llama ProsQA touches zero under near-ceiling coverage. CoLaR retains the separation benefit, while adaptive transfer is mixed: positive on MultiArith, uncertain on SVAMP, and equal on GSM8K and GSM-Hard. Overall, the additional architectures support both separation and allocation.

Table 3: **Architectural generality.** Mean improvement Δ against each method's control; per-dataset results are in Appendix D.

Architecture Backbone	Separation Necessity Separation Δ [range]	Principled Allocation Adaptive Δ [range]
Coconut		
GPT-2	$\mathbf{+5.77}$ [1.79, 10.27]	$\mathbf{+2.24}$ [0.96, 3.03]
Llama-1B	$\mathbf{+7.72}$ [2.83, 12.13]	$\mathbf{+1.37}$ [0.35, 2.95]
CoDI		
GPT-2	$\mathbf{+5.09}$ [1.54, 8.00]	$\mathbf{+2.40}$ [0.98, 3.51]
Llama-1B	$\mathbf{+1.87}$ [0.40, 3.94]	$\mathbf{+0.81}$ [0.32, 1.25]
CoLaR		
Llama-1B	$\mathbf{+9.05}$ [2.17, 12.93]	$\mathbf{+0.41}$ [−0.19, 1.14]

6 Conclusion

We characterized when a single latent state cannot losslessly emulate an explicit beam and introduced Coconut Tree to allocate reasoning width per problem. Our analysis identifies conflation of beams requiring different future outcomes as an obstruction to lossless emulation, and characterizes when mode-specific computation has strict value. Motivated by these results, Coconut Tree uses output-token probabilities to estimate the marginal coverage gain of additional independent paths. A tight universal bound yields a stopping rule that exactly optimizes the penalized local coverage surrogate and guarantees the prescribed width cap $B_{\max}$.

Across Coconut, CoDI, and CoLaR, independent continuations improve pass@K over an equal-compute shared-state control in all 24 evaluated architecture–backbone–dataset conditions. Adaptive allocation provides consistent average coverage gains over fixed-width frontiers at matched mean path count across the evaluated backends. In live Coconut inference with $B_{\max} = 8$, adaptive allocation reduces latency by 13.6–60.0% and estimated dense-model FLOPs by 50.3–84.1% relative to always using the full $B_{\max}$ paths. Together, these results demonstrate the value of preserving distinct latent continuations and allocating their number according to predicted marginal benefit, making latent reasoning width a problem-dependent inference decision.

Limitations **Models.** Public latent-reasoning models remain limited to Coconut, CoDI, and CoLaR checkpoints with GPT-2 or Llama backbones up to 1B parameters, leaving transfer to larger models untested. **Tasks.** All evaluated tasks have verifiable answers. They do not establish transfer to open-ended generation such as creative writing, where there may be no unique target Y and pass@K does not directly measure output quality. **Rule assumption.** The adaptive rule uses output-token probabilities as a proxy for future-equivalence modes under the self-consistent model of Prop. 3. Although the gains and diagnostics support this proxy empirically, token uncertainty need not reflect useful semantic diversity, and no proxy can recover information already lost by a collapsed state.

Reproducibility Statement

Source code is provided alongside the supplementary material. The included `README.md` provides the complete setup, checkpoint downloads, directory structure, and commands for reproducing each experiment. Additional reproducibility details are available in Sec. A.5.

AI disclosure

In this work, we used generative AI tools for improving the text, readability, and refining figures in the paper. Additionally, we used AI for critiquing the paper, in an effort to improve clarity and exposition of the final version. Standard coding agents were used to assist in the development, with code verified by the main author. We have reviewed all AI-assisted work, and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of AI.

Ethics Statement

No ethical concerns arise from this work.

References

Benjamin Patrick Evans, Sumitra Ganesh, and Leo Ardon. Entropy-informed decoding: Adaptive information-driven branching. *Proceedings of the International Conference on Machine Learning*, 2026a.

Benjamin Patrick Evans, Sumitra Ganesh, and Leo Ardon. SEMIE: Semantic entropy-informed decoding. In *ICLR 2026 Workshop on Latent and Implicit Thinking*, 2026b. URL https://openreview.net/forum?id=w8vTkT6NkT.

Halil Alperen Gozeten, M. Emrullah Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=sTPKDKn5ig.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E. Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

Xuan Li, Yining Wang, Yuchen Liu, Guanjun Liu, Delai Qiu, Shengping Liu, Jiaen Liang, Wei Huang, Jun Yu, and Junnan Zhu. Later: Efficient test-time reasoning via latent exploration and explicit verification. *arXiv preprint arXiv:2605.07315*, 2026.

Alex Ning, Yen-Ling Kuo, and Gabe Gomes. Learning when to stop: Adaptive latent reasoning via reinforcement learning. *arXiv preprint arXiv:2511.21581*, 2025.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, Suzhou, China, nov 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of LLM reasoning chains. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/0706261aedab63814a2b73c32564b4c4-Abstract-Conference.html.

Minghan Wang, Ye Bai, Thuy-Trang Vu, Ehsan Shareghi, and Gholamreza Haffari. Gts: Inference-time scaling of latent reasoning with a learnable gaussian thought sampler. *arXiv preprint arXiv:2602.14077*, 2026.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. SIM-CoT: Supervised implicit chain-of-thought. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=6YRJ4jmVQl.

Haoyi Wu, Zhihao Teng, and Kewei Tu. Parallel continuous chain-of-thought with jacobi iteration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 914–926, Suzhou, China, nov 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.47. URL https://aclanthology.org/2025.emnlp-main.47/.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot++: Test-time scaling with soft chain-of-thought reasoning. *arXiv preprint arXiv:2505.11484*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.

Runyang You, Yongqi Li, Meng Liu, Wenjie Wang, Liqiang Nie, and Wenjie Li. Parallel test-time scaling for latent reasoning models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44703–44717, San Diego, California, United States, jul 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.2069. URL https://aclanthology.org/2026.acl-long.2069/.

