MODULAR ADAPTIVE REJECTION SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive language models are increasingly used in combination: one wants to *intersect* two models so that outputs are probable under both, take their *union*, *sharpen* a model, or *contrast* it against a weaker model. Existing approaches for sampling from such combinations are either efficient but not faithful to the distribution induced by the model combination, or faithful but not efficient, paying for every disagreement between the models at every sample. We present *Modular Adaptive Rejection Sampling* (MARS), an exact rejection sampling algorithm that supports complex combinations of distributions and that *learns from its rejections*. MARS generates a sample token by token, keeping for every prefix an upper bound on the total target weight of the sequences that start with that prefix and choosing each next token in proportion to these bounds. When MARS extends a prefix, the bounds of the prefix's possible extensions reveal how much the prefix's own bound overpromised; MARS rejects with exactly that probability and remembers the tighter bound, so the same rejection never happens again. Such bounds, which we call *envelopes*, are closed under intersection, union, sharpening, reweighting, and constraining, which yields an algebra of language models: any globally-normalized distribution built from these operations can be sampled exactly by the same algorithm. On three benchmarks, MARS uses up to $8\times$ fewer forward passes than sequential MC at the particle count prior work reports, while sampling exactly; against rejection sampling, MARS needs up to $4.3\times$ fewer descents per sample, and sometimes samples where rejection sampling times out at 400 attempts.

1 INTRODUCTION

Autoregressive language models (LMs) are increasingly used in combination. One may want to *intersect* models, e.g., to sample completions plausible under two prompts (Chan et al., 2026); take their *union*; *sharpen* a model; *reweight* a model by a verifier (Yang & Klein, 2021; Mudgal et al., 2024); or *contrast* it against a weaker model (Li et al., 2023; Liu et al., 2021). Each operation combines the probabilities the models assign to a *complete* sequence and defines a new *globally normalized* distribution over sequences—an energy-based model whose normalizing constant is a sum over all complete sequences (Hinton, 2002; Khalifa et al., 2021; Mireshghallah et al., 2022). Renormalizing the combination token by token is easy to sample but yields a different, locally normalized distribution: it commits to each token without knowing whether the prefix can be completed compatibly (Lafferty et al., 2001; Andor et al., 2016), and Chan et al. (2026) and Park et al. (2024) find local and global sequence probabilities almost uncorrelated in their prompt intersection and grammar-constraining experiments. Sampling the global distribution is hard for the same reason: the correct probability of the next token depends on the total weight of all completions of the prefix, a sum over exponentially many sequences, while each model exposes only its next-token distribution.

An ideal sampler should be **exact** (fidelity to the global distribution) and use few forward passes per sample (**efficiency**); existing methods achieve one or the other.

Exact methods. Rejection Sampling (RS) draws from a proposal distribution and accepts each sequence with a probability that corrects for the target. To sample the plain product of two models, one may draw from the first model and accept with the sequence's probability under the second. RS is therefore exact, but can be inefficient. For example, for products whose exponents sum to more than one, RS's acceptance rate can be exponentially small even when the models are identical, because the product itself sharpens the sequence distribution.

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.

Approximate methods. *Local reweighting* methods apply the operation to the models' next-token distributions and renormalize at every step, at one forward pass per token (Liu et al., 2021; Yang & Klein, 2021; Dekoninck et al., 2024). These methods sample a different, *locally normalized* distribution (Lafferty et al., 2001; Andor et al., 2016). Sequential Monte Carlo (SMC) methods (Lew et al., 2023; Zhao et al., 2024; Loula et al., 2025; Chan et al., 2026) correct the local proposal toward the global distribution by reweighting and resampling a population of partial sequences, but they are exact only asymptotically, and their bias at any fixed budget is unknown.

Our approach. The landscape thus offers *exactness without efficiency, or efficiency without exactness*. In this paper, we present *Modular Adaptive Rejection Sampling* (MARS), which enriches rejection sampling the ability to *improve its efficiency over time* without giving up exactness. MARS builds on Constrained Adaptive Rejection Sampling (CARS) (Parys et al., 2026), which samples from an LM under a hard constraint by recording constraint-violating prefixes in a trie and subtracting their mass from future draws, so no invalid prefix is sampled twice. To lift CARS to combinations of models, MARS treats what the trie tracks for a prefix as an *upper bound* on the target weight of the prefix's completions in the global distribution. MARS chooses each next token in proportion to these bounds, and extending a prefix for the first time reveals how much its own bound overpromised: MARS rejects with exactly that probability and tightens bound. We call such a family of bounds an *envelope*. The reason MARS is more than a reformulation of CARS is that envelopes compose. Intersection, union, sharpening, reweighting, and constraining all map envelopes to envelopes, so any distribution built from them—including every generalized-mean ensemble that rewards agreement (Chan et al., 2026), which prior work samples only approximately—is sampled exactly by the same algorithm.

We make three contributions. (1) **MARS**, an exact rejection sampler for combinations of autoregressive models whose rejection rate decreases monotonically (Section 3). (2) An **algebra of language models**: compositional envelopes covering intersection, union, sharpening, reweighting, and constraining, and we prove a complementary worst-case barrier for two operations outside this algebra—tempering and contrasting (Section 4). (3) An **evaluation** on BBH word sorting, Spider text-to-SQL, and JSON Schema (Section 5). MARS needs up to $8\times$ fewer forward passes than SMC at the particle count used in prior work, while sampling exactly, and up to $4.3\times$ fewer attempts per sample than RS; where the models disagree most, RS times out after 400 attempts on up to $26/100$ instances, whereas MARS samples successfully.

2 Sampling from Combinations of Language Models

In this section, we formalize the problem of sampling from a combination of autoregressive models, define the operations we want to support (Table 1), state our desiderata, and describe why existing algorithms do not meet them. We follow the notation of Park et al. (2024) and Parys et al. (2026).

Let $\Sigma_\$$ be a set of tokens including an end-of-sequence marker \$, and let $\Sigma = \Sigma_\$ \setminus \{\$\}$. We consider sequences from the set $\Sigma^*\$^?$ (i.e., sequences of tokens that may have \$ only at the end). We write $u \preceq w$ to denote that a sequence u is a prefix of a sequence w. For a set of sequences $\mathcal{L}$ we write $\mathrm{prefix}(\mathcal{L})$ for the set of prefixes of sequences in $\mathcal{L}$—i.e., $\mathrm{prefix}(\mathcal{L}) = \{u \mid \exists w \in \mathcal{L}.\ u \preceq w\}$. An (autoregressive) language model is given by next-token conditional probability distributions $P(ua \mid u)$, where $u \in \Sigma^*$ and $a \in \Sigma_\$$, denoting the probability that u is followed by a. The probability of longer continuations is the product of the conditionals along it, $P(ua_1 \dots a_n \mid u) = \prod_{i=1}^{n} P(ua_1 \dots a_i \mid ua_1 \dots a_{i-1})$, and we write $P(w)$ for $P(w \mid \varepsilon)$. Some proofs rely on fixing a maximum output length L, counting non-\$ tokens, which can be enforced by forcing the L-th token to always be \$. This assumption is made to avoid technical issues concerning nonterminating descents, rather than as a restriction inherent to the envelope constructions.

The other objects we combine with language models are presented in the same way. A *potential* g is given by next-token conditional weights $g(ua \mid u) \in [0, 1]$, extended to sequences by the same product, $g(a_1 \dots a_n) = \prod_i g(a_1 \dots a_i \mid a_1 \dots a_{i-1})$. We do *not* require weights at a prefix to sum to one; when they do, the potential is itself a language model. A *scorer* that assigns a value in $[0, 1]$ to complete sequences only—a verifier—is a potential with $g(ua \mid u) = 1$ for $a \neq \$$ and $g(u\$ \mid u)$ equal to the score. A *constraint*, i.e., a set $\mathcal{L} \subseteq \Sigma^*\$$ of acceptable sequences whose prefixes can be checked, is the $\{0, 1\}$-valued potential $g(ua \mid u) = \mathbb{1}[ua \in \mathrm{prefix}(\mathcal{L})]$, so that $g(w) = \mathbb{1}[w \in \mathcal{L}]$. In general, any weighting of sequences is a potential that reveals all of its weight at \$; the more of the weight a potential reveals early, the more useful it will be for sampling (Section 3).

Table 1: Operations for combining language models P_k (and a potential g). Each row combines the probabilities $P_k(w)$ of a complete sequence into a weight $\varphi(w)$; the target is the globally normalized distribution φ/Z. The last three columns preview our results. The *Envelope* and *Exact* columns are the results of Sections 3 and 4, with E_k an envelope for P_k, and $c = 1/\min_k \omega_k$. The two envelopes marked † may introduce terminal slack even when their input (Section 4). For the last two rows, "poly-cost ⇒ exp. loose" means that any envelope whose expansion cost is polynomial in the maximum output length L has $E(\varepsilon)/Z = 2^{\Omega(L)}$ on some instances (Corollary A.8); correspondingly, every exact sampler with next-token access requires $2^{\Omega(L)}$ queries on some instances (Theorem A.7).

Operation	Target $\varphi(w)$	Envelope $E(u)$	Exact	Used for
Generalized-mean ensembles of Chan et al. (2026), $M_\tau(x) = \left(\sum_k \omega_k x_k^\tau\right)^{1/\tau}$				
Agree	$\min_k P_k(w)$	$\min_k E_k(u)$	✓	mean, $\tau = -\infty$
Agree (mean)	$M_\tau(P_1(w), \dots, P_K(w)), \tau \le 1$	$M_\tau(E_1(u), \dots, E_K(u))$	✓	harmonic, …
Intersect (product)	$\prod_k P_k(w)^{\omega_k}, \sum_k \omega_k \ge 1$	$\prod_k E_k(u)^{\omega_k}$	✓	products of experts
Union	$\sum_k \omega_k P_k(w)$	$\sum_k \omega_k E_k(u)$	✓	mixtures, $\tau = 1$
Cover (mean)	$M_\tau(P_1(w), \dots, P_K(w)), \tau > 1$	$c \sum_k \omega_k E_k(u)^\dagger$	✓	quadratic, maximum, …
Operations beyond generalized means that MARS supports				
Sharpen	$P(w)^\gamma, \gamma \ge 1$	$E(u)^\gamma$	✓	low temperature
Reweight (prefix)	$P(w)\, g(w), g(ua) \le g(u)$	$E(u)\, g(u)$	✓	rewards, preferences
Reweight (output)	$P(w)\, g(w)$	$E(u)^\dagger$	✓	verifiers
Constrain	$P(w)\, \mathbb{1}[w \in \mathcal{L}]$	$E(u)\, \mathbb{1}[u \in \mathrm{prefix}(\mathcal{L})]$	✓	constrained generation
Operations that no exact sampler with next-token access can support efficiently (Theorem A.7)				
Temper	$P(w)^\gamma, \gamma < 1$	poly-cost ⇒ exp. loose	$2^{\Omega(L)}$ calls	high temperature
Contrast	$P_1(w)^\alpha / P_2(w)^\beta$	poly-cost ⇒ exp. loose	$2^{\Omega(L)}$ calls	contrastive decoding

Operations. Table 1 lists the operations we want to support. Each combines one or more language models P_i, and where indicated an additional potential g, to define an unnormalized weight $\varphi(w)$ on complete sequences; the target distribution is φ/Z with $Z = \sum_w \varphi(w)$, which we assume to satisfy $0 < Z < \infty$. The first group of rows is the one-parameter family of *generalized means*by Chan et al. (2026): $M_\tau(x) = (\sum_k \omega_k x_k^\tau)^{1/\tau}$, where the weights $\omega_k > 0$ are fixed constants with $\sum_k \omega_k = 1$, with the usual continuous extensions at $\tau = 0$ and $\tau = \pm\infty$. The operations compose: reweighting an intersection by a verifier and constraining the result is again a target of the same kind. Constrained generation, the setting of CARS, is the single row *constrain* applied to one model. The recurring example of this paper is the intersection of two language models P_1 and P_2, whose target distribution, with exponents $r, s > 0$, is

$$\frac{\varphi(w)}{Z} \qquad \text{with} \qquad \varphi(w) = P_1(w)^r\, P_2(w)^s, \qquad Z = \sum_{w' \in \Sigma^* \$} \varphi(w'). \tag{1}$$

For $r + s = 1$ this is their weighted geometric mean, and for $r = s = 1$ a product of experts.

Exact sampling. We focus on designing an algorithm that samples **exactly from the target distribution** φ/Z while being **more efficient than existing exact methods**. Two properties of the problem shape every method. First, the target is *globally* normalized: Z is a sum over all sequences and is intractable. Second, the models are only available *locally*: a model exposes $P(ua \mid u)$ but not the total target weight $\sum_{w \succeq u} \varphi(w)$ of the completions of a prefix u.

Existing exact methods. Rejection sampling needs a proposal distribution q and a constant Λ with $\varphi(w) \le \Lambda q(w)$ pointwise. It draws $w \sim q$ and accepts it with probability $\varphi(w)/(\Lambda q(w))$. The accepted samples are distributed exactly according to φ/Z, and the overall acceptance probability is Z/Λ. Every generalized mean with $\tau \le 1$ and every weighted product whose exponents sum to at least one is pointwise dominated by a suitable arithmetic mixture of the models, giving a rejection-sampling proposal with $\Lambda = 1$. For the geometric mean, the resulting acceptance probability is the Bhattacharyya coefficient of the two sequence distributions, which can decay exponentially with sequence length when the Bhattacharyya coefficients of their next-token distributions remain bounded away from one. A suitable explicit Λ is likewise available for every other row of Table 1 except tempering and contrasting. For these two operations, a finite Λ still exists for every finite-length instance, so exact rejection sampling is possible in principle. However, this does not lead to an efficient exact method in the next-token query model: every exact sampler requires exponentially many model queries on some instances for tempering and contrasting (Theorem A.7).

Existing approximate methods. Local reweighting applies the operation to the next-token distributions instead of to the sequence probabilities: for the intersection of two models it samples the next token from $P_1(a \mid u)^{\omega_1} P_2(a \mid u)^{\omega_2} / \sum_{a' \in \Sigma_\$} P_1(a' \mid u)^{\omega_1} P_2(a' \mid u)^{\omega_2}$. The resulting distribution over complete sequences is the product of these per-step normalizers away from the target: the normalizer at u rewards prefixes at which the models *agree* on the next token, independently of whether the prefix is likely under either. Chan et al. (2026) show this bias is large in practice: in their prompt-intersection experiment the correlation between local and global ensemble probabilities is 0.09. Sequential Monte Carlo corrects this bias by importance weighting and resampling N partial sequences (Lew et al., 2023; Zhao et al., 2024; Chan et al., 2026), but is correct only as $N \to \infty$.

3 Modular Adaptive Rejection Sampling

Modular Adaptive Rejection Sampling (MARS) maintains a trie $\mathcal{K}$ of *expanded* prefixes. Each node u of the trie stores a mass q_u that upper-bounds the total target weight of all complete sequences extending u. A sample is drawn by descending the trie, choosing each child proportionally to its mass. Whenever the descent enters a prefix for the first time, the prefix is expanded: its bound is replaced by the sum of the masses of its children, which can only be smaller, and the descent continues with probability equal to the ratio of the new bound to the old one—otherwise the sample is rejected and a new descent starts from the root. This lets us retain the exact distributional fidelity of rejection sampling while making every rejection permanent: the tightened bound is stored, so the same disagreement between the models is never paid for twice. This section explains Algorithm 1: what the bounds q_u are, how the trie is maintained and sampled from, and why the result is exact.

