STATE-COARSENED FUTURE-VALIDITY CORRECTION FOR CONSTRAINED DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Grammar-constrained decoding with token masking samples from the *locally renormalized* distribution μ_{proj}, not the true grammar-conditioned distribution μ^*; on prompted Dyck with GPT-2 the total-variation gap is 0.75, and single strings are over-weighted by 16 orders of magnitude. Exact correction requires the future validity $\Phi(y) = P_p(\text{complete legally} \mid y)$ for every prefix, which is intractable; practice therefore falls back on experience — flatten the schema, avoid deep nesting — with no way to tell, for the schema at hand, whether a cheap correction will do. We replace that rule of thumb with an *a priori computable* criterion. We estimate Φ *coarsened onto automaton states* — a lookup table indexed by (DFA/parser state, remaining budget) instead of exact prefixes — and price that approximation exactly. We (i) derive an exact identity $\mathrm{KL}(\mu^* \| q) = \sum_t$ per-step Jensen gaps of the error ratio $\hat{\Phi}/\Phi$ under μ^*, whose second-order form shows the coarsening error is the μ^**-weighted within-state variance of* $\log \Phi$ and the optimal table entry is the μ^*-weighted geometric mean — making the applicability of any state-level table a *computable quantity*; (ii) map this criterion across a grammar family (near-exact on Dyck; the criterion flags, in advance, grammars where Φ depends on content outside the state features) and across model scale, where the effect is *non-monotonic*: sharper content preferences raise within-state variance, so a syntactic coarsening that gains 31% at 1.7B gains nothing at 7B, while a semantically featurized state improves monotonically ($0.567 \to 0.074 \to 0.009$) — scale does not substitute for the right state features, and the criterion says in advance which regime one is in; and (iii) show that with a shared budget of masked rollouts, the state-level table is stably better than an offline prefix-level table (ASAp-keyed) once the budget reaches $N \geq 100$ (all three seeds), because the state table generalizes to unseen prefixes at zero extra forward passes per step. On full-scale models (SmolLM2-1.7B, Qwen3-8B; three seeds each) the correction adds no per-token forward passes (21.5 vs 21.0 calls/sample on Qwen3-8B) and preserves downstream accuracy, with sequence log-probability sharply improved at 1.7B and statistically unchanged at 8B; on content-dependent nested/long JSON and real JSONSchemaBench schemas the identity's prediction is confirmed exactly, giving a boundary computable in advance between where coarse tables suffice and where a learned $\hat{\Phi}$ is required. The same identity reads failures reported across the literature as one term: ASAp's slow, non-monotonic convergence, importance-weight degeneracy in our own estimator ablation and in AWRS (ESS $43/200 \to 6/200$), and, as $\hat{\Phi}/\Phi \to 0$, the zero-estimate collapse of twisted SMC on rare events. Speculative decoding composes: strict acceptance against the *uncorrected* masked verifier target keeps its quality (and its bias) with 3.6–4.5$\times$ fewer verifier forwards per token — $1.9/3.3/1.8\times$ wall-clock at deployment scale (135M drafter, 1.7B verifier, V100) — while moving $\hat{\Phi}$ into the verification target (SPEC-Ω) sidesteps the local-mask impossibility at no extra cost but, with a state lookup, inherits the same $\hat{\Phi}$ failure boundary.

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.

1 INTRODUCTION

Constrained decoding enforces formal-language constraints (JSON schemas, SQL, API grammars) by masking illegal tokens at each step and renormalizing. This is now standard infrastructure (xgrammar-style masking costs only 2–4 μs per step (Li et al., 2026)), but it silently changes the sampling distribution: masking never repays probability already spent walking toward a dead end, so the sampler draws from μ_{proj} rather than $\mu^*(y) \propto p(y)\,\mathbf{1}[y \in G]$, with real consequences in production agent systems (Li et al., 2026). Strict-schema and function-calling APIs (OpenAI, 2024) — dominated by tool selection, routing, and `enum`/boolean fields, exactly the *near-content-free* constraints our state-level correction handles at near-zero cost — make this a tax on the dominant LLM interface, not a benchmark curiosity. That the bias exists is established (Nie et al., 2026); that it can be removed by paying extra decode-time inference is established (Park et al., 2024; Lipkin et al., 2025; Dang et al., 2026). Neither tells a practitioner holding a particular schema whether a *cheap* correction suffices; the operative guidance is heuristic (strict-schema APIs cap nesting depth and property counts outright (OpenAI, 2024)). This paper supplies the missing layer: the exact price of a cheap correction, in closed form and computable *before* decoding, so that the boundary between "a zero-extra-forward table suffices" and "a learned $\hat{\Phi}$ is required" is a quantity one evaluates, not a convention one inherits.