Xiandong Zou, Jing Huang, Jianshu Li, and Pan Zhou. Latent thought flow: Efficient latent reasoning in large language models. *arXiv preprint arXiv:2606.16222*, 2026.

APPENDIX

Appendix A details the experimental protocol, Appendix B expands the resource measurements, Appendix C validates the allocation rule and its proxies, and Appendix D reports architectural transfer.

A EXPERIMENTAL AND IMPLEMENTATION DETAILS

A.1 DATASETS AND CHECKPOINTS

We evaluate the complete held-out splits: 500 ProsQA examples, 580 MultiArith examples, 1,319 GSM8K examples, 1,319 GSM-Hard examples, and 1,000 SVAMP examples. When presenting results, we assign weight of 1 to each dataset, except GSM and GSM-Hard, where we assign each a weighting of 0.5 each to account for the similarity of the two datasets and prevent artificially skewing results towards methods which do well only on GSM8k style problems. ProsQA has a binary answer and explicit reasoning graphs; the four arithmetic tasks use open greedy numeric decoding. The primary COCONUT runs use at most 32 new answer tokens; architecture-transfer adapters preserve the released models' native decoding formats and budgets.

For GPT-2, ProsQA uses the public epoch-16 checkpoint from `bmarti44/coconut-curriculum-checkpoints` at its three-latent curriculum stage, while all four arithmetic datasets use the released `ModalityDance/latent-tts-coconut` math checkpoint with six latent steps. For Llama-3.2-1B, ProsQA uses `connordilgren/llama32-1b-prosqa-coconut`, and all four arithmetic datasets use the GSM8K-trained `connordilgren/llama32-1b-gsm8k-coconut`; both released Llama checkpoints use six latent steps. The raw Llama state dictionaries are loaded into the Llama-3.2-1B-Instruct architecture with its tokenizer and three COCONUT tokens in their training order.

The CODI transfer uses the original authors' released inference wrapper and separate ProsQA- and GSM-trained checkpoints for both GPT-2 and Llama-3.2-1B, with six latent steps. Each GSM-trained checkpoint is evaluated without retraining on all four arithmetic datasets. The COLAR transfer uses the released `ModalityDance/latent-tts-colar` 1B Llama checkpoint, its native `Thinking speed: 2` prompt, learned latent head, end-marker stopping, and variable latent horizon. We evaluate it on the four arithmetic datasets for which it was released. We do not report COLAR on ProsQA because the released model and inference interface support arithmetic reasoning rather than ProsQA's graph-reasoning task; treating a custom cross-task answer adapter as architectural transfer would confound the comparison.

A.2 HYPERPARAMETERS

All stochastic path runs use dropout probability 0.1. GPT-2 and CODI activate their native dropout modules. Llama-3.2 has no ordinary `nn.Dropout` modules, so its stochastic transitions activate attention dropout at the same probability; the COLAR adapter installs the same probability in its latent transition modules. Deterministic paths and scoring disable dropout. This applies the same Monte Carlo dropout intervention rather than introducing a model-specific allocation parameter.

The open-answer proxy and its ablations require no additional stochastic continuations. One deterministic feature pass records vocabulary-head statistics at every non-special generated position, using batch size one for ProsQA and the canonical batch size eight for all four arithmetic datasets. The main max-D_2 proxy selects the position with the largest collision-effective support; the terminal proxy selects the last position, and the mean proxy averages its C_B curves across positions. All width rules are evaluated on the same nested path banks, without using gold answers to choose widths.

We evaluate three stochastic generation seeds (0, 1, and 2) on every example in every reported model–dataset condition, giving 1,500 ProsQA, 1,740 MultiArith, 3,957 GSM8K, 3,957 GSM-Hard, and 3,000 SVAMP example–seed trials per applicable checkpoint. To make every trial deterministic and resumable, the runner initializes its random state with the sum of the run seed and example index. Reported point estimates first average paired outcomes over seeds within each example. Confidence intervals then resample examples as clusters while retaining all three seeds, so repeated generations of one example are not treated as independent test observations.

A.3 COMPUTE MATCHING AND UNCERTAINTY

Both ProsQA arms use eight proposal transitions, the same latent horizon, and the same candidate-scoring rows. They average 181.824 backbone rows per example for GPT-2 and 198.880 for Llama-1B. All four arithmetic datasets use the same protocol followed by open decoding. If EOS makes one arm finish first, unused forward passes pad the shorter arm after its answer is complete. The two arms therefore have zero paired row mismatches in all ten model–dataset conditions. For GPT-2/Llama-1B, respectively, MultiArith averages 81.076/88.129 rows and GSM8K averages 138.287/91.997 rows; GSM-Hard averages 162.622/100.980 rows; and SVAMP averages 82.899/89.328 rows. The CODI GPT-2 conditions average 253.824, 104.257, 107.522, 122.183, and 106.672 rows on ProsQA, MultiArith, GSM8K, GSM-Hard, and SVAMP; the corresponding CODI Llama-1B conditions average 246.880, 112.000, 113.961, 125.544, and 113.005 rows. The COLAR arithmetic conditions average 122.097, 164.983, 292.962, and 107.605 rows. Every reported architecture-transfer condition has zero paired row mismatches.

Confidence intervals use a paired cluster bootstrap over test examples. Each resampled cluster retains all three stochastic seeds, and seed-level outcomes are averaged within the sampled example; this preserves the dependence among repeated generations and makes the number of test examples, rather than the number of runs, the inferential sample size. Separate-path intervals use 10,000 replicates with bootstrap seed 0. Collision-marginal intervals use 5,000 replicates with seed 20260731; local widths are retained and the compute-matched fixed-width frontier is recomputed inside every replicate. The same paired procedure is applied independently at every value $B_{\max} = 1, \ldots, 8$ in the sensitivity sweep. Fig. 4 shows marginal bands for adaptive-minus-width-one and frontier-minus-width-one coverage from these replicates; Table 2 uses the paired interval for their difference.