Envelopes. The bounds stored in the trie come from an *envelope*, a function on prefixes that upper-bounds the target weight on complete sequences and whose children never carry more total bound than their parent.

Definition 3.1 (Envelope)**.** Let $\varphi\colon \Sigma^*\$ \to \mathbb{R}_{\geq 0}$ be a target weight. A function $E\colon \Sigma^*\$^? \to \mathbb{R}_{\geq 0}$ is an *envelope* for φ if **(i)** $E(w\$) \geq \varphi(w\$)$ for every complete sequence $w\$$, and **(ii)** $E(u) \geq \sum_{a \in \Sigma_\$} E(ua)$ for every prefix $u \in \Sigma^*$.

Every target with $0 < Z < \infty$ admits one envelope for free: the *tight* envelope $E^\star(u) = \sum_{w\$ \succeq u} \varphi(w\$)$, the exact remaining target mass below u, which satisfies **(i)** and meets **(ii)** with equality. $E^\star$ is useless in practice for the reason it is interesting in theory: evaluating it at a single prefix is the global summation that sampling from φ was supposed to avoid. An envelope is useful when it is both cheap to evaluate and sufficiently tight. Locality concerns the former; the root ratio $E(\varepsilon)/Z$ is the basic measure of the latter, since Theorem A.3 guarantees that an iteration succeeds with probability at least $Z/E(\varepsilon)$. We call an envelope *local* if $E(u)$ can be evaluated from the information revealed by the component models or potentials at u, without querying any strict descendant of u. The envelope constructions used throughout this paper are local. Their tightness, however, is a separate question, and will be central to both the positive and negative results of Section 4.

Conditions **(i)** and **(ii)** make an envelope an upper bound: iterating **(ii)** down the trie and using **(i)** at complete sequences shows that $E(u)$ dominates the total target weight of any finite set of complete sequences extending u, and in particular that $Z \leq E(\varepsilon)$. One should think of $E(u)$ as a *budget* for the subtree rooted at u: the budget may be split among the children without going over, while at a complete sequence MARS uses the exact target weight and any excess becomes terminal slack.

The prefix weight $\vec{g}(u) = g(u)$ of a potential is an envelope for g whenever g is sub-stochastic, i.e., $\sum_{a \in \Sigma_\$} g(ua \mid u) \leq 1$: condition **(ii)** reads $g(u) \geq g(u) \sum_{a \in \Sigma_\$} g(ua \mid u)$. For a language model this holds with equality, and $\vec{P}_k(u) = P_k(u)$ is exactly the total probability of the completions of u. For the intersection target of Equation (1), with language models P_1 and P_2, we take

$$E(u) \;=\; P_1(u)^r \, P_2(u)^s. \tag{2}$$

Condition **(i)** holds by definition. Condition **(ii)** holds whenever $r + s \geq 1$: for $r + s = 1$ it is Hölder's inequality, $\sum_{a \in \Sigma_\$} P_1(ua \mid u)^r P_2(ua \mid u)^s \leq \big(\sum_{a \in \Sigma_\$} P_1(ua \mid u)\big)^r \big(\sum_{a \in \Sigma_\$} P_2(ua \mid u)\big)^s \leq 1$, and the case $r + s > 1$ reduces to it (Section 4). The hard-constraint envelope $E(u) = P(u) \cdot \mathbb{1}[u \in \mathrm{prefix}(\mathcal{L})]$ is obtained by multiplying the model envelope by the prefix mask of the constraint.

Algorithm 1 MARS algorithm. The bounds it samples from are Equation (3); the envelope E is any function satisfying Definition 3.1. EXPANDED returns the prefixes the descent expanded, which is recorded by the descent itself and is well defined whether or not it survived.

1: **Input:** Envelope E for the target weight φ
2: **Output:** A stream of samples drawn from the target distribution φ/Z
3: $\mathcal{K} \leftarrow \emptyset$ ▷ *initialize expanded prefixes*
4: **while** true **do**
5: $w \sim Q^{\mathcal{K}}$ ▷ *sample left to right from the trie's bounds; $\perp$ if rejected*
6: **if** $w \neq \perp$ **then yield** w ▷ *a surviving descent is a sample*
7: $\mathcal{K} \leftarrow \mathcal{K} \cup \text{EXPANDED}(\textit{descent})$ ▷ *expand what w visited; new masses by Equation (3)*

Example 3.2 (Arithmetic Expressions). Our running example intersects two tiny models over the alphabet $\Sigma = \{\texttt{0}, \texttt{1}, \texttt{+}\}$, in which the next-token distribution depends only on the last token. The first model, P_1, is a language model that has learned arithmetic only "loosely": at the start it emits `0`, `1`, `+` with probabilities 0.4, 0.3, 0.3; after a digit it emits `0`, `1`, `+`, `$` with 0.1, 0.1, 0.5, 0.3; and after a `+` it emits `0`, `1`, `+` with 0.5, 0.4, 0.1. The second model, P_2, only produces syntactically valid sums of digits and prefers short ones: at the start or a `+` it gives probability 0.5 to each digit, and after a digit it gives probability 0.6 to `$` and 0.4 to `+`. So `1$` has probability 0.3 under P_2, `1+0$` has probability 0.06, and `0++` has probability 0. Their intersection combines P_1's preferences among well-formed sums with P_2's preference for short ones and gives everything else weight 0. Our target is the geometric mean $\varphi(w) = P_1(w)^{1/2} P_2(w)^{1/2}$, i.e., Equation (1) with $r = s = 1/2$.

In Example 3.2, the envelope of Equation (2) gives $E(\varepsilon) = 1$; $E(\texttt{0}) = \sqrt{0.4 \cdot 0.5} \approx 0.447$; $E(\texttt{1}) = \sqrt{0.3 \cdot 0.5} \approx 0.387$; and $E(\texttt{+}) = \sqrt{0.3 \cdot 0} = 0$. The children of ε sum to $0.834 < 1 = E(\varepsilon)$: the envelope is not tight at the root; the gap, 0.166, is what MARS has to pay for in rejections. The gap is not due to the excluded token `+` alone—the models also disagree on how likely `0` and `1` are.

The MARS algorithm. MARS maintains a prefix-closed set $\mathcal{K} \subseteq \Sigma^{\leq L}$ of *expanded* prefixes. The *boundary* $\mathcal{B}$ of $\mathcal{K}$ consists of the sequences that are not expanded but whose parent is, together with the leaves $u\$$ of expanded nodes u. Every node u in $\mathcal{K} \cup \mathcal{B}$ carries a mass q_u defined by

$$q_u = \sum_{a \in \Sigma_\$} q_{ua} \quad \text{if } u \in \mathcal{K}, \qquad q_u = E(u) \quad \text{if } u \in \mathcal{B} \cap \Sigma^*, \qquad q_u = \varphi(u) \quad \text{if } u \in \mathcal{B} \cap \Sigma^*\$. \tag{3}$$

The mass of an expanded node is inherited from its children, the mass of an unexpanded proper prefix is the envelope's promise for it, and the mass of a complete sequence is its exact target weight. Initially, $\mathcal{K} = \emptyset$, the boundary is $\{\varepsilon\}$, and $q_\varepsilon = E(\varepsilon)$. The values q_u are stored in the trie and change only when a prefix is expanded during a descent (Line 5): expanding a boundary prefix u replaces $q_u = E(u)$ with $\varphi(u\$) + \sum_{a \in \Sigma} E(ua)$, and by conditions **(i)** and **(ii)** of Definition 3.1 this is a *decrease*, which is propagated upward. The cost of an expansion is one evaluation of each component at u—one forward pass for each language-model component—plus a trie update linear in $|u|$; in memory it costs two vectors over the vocabulary, held on the host.

Two facts about the masses drive everything that follows. Every mass satisfies $q_u \leq E(u)$, and q_u dominates the total target weight of all complete sequences below u; i.e., $Z \leq q_\varepsilon \leq E(\varepsilon)$ at all times. Sampling from $Q^{\mathcal{K}}$ (Line 5) is a descent from the root that chooses, at each expanded node u, the child ua with probability q_{ua}/q_u. Reaching a leaf $w\$$ yields it. Reaching a boundary node u that is not a leaf triggers its expansion and a coin flip: the descent survives with probability $q_u^{\text{new}}/q_u^{\text{old}}$, the factor by which the bound at u was tightened; otherwise the descent returns $\perp$ and Line 7 records the prefixes it expanded, so that the next descent starts from the tightened trie. By the standing length bound, every descent terminates after finitely many expansions. The probability of a successful descent along a fixed path telescopes. Indeed, when the descent enters a node that is expanded for the first time, the transition into the node uses its old mass q_u^{old}; surviving the expansion contributes the factor $q_u^{\text{new}}/q_u^{\text{old}}$; and the transition to the next node uses q_u^{new} in the denominator. Thus the old and new masses both cancel. At a node that was expanded before the descent, there is no survival test and its mass cancels directly between the transition into and out of the node. Consequently all intermediate masses cancel, leaving only the leaf mass $q_{w\$} = \varphi(w\$)$ and the root mass q_ε at the start of the descent. Hence a descent from a state $\mathcal{K}$ yields $w\$$ with probability $\varphi(w\$)/q_\varepsilon$, as stated in Lemma A.2; this identity is the key to the exactness of MARS.

In Example 3.2, the first descent expands the root: q_ε drops from 1 to 0.834 and the descent survives with probability 0.834. Suppose it then chooses `0` (probability $0.447/0.834 \approx 0.54$) and expands it: the children of `0` have bounds $E(\texttt{0\$}) = \varphi(\texttt{0\$}) = \sqrt{0.4 \cdot 0.3 \cdot 0.5 \cdot 0.6} \approx 0.190$, $E(\texttt{0+}) = \sqrt{0.4 \cdot 0.5 \cdot 0.5 \cdot 0.4} = 0.200$, and $E(\texttt{00}) = E(\texttt{01}) = 0$, so q_0 drops from 0.447 to 0.390, the decrease 0.057 is propagated to the root ($q_\varepsilon = 0.777$), and the descent survives with probability $0.390/0.447 \approx 0.87$. Finally it chooses \$ with probability $0.190/0.390 \approx 0.49$ and yields `0$`. The product of the factors is $0.834 \cdot 0.54 \cdot 0.87 \cdot 0.49 \approx 0.190 = \varphi(\texttt{0\$})/q_\varepsilon$ with $q_\varepsilon = 1$ the root mass *at the start* of the descent. Figure 1 shows the trie afterwards. The second descent starts at $q_\varepsilon = 0.777$: it never pays for the root or `0` again, and yields `0$` with probability $0.190/0.777 \approx 0.24$, instead of 0.19.

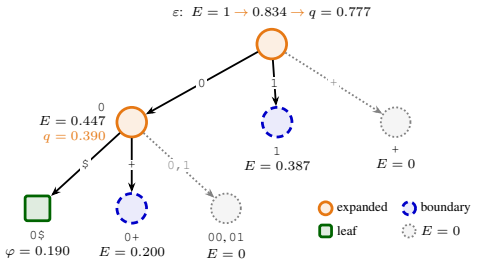


Figure 1: The MARS trie for Example 3.2 after a first descent that yielded `0$`. Colour marks which case of Equation (3) a prefix is in: expanded: its bound has been replaced by the total of its children; on the boundary: holding a bound that has not yet been paid for, i.e., where the remaining slack lives; a leaf: carrying its exact target mass by Equation (3); or ruled out, with envelope 0 and never visited. The root mass has fallen from 1 to 0.777; the difference is the rejection mass this descent paid for, and it will never be paid again.

The survival probability at an expansion of u is $\big(\varphi(u\$) + \sum_{a \in \Sigma} E(ua)\big)/E(u)$. For the envelope of Equation (2) with $r = s = 1/2$, which is exact at complete sequences, this is $\sum_{a \in \Sigma_\$} \sqrt{P_1(ua \mid u) P_2(ua \mid u)}$, the *Bhattacharyya coefficient* of the next-token distributions of the two models at u, and the rejection probability $1 - \mathrm{BC}(P_1(\cdot \mid u), P_2(\cdot \mid u))$ is their squared Hellinger distance. MARS thus rejects exactly where, and exactly as much as, the two models disagree on the *next token*, and each such disagreement is paid for once. Every node where the two distributions differ contributes to the rejection mass.

In Section A we prove, and verify in Lean (Section B), that when E is an envelope for φ with $Z > 0$, MARS samples exactly from φ/Z. We also prove that the root mass q_ε never increases, so the probability Z/q_ε that an iteration yields a sample never decreases; since that probability is at least $Z/E(\varepsilon) > 0$, MARS produces a sample after finitely many iterations almost surely. Finally, the mass a descent removes, weighted by the root mass at the time, never increases in expectation (Theorem A.9), and that on hard constraints MARS coincides with CARS (Section C).

4 An Algebra of Envelopes

Algorithm 1 takes an envelope as input and never looks at how it was built. In this section we show whether one can obtain an envelope that is both cheap to evaluate and sufficiently tight to make MARS efficient, yielding simple local constructions for every row of Table 1 except the last two.

Throughout, $E, E', E_1, \ldots, E_K$ are envelopes for targets $\varphi, \varphi', \varphi_1, \ldots, \varphi_K$, and a sub-stochastic potential g provides the *base* envelope $\vec{g}(u) = g(u)$ (Section 3). Because Definition 3.1 does not restrict the range of E to $[0, 1]$, multiplying an envelope by a factor $c \geq 1$ preserves an envelope for the same target; only the ratio $Z/E(\varepsilon)$ is affected.

Envelopes are closed under weighted products whose exponents sum to at least one, under minima and mixtures, and under multiplication by prefix-monotone scorers such as the masks of hard constraints; they are not closed under powers below one or quotients (Proofs in Proposition A.4). These closure properties directly give several rows of Table 1: *intersect* and *sharpen* are weighted products, *agree* and *union* are minima and mixtures, *prefix reweighting* and *constraining* use prefix-monotone scorers, and *output reweighting* uses target domination. The generalized-mean constructions are treated below.

Loose terminal bounds. An envelope need not be exact at a complete sequence. When a proper prefix w is expanded, its \$ child is assigned the exact mass $\varphi(w\$)$ rather than the envelope bound $E(w\$)$. Hence any terminal slack $E(w\$) - \varphi(w\$)$ contributes directly to the decrease of q_w, propagated to its ancestors and remembered exactly like ordinary envelope slack. This is relevant for output-only verifiers and for the coverage regime of generalized means (e.g., max of two models).

Generalized means. The generalized mean M_τ interpolates between the operations above: it is the minimum at $\tau \to -\infty$, the weighted geometric mean at $\tau = 0$, the mixture at $\tau = 1$, and the maximum as $\tau \to +\infty$. For $\tau < 1$ it rewards *agreement* and for $\tau > 1$ *coverage*, the same split Chan et al. (2026) derive from the α-divergences the means minimize. Since MARS samples exactly from any target for which it is given an envelope (Theorem A.3), sampling either regime exactly reduces to building one. For $\tau \leq 1$ the generalized mean is concave and positively homogeneous, hence superadditive (Lemma A.5); applied to envelopes, it therefore preserves the branching condition **(ii)** from Section 3, so $M_\tau(E_1, \ldots, E_K)$ is an envelope for $M_\tau(\varphi_1, \ldots, \varphi_K)$ and MARS samples the agreement regime exactly (Corollary A.6). For $\tau > 1$ the mean is subadditive and can violate **(ii)**, so MARS instead uses a dominating mixture. At complete sequences M_τ is at most $c \sum_k \omega_k E_k$, where $c = 1/\min_k \omega_k$; this scaled mixture is an envelope for $c \sum_k \omega_k \varphi_k$ (Proposition A.4(2)), hence, by target domination (Proposition A.4(4)), also for $M_\tau(\varphi_1, \ldots, \varphi_K)$, and MARS samples the coverage regime exactly as well. The mixture is tight under branching at every incomplete prefix, so the slack it adds appears only at complete sequences, where each sequence keeps at least a $1/c = \min_k \omega_k$ fraction of its bound when the input envelopes are exact there. Plain rejection sampling from the mixture therefore also works, with acceptance probability at least $\min_k \omega_k$, but it pays that terminal slack again on every descent; MARS removes it from the parent's mass the first time the prefix is expanded and never pays it again, so the trie earns its keep in the coverage regime too (Section D).