Scale is not the fix, and can be the problem. Stronger models put sharper preferences on *content* — exactly the part of Φ a syntactic state cannot see — so the coarsening's advantage peaks at SmolLM2-1.7B (-31% TV on JSON array) and is gone by Qwen2.5-7B ($0.352 \to 0.352$), and on a nested schema at Qwen3-8B the lookup moves TV from 0.654 to 1.000 while accuracy barely moves ($0.996 \to 0.979$), so an accuracy dashboard hides it. Give the state the semantic feature Φ depends on and the gain instead grows monotonically with scale ($0.567 \to 0.074 \to 0.009$, mini-SQL). The deployment question is not "how large a model" but "does my state see what Φ depends on" (§4.2, §4.5) — which is what the criterion answers, before decoding.

A two-token example. Vocabulary $\{a, b\}$, legal strings $\{aa,\ b\}$, $p(a) = 0.9$, $p(b) = 0.1$, and after a only $p(a \mid a) = 0.05$ on the sole legal continuation. The target weights whole strings ($\mu^*(aa) \propto 0.045$ vs $\mu^*(b) \propto 0.1$: b wins with probability 0.69), but masked decoding commits to a with probability 0.9 — nothing is illegal *yet* — and, walled in, completes aa: a 31% string is emitted 90% of the time. One number per prefix repairs this: weighting step one by the future validity $\Phi(a) = 0.05$, $\Phi(b) = 1$ recovers μ^* exactly. Everything in this paper is about estimating that number cheaply — and knowing, in advance, when a cheap estimate must fail. At real scale the distortion is not a curiosity: on prompted Dyck with GPT-2 the masked sampler sits at TV 0.75 from μ^* and over-weights single strings by 16 orders of magnitude.

The exact fix is the Doob transform $\mu^*(a \mid y) \propto p(a \mid y)\,\Phi(ya)$ with $\Phi(y) = P_p(\text{legal completion} \mid y)$, which is intractable — and every practical corrector is, implicitly or explicitly, an estimator of this one quantity: GAD-style methods learn it per exact prefix online (Park et al., 2024), lookahead methods roll it out afresh at every token (Nakshatri et al., 2024), twisted SMC learns it as a twist network and pays K particles (Zhao et al., 2024; Dang et al., 2026), and rejection- or MCMC-based methods encode it in extra accept/reject forward passes (Lipkin et al., 2025; Anaya Gonzalez et al., 2025). The literature thus occupies a one-dimensional design space — how much decode-time inference is paid for how good a Φ estimate — but only its expensive end, and no existing method can say *in advance* what a cheaper estimate would cost. This paper prices the whole axis, and then solves its cheapest point.

Generality. Our object is the exact price of *any* future-validity approximation: identity (1) gives $\mathrm{KL}(\mu^* \| q)$ in closed form for every estimator $\hat{\Phi}$, and the within-state variance of $\log \Phi$ decides *before* decoding whether a zero-extra-forward state table suffices. The Φ viewpoint was named by Nie et al. (2026), whose analysis lives at the two *endpoints* of the axis — exact Φ and no correction (μ_{proj}) — on enumerable languages; these are precisely the two limits of identity (1) ($\hat{\Phi} = \Phi$: every Jensen gap vanishes; $\hat{\Phi} \equiv 1$: the identity prices masked decoding itself). Everything strictly between the endpoints — every practical approximate corrector, and whether its cost is knowable in advance — is priced only here. We study the middle point: **coarsen Φ onto the constraint automaton's state space**, estimated once from masked rollouts, then applied as a zero-extra-forward lookup at decode

time — with experiments at real vocabularies and scales (49k tokens, Qwen3-8B), not enumerable toy languages.

Contributions.