The branch-score diagnostic evaluates all seven prospective additions per example. Fixed bands have boundaries 10^{-3}, 10^{-2}, $\lambda(8)$, and 10^{-1}. Its 5,000 bootstrap replicates resample examples while retaining all widths and stochastic runs from each selected example. Intervals for the above-minus-below-threshold comparison are recomputed from the two conditional means within every replicate.

A.4 PATH GENERATION AND SHARED-STATE CONTROL

The implementation extends the released LATENTTTS / COCONUT code with branch scoring and a minimal path runner. For each example, it first performs deterministic width-one latent reasoning. It then draws a bank of $B_{\max} = 8$ first latent state–cache pairs using dropout and continues each pair independently through the full latent horizon. For the equal-compute shared-state control, the runner restores the same downstream random-number state, makes eight copies of the first proposal, and continues every copy for the same horizon. The two arms therefore differ in whether their latent futures remain separate, rather than in the number of proposal or continuation steps.

Each completed path scores the two ProsQA candidates by sequence log likelihood or uses open numeric decoding for the arithmetic tasks. Separation coverage is evaluated on nested stochastic-path prefixes. For allocation, both adaptive and fixed-width policies instead use the deterministic first prediction followed by the first $B - 1$ independent stochastic predictions (Alg. 1). These experiments replay the stored paths, so increasing width never changes an earlier path or requires further model inference during analysis.

Inference implementation. In Alg. 1, INITIALIZE retains the prompt inputs before the first latent transition. COMPLETEPATH applies the remaining latent transitions and terminal prediction procedure, returning a task-normalized answer and recorded scores or probabilities $\mathcal{L}$. The `det/stoch` flag controls latent-transition stochasticity; terminal prediction uses candidate scoring or greedy decoding in both cases. For the token proxy, $\mathcal{I}$ excludes positions whose generated token is a special token. Live inference falls back to all generated positions if none qualify. Position filtering does not alter the full-vocabulary distributions $q_{s_0,t}$.

A.5 REPRODUCIBILITY

The supplementary `README.md` provides the complete setup, checkpoint downloads, directory structure, and commands for reproducing each experiment. Our implementation extends the publicly

released LATENTTTS codebase[2], retaining its latent-generation interface and model wrappers while adding the shared-state control, branch scoring, adaptive width allocation, and experiment logging used in this work. The implementation is Python/PyTorch based, with dependencies listed in `requirements-minimal.txt`.

The repository includes the normalized evaluation data and stored path, proxy, and compute records used in the paper, so all analyses, tables, and figures can be regenerated on CPU without model inference. Regenerating model outputs requires the public checkpoints and a CUDA-capable GPU. All evaluated models fit on a single 24 GB-class GPU; the reported runs used NVIDIA A10G devices. No multi-GPU execution or additional model training is required. Model–dataset groups are independently resumable and may be run serially on one GPU or distributed across machines.

The stored records also permit direct verification of the experimental controls: independent and shared arms match proposal count, latent horizon, scoring work, random stream, and backbone rows, while adaptive replay preserves earlier paths and compares against the fixed-width frontier at matched mean compute.

B WALL-CLOCK AND RESOURCE MEASUREMENTS

B.1 MEASUREMENT PROTOCOL

We benchmark the two primary COCONUT backbones on one NVIDIA A10G (23 GB) in bfloat16. A fixed pseudorandom sample of 32 test examples from each dataset is measured three times at every realized width $B = 1, \ldots, 8$. Before measurement, every model–width batch shape is warmed once; width–policy blocks are then randomly ordered to reduce thermal and ordering bias. Each example is synchronized before and after inference. End-to-end time includes tokenization, host-to-device transfer, prompt embedding, deterministic latent reasoning, the open-answer or candidate score, width allocation, and all selected continuations, but excludes model loading.

Fixed-width runs do not calculate the adaptive proxy. Separate score-enabled runs measure the method at every realized width; for each cap, the analysis uses the actually measured record at that example's selected width and adds the measured stopping-rule CPU time. Thus no latency is interpolated from path count. Confidence bands resample examples as clusters while retaining all three timing repetitions. GPU utilization, memory, power, temperature, and clocks are sampled every 200 ms over each block after a loaded-model idle baseline. The analytical FLOP counter uses the observed batch, query, and cache lengths for every forward pass and includes dense attention projections, QK/AV products, MLPs, and the vocabulary head. It excludes embeddings, normalization, activations, sampling, and proxy elementwise operations.

B.2 ENDPOINT RESOURCE TELEMETRY

Table 4: **Live resource use at $B_{\max} = 8$.** Values are dataset-weighted means. Peak memory is live CUDA allocation above the loaded-model baseline; energy is gross GPU-board energy estimated from sampled power. Adaptive rows include score and allocation overhead.

Backbone	Policy	Mean B	Time (ms)	TFLOPs	Peak MiB	GPU util.	Power (W)	Energy (J)
GPT-2	Full width	8.000	170.2	0.285	293	27.8%	80.6	12.97
	Adaptive	**2.668**	**130.2**	**0.096**	**84**	**20.3%**	**74.3**	**9.44**
Llama-1B	Full width	8.000	414.7	2.625	347	54.4%	123.2	51.29
	Adaptive	**2.156**	**278.3**	**0.558**	**72**	**45.1%**	**110.7**	**30.31**

The deterministic first path is an irreducible latency floor. For GPT-2, deterministic/extra-branch time is 80.5/89.3 ms at full width and 80.8/48.9 ms adaptively; for Llama-1B it is 187.0/227.2 ms and 187.5/90.4 ms, respectively. Tokenization plus prompt embedding is below 0.5 ms. Against a same-width run without scoring, the proxy adds only 0.56 ms for GPT-2 and 0.36 ms for Llama-1B; local allocation adds 3.7 microseconds. Median per-example timing variation across repetitions is below 0.8% for every policy and backbone.