Composing operations. The constructions above compose. For example, a model sharpened by an exponent $\gamma \geq 1$, constrained to a set of acceptable outputs $\mathcal{L}$, and reweighted by a verifier g on complete outputs has target $P(w)^\gamma \, \mathbb{1}[w \in \mathcal{L}] \, g(w)$. The function $(\vec{P}(u))^\gamma \, \mathbb{1}[u \in \mathrm{prefix}(\mathcal{L})]$ is an envelope before applying the verifier, by weighted-product and prefix-monotone-scorer closure; since $g(w) \leq 1$, target domination (Proposition A.4(4)) shows that the same function is also an envelope for the verifier-reweighted target. When an incomplete sequence w is expanded, its \$ child receives the exact target mass, including the verifier score, so any resulting terminal slack is learned in the usual way. The *prompt intersection* experiment of Chan et al. (2026)—sample completions that are probable under two different prompts—is the *intersect* row with $K = 2$, which MARS samples exactly rather than through the locally normalized product whose bias motivated their paper. For the agreement operators, the rejection mass at a prefix has a direct divergence interpretation between the models' next-token distributions (the Hellinger distance for the geometric mean and the total variation distance for the minimum), and the trie ensures that this mass is paid only once.

5 Evaluation

We evaluate MARS on efficiency and sample quality against the other ways of sampling an ensemble. We answer: (i) what learning from rejections buys over rejection sampling with the same prefix cache, and (ii) what exactness costs against an approximate sampler.

Benchmarks and Baselines. We use BBH word sorting (Suzgun et al., 2023), from Chan et al. (2026), and text-to-SQL over Spider (Yu et al., 2018) and JSON Schema (Geng et al., 2025), standard in constrained decoding (Parys et al., 2026). Every target ensembles (minimum, product, mixture, and maximum) one model under two prompts, the *within-model* setting of Chan et al. (2026) (Sections D and I). *Rejection sampling* (RS) is MARS without learning and is *exact*. *SMC* is the state-of-the-art *approximate* method for sampling model combinations by Chan et al. (2026), run through their own implementation over our envelope at the settings they report ($N = 10$ particles, Section I). Caching is matched across all three: MARS and RS keep every expanded prefix for an instance, and our adaptor gives SMC an unbounded prefix cache of its own, so no lookup ever evicts.

Metrics. We measure LM *forward passes per sample* and *descents per sample*, the number of tries from the root that one accepted sample takes (this last metric only makes sense for RS and MARS, since SMC never rejects a sample). We count one forward pass whenever one component model under one prompt is evaluated at a prefix to obtain its full next-token distribution. For SMC, the cost of one returned sample is the total number of such uncached evaluations made by all N particles over the entire SMC sweep. For MARS and RS, the same count is accumulated over all descents needed to obtain the accepted sample. An exact sampler has no deterministic bound on the number of descents required before acceptance, so MARS and RS both give up on a sample after 400 rejected descents; we call this event a *timeout* and report the number of instances on which one occurs. We include the

Table 2: BBH word sorting (five seeds) with Llama-3.1-8B-Instruct, 100 instances and five samples per instance, 95% intervals over seeds. P_1 and P_2 are the same model under two prompts. Forward passes are per accepted sample and count all uncached component-model evaluations; for SMC, they are summed over all particles in the sweep. Ratios to MARS are in parentheses. For RS and MARS, descents per sample are the tries from the root one accepted sample takes.

	Accuracy			Forward passes per sample			Descents per sample	
Target $\varphi(w)$	MARS	RS	SMC	MARS	RS	SMC	MARS	RS
BBH word sorting								
$P_1(w)$ (single prompt)	$.39_{\pm.02}$	$.40_{\pm.02}$	$.42_{\pm.01}$	$21.3_{\pm0.6}$	$21.4_{\pm0.3}$ (1.00×)	$113_{\pm2}$ (5.3×)	$1.00_{\pm0.00}$	$1.00_{\pm0.00}$ (1.00×)
$\min\big(P_1(w), P_2(w)\big)$	$.49_{\pm.01}$	$.48_{\pm.02}$	$.50_{\pm.01}$	$46.1_{\pm1.3}$	$47.1_{\pm2.2}$ (1.02×)	$154_{\pm3}$ (3.4×)	$1.58_{\pm0.05}$	$4.57_{\pm0.54}$ (2.9×)
$\sqrt{P_1(w)\,P_2(w)}$	$.48_{\pm.01}$	$.46_{\pm.01}$	$.47_{\pm.01}$	$44.4_{\pm1.7}$	$45.5_{\pm1.0}$ (1.02×)	$188_{\pm3}$ (4.2×)	$1.34_{\pm0.03}$	$1.90_{\pm0.11}$ (1.42×)
$\frac{1}{2}P_1(w) + \frac{1}{2}P_2(w)$	$.40_{\pm.01}$	$.41_{\pm.02}$	$.42_{\pm.01}$	$44.1_{\pm0.8}$	$44.9_{\pm0.7}$ (1.02×)	$250_{\pm5}$ (5.7×)	$1.00_{\pm0.00}$	$1.00_{\pm0.00}$ (1.00×)
$\max\big(P_1(w), P_2(w)\big)$	$.36_{\pm.01}$	$.36_{\pm.01}$	$.41_{\pm.01}$	$52.5_{\pm2.0}$	$52.5_{\pm0.7}$ (1.00×)	$251_{\pm2}$ (4.8×)	$1.14_{\pm0.01}$	$1.41_{\pm0.04}$ (1.23×)

forwards passes of timed-out samples in our cost. *Accuracy* is the fraction of correct draws according to the task being solved, one draw per sample.

Table 2 reports both comparisons on word sorting; Table 3 in Section E reports Spider and Table 4 in Section F JSON Schema, which sit at the easy end and in the middle of the three and are drawn on below. The rows that isolate the learning claim are the minimum, the square root, and the maximum (this last one only learns from rejected leaves). On the single prompt and the weighted average there is nothing to learn and MARS and RS are the same algorithm.

Accuracy. MARS and RS have similar accuracy; they sample the same distribution. Because of its approximate sampling, SMC has higher accuracy over the maximum: .41 against .36 on word sorting, and .52 against .56 on JSON Schema. On word sorting and JSON, we see the same pattern as in Chan et al. (2026): minimum and product improve on the single-prompt, the mixture does not. Spider's accuracy seems unaffected by the operator. Similar results hold across different models (Table 5).

MARS uses fewer forward passes than SMC: 3.4–5.7× on word sorting, 4.4–6.6× on Spider and 6.9–8.0× on JSON Schema, with the consensus operators at the low end on word sorting. So does RS, whose shared cache keeps it within 7% of MARS's forward passes. With the cache matched RS and MARS incur similar numbers of forward passes, but for opposite reasons. RS's passes are few because RS cannot leave the region that keeps causing rejections: with nothing learned, each descent returns to the same cached prefixes and is rejected there again, which costs almost nothing in terms of forward passes, but also produces nothing. MARS's tightened bounds move it out of that region, onto prefixes it has not expanded yet, which is where its passes go and where its samples come from.

MARS needs fewer descents per sample than RS, and it samples where RS times out. While the difference in numbers of forward passes is small, MARS needs at most 1.58 *descents per sample*, whereas RS can require up to 4.57 on word sorting; SMC needs $N = 10$ by construction. That is, on word sorting MARS needs 2.9× fewer descents per sample than RS on the minimum, the strictest consensus operator and the one that rejects most, and 1.4× fewer on the product; this is what Theorem A.3 predicts, since the trie can only tighten a bound that a rejection exposes. On the rows that never reject RS and MARS need the same number of samples. The maximum is not one of them, though its slack is of a different kind: the max envelope is a dominating mixture that is exact under branching, so the additional slack introduced by domination is terminal slack. MARS removes that slack when the corresponding prefix is first expanded and remembers the resulting decrease, whereas RS pays the same slack again on later descents. For maximum, MARS needs 1.2× fewer descents per sample than RS on word sorting, and its root mass falls to 0.80 (Section D). On JSON Schema the saving is 1.0–2.4× and on Spider 1.0–1.1×, largest on the minimum in both, and on Qwen2.5-7B and phi-4 it reaches 4.3× (Section G). The small ratios on Spider and JSON are expected: two prompts asking for SQL against one schema, or for a document against one JSON Schema, agree on nearly every token, so the envelope is close to tight before the first descent and the trie has little to remove. Where the prompts disagree most, waste decides whether a sample arrives at all: on Qwen2.5-7B's minimum ensemble **RS times out at 400 attempts on 26 of 100 instances, and on its product on 9, whereas MARS samples successfully on every instance** (Table 5 in Section G); on the cross-model targets of Section H, RS times out on twice as many instances as MARS.

SMC's cost depends on its particle count. To see the trade-off between particle counts, accuracy, and cost, we run SMC at $N \in \{1, 2, 5, 10\}$ on the minimum ensemble for word sorting. Figure 2 shows the trade-off, with forward passes per sample against empirical KL divergence from the target distribution, the quantity Parys et al. (2026) use to compare exact and approximate samplers. SMC is far from the target at one particle, reaches the exact samplers at about five particles, at $2.8\times$ MARS's cost, and is indistinguishable from there above; ten particles cost $4.1\times$ on this instance set and $3.4\times$ on the larger one of Table 2.

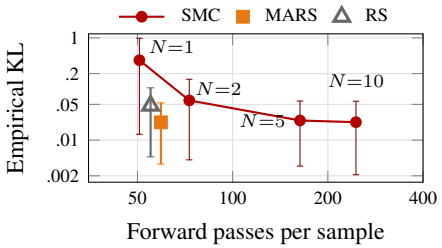


Figure 2: BBH minimum ensemble.

Diminishing returns. Theorem A.9 says that the rejection rate a descent removes, weighted by the root mass at the time, never increases in expectation—i.e., the trie learns most from its first rejections. Figure 3 shows the first descent removes .45 of the envelope's mass on the minimum and the first three descents remove 91% of what the first thirty samples together remove. Because the trie is shared across samples, waste falls as it fills in: on the minimum ensemble MARS's descents per sample fall from 3.8 on the first sample to about 1.5 thereafter, while RS needs more than 4 descents per sample, since nothing it learns carries over.

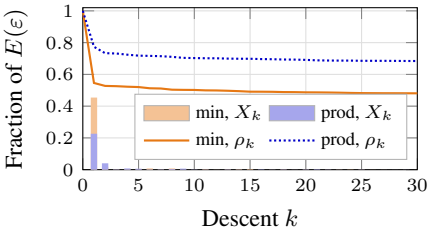


Figure 3: Word sorting. Root mass ρ_k and the mass each try removes, the X_k of Theorem A.9, as fractions of $E(\varepsilon)$.

A last experiment: two different models. Our hardest setting ensembles two *different* models that share a tokenizer, Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct, on 30 word-sorting instances under the minimum and the product, with the same budget of 400 rejected descents (Section H). Learning now decides whether a sample arrives at all: RS times out on 21 and 17 of the 30 instances against MARS's 10 and 9, and where both finish RS needs $13\times$ and $10\times$ the descents per sample for about the same forward passes. That MARS still times out on a third of them is the honest limit of the method: the trie helps, but it does not make exact sampling reliable on a target this hard. Against SMC the comparison inverts—ten particles cost a fixed 208 and 235 passes per draw against the 918 and 941 an exact sample costs MARS—and the only fidelity evidence we could get is indirect (Section H), so we report the cost without a fidelity claim. As Parys et al. (2026) observe for CARS, when rejected prefixes share no structure an approximate method may be the practical choice.

6 Additional Related Work

We discuss work not already covered in Sections 1 and 2. Our algebra of envelopes (Section 4) plays for exact global sampling the role that model arithmetic (Dekoninck et al., 2024) plays for local reweighting, which composes next-token distributions and so samples a locally normalized approximation of the composed distribution (Section 2) rather than the distribution itself. Our work is a direct generalization of CARS (Parys et al., 2026), which in turn extends adaptive rejection sampling (Gilks & Wild, 1992; Mansinghka et al., 2009) from rejected sequences to rejected prefixes. All three are exact; what changes is what is remembered between samples. CARS remembers a set of invalid prefixes and is limited to hard constraints; MARS remembers an upper bound per visited prefix and handles any target with an envelope, reducing to CARS when the constraint is hard (Section C).

Twisted SMC (Zhao et al., 2024) learns per-prefix twist functions that play a role similar to our envelopes, but they are approximate and are used for reweighting rather than for exact pruning. Chan et al. (2026) apply byte-level SMC to what they call f-ensembles, targets of the form $\varphi(w) = f(P_1(w), \ldots, P_K(w))$, and introduce the generalized-mean family we sample from exactly in Section 4. In our experiments, all models in a target share a token space, so no byte-level conversion is involved; different tokenizers would need the trie keyed on bytes, the machinery of Chan et al. (2026), which their own results show costs accuracy.

Reproducibility Statement

The statements of Sections 3 and 4 are formalized and machine-checked in Lean 4 with Mathlib, with the exception of the query lower bound (Theorem A.7), which quantifies over all algorithms with next-token access and is proved on paper, and of the step identifying a descent with a draw from the phantom model in the proof of Theorem A.9, which is argued in prose as it is for CARS; Section B states the scope precisely and maps each result to its Lean statement, and the formalization will be released with the code. Every checkpoint is identified by its Hugging Face revision in Section I, together with the library versions and the commit of our sampler; the run scripts, per-instance outputs and the tables' generating code will be released with it.

AI Use Statement

In this work, we used generative AI tools (large language models) for the following tasks with required disclosure: helping develop the theoretical framework, including defining the envelopes for the operators of the algebra of language models (Section 4); formulating mathematical claims and providing ingredients of their proofs; assisting in the writing of proofs, both the proofs of the appendix and the Lean 4 formalization of Section B, which LLMs wrote; refining research hypotheses; designing and giving feedback on the experiments; implementing the sampler and the evaluation code and running the experiments; and interpreting the experimental results. We have not used generative AI tools to generate synthetic data sets, and translation, dataset cleaning and reformatting, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create and edit software code, produce the figures and tables from the experimental data, draft and edit parts of the paper, suggest its structure, and search for, verify and format references. We have reviewed all AI-assisted work. Every formal proof is checked by the Lean kernel, so the correctness of the formalized results does not depend on the model that produced them; the paper proofs, including that of the one result proved only on paper (Theorem A.7), were checked by the authors; the interface between our envelopes and the SMC baseline was checked against independently computed quantities on a model whose target distribution is known exactly; and every reference was checked against its publisher or arXiv record. The authors directed the work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2442–2452, 2016.