1. **Exact error identity** (§3), beyond the estimator-specific bounds of Nie et al. (2026): for any $\hat{\Phi}$, $\mathrm{KL}(\mu^* \| q)$ equals the sum of per-step Jensen gaps of $r = \hat{\Phi}/\Phi$ under μ^*; to second order, $\mathrm{KL} \approx \frac{1}{2}\sum_t \mathbb{E}\,\mathrm{Var}_{\mu^*}[\log r]$, with the μ^*-weighted geometric mean as the optimal table value. Verified numerically to machine precision on four grammars.
2. **A predictive applicability map** (§4.1): across a grammar family the same method ranges from near-exact (Dyck: TV $0.900 \to 0.0011$, Table 1) to criterion-flagged (required-fields schema), and the identity explains — and predicts in advance — why: coarsening error = dependence of Φ on content not captured by the state features.
3. **A unifying taxonomy** (§5): the Doob view makes every distribution-correcting decoder a Φ-estimator; existing methods (GAD, AWRS, SMC/P-GCD, DCCD) are located on the two axes of estimator and extra decode-time inference, and identity (1) prices the explicit-$\hat{\Phi}$ points of that space.
4. **Budget-matched comparison vs. prefix-level estimation** (§4.3): on a real-vocabulary (49k tokens) budgeted JSON task, the state-level table beats an offline prefix-frequency table at every rollout budget $N \geq 100$ (per-seed, all wins) and matches it below; most state buckets appear early ($40/44$ by $N{=}30$) but TV continues to improve through $N \geq 100$ ($0.446 \to 0.251$).
5. **Composition with speculative decoding** (§4.6): across three budgeted JSON tasks, strict speculation against the *uncorrected* masked target (μ^V_{proj}; bias unchanged) matches the masked sampler's quality with 3.6–4.5× fewer verifier forward passes per token (E29d grid), while a state-lookup $\hat{\Phi}$-weighted verifier target inherits the lookup's content-dependence failure — losing quality *and* speed.
6. **Deployability of the criterion** (App. H, App. I): the criterion runs as an offline pilot → variance gate → zero-extra-forward serving pipeline (TAB-Ω), or as a corrected speculative verification target (SPEC-Ω); Theorem 1 identifies the exactly optimal table as the solution of a convex program, of which contribution 1's geometric mean is the second-order approximation.

2 Setup

Base model p, legal set G (finite language, EOS-terminated), target $\mu^*(y) = p(y)\mathbf{1}[y \in G]/Z$. Masked decoding μ_{proj} renormalizes p over legal next tokens. Doob: $\mu^*(a \mid y) = p(a \mid y)\Phi(ya)/\Phi(y)$. Approximate decoder $q(a \mid y) \propto p(a \mid y)\hat{\Phi}(ya)$. State coarsening: abstraction $\sigma(y) =$ automaton state (optionally × remaining budget × semantic features), $\hat{\Phi}(y) = f(\sigma(y))$.

3 Theory: What does coarsening cost?

Let $r(ya) = \hat{\Phi}(ya)/\Phi(ya)$. Then, exactly and for any $\hat{\Phi}$,

$$\mathrm{KL}(\mu^* \| q) = \sum_t \mathbb{E}_{y_{<t}\sim\mu^*}\Big[\log \mathbb{E}_{a\sim\mu^*(\cdot|y_{<t})} r(y_{<t}a) - \mathbb{E}_{a\sim\mu^*(\cdot|y_{<t})} \log r(y_{<t}a)\Big]. \tag{1}$$

In words: at each step you pay only for how much the *relative* error of $\hat{\Phi}$ varies across the next-token branches that μ^* actually takes — being wrong is free, being *unevenly* wrong where it matters is what costs. Consequences: (a) any per-step constant rescaling of $\hat{\Phi}$ is harmless — only *within-step branch variation* of r under μ^* costs KL; (b) fluctuations on μ^*-negligible prefixes are invisible — this is why unweighted within-state dispersion fails to predict TV (Dyck has rel-std 0.90 yet TV 0.001); (c) to second order, $\mathrm{KL} \approx \frac{1}{2}\sum_t \mathbb{E}\,\mathrm{Var}_{\mu^*}[\log r]$, so for a lookup table the optimal entry is $\log f(s) = \mathbb{E}_{\mu^*}[\log \Phi \mid \sigma = s]$ and the residual error is the μ^*-weighted within-state variance of $\log \Phi$ (Figure 1). Pinsker converts to TV. The geometric mean minimizes this surrogate *state by*

state; the entries of sibling states reached from one prefix are coupled through that step's normalizer, so it is an upper bound on the optimal table's KL, not the minimizer. The exact optimum is a convex program with a moment-matching characterization (Theorem 1, App. H; Table 8 measures the gap, 3×10^{-5}–0.026 nats), where the theory is also packaged as a pilot→gate→serve pipeline (TAB-Ω). In practice the criterion is itself estimated, so it is a computable indicator rather than a decision procedure (App. D).