The gross-energy estimate combines shorter execution with lower mean power. Loaded-model idle power is 56–57 W; subtracting it reduces GPT-2 energy from 3.27 to 2.03 J and Llama-1B

[2] `https://github.com/ModalityDance/LatentTTS`

energy from 27.86 to 14.59 J. We treat gross energy as primary because the A10G does not expose a cumulative-energy counter. GPU core and memory clocks remain fixed at 1710 and 6250 MHz, temperatures stay between 31 and 37 °C, and the PCIe link remains Gen4 ×8, giving no evidence of thermal throttling. Board-level memory remains near 1.2/3.9 GiB for GPT-2/Llama because the warmed PyTorch allocator retains reservations; the live-allocation column in Table 4 is therefore the meaningful width-dependent memory measure.

B.3 Per-dataset scaling

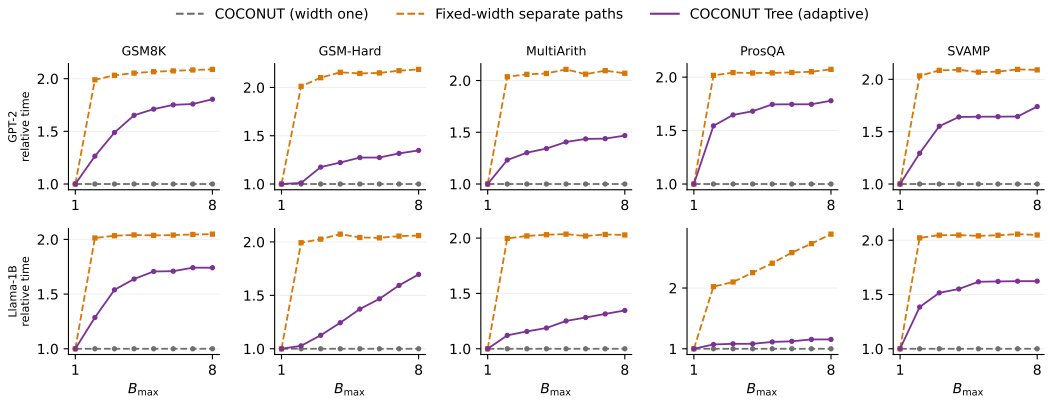


Figure 8: **Per-dataset wall-clock scaling.** Rows give GPT-2 and Llama-1B; columns give the five datasets. Each condition is normalized by its own width-one time. Full fixed width is orange, COCONUT TREE is purple, and the width-one reference is gray.

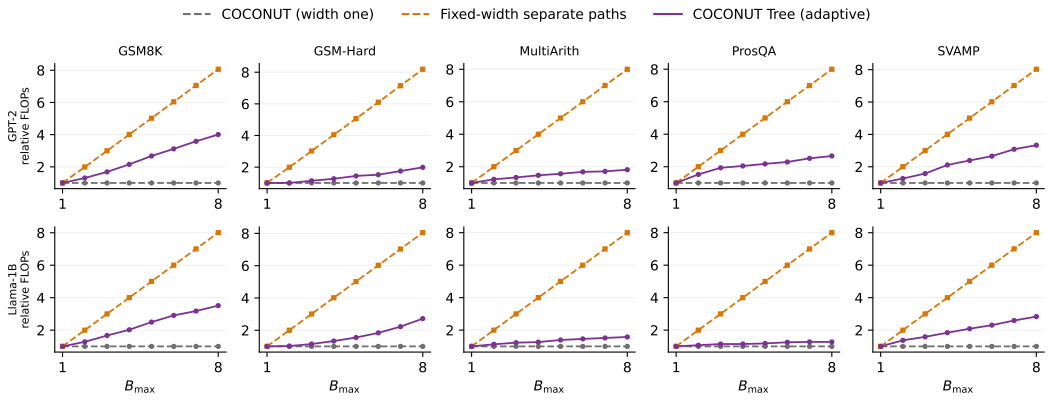


Figure 9: **Per-dataset dense-model FLOP scaling.** The full-width policy grows linearly with $B_{\max}$, while the adaptive slope follows the heterogeneous realized widths. FLOPs use measured forward shapes and the analytical definition in this subsection.

Across individual conditions at $B_{\max} = 8$, adaptive reductions range from 13.6–60.0% in wall-clock time, 50.3–84.1% in dense-model FLOPs, 53.9–83.6% in peak live allocation, and 15.4–68.1% in gross energy. The benefit is therefore not driven by a single dataset or backbone, but is consistent across configurations.

Measurement Limitation **Hardware-dependent latency** Live measurements use one NVIDIA A10G and the implementation's batched branch execution. Different accelerators, serving batch sizes, and concurrent workloads can change wall-clock speedups even when model FLOPs are unchanged. Board energy is estimated from sampled power rather than a hardware cumulative-energy counter. The FLOP and live allocation reductions are therefore more hardware-stable than the precise latency and energy values.

C Allocation Diagnostics and Ablations

C.1 Cap sensitivity

Table 5 reports the complete local sweeps used in Fig. 4, while Fig. 10 summarizes how their advantage changes with the cap. Each row changes only $B_{\max}$; the threshold is always set analytically as $\lambda(B_{\max}) = B_{\max}^{B_{\max}}/(B_{\max}+1)^{B_{\max}+1}$, and the same setting is applied to every model and dataset. The comparator is the compute-matched fixed-width frontier: the upper boundary of the convex hull of the fixed-width operating points $B \in \{1, \ldots, 8\}$. At noninteger mean widths, linear interpolation between fixed-width points matches aggregate compute exactly. The frontier is recomputed at the realized mean width inside each bootstrap replicate. Within each model block, bold marks the better coverage between Coconut Tree and its compute-matched fixed-width frontier; ties are both bolded.