Robin Shing Moon Chan, Tianyu Liu, Samuel Kiegeland, Clemente Pasti, Jacob Hoover Vigly, Timothy J. O'Donnell, Ryan Cotterell, and Tim Vieira. Ensembling language models with sequential Monte Carlo, 2026. URL https://arxiv.org/abs/2603.05432. arXiv:2603.05432.

Jasper Dekoninck, Marc Fischer, Luca Beurer-Kellner, and Martin Vechev. Controlled text generation via language model arithmetic. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *The Twelfth International Conference on Learning Representations*, Vienna, Austria, May 2024. URL https://openreview.net/forum?id=SLw9fp4yI6.

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. JSONSchemaBench: A rigorous benchmark of structured outputs for language models, 2025. URL https://arxiv.org/abs/2501.10868. arXiv:2501.10868.

W. R. Gilks and P. Wild. Adaptive rejection sampling for Gibbs sampling. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(2):337–348, 1992. doi: 10.2307/2347565.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783. arXiv:2407.21783.

G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1934.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002. doi: 10.1162/089976602760128018.

Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. A distributional approach to controlled text generation. In *International Conference on Learning Representations*, 2021.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 282–289, 2001.

Alexander K. Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash K. Mansinghka. Sequential Monte Carlo steering of large language models using probabilistic programs. In *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*, 2023. URL https://openreview.net/forum?id=Ul2K0qXxXy.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12286–12312, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.687.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. DExperts: Decoding-time controlled text generation with experts and anti-experts. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6691–6706, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.522.

João Loula, Benjamin LeBrun, Li Du, Ben Lipkin, Clemente Pasti, Gabriel Grand, Tianyu Liu, Yahya Emara, Marjorie Freedman, Jason Eisner, Ryan Cotterell, Vikash Mansinghka, Alexander Lew, Tim Vieira, and Timothy O'Donnell. Syntactic and semantic control of large language models via sequential Monte Carlo. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xoXn62FzD0.

Vikash Mansinghka, Daniel Roy, Eric Jonas, and Joshua Tenenbaum. Exact and approximate sampling by systematic stochastic search. In David van Dyk and Max Welling (eds.), *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 400–407, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR. URL https://proceedings.mlr.press/v5/mansinghka09a.html.

Fatemehsadat Mireshghallah, Kartik Goyal, and Taylor Berg-Kirkpatrick. Mix and match: Learning-free controllable text generation using energy language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 401–415, 2022.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, Yaguang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, Jilin Chen, Alex Beutel, and Ahmad Beirami. Controlled decoding from language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 36486–36503. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/mudgal24a.html.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 24547–24568. Curran Associates, Inc., 2024. doi: 10.52202/079017-0774.

Paweł Parys, Sairam Vaidya, Taylor Berg-Kirkpatrick, and Loris D'Antoni. Constrained adaptive rejection sampling. In *International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=gLRqqQOM9K.

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, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115. arXiv:2412.15115.

Kevin Yang and Dan Klein. FUDGE: Controlled text generation with future discriminators. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3511–3535, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.276.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3911–3921, Brussels, Belgium, Oct–Nov 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1425.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Baker Grosse. Probabilistic inference in language models via twisted sequential Monte Carlo. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 60704–60748. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhao24c.html.

Appendix

A PROOFS

The proofs in this appendix follow the structure of our Lean formalization (Section B): each result corresponds to a group of Lean statements, in the same order, and most arguments are the ones the formalization checks, written out in prose. The mass bounds of Lemma A.1, the telescoping identity of Lemma A.2 and Theorem A.3 follow `LeanTheorems/Mars.lean`; the closure properties of Proposition A.4 follow `MarsAlgebra.lean`; the power-mean results of Lemma A.5 and Corollary A.6 follow `MarsMeans.lean` and `MarsClauses.lean`; and the diminishing-returns theorem follows `MarsDiminishing.lean`. Two proofs depart from this. Theorem A.7 has no Lean counterpart and is proved on paper only. Lemma A.5 is proved here from concavity and homogeneity, whereas the Lean proof obtains superadditivity directly from a two-point inequality. Section B gives the exact correspondence and explains both exceptions.

Notation. Fix an envelope E for φ. For a prefix-closed set $\mathcal{K} \subseteq \Sigma^{\le L}$ of expanded prefixes, its boundary $\mathcal{B}(\mathcal{K})$ consists of the prefixes $v \notin \mathcal{K}$ whose parent is in $\mathcal{K}$ (or $v = \varepsilon$ if $\mathcal{K} = \emptyset$), together with the leaves $u\$$ for $u \in \mathcal{K}$. The masses $q_u = q_u(\mathcal{K})$ are defined by Equation (3): $q_u = E(u)$ for incomplete sequences $u \in \mathcal{B}(\mathcal{K})$, $q_{w\$} = \varphi(w\$)$ for complete sequences $w\$ \in \mathcal{B}(\mathcal{K})$, and $q_u = \sum_{a \in \Sigma_\$} q_{ua}$ for $u \in \mathcal{K}$. All sums below are over $\Sigma_\$$ unless stated otherwise.

Lemma A.1. *For every state $\mathcal{K}$ and every prefix $u \in \mathcal{K} \cup \mathcal{B}(\mathcal{K})$:*

(a) $q_u \le E(u)$*;*

(b) the total target weight of all complete sequences extending u is at most q_u, that is, $\sum_{w\$ \succeq u} \varphi(w\$) \le q_u$*;*

(c) if $\mathcal{K} \subseteq \mathcal{K}'$ then $q_\varepsilon(\mathcal{K}') \le q_\varepsilon(\mathcal{K})$.

In particular $Z \le q_\varepsilon(\mathcal{K}) \le E(\varepsilon)$ for every $\mathcal{K}$.

Proof. (a) By induction on the trie below u. If $u \in \mathcal{B}(\mathcal{K})$ is incomplete, then $q_u = E(u)$. If $u = w\$ \in \mathcal{B}(\mathcal{K})$ is complete, then $q_u = \varphi(w\$) \le E(w\$)$ by condition **(i)**. If $u \in \mathcal{K}$, then $q_u = \sum_a q_{ua} \le \sum_a E(ua) \le E(u)$ by the induction hypothesis and condition **(ii)** of Definition 3.1.

(b) We prove by induction on the remaining depth both that

$$\sum_{w\$ \succeq u} \varphi(w\$) \le E(u)$$

for every incomplete sequence u, and that

$$\sum_{w\$ \succeq u} \varphi(w\$) \le q_u$$

for every $u \in \mathcal{K} \cup \mathcal{B}(\mathcal{K})$. If $u = w\$$ is complete, the second claim holds with equality, $q_u = \varphi(w\$)$. Now let u be incomplete. Splitting the complete continuations into $u\$$ itself and those extending ua, $a \in \Sigma$, and using the induction hypothesis gives

$$\sum_{w\$ \succeq u} \varphi(w\$) \le \varphi(u\$) + \sum_{a \in \Sigma} E(ua) \le E(u\$) + \sum_{a \in \Sigma} E(ua) \le E(u),$$

by conditions **(i)** and **(ii)**. If $u \in \mathcal{B}(\mathcal{K})$ is incomplete, then $q_u = E(u)$, so the second claim follows. If $u \in \mathcal{K}$, then, using the induction hypothesis for the non-EOS children and $q_{u\$} = \varphi(u\$)$,

$$\sum_{w\$ \succeq u} \varphi(w\$) \le_{u\$} + \sum_{a \in \Sigma} q_{ua} = q_u.$$

(c) It suffices to treat $\mathcal{K}' = \mathcal{K} \cup \{v\}$ with $v \in \mathcal{B}(\mathcal{K})$ an incomplete sequence; the general case follows by adding the elements of $\mathcal{K}' \setminus \mathcal{K}$ one at a time in order of length. Expanding v replaces $q_v = E(v)$

by

$$\varphi(v\$) + \sum_{a\in\Sigma} E(va) \le E(v\$) + \sum_{a\in\Sigma} E(va) \le E(v)$$

by conditions **(i)** and **(ii)**, and leaves the masses of all other boundary nodes unchanged, so every ancestor's mass, being a sum, decreases by the same amount.

The final claims follow from (a)–(b) at $u = \varepsilon$. □

Lemma A.2 (Telescoping)**.** *A descent started at the root in state $\mathcal{K}$ yields the complete sequence $w\$$ with probability $\varphi(w\$)/q_\varepsilon$, where q_ε is the root mass in state $\mathcal{K}$.*

Proof. Let $w\$ = a_1 \dots a_n\$$ and let $u_i = a_1 \dots a_i$, so that $u_0 = \varepsilon$ and $u_n = w$; set also $u_{n+1} = w\$$. Consider the descent along this path. For each node u_i, let q_i^{old} be its mass when the descent reaches it, before any expansion performed during this descent, and let q_i^{new} be its mass after such an expansion, if one occurs; set $q_i^{\text{new}} = q_i^{\text{old}}$ otherwise. If some $q_i^{\text{old}} = 0$, then the descent reaches u_i with probability 0. If some $q_i^{\text{new}} = 0$ while $q_i^{\text{old}} > 0$, then the descent survives the expansion of u_i with probability 0. In either case, the descent yields $w\$$ with probability 0; moreover, by Lemma A.1(b) we have $\varphi(w\$) = 0$, so the claim holds. We may therefore assume that all masses below are positive. If a node u_i is expanded upon being reached, the descent survives with probability $q_i^{\text{new}}/q_i^{\text{old}}$; if u_i was already expanded, no survival test is performed, and we regard its survival probability as the same value $1 = q_i^{\text{new}}/q_i^{\text{old}}$. Once u_i is expanded, the descent chooses its next node u_{i+1} with probability $q_{i+1}^{\text{old}}/q_i^{\text{new}}$. Thus, for every $i \in \{0, \dots, n\}$, the probability of surviving the possible expansion of u_i and then moving from u_i to u_{i+1} is

$$\frac{q_i^{\text{new}}}{q_i^{\text{old}}} \frac{q_{i+1}^{\text{old}}}{q_i^{\text{new}}} = \frac{q_{i+1}^{\text{old}}}{q_i^{\text{old}}}.$$

Therefore, the probability of following the entire path to $w\$$ and surviving every expansion along it is

$$\prod_{i=0}^{n} \frac{q_{i+1}^{\text{old}}}{q_i^{\text{old}}} = \frac{q_{n+1}^{\text{old}}}{q_0^{\text{old}}}.$$

Now $q_0^{\text{old}} = q_\varepsilon$ is the root mass at the start of the descent, since $u_0 = \varepsilon$. Moreover, $u_{n+1} = w\$$ is a leaf, so $q_{n+1}^{\text{old}} = \varphi(w\$)$ by the definition of the masses in Equation (3). Hence the descent yields $w\$$ with probability $\frac{\varphi(w\$)}{q_\varepsilon}$, as claimed. □

Theorem A.3 (Soundness, Monotonicity, Termination; precise form of the guarantees stated in Section 3)**.** *Let E be an envelope for φ with $Z = \sum_w \varphi(w\$) > 0$. Then:*

- *(a)* (Soundness) *At every iteration of Algorithm 1, conditioned on a sequence being yielded in Line 6, the yielded sequence is distributed according to the target φ/Z.*
- *(b)* (Monotonicity) *The root mass q_ε never increases, and the probability that some sequence is yielded in an iteration, Z/q_ε, never decreases.*
- *(c)* (Termination) *The probability that some sequence is yielded in an iteration is at least $Z/E(\varepsilon) > 0$, so Algorithm 1 yields a sample after finitely many iterations almost surely.*

Proof. Fix an iteration and let $\mathcal{K}$ be the state of the trie at its start, with root mass q_ε.

(a) By Lemma A.2, the iteration yields a sequence $w\$$ with probability $\varphi(w\$)/q_\varepsilon$ for every w. Summing over all complete sequences $w\$$, it yields *some* sequence with probability Z/q_ε, and conditioned on that event it yields $w\$$ with probability $\varphi(w\$)/Z$, independently of $\mathcal{K}$. Since this holds at every iteration, it holds for the overall output.

(b) The state at the start of the next iteration is a superset of $\mathcal{K}$, so by Lemma A.1(c) its root mass is at most q_ε, and by (a) its yield probability Z/q_ε is at least the current one.

(c) By Lemma A.1, $Z \le q_\varepsilon \le E(\varepsilon)$, so the yield probability Z/q_ε is at least $Z/E(\varepsilon)$, which is positive because $Z > 0$. The probability that none of the first n iterations yields is therefore at most $(1 - Z/E(\varepsilon))^n \to 0$. □

Proposition A.4 (Closure)**.** *The following are envelopes for the indicated targets.*

(1) Weighted products. $\prod_k E_k^{\omega_k}$ *for* $\prod_k \varphi_k^{\omega_k}$*, whenever* $\omega_k > 0$ *and* $\sum_k \omega_k \geq 1$*. In particular, products* $E \cdot E'$ *for* $\varphi \cdot \varphi'$*, and powers* E^γ *for* φ^γ *with* $\gamma \geq 1$*.*

(2) Minima and mixtures. $\min(E, E')$ *for* $\min(\varphi, \varphi')$*, and* $\sum_k \omega_k E_k$ *for* $\sum_k \omega_k \varphi_k$ *with* $\omega_k \geq 0$*.*

(3) Prefix-monotone scorers. $E \cdot g$ *for* $\varphi \cdot g$*, for every* $g \colon \Sigma^*\$^? \to [0, 1]$ *with* $g(ua) \leq g(u)$*. This includes masks* $g(u) = \mathbb{1}[u \in \mathrm{prefix}(\mathcal{L})]$ *(hard constraints), and scorers defined only on complete sequences, by setting* $g \equiv 1$ *on proper prefixes.*

(4) Target domination. *If* E *is an envelope for* ψ *and* $\varphi(w\$) \leq \psi(w\$)$ *for every complete sequence* $w\$$*, then* E *is also an envelope for* φ*.*

Envelopes are not *closed under powers* E^γ *with* $\gamma < 1$*, nor under weighted quotients* $E^\alpha/(E')^\beta$ *with* $\alpha, \beta > 0$*: these transformations need not satisfy condition* ***(ii)****, and the corresponding targets need not be normalizable.*

Proof. We first verify condition **(i)**. For weighted products it follows from $E_k(w\$) \geq \varphi_k(w\$)$ and $\omega_k > 0$; for minima and mixtures from monotonicity in each argument; and for prefix-monotone scorers from $g(w\$) \geq 0$. For target domination, $E(w\$) \geq \psi(w\$) \geq \varphi(w\$)$. We now verify condition **(ii)** at a prefix u. Write $x_k(a) = E_k(ua)$ for $a \in \Sigma_\$$, so that $\sum_a x_k(a) \leq E_k(u) =: X_k$.

(1) Suppose first $s := \sum_k \omega_k = 1$. For each k let $y_k(a) = x_k(a)/\sum_{a'} x_k(a')$ (or 0 if the sum is 0). The weighted AM-GM inequality gives $\prod_k y_k(a)^{\omega_k} \leq \sum_k \omega_k y_k(a)$ for each a; summing over a, $\sum_a \prod_k y_k(a)^{\omega_k} \leq \sum_k \omega_k \sum_a y_k(a) \leq 1$. Multiplying back by $\prod_k (\sum_a x_k(a))^{\omega_k}$ yields the generalized Hölder inequality $\sum_a \prod_k x_k(a)^{\omega_k} \leq \prod_k \left(\sum_a x_k(a)\right)^{\omega_k} \leq \prod_k X_k^{\omega_k}$. If $s > 1$, then since $0 \leq x_k(a) \leq X_k$ we have $x_k(a)^{\omega_k} \leq x_k(a)^{\omega_k/s} X_k^{\omega_k(1-1/s)}$; summing over a and applying the case $s = 1$ to the exponents ω_k/s gives $\sum_a \prod_k x_k(a)^{\omega_k} \leq \prod_k X_k^{\omega_k(1-1/s)} \prod_k X_k^{\omega_k/s} = \prod_k X_k^{\omega_k}$. Products are the case $\omega_k = 1$, and powers $\gamma \geq 1$ the case $K = 1$, $\omega_1 = \gamma$.