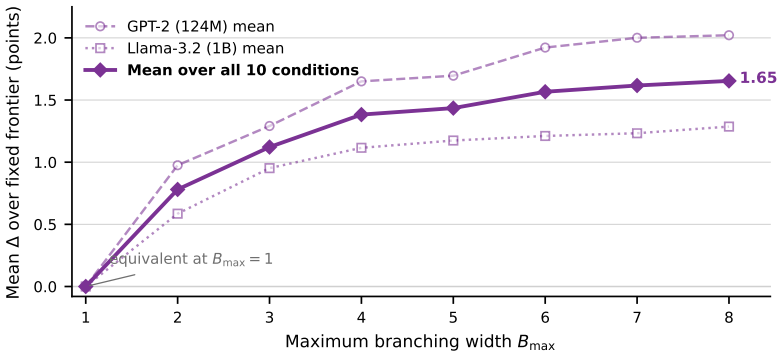


Figure 10: **Adaptive advantage increases with the available cap.** Δ is Coconut Tree coverage minus the compute-matched fixed-width frontier, averaged equally over the five datasets within each backbone. The bold purple curve averages all ten model–dataset conditions. At $B_{\max} = 1$, both policies are exactly width one and $\Delta = 0$ by construction. Increasing the cap permits more heterogeneous per-example allocations, and the mean advantage grows monotonically over the evaluated range.

The aggregate gain rises from exactly zero at $B_{\max} = 1$ to 0.78 points at $B_{\max} = 2$, 1.12 at $B_{\max} = 3$, 1.38 at $B_{\max} = 4$, and 1.65 at $B_{\max} = 8$. At the largest cap, the corresponding backbone-specific means are 2.02 points for GPT-2 and 1.29 for Llama-1B. This descriptive trend does not require every individual condition to improve at every step; rather, a larger ceiling gives the local rule more scope to vary width across examples, while the matched fixed-width comparator remains input-agnostic. The full condition-level results and intervals are retained in Table 5.

C.2 Per-condition branch-score validation

Fig. 11 gives the complete dataset and backbone breakdown underlying the aggregate diagnostic in Fig. 7. Each condition is normalized by its own largest observed score-band gain, so the curves test ranking rather than absolute task difficulty. The highest-score band has the largest point-estimated gain in every condition.

C.3 Allocation-rule ablations

To ensure the adaptive advantage is not a generic consequence of mapping any information number directly to width, we perform ablations on the chosen stopping criterion. We compare the proposed marginal rule with three direct information-number baselines: D_1, D_2, and D_∞.

For a discrete distribution $r = (r_i)_i$, let

$$D_1(r) = \exp\left(-\sum_i r_i \log r_i\right), \qquad D_2(r) = \frac{1}{\sum_i r_i^2}, \qquad D_\infty(r) = \frac{1}{\max_i r_i}. \tag{13}$$

Table 5: **COCONUT TREE sensitivity to the sole hyperparameter across both backbones.** Coverage and adaptive-minus-fixed-frontier difference (Δ), in percentage points, for every cap. The threshold is derived analytically rather than fitted. Within each backbone, the higher of ours and the compute-matched fixed frontier is bold; ties are both bold.

			GPT-2 (124M)				*Llama-3.2 (1B)*			
Dataset	$B_{\max}$	λ	Mean B	Ours	Fixed	Δ [95% CI]	Mean B	Ours	Fixed	Δ [95% CI]
ProsQA	1	.250000	1.000	**79.40**	**79.40**	+0.00 [+0.00, +0.00]	1.000	**97.60**	**97.60**	+0.00 [+0.00, +0.00]
	2	.148148	1.370	**82.80**	80.66	+2.14 [+1.47, +2.86]	1.032	**97.73**	97.61	+0.12 [−0.01, +0.31]
	3	.105469	1.572	**83.67**	81.34	+2.32 [+1.56, +3.13]	1.048	**97.73**	97.62	+0.11 [−0.02, +0.30]
	4	.081920	1.706	**84.13**	81.80	+2.33 [+1.56, +3.17]	1.062	**97.87**	97.62	+0.24 [−0.01, +0.56]
	5	.066980	1.798	**84.27**	82.11	+2.15 [+1.41, +2.97]	1.078	**98.00**	97.63	+0.37 [+0.04, +0.79]
	6	.056653	1.942	**85.07**	82.60	+2.46 [+1.62, +3.41]	1.086	**98.00**	97.63	+0.37 [+0.04, +0.78]
	7	.049087	2.054	**85.40**	82.90	+2.50 [+1.63, +3.46]	1.110	**98.00**	97.64	+0.36 [+0.03, +0.76]
	8	.043305	2.182	**85.80**	83.13	+2.67 [+1.78, +3.62]	1.118	**98.00**	97.65	+0.35 [+0.03, +0.76]
MultiArith	1	.250000	1.000	**80.86**	**80.86**	+0.00 [+0.00, +0.00]	1.000	**83.97**	**83.97**	+0.00 [+0.00, +0.00]
	2	.148148	1.183	**82.24**	81.32	+0.92 [+0.36, +1.49]	1.219	**85.34**	84.33	+1.01 [+0.49, +1.59]
	3	.105469	1.376	**83.51**	81.81	+1.69 [+0.99, +2.42]	1.400	**86.49**	84.63	+1.86 [+1.11, +2.66]
	4	.081920	1.528	**84.66**	82.20	+2.46 [+1.61, +3.32]	1.507	**86.95**	84.81	+2.14 [+1.30, +3.02]
	5	.066980	1.684	**85.29**	82.59	+2.69 [+1.78, +3.61]	1.634	**87.47**	85.02	+2.45 [+1.61, +3.31]
	6	.056653	1.834	**85.98**	82.97	+3.00 [+2.01, +4.00]	1.741	**87.70**	85.20	+2.50 [+1.64, +3.36]
	7	.049087	1.983	**86.44**	83.35	+3.09 [+2.09, +4.06]	1.829	**88.05**	85.35	+2.70 [+1.81, +3.57]
	8	.043305	2.128	**86.67**	83.64	+3.03 [+2.05, +3.96]	1.919	**88.45**	85.50	+2.95 [+2.02, +3.89]
GSM8K	1	.250000	1.000	**34.04**	**34.04**	+0.00 [+0.00, +0.00]	1.000	**36.09**	**36.09**	+0.00 [+0.00, +0.00]
	2	.148148	1.239	**35.30**	34.85	+0.45 [+0.12, +0.81]	1.346	**37.45**	36.88	+0.58 [+0.29, +0.88]
	3	.105469	1.506	**36.44**	35.76	+0.69 [+0.25, +1.12]	1.683	**38.77**	37.64	+1.13 [+0.74, +1.53]
	4	.081920	1.829	**37.93**	36.85	+1.08 [+0.59, +1.58]	2.043	**39.85**	38.44	+1.41 [+0.95, +1.87]
	5	.066980	2.192	**39.15**	37.88	+1.27 [+0.74, +1.82]	2.428	**40.66**	39.15	+1.51 [+1.08, +1.97]
	6	.056653	2.559	**40.11**	38.74	+1.37 [+0.82, +1.90]	2.774	**41.27**	39.79	+1.48 [+1.04, +1.93]
	7	.049087	2.933	**41.02**	39.62	+1.40 [+0.79, +1.98]	3.123	**41.70**	40.36	+1.34 [+0.90, +1.77]
	8	.043305	3.289	**41.65**	40.33	+1.32 [+0.76, +1.88]	3.462	**42.15**	40.79	+1.36 [+0.93, +1.78]
GSM-Hard	1	.250000	1.000	**7.66**	**7.66**	+0.00 [+0.00, +0.00]	1.000	**8.19**	**8.19**	+0.00 [+0.00, +0.00]
	2	.148148	1.068	**8.04**	7.73	+0.31 [+0.09, +0.57]	1.108	**8.59**	8.26	+0.33 [+0.14, +0.57]
	3	.105469	1.149	**8.26**	7.82	+0.45 [+0.17, +0.78]	1.255	**8.90**	8.36	+0.54 [+0.25, +0.87]
	4	.081920	1.284	**8.62**	7.96	+0.66 [+0.34, +1.04]	1.434	**9.20**	8.47	+0.73 [+0.40, +1.09]
	5	.066980	1.448	**9.02**	8.13	+0.89 [+0.53, +1.30]	1.661	**9.33**	8.62	+0.70 [+0.39, +1.05]
	6	.056653	1.657	**9.27**	8.35	+0.92 [+0.51, +1.38]	1.898	**9.48**	8.78	+0.70 [+0.37, +1.04]
	7	.049087	1.898	**9.55**	8.61	+0.94 [+0.53, +1.39]	2.168	**9.55**	8.93	+0.62 [+0.30, +0.94]
	8	.043305	2.174	**9.78**	8.82	+0.96 [+0.53, +1.42]	2.449	**9.60**	9.08	+0.52 [+0.24, +0.81]
SVAMP	1	.250000	1.000	**35.60**	**35.60**	+0.00 [+0.00, +0.00]	1.000	**48.50**	**48.50**	+0.00 [+0.00, +0.00]
	2	.148148	1.194	**37.20**	36.14	+1.06 [+0.61, +1.54]	1.303	**50.00**	49.12	+0.88 [+0.52, +1.30]
	3	.105469	1.462	**38.20**	36.89	+1.31 [+0.79, +1.83]	1.564	**50.77**	49.65	+1.12 [+0.70, +1.58]
	4	.081920	1.768	**39.47**	37.75	+1.72 [+1.16, +2.30]	1.825	**51.23**	50.18	+1.06 [+0.68, +1.48]
	5	.066980	2.058	**39.97**	38.49	+1.47 [+0.90, +2.06]	2.066	**51.43**	50.59	+0.84 [+0.46, +1.24]
	6	.056653	2.361	**40.83**	38.98	+1.86 [+1.24, +2.45]	2.283	**51.80**	50.79	+1.01 [+0.59, +1.42]
	7	.049087	2.662	**41.53**	39.46	+2.07 [+1.41, +2.66]	2.493	**52.13**	50.98	+1.16 [+0.70, +1.60]
	8	.043305	2.943	**42.03**	39.91	+2.12 [+1.46, +2.67]	2.689	**52.40**	51.15	+1.25 [+0.73, +1.72]

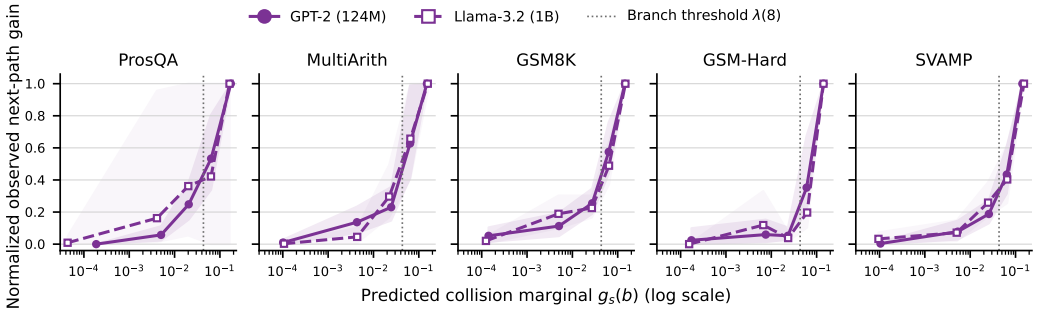


Figure 11: **Per-condition branch-score validation.** Points show realized marginal expansion gains within five score bands computed before expansion; shading gives example-cluster bootstrap 95% intervals. Solid circles denote GPT-2 and dashed open squares denote Llama-1B. The dotted line marks the derived threshold $\lambda(8)$.