(2) $\sum_a \min(x(a), x'(a)) \leq \min\left(\sum_a x(a), \sum_a x'(a)\right) \leq \min(X, X')$, and mixtures are linear.

(3) $\sum_a E(ua)g(ua) \leq g(u) \sum_a E(ua) \leq g(u)E(u)$, using $g(ua) \leq g(u)$ for every $a \in \Sigma_\$$.

(4) Condition **(ii)** is unchanged, since the envelope E itself is unchanged.

For the negative statements, take $K = 1$ and a prefix u with two children of mass $1/2$ each: $(1/2)^\gamma + (1/2)^\gamma > 1$ for $\gamma < 1$. For weighted quotients, let

$$E(u) = E'(u) = 1, \qquad E(ua_0) = E(ua_1) = \frac{1}{2},$$

and

$$E'(ua_0) = \eta, \qquad E'(ua_1) = 1 - \eta$$

for $0 < \eta < 1$. Then

$$\frac{E(u)^\alpha}{E'(u)^\beta} = 1,$$

whereas the sum over the two children is

$$2^{-\alpha}\left(\eta^{-\beta} + (1-\eta)^{-\beta}\right),$$

which exceeds 1 for sufficiently small η. Thus $E^\alpha/(E')^\beta$ need not satisfy condition **(ii)** for any $\alpha, \beta > 0$. □

Lemma A.5. *For* $\tau \leq 1$ *and weights* $\omega_k \geq 0$ *with* $\sum_k \omega_k = 1$*,* M_τ *is concave, positively homogeneous of degree one, and therefore superadditive:* $M_\tau(x + y) \geq M_\tau(x) + M_\tau(y)$*. For* $\tau > 1$ *it is convex and subadditive.*

Proof. Concavity of the weighted power mean M_τ on $\mathbb{R}^K_{\geq 0}$ for $\tau \leq 1$, and convexity for $\tau \geq 1$, are classical (Hardy et al., 1934); the cases $\tau \in \{-\infty, 0\}$ are the limits (minimum and weighted geometric mean) and are handled directly. Homogeneity is immediate from the definition. For a concave, positively homogeneous h, $h(x+y) = 2\,h\big(\frac{x+y}{2}\big) \geq 2\big(\frac{1}{2}h(x) + \frac{1}{2}h(y)\big) = h(x) + h(y)$; for a convex one the inequality reverses. □

Corollary A.6. *For $\tau \leq 1$, $M_\tau(E_1, \ldots, E_K)$ is an envelope for $M_\tau(\varphi_1, \ldots, \varphi_K)$, and MARS samples exactly from the τ-ensemble. If each E_k is exact at complete sequences, then so is $M_\tau(E_1, \ldots, E_K)$. For $\tau > 1$, $c\sum_k \omega_k E_k$ with $c = 1/\min_k \omega_k$ is an envelope for $M_\tau(\varphi_1, \ldots, \varphi_K)$, by mixture closure and target domination (Proposition A.4(2),(4)), and MARS again samples exactly from the τ-ensemble. If each E_k is exact at complete sequences, then at every complete sequence the exact target mass is at least a $1/c$ fraction of its envelope bound.*

Proof. For $\tau \leq 1$, condition **(i)** follows from monotonicity of M_τ: at every complete sequence $w\$$,

$$M_\tau(E_1(w\$), \ldots, E_K(w\$)) \geq M_\tau(\varphi_1(w\$), \ldots, \varphi_K(w\$)).$$

For condition **(ii)**, superadditivity and monotonicity of M_τ give

$$\begin{aligned}\sum_a M_\tau(E_1(ua), \ldots, E_K(ua)) &\leq M_\tau\Big(\sum_a E_1(ua), \ldots, \sum_a E_K(ua)\Big) \\ &\leq M_\tau(E_1(u), \ldots, E_K(u)).\end{aligned}$$

If each E_k is exact at complete sequences, then the first inequality above is an equality there, so the resulting envelope is also exact at complete sequences.

For $\tau > 1$, fix a complete sequence $w\$$ and write $x_k = \varphi_k(w\$)$. For every k,

$$x_k \leq (c\,\omega_k)x_k \leq c\sum_j \omega_j x_j,$$

hence

$$\sum_k \omega_k x_k^\tau \leq \Big(c\sum_j \omega_j x_j\Big)^\tau$$

and therefore

$$M_\tau(x_1, \ldots, x_K) \leq c\sum_j \omega_j x_j.$$

Thus

$$M_\tau(\varphi_1(w\$), \ldots, \varphi_K(w\$)) \leq c\sum_k \omega_k \varphi_k(w\$).$$

The mixture $\sum_k \omega_k E_k$ scaled by c is an envelope for $c\sum_k \omega_k \varphi_k$ by Proposition A.4(2), and the pointwise domination above shows, by Proposition A.4(4), that it is also an envelope for $M_\tau(\varphi_1, \ldots, \varphi_K)$.

Finally, suppose that each E_k is exact at complete sequences. For every complete sequence $w\$$,

$$\sum_k \omega_k \varphi_k(w\$) \leq M_\tau(\varphi_1(w\$), \ldots, \varphi_K(w\$))$$

for $\tau \geq 1$ by Jensen's inequality. Hence

$$M_\tau(\varphi_1(w\$), \ldots, \varphi_K(w\$)) \geq \frac{1}{c}\left(c\sum_k \omega_k E_k(w\$)\right).$$

Thus the exact target mass is at least a $1/c$ fraction of its envelope bound. □

Say that a sampler has *next-token access* if it interacts with the models only through queries $P_k(\cdot \mid u)$ at prefixes u of its choice, and that it is *exact* if its output is distributed as the target for every choice of models.

Theorem A.7 (Lower bound for tempering and contrasting)**.** *Let $|\Sigma| \geq 2$ and let all models have length bound L.*

(a) For tempering, $\varphi = P^\gamma$ with $0 < \gamma < 1$, every exact sampler with next-token access makes at least $\frac{1}{4}|\Sigma|^{\lfloor L(1-\gamma)/(2-\gamma)\rfloor}$ queries in expectation on some model before producing its first sample.

(b) For contrasting, $\varphi = P_1^\alpha / P_2^\beta$ with $\alpha, \beta > 0$, every exact sampler with next-token access makes at least $\frac{1}{4}|\Sigma|^{L-1}$ queries in expectation on some pair of models, even though the ratio P_1/P_2 is at most $|\Sigma|^{(L-1)/\beta}$ on the support of P_1 for every pair in the construction.

Proof. Write $\sigma = |\Sigma|$ and fix a token $a_0 \in \Sigma$. A sampler A is a randomized algorithm; on a given model it makes a (random) sequence of queries and then outputs a sequence. We say that A *touches* a prefix v on a run if it queries some prefix u with $v \preceq u$. The coupling used in both parts is the following. Let P and P' be models (or tuples of models) whose next-token distributions agree at every prefix u with $v \not\preceq u$. Running A with the same random bits on P and P', the two runs receive identical answers, and hence make identical queries and outputs, until the first query that touches v. Hence the total-variation distance between the output distributions of A on P and on P' is at most $\Pr_P[A \text{ touches } v]$. If A is exact on both, this distance equals the distance between the two targets.

(a) Let $k = \lfloor L(1-\gamma)/(2-\gamma)\rfloor$ and $K = \sigma^k$; note that $k \le (L-k)(1-\gamma)$, i.e., $\sigma^{(L-k)(1-\gamma)} \ge K$. The base model P_0 chooses a prefix of length k uniformly among the K possibilities, then continues with $a_0^{L-k}\$$ deterministically. For a prefix v of length k, the model P_v agrees with P_0 except below v, where it continues with a uniformly random sequence of length $L-k$ followed by \$. Under P_0, the target P^γ is uniform over the K leaves, each of weight $K^{-\gamma}$, with $Z_0 = K^{1-\gamma}$. Under P_v, the $K-1$ unmodified leaves keep weight $K^{-\gamma}$, while the σ^{L-k} leaves below v have probability $(K\sigma^{L-k})^{-1}$ each and total weight $\sigma^{(L-k)(1-\gamma)}K^{-\gamma} \ge K \cdot K^{-\gamma} = Z_0$. So the subtree of v carries at least half of the target mass under P_v and $1/K$ of it under P_0, and the two targets are at total-variation distance at least $\frac{1}{2} - \frac{1}{K} \ge \frac{1}{4}$ (for $K \ge 4$; for smaller K the bound is trivial). By the coupling, $\Pr_{P_0}[A \text{ touches } v] \ge \frac{1}{4}$ for every v. A query at a prefix of length at least k touches exactly one v, and a shorter query touches none, so the expected number of queries of A on P_0 is at least $\sum_v \Pr_{P_0}[A \text{ touches } v] \ge K/4$.

(b) Let $k = L-1$ and $K = \sigma^k$, and let $P_1 = P_2 = P_0$ be the base model of (a), so that the target P_1^α/P_2^β is uniform over the K leaves, each of weight $K^{\beta-\alpha}$. For a prefix v of length k, let $P_2^{(v)}$ agree with P_2 except at v, where it gives probability $\eta = K^{-1/\beta}$ to a_0 and $1-\eta$ to some other token a_1 (followed by \$). The pair $(P_1, P_2^{(v)})$ agrees with (P_1, P_2) at every prefix not extending v. Its target gives the leaf $va_0\$$ weight

$$\frac{(1/K)^\alpha}{(\eta/K)^\beta} = K^{\beta-\alpha}\eta^{-\beta} = K \cdot K^{\beta-\alpha},$$

while every other leaf in the support of P_1 has weight $K^{\beta-\alpha}$. Moreover, the ratio $P_1/P_2^{(v)}$ is at most

$$\eta^{-1} = K^{1/\beta} = \sigma^{(L-1)/\beta}$$

on the support of P_1. Thus the subtree of v carries target mass

$$\frac{K}{2K-1} \ge \frac{1}{2},$$

against $1/K$ under (P_1, P_2), and the rest of the argument is as in (a). □

Corollary A.8 (Envelope cost–tightness trade-off)**.** *Suppose an envelope E for a target with length bound L can be expanded at any prefix using at most $C(L)$ next-token queries. Then MARS produces its first sample using at most*

$$(L+1) \cdot C(L) \cdot \frac{E(\varepsilon)}{Z}$$

next-token queries in expectation. Consequently, for every such envelope construction for tempering, $\varphi = P^\gamma$ with $0 < \gamma < 1$, there is a model for which

$$\frac{E(\varepsilon)}{Z} \ge \frac{|\Sigma|^{\lfloor L\cdot(1-\gamma)/(2-\gamma)\rfloor}}{4 \cdot (L+1) \cdot C(L)}.$$

For the contrast target $\varphi = P_1^\alpha / P_2^\beta$, with $\alpha, \beta > 0$, there is likewise a pair of models for which

$$\frac{E(\varepsilon)}{Z} \geq \frac{|\Sigma|^{L-1}}{4 \cdot (L+1) \cdot C(L)}.$$

In particular, if $C(L)$ is polynomial in L, then $E(\varepsilon)/Z$ is exponential in L on some instances.

Proof. By Theorem A.3, at every iteration the probability of producing a sample is at least $Z/E(\varepsilon)$, independently of the previous history. Hence the expected number of iterations before the first sample is at most $E(\varepsilon)/Z$. Since a descent expands at most $L+1$ prefixes and each expansion uses at most $C(L)$ next-token queries, the expected number of queries before the first sample is at most

$$(L+1) \cdot C(L) \cdot \frac{E(\varepsilon)}{Z}.$$

Applying the lower bounds of Theorem A.7 to this exact sampler gives the two stated inequalities. □

Locality is not the obstacle. Corollary A.8 is about tightness, not locality, and it is worth separating the two. Under the standing length bound, tempering does admit a trivial local envelope: writing N_u for the number of complete continuations of a prefix u that the bound allows, $E(u) = N_u^{1-\gamma} P(u)^\gamma$ is an envelope for P^γ. Condition **(i)** holds because $N_{w\$} = 1$, and condition **(ii)** is Hölder's inequality with exponents $1/(1-\gamma)$ and $1/\gamma$ applied to $\sum_{a \in \Sigma_\$} N_{ua}^{1-\gamma} P(ua)^\gamma$, using $\sum_{a \in \Sigma_\$} N_{ua} = N_u$ and $\sum_{a \in \Sigma_\$} P(ua) = P(u)$. This bound is computed at u from one next-token distribution, so it is as local as any envelope in Proposition A.4; what it is not is tight, and an envelope may be cheap to expand and still have $E(\varepsilon)/Z$ exponentially large. The corollary says that this is unavoidable rather than an artifact of the construction: were some envelope for tempering both polynomial to expand and polynomially tight, MARS run on it would be an exact sampler with next-token access making polynomially many queries, which Theorem A.7 forbids. Tempering and contrasting therefore sit outside the algebra of Proposition A.4 for a reason no envelope construction can repair.

A.1 Diminishing returns

Theorem A.9 (Diminishing returns)**.** *Let E be an envelope for φ with $Z > 0$, let K_k be the trie after k descents, $\rho_k = q_\varepsilon(K_k)/E(\varepsilon)$ its normalized root mass, and $X_k = \rho_{k-1} - \rho_k$ the mass removed by the k-th descent. Then (a) for every run and every $k \geq 1$, $\rho_k \, \mathbb{E}[X_{k+1} \mid K_k] \leq \rho_{k-1} \, \mathbb{E}[X_k \mid K_{k-1}]$, and (b) $\mathbb{E}[\rho_{k-1} X_k]$ is non-increasing in k.*

Proof. Fix an envelope E for φ with $Z > 0$, and write

$$s(v) = E(v) - \varphi(v\$) - \sum_{a \in \Sigma} E(va) \geq 0$$

for the slack of an incomplete sequence v, where nonnegativity follows from conditions **(i)** and **(ii)**. For every nonempty trie state K, summing s over K telescopes:

$$\sum_{v \in K} s(v) = E(\varepsilon) - \sum_{u \in \partial K \cap \Sigma^*} E(u) - \sum_{w\$ \in \partial K} \varphi(w\$) = E(\varepsilon) - q_\varepsilon(K).$$

The same identity holds trivially for the initial state $K = \emptyset$. Hence, for every trie state,

$$q_\varepsilon(K) = E(\varepsilon) - \sum_{v \in K} s(v). \tag{4}$$

In particular, if a descent starting from K expands a set D of previously unexpanded sequences, then its decrease in normalized root mass is

$$\frac{q_\varepsilon(K) - q_\varepsilon(K \cup D)}{E(\varepsilon)} = \frac{1}{E(\varepsilon)} \sum_{v \in D} s(v). \tag{5}$$

We next compute the probability that a given sequence is expanded. For every $v \notin K$,

$$\Pr[v \text{ is expanded by the next descent} \mid K] = \frac{E(v)}{q_\varepsilon(K)}. \tag{6}$$

Indeed, let u be the first unexpanded sequence on the path from ε to v. If $u = \varepsilon$, the claim is immediate, since $q_\varepsilon(K) = E(\varepsilon)$. Otherwise all strict prefixes of u on this path are expanded, so their transition probabilities telescope and the probability of reaching u is

$$\frac{q_u}{q_\varepsilon(K)} = \frac{E(u)}{q_\varepsilon(K)}.$$

Now suppose an unexpanded u on the path has been reached. Expanding it changes its mass from $E(u)$ to

$$q_u^{\text{new}} = \varphi(u\$) + \sum_{b\in\Sigma} E(ub).$$

For the next token a on the path to v, the probability of surviving this expansion and then choosing a is

$$\frac{q_u^{\text{new}}}{E(u)} \cdot \frac{E(ua)}{q_u^{\text{new}}} = \frac{E(ua)}{E(u)}.$$

Thus induction along the remainder of the path gives (6). If one of the relevant envelope values is zero, condition **(ii)** together with nonnegativity makes all corresponding descendant masses zero, and the same identity holds with both sides equal to zero.