Here, D_1 is the Shannon effective support used by the EDEN allocation rule (Evans et al., 2026a;b). Each baseline uses

$$B(s) = \min\{B_{\max}, \lfloor D_\alpha(\hat{p}_s) + 1/2 \rfloor\}, \qquad \alpha \in \{1, 2, \infty\}.$$

The effective mode count is rounded to its nearest feasible integer, with no threshold, exponent, or dataset-level budget. All methods share $B_{\max} = 8$. Because their natural mean widths differ, Table 6 reports coverage gain over the compute-matched fixed-width frontier at each rule's own mean width.

The proposed collision-marginal stopping has the largest point-estimated gain in all six original model–dataset conditions, whereas rounded D_1, D_2, and D_∞ do not significantly improve over the compute-matched frontier in general. On both GSM8K backbones, none of the three direct effective-count mappings shows a significant improvement over its compute-matched frontier, whereas marginal stopping does. Effective support is therefore useful evidence of multiplicity, but its rounded cardinality is not by itself a reliable stopping rule.

Table 6: **Parameter-free allocation baselines under a common cap.** Each cell gives adaptive-minus-fixed-frontier coverage in percentage points, its paired 95% interval, and mean width in parentheses. Results are aggregated over stochastic runs. Purple marks the proposed marginal rule.

Dataset	**Marginal rule (ours)**	round(D_∞)	round(D_2)	round(D_1)
GPT-2 (124M)				
ProsQA	+2.67 [1.78, 3.62] (2.182)	+2.12 [1.28, 3.00] (1.160)	+2.15 [1.45, 2.89] (1.328)	+2.00 [1.37, 2.67] (1.412)
MultiArith	+3.03 [2.05, 3.96] (2.128)	+2.39 [1.47, 3.34] (1.462)	+2.94 [1.91, 3.97] (1.859)	+2.77 [1.85, 3.65] (2.257)
GSM8K	+1.32 [0.76, 1.88] (3.289)	−0.38 [−1.07, 0.26] (3.469)	−0.31 [−1.03, 0.34] (4.635)	+0.44 [−0.13, 0.98] (5.362)
Llama-3.2 (1B)				
ProsQA	+0.35 [0.03, 0.76] (1.118)	+0.06 [−0.00, 0.20] (1.008)	+0.12 [−0.01, 0.32] (1.022)	+0.12 [−0.01, 0.31] (1.038)
MultiArith	+2.95 [2.02, 3.89] (1.919)	+1.00 [0.49, 1.57] (1.191)	+1.87 [1.13, 2.64] (1.397)	+2.23 [1.42, 3.06] (1.698)
GSM8K	+1.36 [0.93, 1.78] (3.462)	+0.00 [−0.48, 0.47] (2.766)	+0.34 [−0.18, 0.78] (3.813)	+0.16 [−0.34, 0.62] (4.585)

C.4 Mode-distribution proxy ablation

Table 7 isolates the distribution supplied to the marginal rule. On ProsQA, task-valid answer candidates provide a direct semantic distribution. We compare it with vocabulary distributions at the terminal generated position, averaged along the greedily generated continuation, or at the position with largest collision-effective support D_2. The open-answer tasks do not have an enumerable task-valid candidate set, so only vocabulary proxies are available.

We also test a position-free marginal envelope. For every prospective branch, it uses

$$\widetilde{g}_s(B) = \max_t \sum_v q_{s,t}(v)^2 \big(1 - q_{s,t}(v)\big)^B. \tag{14}$$

The rule continues while $\widetilde{g}_s(B) > \lambda(B_{\max})$. Each position's marginal is nonincreasing in B and satisfies Eq. (11); therefore their maximum is also nonincreasing, obeys the same bound, and retains the universal cap. This is the most direct extension of the stopping argument to multiple generated positions, and requires no additional model pass.

Answer candidates are strongest on GPT-2 ProsQA and tied with the max-D_2 token proxy on the near-ceiling Llama model. For open-answer tasks, max-D_2 is the main operational proxy because it selects one distribution by the collision quantity already used in the theory. Terminal, max-D_2, and max-marginal rules are effectively tied on MultiArith for both backbones. On GSM8K, max-D_2 gives the largest point-estimated gain for both GPT-2 (+1.32 points) and Llama-1B (+1.36 points), while nearby proxies remain competitive. The single-distribution max-D_2 rule therefore remains the closest operational match to the theoretical $C_B(p_s)$ objective across models.

Table 7: **Semantic-proxy ablation.** The marginal stopping rule is held fixed. Cells report gain over the compute-matched fixed-width frontier, paired 95% interval, and mean width in parentheses. A dash indicates that exact task-valid answer candidates are unavailable. Purple marks the operational proxy used by COCONUT TREE (ours) in each dataset.

Dataset	Answer candidates	Terminal token	Mean token	Max-D_2 token	Max marginal
GPT-2 (124M)					
	$+2.67$ $[1.78, 3.62]$	$+0.00$ $[0.00, 0.00]$	$+1.20$ $[0.55, 1.88]$	$+0.27$ $[-0.08, 0.60]$	$+0.30$ $[-0.03, 0.61]$
ProsQA	(2.182)	(1.000)	(1.256)	(3.778)	(3.798)
		$+3.04$ $[2.05, 3.96]$	$+2.16$ $[1.32, 3.04]$	$+3.03$ $[2.05, 3.96]$	$+3.02$ $[2.04, 3.95]$
MultiArith	—	(2.122)	(1.509)	(2.128)	(2.129)
		$+0.34$ $[-0.35, 1.07]$	$+0.99$ $[0.53, 1.46]$	$+1.32$ $[0.76, 1.88]$	$+1.14$ $[0.67, 1.62]$
GSM8K	—	(2.231)	(1.848)	(3.289)	(3.706)
Llama-3.2 (1B)					
	$+0.35$ $[0.03, 0.76]$	$+0.00$ $[0.00, 0.00]$	$+0.00$ $[0.00, 0.00]$	$+0.35$ $[0.03, 0.75]$	$+0.35$ $[0.03, 0.75]$
ProsQA	(1.118)	(1.000)	(1.000)	(1.132)	(1.132)
		$+2.97$ $[2.04, 3.91]$	$+1.16$ $[0.58, 1.77]$	$+2.95$ $[2.02, 3.89]$	$+2.95$ $[2.02, 3.89]$
MultiArith	—	(1.905)	(1.272)	(1.919)	(1.919)
		$+1.33$ $[0.85, 1.82]$	$+0.64$ $[0.34, 0.95]$	$+1.36$ $[0.93, 1.78]$	$+1.32$ $[0.90, 1.71]$
GSM8K	—	(3.184)	(1.517)	(3.462)	(3.538)