Let

$$\rho(K) = \frac{q_\varepsilon(K)}{E(\varepsilon)}$$

and let X_{next} denote the normalized root mass removed by the next descent from K. By (5), linearity of expectation, and (6),

$$\begin{aligned}\mathbb{E}[X_{\text{next}} \mid K] &= \frac{1}{E(\varepsilon)} \sum_{v\notin K} s(v)\ \Pr[v \text{ is expanded} \mid K] \\ &= \frac{1}{E(\varepsilon) q_\varepsilon(K)} \sum_{v\notin K} E(v)s(v). \end{aligned} \tag{7}$$

Multiplying by $\rho(K)$ yields the key identity

$$\rho(K)\,\mathbb{E}[X_{\text{next}} \mid K] = \frac{1}{E(\varepsilon)^2} \sum_{v\notin K} E(v)s(v). \tag{8}$$

Now let $K \subseteq K'$. Since every term $E(v)s(v)$ is nonnegative,

$$\sum_{v\notin K'} E(v)s(v) \le \sum_{v\notin K} E(v)s(v).$$

Together with (8), this gives

$$\rho(K')\,\mathbb{E}[X_{\text{next}} \mid K'] \le \rho(K)\,\mathbb{E}[X_{\text{next}} \mid K]. \tag{9}$$

Along every run $K_{k-1} \subseteq K_k$. Applying (9) with $K = K_{k-1}$ and $K' = K_k$ gives

$$\rho_k\,\mathbb{E}[X_{k+1} \mid K_k] \le \rho_{k-1}\,\mathbb{E}[X_k \mid K_{k-1}],$$

which proves part (a).

Finally, taking expectations and using the tower property,

$$\begin{aligned}\mathbb{E}[\rho_k X_{k+1}] &= \mathbb{E}[\rho_k\,\mathbb{E}[X_{k+1} \mid K_k]] \\ &\le \mathbb{E}[\rho_{k-1}\,\mathbb{E}[X_k \mid K_{k-1}]] \\ &= \mathbb{E}[\rho_{k-1} X_k].\end{aligned}$$

Thus $\mathbb{E}[\rho_{k-1}X_k]$ is non-increasing in k, proving part (b). □

B MACHINE-CHECKED PROOFS

All statements of Sections 3 and 4 have been formalized and machine-checked in Lean 4 with Mathlib, with two exceptions discussed below. The formalization is in the files `LeanTheorems/Mars.lean`, `LeanTheorems/MarsAlgebra.lean`, `LeanTheorems/MarsMeans.lean`, `LeanTheorems/MarsClauses.lean` and `LeanTheorems/MarsDiminishing.lean` of the accompanying repository, and contains no `sorry`.

The formalization models tokens as a finite type and incomplete prefixes as lists. An envelope is represented canonically by a prefix bound E, a target weight φ on complete sequences, and the inequality

$$\varphi(u\$) + \sum_{a\in\Sigma} E(ua) \le E(u).$$

This representation retains exactly the part of an envelope that MARS uses. Conditions **(i)** and **(ii)** of Definition 3.1 imply the displayed inequality. Conversely, any object satisfying this inequality determines a valid envelope in the sense of Definition 3.1, for example by choosing the minimal admissible terminal bounds $E(w\$) = \varphi(w\$)$. Larger terminal bounds allowed by condition **(i)** are not represented separately in Lean, because Equation (3) assigns every complete sequence its exact target mass $\varphi(w\$)$. Since Equation (3) always assigns a complete sequence its exact target mass, the formalization need not store a separate, possibly loose, terminal envelope value. The trie is a finite prefix-closed set with the masses of Equation (3) defined by well-founded recursion; and a descent is the product of its factors along a path. The Lean development is formulated over unbounded lists and therefore retains a sink that accounts for possible mass at infinity. Under the standing finite-length assumption of the paper, this sink has probability zero.

Lemma A.2 is `returnProb_eq`; Theorem A.3(a) is `soundness`; (b) is `q_nil_insert_le` and `tsum_returnProb_le_insert`, which compare the trie before and after the expansion of a boundary node; (c) is `tsum_returnProb_ge`, `run_no_return_le` and `no_return_tendsto_zero`; and summability of φ is derived rather than assumed (`summable_φ`). The incremental update of Line 7 is certified by `q_insert_self`, `q_insert_eq_of_incomparable` and `q_insert_eq_of_prefix`: expanding v sets q_v to $\varphi(v\$) + \sum_{a\in\Sigma} E(va)$, leaves every node incomparable with v unchanged, and decreases every ancestor of v by exactly the decrease at v, so the masses defined by Equation (3) can be maintained by propagating that one number upward.

Target domination, Proposition A.4(4), is `targetDomination`: it keeps the prefix bounds E unchanged and replaces the target by a pointwise smaller target on complete sequences. The theorem `targetDomination_q_le` says that, for a fixed trie state, this cannot increase any node mass, while `targetDomination_soundness` is `soundness` applied to the dominated target. Output-only reweighting is `stringScorerEnvelope` and `stringScorer_soundness`, obtained from target domination because the verifier score lies in $[0, 1]$.

The remaining clauses of Proposition A.4 are formalized as constructions of envelopes (`wprod`, `prod`, `pow`, `min2`, `mix`, `scorer`, `mask`), with the generalized Hölder inequality proved from Mathlib's weighted AM-GM inequality. The superadditivity statement of Lemma A.5, which is the part the corollary uses, is `pmean_add_le`, proved for every $\tau \le 1$, $\tau \ne 0$, from the two-point concavity of $t \mapsto t^\tau$ (Mathlib's `Real.concaveOn_rpow`) for $0 < \tau \le 1$ and, for $\tau < 0$, from its two-point convexity derived from the concavity of $\log$ and the convexity of $\exp$; the power mean is defined to be 0 when $\tau < 0$ and some entry is 0, which is the convention the harmonic row of Table 1 needs. The remaining clauses of Lemma A.5 are in `MarsClauses.lean`: homogeneity is `pmean_smul`; concavity for $\tau \le 1$, $\tau \ne 0$, is `pmean_concaveOn`, derived from superadditivity and homogeneity; subadditivity and convexity for $\tau \ge 1$ are `pmean_add_ge` and `pmean_convexOn`, from Minkowski's inequality (Mathlib's `Real.Lp_add_le_of_nonneg`); and the limit cases are `geomMean_add_le`, `geomMean_smul` and `geomMean_concaveOn` for the weighted geometric mean ($\tau = 0$, from the weighted AM-GM inequality) and `minMean_add_le`, `minMean_smul` and `minMean_concaveOn` for the minimum ($\tau = -\infty$). The negative clause of Proposition A.4 is `not_envelope_rpow` and `not_envelope_div`: for the two-letter envelope `halves` with $E(\varepsilon) = 1$ and $E(a) = 1/2$, no envelope has bound E^γ for any $\gamma < 1$, nor bound E/E, since both violate condition **(ii)** at the root.

For $\tau \leq 1$, Corollary A.6 is `Envelope.gmean` and `consensus_gmean_soundness` (with `wprod` for $\tau = 0$). For $\tau > 1$, the scaled-mixture envelope followed by target domination is `coverageTargetEnvelope`, exactness is `coverage_soundness`, and the $1/c$ lower bound on the ratio of the exact target mass to the scaled-mixture terminal bound is `coverageTarget_fraction`.

Theorem A.9 is formalized in `MarsDiminishing.lean`. The prefix slack is

$$\texttt{slack}(v) = E(v) - \varphi(v\$) - \sum_{a \in \Sigma} E(va),$$

and `q_nil_sub_insert_eq_slack` proves that expanding a boundary prefix decreases the root mass by exactly this amount. The lemmas `reach_eq` and `expandProb_eq` show that, for every unexpanded prefix v, the probability that the next descent expands v is

$$\frac{E(v)}{q_\varepsilon(K)}.$$

The quantity `remainingGain` is $\sum_{v \notin K} E(v)s(v)$, and `expectedRemoval_eq` computes the expected normalized root-mass decrease of the next descent from these expansion probabilities. The key identity

$$\rho(K)\,\mathbb{E}[X_{\text{next}} \mid K] = \frac{1}{E(\varepsilon)^2} \sum_{v \notin K} E(v)s(v)$$

is `rho_mul_expectedRemoval_eq`. Since every summand is nonnegative, `remainingGain_antitone` shows that the right-hand side can only decrease as the trie grows. Thus `mars_diminishing_pathwise` proves part (a) in the statewise form for arbitrary prefix-closed states $K \subseteq K'$, and `mars_diminishing_run` gives the corresponding antitonicity along any nested sequence of trie states. Finally, `mars_diminishing_weighted` proves the finite weighted-average form of the same inequality; under the standing finite-length assumption, part (b) follows by weighting the finitely many histories at a fixed iteration by their probabilities and applying the tower property.

The soundness of the intersection envelope of Equation (2) is `section1_soundness` specialized to a language model as the second component; the more general sub-stochastic-potential version is what the Lean statement proves. The theorem `section1_soundness'` covers an arbitrary $[0, 1]$-valued potential g with $r \geq 1$. The reduction to CARS is `hard_cars_soundness`.

The two exceptions are as follows. First, Theorem A.7 is a statement about arbitrary query algorithms and is proved on paper only. It quantifies over every randomized algorithm with next-token access, so formalizing it means first formalizing that computational model—a type of oracle algorithms, the coupling of two runs on models that agree outside a subtree, and the resulting bound on the total-variation distance between their outputs—which is a development in its own right and beyond the scope of this paper; the statement is not out of reach of Lean, only out of proportion to it. Second, the formalization proves the statewise inequality underlying Theorem A.9, together with its finite weighted-average form, rather than defining the entire MARS run as a stochastic process. Under the paper's standing finite-length bound there are only finitely many histories at every fixed iteration, so part (b) follows from the formalized weighted inequality by weighting those histories by their probabilities and applying the tower property. The same finite-length bound separately guarantees that every descent terminates after finitely many steps.

C MARS GENERALIZES CARS

For a hard constraint $\mathcal{L}$ and the envelope $E(u) = P(u) \cdot \mathbb{1}[u \in \text{prefix}(\mathcal{L})]$, expanding a prefix u sets the masses of its invalid children to 0 and the survival probability to $\sum_{a \in \Sigma_\$:\, ua \in \text{prefix}(\mathcal{L})} P(ua \mid u)$, the LM's probability of choosing a valid next token. This is precisely the pruning performed by CARS's update strategy, which adds to its set of invalid prefixes every invalid continuation of every prefix of the current sample. The only difference is cosmetic: CARS discovers an invalid continuation by drawing it and then rejecting, whereas MARS flips a coin with the same probability at the moment the parent is expanded and never draws the invalid token. Consequently, on hard constraints MARS and CARS visit the same prefixes with the same probabilities and have the same acceptance rate in every state.

Conversely, MARS adds to CARS exactly what is needed for combinations of models: a mass that can be strictly between 0 and $P(u)$, and a rejection that can be partial. Remarkably, this requires no new data structure and no new proof technique; what it requires is the right notion of what the trie stores—an envelope rather than a set of invalid prefixes—and that notion turns out to be closed under a rich algebra of operations (Section 4).

D Additional Evaluation Data

This appendix describes the benchmarks in full and gives the measurements behind Section 5 that the section does not report.

Benchmarks. We use three tasks the two literatures MARS sits between already use. BBH word sorting (Suzgun et al., 2023) asks a model to sort a list of words alphabetically; it is the task Chan et al. (2026) evaluate their ensembles on, and it has a single correct answer with enough freedom in how to reach it that two prompts disagree. Text-to-SQL over Spider (Yu et al., 2018) pairs a question with a database schema and asks for the SQL that answers it, and JSON Schema (Geng et al., 2025) asks for a document that validates against a given schema. Both are standard in the constrained-decoding literature Parys et al. (2026); here they carry an ensemble target rather than a grammar. Targets are the generalized-mean ensembles of Table 1 over $K = 2$ language models—minimum, product, mixture and maximum—together with the single-model baseline each is built from, in the *within-model* setting of Chan et al. (2026): one model under two prompts. Models, prompts, instance and seed counts, sampler settings and hardware are collected in Section I.

How Figure 2 is measured. Its divergence axis is the KL between the empirical and the exact distribution restricted to the outputs the methods produced—what a finite sample can estimate, not the KL from the full target—on 12 short-answer instances at 1500 samples each, where outputs repeat. Its cost axis is 30 instances at three samples each; points are means with bootstrap 95% intervals, the exact samplers pooling two runs each, and one instance with 360 distinct answers dominates every mean, so the intervals are wide. RS's point comes from separate runs of rejection sampling with MARS's prefix cache, on the same instances and settings. One of its two divergence runs drew, once, a sequence that ran to the 128-token bound on an instance whose other answers are two words; the model is forced to terminate at that bound, so the draw is legitimate, but a single sequence of negligible individual probability contributes 0.6 to that instance's KL and lifts RS's mean to 0.047. Without that instance RS and MARS, measured in the same runs, sit together at 0.024 and 0.025.

Acceptance rate and root mass. MARS's acceptance rate is .91–1.00 on four of the five Spider rows, against far less on the word-sorting consensus operators, and the root mass barely moves on Spider while it falls steadily on word sorting. The maximum is the exception in acceptance rate: it is the lowest on Spider, at .61. Its slack is mostly terminal rather than branching slack: the dominating-mixture envelope is tight under branching but can be loose at complete sequences. When a prefix is first expanded, MARS assigns its EOS child the exact target mass and removes this terminal slack from the parent mass, so the decrease is propagated to the root and remembered. The root mass therefore falls to 0.84, and RS, with the same cache, needs $1.09\times$ the descents per sample that MARS does. Forward passes stay level ($0.98\times$), because the descents MARS saves are mostly cheap revisits of cached prefixes; the trie that remembers the resulting decreases holds 11% more expanded prefixes (Section J).

Cost over thirty samples. On the ten instances of Figure 3, rejection sampling with the same cache spends the same forward passes as MARS over thirty samples (1246 each on the minimum ensemble, 1111 against 1168 on the product), about 40 per sample after the first, but not the same descents: the cumulative ratio of RS's descents to MARS's grows from $1.1\times$ at the first sample to $2.7\times$ at the fifth and $4.4\times$ at the thirtieth on the minimum, and from $1.4\times$ to $1.6\times$ on the product. Over the same run the root mass falls from 0.51 to 0.47 on the minimum and from 0.74 to 0.67 on the product. Table 2, at five samples per instance, therefore understates the gap in wasted samples on workloads that draw more.