D ARCHITECTURE TRANSFER

We evaluate architectural transfer using additional latent-reasoning models. Checkpoint and adapter details are given in Appendix A. Every transfer condition uses the complete test split, stochastic seeds 0, 1, and 2, $B_{\max} = 8$, the same derived threshold, compute matching, and example-cluster bootstrap as the primary experiments. ProsQA uses exact candidate probabilities, while arithmetic tasks use the max-D_2 vocabulary proxy. No component of the allocation rule is retuned.

Per-dataset results. Table 8 reports both transfer tests. Separation compares independent and shared-state continuation at $B = 8$ with equal model compute. Adaptive allocation is compared with the compute-matched fixed-width frontier at its realized mean width.

Table 8: **Per-dataset architectural transfer.** Separation compares separate and shared-state continuation at $B = 8$. Adaptive results compare COCONUT TREE with the compute-matched fixed-width frontier. Coverage and differences are percentage points. Bold coverage is higher within each comparison; bold differences have paired 95% intervals excluding zero.

	Separation Necessity			**Adaptive Efficiency**			
Dataset	Shared	Separate	Δ [95% CI]	Mean width	**COCONUT TREE** (ours)	Fixed frontier	Δ [95% CI]
CODI *with GPT-2 (124M)*							
ProsQA	82.9	**90.9**	$+8.0$ $[+6.2, +9.9]$	1.466	**86.40**	82.89	$+3.51$ $[+2.39, +4.63]$
MultiArith	94.4	**97.7**	$+3.3$ $[+2.3, +4.4]$	1.414	**95.52**	92.38	$+3.14$ $[+2.11, +4.19]$
GSM8K	46.2	**52.5**	$+6.3$ $[+5.3, +7.3]$	2.886	**48.72**	46.92	$+1.80$ $[+1.16, +2.45]$
GSM-Hard	10.3	**11.8**	$+1.5$ $[+1.0, +2.1]$	1.684	**10.77**	9.79	$+0.98$ $[+0.60, +1.40]$
SVAMP	43.6	**48.8**	$+5.2$ $[+4.1, +6.3]$	2.792	**46.63**	45.08	$+1.55$ $[+1.07, +2.06]$
CODI *with Llama-1B*							
ProsQA	99.0	**99.4**	$+0.4$ $[+0.1, +0.8]$	1.040	**99.33**	99.01	$+0.32$ $[+0.00, +0.87]$
MultiArith	96.8	**97.7**	$+0.9$ $[+0.3, +1.6]$	1.179	**97.53**	96.78	$+0.75$ $[+0.17, +1.45]$
GSM8K	57.2	**61.2**	$+3.9$ $[+3.2, +4.7]$	2.552	**59.16**	57.91	$+1.25$ $[+0.83, +1.68]$
GSM-Hard	13.2	**14.2**	$+1.0$ $[+0.7, +1.4]$	2.024	**13.67**	13.02	$+0.65$ $[+0.35, +0.94]$
SVAMP	61.3	**65.1**	$+3.7$ $[+2.8, +4.7]$	2.392	**63.63**	62.41	$+1.22$ $[+0.76, +1.68]$
COLAR *with Llama-1B (arithmetic)*							
MultiArith	82.2	**95.1**	$+12.9$ $[+11.0, +14.9]$	1.097	**93.22**	92.07	$+1.14$ $[+0.52, +1.87]$
GSM8K	34.3	**39.9**	$+5.6$ $[+4.1, +7.0]$	1.781	42.63	**42.66**	-0.03 $[-0.46, +0.29]$
GSM-Hard	7.6	**9.7**	$+2.2$ $[+1.4, +3.0]$	2.024	9.45	**9.64**	-0.19 $[-0.48, +0.01]$
SVAMP	53.2	**63.6**	$+10.4$ $[+8.7, +12.0]$	1.593	**56.73**	56.53	$+0.20$ $[-0.42, +0.75]$

Separation Necessity. Separate continuation exceeds the equal-compute shared-state control in all fourteen transfer conditions, with every paired interval excluding zero. Gains range from 0.4–8.0 points across the ten CODI conditions and 2.2–12.9 points across the four supported COLAR conditions. Together with the ten COCONUT conditions in Table 1, this gives positive separation effects in all 24 evaluated conditions.

Adaptive Efficiency. Adaptive allocation transfers strongly to CoDI: all ten point estimates are positive and nine intervals exclude zero. The remaining Llama ProsQA interval touches zero in a near-ceiling condition with 99.0% width-one coverage. Transfer to CoLAR is mixed: MultiArith improves clearly, SVAMP has a small uncertain positive estimate, and GSM8K and GSM-Hard match the compute-matched fixed-width frontier.

For CoLAR, above-threshold expansion opportunities yield 13.69 versus 0.70 points of marginal coverage on MultiArith (difference CI $[+7.00, +20.21]$), but 1.78 versus 1.41 points on GSM8K (CI $[-0.15, +0.93]$). Separation therefore transfers broadly, while the strength of adaptive gains varies with how accurately the zero-extra-pass output-token distribution ranks marginal expansion value.