Rejections per accepted sample. Figure 4 gives the rejection schedule on the same instances. MARS's rejections per accepted sample fall from 1.10 over the first five samples to 0.4–0.5 thereafter

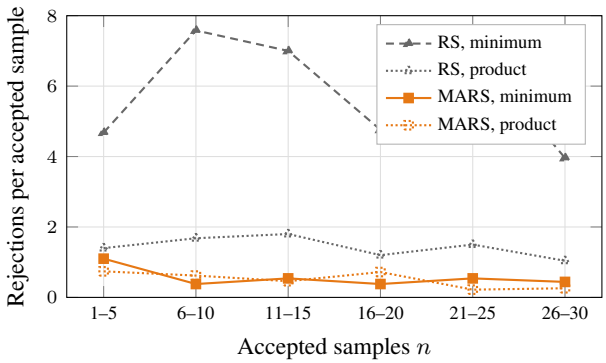


Figure 4: Rejections per accepted sample in blocks of five samples, mean over the ten instances of Figure 3. MARS's rate falls within the first block and stays low; rejection sampling's does not change, since it learns nothing.

Table 3: Text-to-SQL on Spider with Llama-3.1-8B-Instruct, within-model setting, 100 instances, five samples per instance, three seeds, 95% intervals over seeds. P_1 and P_2 are the same model under two different prompts, as in Table 2, and the ensemble rows are the generalized means at $\tau = -\infty$, 0, 1 and $+\infty$. Columns as in Table 2: accuracy is the fraction of draws that execute to the right answer, cost is forward passes per accepted sample, with the ratio to MARS in parentheses, and descents per sample are the tries from the root one accepted sample takes. RS shares MARS's prefix cache, and no method timed out on any instance. SMC is at $N = 10$ particles with the prefix cache unbounded.

	Accuracy			Forward passes per sample			Descents per sample	
Target $\varphi(w)$	MARS	RS	SMC	MARS	RS	SMC	MARS	RS
$P_1(w)$ (single prompt)	.56±.01	.57±.01	.58±.03	22.8±0.1	23.0±0.4 (1.01×)	146±0 (6.4×)	1.00±0.00	1.00±0.00 (1.00×)
$\min\big(P_1(w), P_2(w)\big)$	.60±.02	.60±.01	.60±.02	47.5±0.5	46.6±1.1 (0.98×)	291±5 (6.1×)	1.11±0.03	1.23±0.03 (1.10×)
$\sqrt{P_1(w)\,P_2(w)}$	.58±.03	.58±.01	.58±.02	45.9±1.5	45.1±0.3 (0.98×)	303±6 (6.6×)	1.03±0.01	1.03±0.00 (1.00×)
$\frac{1}{2}P_1(w) + \frac{1}{2}P_2(w)$	.60±.02	.58±.00	.59±.01	46.2±1.1	46.4±0.9 (1.00×)	300±6 (6.5×)	1.00±0.00	1.00±0.00 (1.00×)
$\max\big(P_1(w), P_2(w)\big)$	.58±.01	.58±.03	.59±.02	68.0±1.3	66.7±1.2 (0.98×)	299±1 (4.4×)	1.51±0.03	1.65±0.01 (1.09×)

on the minimum ensemble, and from 0.74 to about 0.25 on the product, while rejection sampling stays at its initial rate, 4 to 8 and 1.0 to 1.8.

E THE SECOND BENCHMARK: TEXT-TO-SQL ON SPIDER

The second task pairs a question with a database schema and asks for the SQL that answers it (Yu et al., 2018). Table 3 repeats the within-model experiment of Section 5 with the same model, the same five targets and the same settings, at three seeds.

Spider is the easy end of the three benchmarks. Two prompts asking for SQL against one schema agree on nearly every token, so the envelope is close to tight before the first descent and the trie has little to remove: MARS's acceptance rate is .91–1.00 on four of the five rows and the root mass ends at 0.84, against 0.80 on word sorting (Section D). Against rejection sampling with the same cache the two therefore spend the same forward passes to within 2% on every row, and MARS's saving in wasted samples is correspondingly small: RS needs 1.10× the descents per sample on the minimum and 1.09× on the maximum, and exactly one descent per sample, like MARS, on the single prompt and the mixture, which never reject. Accuracy is flat across the operators—every cell sits within .56–.60—and MARS, RS and SMC agree on every row to within .02. SMC costs 4.4–6.6× the forward passes of MARS, between the 3.4–5.7× of word sorting and the 6.9–8.0× of JSON Schema (Section F).

Table 4: JSON Schema on Llama-3.1-8B-Instruct, within-model setting, 100 instances, five samples per instance, one seed. P_1 and P_2 are the same model under two different prompts, as in Table 2, and the ensemble rows are the generalized means at $\tau = -\infty$, 0, 1 and $+\infty$. Columns as in Table 2: accuracy is the fraction of draws that validate against the schema, cost is forward passes per accepted sample pooled over the row, with the ratio to MARS in parentheses, and descents per sample are the tries from the root one accepted sample takes, pooled the same way. RS shares MARS's prefix cache. SMC is at $N = 10$ particles with the prefix cache unbounded.

	Accuracy			Forward passes per sample			Descents per sample	
Target $\varphi(w)$	MARS	RS	SMC	MARS	RS	SMC	MARS	RS
$P_1(w)$ (single prompt)	.57	.56	.56	56.4	56.0 (0.99×)	445 (7.9×)	1.00	1.00 (1.00×)
$\min\big(P_1(w), P_2(w)\big)$	.61	.59	.58	112	115 (1.02×)	856 (7.6×)	1.44	3.40 (2.4×)
$\sqrt{P_1(w)\,P_2(w)}$	.58	.56	.55	114	118 (1.04×)	880 (7.8×)	1.27	1.89 (1.49×)
$\frac{1}{2}P_1(w) + \frac{1}{2}P_2(w)$	.54	.53	.54	115	121 (1.05×)	924 (8.0×)	1.00	1.00 (1.00×)
$\max\big(P_1(w), P_2(w)\big)$	.56	.50	.52	134	143 (1.07×)	919 (6.9×)	1.15	1.23 (1.07×)

F THE THIRD BENCHMARK: JSON SCHEMA

The third task is structured generation: given a JSON Schema from JSONSchemaBench (Geng et al., 2025), produce a document that validates against it. Table 4 repeats the within-model experiment of Section 5 with the same model, the same five targets and the same settings, at one seed. Why JSON Schema is one seed, and what one seed does and does not resolve, is explained with the other run settings in Section I.

JSON Schema sits between the other two tasks. Against rejection sampling with the same cache, the two spend the same forward passes to within 7% on every row, and MARS's saving is in wasted samples: RS needs 2.4× the descents per sample on the minimum, the strictest consensus operator, 1.5× on the product and 1.07× on the maximum, and exactly one descent per sample, like MARS, on the single prompt and the mixture, which never reject. The maximum gains least of the three benchmarks: two prompts against one schema agree so closely that the mixture MARS starts from is nearly exact at the complete sequences too, and its root mass ends at 0.98 against 0.84 on Spider and 0.80 on word sorting. A schema constrains the output almost as tightly as a SQL grammar does, so two prompts disagree less than on word sorting but more than on Spider, and no instance timed out for either sampler. MARS and RS agree to within .02 on four of the five rows, inside what one seed can resolve. The maximum is the exception, at .56 against .50: the two sample the same distribution, so a gap of .06—roughly twice the standard error of a difference between two cells, and the largest MARS–RS gap anywhere in this paper—is more than one seed comfortably explains. It is the row whose envelope has terminal slack but no additional branching slack, and we report it rather than pool it away; a second seed would settle whether the difference is simply the tail of the sampling noise. SMC costs 6.9–8.0× the forward passes of MARS on every row, above the 3.4–5.7× of word sorting and the 4.4–6.6× of Spider (Section E), and its accuracy agrees with MARS's to within .04 on each.

G OTHER MODELS

Table 5 repeats the within-model experiment of Table 2 on Qwen2.5-7B-Instruct and phi-4, with the same task, prompts, instance set and settings, at one seed. Without replicate seeds there are no intervals, so accuracy differences of a few points should be read as noise; 500 draws per cell put the standard error of an accuracy near .02.

Two findings carry over from Llama unchanged. Against rejection sampling with the same cache, MARS wastes far fewer samples where the envelope rejects most: RS needs 4.3× and 4.0× the descents per sample on the minimum and 2.7× and 2.2× on the product, and the same number on the single prompt and the mixture; on the maximum, where the slack sits at the complete sequences rather than in the interior, it needs 1.04× and 1.07×. Forward passes stay within 4% of MARS's on every row where RS never times out except phi-4's product (1.17×), and fall below them on Qwen2.5-7B's

minimum and product, which is where RS gives up (below). Against SMC at ten particles, MARS is $7.4\times$ to $8.9\times$ cheaper on the single prompt, the mixture and the maximum for both models.

The gap to SMC narrows on the consensus operators, and on Qwen2.5-7B it inverts: on the minimum SMC spends fewer forward passes per sample than MARS (438 against 489), and on the product MARS is only $1.3\times$ cheaper. These are the targets on which Qwen2.5-7B's two prompts disagree most, as rejection sampling shows: it times out on 26 of 100 instances for the minimum and 9 for the product, where MARS times out on none. Its forward passes there are lower than MARS's only because a timed-out instance costs RS almost nothing: its 400 rejected descents return to the same cached prefixes, the region it cannot leave, while MARS spends passes moving out of it and produces the samples. A disagreement that costs rejection sampling whole descents costs SMC only reweighted particles, so where disagreement is extreme the price of exactness is real, and Table 5 is where it shows.

Table 5: BBH word sorting on Qwen2.5-7B-Instruct and phi-4, within-model setting, 100 instances, five samples per instance, one seed. P_1 and P_2 are the same model under two different prompts, as in Table 2, and the ensemble rows are the generalized means at $\tau = -\infty$, 0, 1 and $+\infty$. Columns as in Table 2: accuracy is the fraction of correct draws, cost is forward passes per accepted sample pooled over the row (total passes over total accepted samples), with the ratio to MARS in parentheses; descents per sample are pooled the same way, and RS shares MARS's prefix cache. The last two columns count the instances, out of 100, on which a method timed out, that is, gave up on a sample after 400 rejected descents; SMC never rejects and so never times out. SMC uses the unbounded prefix cache, as in Table 2.

		Accuracy			Forward passes per sample			Descents per sample		Timed out	
Model	Target $\varphi(w)$	MARS	RS	SMC	MARS	RS	SMC	MARS	RS	MARS	RS
Qwen2.5-7B	$P_1(w)$ (single prompt)	.14	.13	.15	35.1	35.8 (1.02×)	314 (8.9×)	1.00	1.00 (1.00×)	0	0
	$\min\big(P_1(w), P_2(w)\big)$	.15	.16	.18	489	359 (0.73×)	438 (0.9×)	18.80	81.26 (4.3×)	0	26
	$\sqrt{P_1(w)\,P_2(w)}$	.15	.15	.18	393	315 (0.80×)	516 (1.3×)	13.04	35.12 (2.7×)	0	9
	$\frac{1}{2}P_1(w) + \frac{1}{2}P_2(w)$	.14	.15	.14	80.8	78.0 (0.96×)	666 (8.2×)	1.00	1.00 (1.00×)	0	0
	$\max\big(P_1(w), P_2(w)\big)$	.14	.14	.15	76.7	79.2 (1.03×)	669 (8.7×)	1.02	1.07 (1.04×)	0	0
phi-4	$P_1(w)$ (single prompt)	.24	.23	.24	30.2	30.8 (1.02×)	226 (7.5×)	1.00	1.00 (1.00×)	0	0
	$\min\big(P_1(w), P_2(w)\big)$	.29	.27	.28	135	138 (1.02×)	338 (2.5×)	3.98	16.08 (4.0×)	0	1
	$\sqrt{P_1(w)\,P_2(w)}$	.27	.26	.27	108	127 (1.17×)	400 (3.7×)	2.67	5.89 (2.2×)	0	0
	$\frac{1}{2}P_1(w) + \frac{1}{2}P_2(w)$	.23	.22	.25	62.2	63.4 (1.02×)	470 (7.6×)	1.00	1.00 (1.00×)	0	0
	$\max\big(P_1(w), P_2(w)\big)$	.21	.20	.24	64.0	63.5 (0.99×)	479 (7.5×)	1.08	1.15 (1.07×)	0	0

H THE HARDEST SETTING: AN ENSEMBLE OF TWO DIFFERENT MODELS

Every target in Section 5 and in the appendices above pairs one model with itself under two prompts. This appendix reports the hardest setting we run: an ensemble of two *different* models, Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct, which share a tokenizer, so their ensemble under one prompt is a target MARS samples exactly. We run the minimum and the product on 30 word-sorting instances at three samples each, one seed, with the same budget of 400 rejected descents; SMC never rejects and so never times out. Because RS gives up on most instances, Table 6 pools descents over all accepted samples and reports forward passes only on the instances both exact methods complete: pooled passes would reward giving up, since an abandoned instance's 400 descents retrace cached prefixes and cost RS almost nothing.

Learning pays most here on the instances both samplers complete, and it decides whether most samples arrive at all. On the common subset RS needs $13\times$ and $10\times$ the descents per sample of MARS (26.0 against 2.0, 35.2 against 3.5) for about the same forward passes, and overall it times out on 21 and 17 of the 30 instances, twice MARS's 10 and 9. MARS still times out on a third of the instances, so the trie helps but does not make exact sampling reliable on a target this hard. Against SMC the comparison inverts: a sweep of ten particles costs a fixed 208 and 235 passes per draw, a quarter of what an exact sample costs MARS over all 30 instances (918 and 941). Whether those cheap draws are close to the target we can check only where outputs repeat: on twelve short-answer instances at 300 samples, SMC at five and ten particles sits on MARS's noise floor (paired KL differences $+0.001$ and -0.002), but those instances have under three distinct answers each. In

Table 6: Cross-model ensembles of Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct on word sorting, 30 instances, three samples, one seed; a method times out on an instance after 400 rejected descents. Descents per sample are total descents over total accepted samples; SMC never rejects, so it has none. The common subset is the 9 (minimum) and 13 (product) instances both exact methods completed.

	Instances timed out			Descents, pooled		Passes, common subset		
Target	MARS	RS	SMC	MARS	RS	MARS	RS	SMC
$\min(P_{7B}, P_{14B})$	10	21	0	83.4	327 (3.9×)	24.7	27.8 (1.1×)	56.0 (2.3×)
$\sqrt{P_{7B}\,P_{14B}}$	9	17	0	85.4	211 (2.5×)	53.7	56.1 (1.0×)	94.4 (1.8×)

the regime that costs 900 passes the only evidence is indirect, SMC's draws averaging 35.6 tokens against 27.5 for an exact sample (RS's 20.1 reflect only the easy instances it solved), so we report cross-model cost without a fidelity claim. As Parys et al. (2026) observe for CARS, when rejected prefixes share no structure an approximate method may be the practical choice.

I Experimental settings

Runs. Every within-model benchmark uses 100 instances and five samples per instance, at five seeds on word sorting, three on Spider and one on JSON Schema; the cross-model rows of Section H use 30 instances at three samples and one seed. The main tables run Llama-3.1-8B-Instruct (Grattafiori et al., 2024); Section G repeats the experiment on Qwen2.5-7B-Instruct and phi-4, and the cross-model rows pair Qwen2.5-7B-Instruct with Qwen2.5-14B-Instruct (Yang et al., 2025). The seed counts are set by the cost of the SMC baseline, not of MARS. SMC uses between 3.4 and 8.0 times the forward passes of MARS on every row of every task (Table 2, Section F), and with its prefix cache unbounded it peaks at 31–37 GB on the ensemble rows, so its shards need a 48 GB card, while MARS and rejection sampling keep their trie in host memory and need the weights plus one forward pass on the GPU, 15–17 GB for Llama-3.1-8B-Instruct except on a JSON Schema instance with a 13k-token prompt (Section J); on 22 GB A10G instances every JSON Schema SMC shard ran out of memory after two to three hours of compute. JSON Schema, whose outputs are the longest, is the extreme: one SMC shard takes about 90 minutes on an L40S, and each further seed is 100 MARS/RS shards plus 100 SMC shards, roughly 225 GPU-hours with the SMC half confined to shared 48 GB cards we could not hold for the two days it would take, so we stopped at one seed there. What one seed can and cannot support: 500 draws per cell put the standard error of an accuracy near .02, so MARS–RS gaps of at most .03 are not resolved, while the cost ratios are the quantity that did not move across seeds on the other two tasks (Spider's RS-to-MARS ratios, in forward passes and in descents per sample, vary by at most .1 over three seeds).

Prompts. For word sorting we use the two prompts of Chan et al. (2026) verbatim, a zero-shot instruction and a one-shot arrow format. Spider and JSON Schema are not theirs; for each we pair the benchmark's own system prompt with a second one we wrote asking for the same output in fewer words, over identical few-shot examples and user message.

Samplers. Rejection sampling is MARS with its bounds never tightened: it keeps MARS's prefix cache, so a revisited prefix costs no forward pass, but every descent faces the original envelope. SMC is `genlm-control` at the $N = 10$ particles and effective-sample-size threshold of 0.9 that Chan et al. (2026) report, with an unbounded prefix cache so that no lookup evicts. One SMC sweep returns one draw; its forward-pass cost is the total number of uncached component-model evaluations requested by all particles during that sweep.

Models. Table 7 lists every checkpoint used in the paper by its Hugging Face revision, so each run can be repeated against the same weights. All models run in `bfloat16` at temperature 1, with top-$p = 1$ and top-$k = 0$. The maximum output length is 128 new tokens on each of the three benchmarks, counted as in Section 2; the longest draws reach it, while word sorting averages about 35 (Section J). Below this limit, MARS, rejection sampling, and SMC all use the model's full next-token distribution; at the limit, EOS is forced as specified in Section 2. The two models of a within-model

target are the same checkpoint under two prompts; the two of a cross-model target share a tokenizer, which MARS checks before building the trie.

Table 7: Checkpoints. The revision is the Hugging Face commit the weights were loaded from; the same revisions were used on both clusters the experiments ran on.

Model	Hugging Face id	Revision	Used in
Llama-3.1-8B-Instruct	`meta-llama/Llama-3.1-8B-Instruct`	`0e9e39f2`	Section 5 main tables
Qwen2.5-7B-Instruct	`Qwen/Qwen2.5-7B-Instruct`	`a09a3545`	Section G, cross-model
Qwen2.5-14B-Instruct	`Qwen/Qwen2.5-14B-Instruct`	`cf98f3b3`	cross-model
phi-4	`microsoft/phi-4`	`2db69c1c`	Section G

Software and hardware. The samplers run on `transformers` 4.57.6 and `torch` 2.5.1 (CUDA 12.1), the models served by Hugging Face transformers in `bfloat16` with no compiled inference engine, so a forward pass through a given component model is the same computation for every method. MARS and rejection sampling are our implementation; the SMC baseline is `genlm-control` 0.3.0, the implementation of Chan et al. (2026), driven through the same envelope. Every run uses a single GPU. The single-model rows need the weights plus one forward pass—15–17 GB for Llama-3.1-8B-Instruct, and 30.9 GB on the one JSON Schema instance whose prompt is 13,274 tokens long—while the cross-model ensembles hold two models at once and peak at 42.2 GB, of which 41.7 GB are the weights of the two models. Those rows, and SMC's unbounded prefix cache at 31–37 GB, therefore need a 48 GB card; the rest fit in half that (Section J).

J Computational overhead

MARS buys its forward passes with bookkeeping: it keeps a trie in host memory, writes a vector of bounds at every prefix it expands, and propagates every decrease in node mass to the root. This appendix measures what that costs, in time and in memory.

J.1 Time

What an expansion costs. Expanding a prefix u costs one evaluation of each component at u—one forward pass for each language-model component—one write of the ratio vector $\log E(ua) - \log E(u)$ at the new node, and one propagation of the resulting decrease in node mass to the root: $|u|$ ancestors, each a reduction over $\Sigma_{\$}$ and one scalar write, by Equation (3).

Outside the forward passes the work is therefore $O(|u| \cdot |\Sigma_{\$}|)$ host floating-point operations, and since each level reads two `float32` vectors over the vocabulary, a few tens of megabytes of host memory traffic for a prefix of the length these tasks generate—at most 128 tokens, and about 35 on average on word sorting. A descent that stays inside the expanded part of the trie is cheaper still: one categorical draw over $\Sigma_{\$}$ per level and no forward pass at all. The comparison to make is against one forward pass of an 8B-to-14B model in `bfloat16` with no compiled inference engine (Section I), which is 45–250 ms on the cards we ran on.

Measured. Every run that produced Table 2 and Section G drew each instance twice in the same process on the same card, once with MARS and once with the same sampler with its learning switched off—the bounds are never tightened and the trie is discarded between descents—so the two differ in nothing but whether what a rejection teaches survives.[1] Dividing each method's wall clock by its forward passes gives what one pass costs under each, and the ratio of the two is what remembering adds. Because the two are paired within a process, the ratio does not depend on which cluster the shard ran on. Table 8 reports it over the 3,382 instances on which both methods made at least 200 passes, a floor that keeps the per-instance setup from dominating the quotient.

A MARS forward pass costs 1.010 times one without the trie at the median over all 3,382 instances, 1.034 at the third quartile and 1.072 at the ninetieth percentile; 83% of the instances are within 5% and 93% within 10%. The two reads that isolate the bookkeeping best are Spider and JSON Schema,

[1] This is not the rejection sampling of Table 2, which also keeps MARS's prefix cache and therefore makes fewer passes; only the cost of one pass is read off here, and the cache does not change it.

Table 8: Wall clock per forward pass, paired per instance against the same sampler with its learning switched off. The ratio is per instance; "ms per node" divides MARS's wall clock in excess of that instance's measured pass cost by the number of prefixes it expanded, and is an upper bound on the trie's share, since it also absorbs the per-instance setup, which MARS, running first, pays for.

				ms per forward pass		Ratio		
Task	Target	Models	Instances	no learning	MARS	median	90th pct.	ms per node
Word sorting	single model	Llama-8B	67	65	67	1.040	1.11	2.5
Word sorting	single model	phi-4	33	96	98	1.015	1.22	1.4
Word sorting	single model	Qwen-7B	37	49	50	1.009	1.08	0.4
Word sorting	within-model	Llama-8B	937	56	59	1.033	1.08	3.7
Word sorting	within-model	phi-4	257	89	91	1.012	1.07	2.2
Word sorting	within-model	Qwen-7B	281	45	46	1.025	1.37	2.2
Word sorting	cross-model	Qwen-7B + Qwen-14B	39	194	203	1.043	1.14	16.2
Spider	single model	Llama-8B	32	247	250	1.003	1.02	0.6
Spider	within-model	Llama-8B	610	147	148	1.004	1.02	1.4
JSON Schema	single model	Llama-8B	182	197	205	1.003	1.09	0.6
JSON Schema	within-model	Llama-8B	907	188	191	1.004	1.03	1.7

at 1.003–1.004: their prompts are long enough (430–1,740 tokens, and one of 13,274) that a pass costs the same wherever in the generated prefix it falls. Word sorting, whose prompts are short, sits higher, at 1.009–1.043, and part of that is not bookkeeping at all: our implementation recomputes a prefix without a key-value cache (the *GPU* paragraph below), so a pass deep in a sequence costs more than a shallow one, and MARS's passes are the deeper ones—a sampler that expands each prefix once never repeats the cheap passes near the root, which is the same saving the forward-pass counts already credit it with. The ratio is flat in the size of the trie, which is the property the diminishing-returns argument needs: 1.013 on the 2,273 instances with 100–300 expanded nodes, 1.006 on the 1,011 with 300–1,000, 1.021 on the 69 with 1,000–3,000, and 1.032 on the 29 with more than 3,000, where the cross-model target and its two larger vocabularies dominate. What is left in the tail is small instances, where a fixed per-instance cost—tokenizing the prompt, laying out the plan—is divided by few passes; on the cross-model row that cost is several seconds, which is why its ms-per-node figure is the largest in the table and why it should be read as an upper bound rather than as the trie's own cost.

End to end. Across the 57 cells of task, target, model and seed in which both samplers ran, MARS's saving in wall clock per accepted sample is between 0.87 and 1.02 of its saving in forward passes per accepted sample, with a median of 0.95. The cost unit this paper reports is therefore the one the clock shows: on word sorting under the minimum, for instance, MARS takes 3.57–4.91 times fewer passes than the non-learning sampler across the five seeds and 3.36–4.46 times less time. The missing few percent is the overhead of Table 8.

J.2 Memory

MARS keeps two kinds of state: the models, together with the activations of one forward pass, on the GPU, and the trie in host memory. We measure both.

The trie. An expanded node u stores the ratios $\log E(ua) - \log E(u)$ for every token a, one `float32` vector over the vocabulary $\Sigma_\$$, and a second vector of the same size, the correction that Equation (3) writes when a bound below u is tightened. Every descent that expands a node propagates the new bound to the root, so every expanded node except the leaves of the expanded part of the trie carries both vectors, $8|\Sigma_\$|$ bytes: 1.0 MB for Llama-3.1 ($|\Sigma_\$| = 128{,}256$), 1.2 MB for Qwen2.5 (151,665) and 0.8 MB for phi-4 (100,352). A target over two models costs no more per node than a single model, since the trie stores the ratios of the combined envelope, not one vector per model. With E expansions and D descents on an instance, at most D expanded nodes are leaves, so the trie occupies between $(2E - D) \cdot 4|\Sigma_\$|$ and $2E \cdot 4|\Sigma_\$|$ bytes. Every run logs E and D, so Table 9 covers every instance of every MARS run behind Section 5 and Section G, at every seed we ran; it reports the upper end, and the lower end is at least 93% of it on the instance each row's maximum comes from. The trie persists across the five draws of an instance and is discarded between instances, so each figure is that instance's peak.

Table 9: Trie size per instance, over all instances, operations and seeds we ran for each row. Nodes are expanded prefixes; memory is the upper bound $2E \cdot 4|\Sigma_{\$}|$, checked against the exact size in the text.

				Nodes		Trie memory		
Task	Target	Models	Instances	median	max	median	90th pct.	max
Word sorting	single model	Llama-8B	500	82	420	80 MB	225 MB	411 MB
Word sorting	single model	phi-4	100	150	419	114 MB	229 MB	321 MB
Word sorting	single model	Qwen-7B	100	159	483	184 MB	408 MB	559 MB
Word sorting	within-model	Llama-8B	2,000	92	834	90 MB	237 MB	816 MB
Word sorting	within-model	phi-4	400	164	3,702	126 MB	348 MB	2.8 GB
Word sorting	within-model	Qwen-7B	400	208	9,216	240 MB	2.1 GB	10.4 GB
Word sorting	cross-model	Qwen-7B + Qwen-14B	60	326	3,915	377 MB	3.3 GB	4.4 GB
Spider	single model	Llama-8B	300	101	538	99 MB	206 MB	526 MB
Spider	within-model	Llama-8B	1,200	105	756	103 MB	272 MB	740 MB
JSON Schema	single model	Llama-8B	300	250	635	245 MB	565 MB	621 MB
JSON Schema	within-model	Llama-8B	1,080	262	1,095	256 MB	587 MB	1.0 GB

The median trie is between 80 and 380 MB on every row, and the 90th percentile stays below 600 MB on every row except the two Qwen-7B ensembles of word sorting. The large tries are the instances on which the two prompts or models disagree most: the ten Qwen-7B within-model instances with more than 5,000 nodes, all under the minimum or the product, needed 340–730 descents for their five samples, against a median of 5.5 on that row. Growth slows as the trie fills in: from the first to the fifth accepted sample the number of nodes grows by a median factor of 3.2 over 6,380 instances, against 5 if every sample cost the same, because later descents mostly follow prefixes that are already expanded, in line with the diminishing returns of Theorem A.9. Where memory is scarce the trie can be frozen: MARS with a fixed set of expanded prefixes samples from a fixed proposal and rejects exactly as rejection sampling does from it, so its samples stay exact and only the further reduction in cost is lost.

Measured. To check the bound and to measure what it cannot give, we reran three shards with instrumentation, MARS only, with the same settings on one L40S: JSON Schema within-model under the mixture and word sorting within-model under the maximum, both on Llama-3.1-8B-Instruct, and the cross-model Qwen-7B + Qwen-14B product, 21 instances in all. The exact trie size, summed over the storage of every stored vector, lies inside the interval on all 21, and the number of nodes without a correction vector is at most D on each. The process's peak resident memory, which includes the trie, the Python runtime and the CUDA context, is 4.5–5.5 GB.

The maximum is the row where terminal slack matters most. To isolate the effect of remembering this slack in the trie, we compare the current MARS update with a post-hoc leaf-rejection variant. In MARS, when a prefix w is first expanded, its EOS child receives the exact target mass $\varphi(w\$)$, so the terminal slack $E(w\$) - \varphi(w\$)$ is removed from q_w, propagated to the root, and never paid again. In the post-hoc variant, the expansion instead keeps the leaf mass $E(w\$)$, and a descent that later reaches $w\$$ is accepted with probability $\varphi(w\$)/E(w\$)$. A rejection at the leaf therefore leaves the trie masses unchanged, so the same terminal slack can be paid again on later descents.

Remembering terminal slack trades host memory for fewer descents. Over this shard's ten instances the MARS trie holds $1{,}380$ expanded prefixes against $1{,}242$ for the post-hoc variant and grows from $1{,}256$ to $1{,}395$ MB, a median of 143 MB per instance against 115 MB, while the descents a sample costs fall from 1.32 to 1.10. The root mass, which remains 1.00 in the post-hoc variant because terminal rejections do not change the trie, ends between 0.51 and 1.00 under MARS. The interval above is stated in E and D, both of which move, and holds on all ten either way.

GPU. The trie never touches the GPU. There MARS holds the weights and the activations of one forward pass, and our implementation recomputes each prefix without a key-value cache and keeps the `float32` log-probabilities of every position, so the transient grows with the length of prompt plus prefix. With 15.0 GB of Llama weights, the peak is 15.2–17.2 GB on 19 of the 20 Llama instances and 30.9 GB on the JSON Schema instance whose prompt, which contains the schema, is 13,274 tokens long; the other prompts of that shard are 430–1,740 tokens. The cross-model target peaks at 42.2 GB, of which 41.7 GB are the weights of the two models. Only the positions of the generated prefix are needed, so keeping log-probabilities for those alone would remove most of the

transient; this is a property of our implementation, not of MARS. SMC is different in kind: with its prefix cache unbounded it holds 31–37 GB on the GPU on the ensemble rows (Section I), and that cache grows with the number of prefixes explored, whereas the MARS trie grows in host memory, where it is cheap.