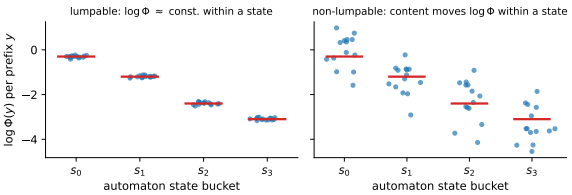

Figure 1: The lumpability picture (schematic). Each dot is a prefix, grouped by its automaton state; the red line is the lookup-table entry. Left: lumpable — $\log \Phi$ nearly constant within each state (Dyck). Right: non-lumpable — generated *content* moves $\log \Phi$ within a state; by identity (1) the μ^*-weighted spread around the red line is the irreducible KL cost of any state-level table.

Numerical verification (four grammars $\times$ all abstractions, exact computation): identity (1) holds to $\sim 10^{-16}$; the bound holds and is tight (e.g. JSON array TV $0.217 \leq$ bound 0.275); switching table entries to the weighted geometric mean drives good abstractions to near-exact (Dyck $0.0011 \rightarrow 3 \times 10^{-5}$; SQL semantic abstraction $0.034 \rightarrow 0.0012$).

One mechanism behind several reported failures. Read through (1), failures reported for different correctors are the same term. ASAp (Park et al., 2024) refines sound per-prefix overapproximations of Φ only along visited paths, so r is *uneven across sibling branches*; the identity predicts a KL that need not fall as individual estimates improve — matching their benchmarks where KL "barely improves" after 2,000 iterations and stays above masking's (their Fig. 5c). Our own μ^*-weighted estimator (App. G) and our same-task AWRS-IS ablation (App. K: AWRS's weight estimate used as sequential importance weights, not its exact rejection loop; ESS $43/200 \rightarrow 6/200$ from 135M to 1.7B) both degrade through importance weights concentrating on few rollouts — within-bucket variance of r. In the limit, a twist that assigns ≈ 0 to a branch μ^* takes gives $r \rightarrow 0$, $\log r \rightarrow -\infty$, and a divergent gap: the zero-estimate collapse of standard twisted SMC on rare events (Liu et al., 2026) is the identity's boundary case, not a separate pathology.

4 Experiments

The experiments test the identity's quantitative predictions (exact verification, predicted success/failure regimes, finite-budget estimation) rather than broad leaderboard comparisons.

4.1 Grammar family (exact, GPT-2)

Grammar	mask TV	state-Φ TV (#states)	+semantic feature	exact Φ
Dyck L=10	0.900	**0.0011** (21)	—	$\sim 10^{-16}$
JSON array L=7	0.510	0.217 (10)	—	$\sim 10^{-15}$
required-fields schema	0.574	0.286 (21)	0.249 (60)	$\sim 10^{-16}$
mini-SQL (WHERE $\subseteq$ SELECT)	0.993	0.322 (25)	**0.034** (72)	$\sim 2 \times 10^{-16}$

Table 1: State-abstraction error across a grammar family (GPT-2, exact, unprompted; §1's Dyck 0.75 is the prompted variant).

The spectrum is continuous and explained by §3: one semantic feature (selected-column set) removes 90% of the SQL error because Φ depends mostly on that set; it fails on required-fields because the residual bias lives in the LM's preferences over concrete values.

4.2 MODEL SCALE

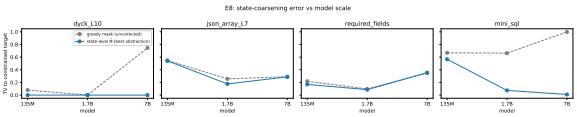

Figure 2: State-coarsening error vs. model scale (SmolLM2-135M, SmolLM2-1.7B, Qwen2.5-7B fp16).

Re-running Table 1 across scales shows the hypothesis "bigger models have smoother Φ" is **false**, and the scale effect is non-monotonic. On Dyck the state correction stays essentially exact at every scale while the mask bias varies wildly ($0.079 \to 0.0003 \to 0.746$); SmolLM2-1.7B shows the largest relative coarsening gain (JSON array $0.257 \to 0.177$, -31%), but at Qwen2.5-7B the gain vanishes ($0.352 \to 0.352$): stronger models have sharper content preferences, increasing within-state variance. Mini-SQL is the exception that proves the rule: the *semantic* abstraction (adding the selected-column set to the state) improves *monotonically* with scale — TV $0.567 \to 0.074 \to 0.009$ from 135M to Qwen2.5-7B (Table 7) — because there the content Φ depends on *is* the semantic feature. Rule: coarsening is adequate exactly when Φ is a function of the (featurized) grammar state — a property of the constraint, not the model.

4.3 ROLLOUT BUDGET VS. PREFIX-LEVEL BASELINE

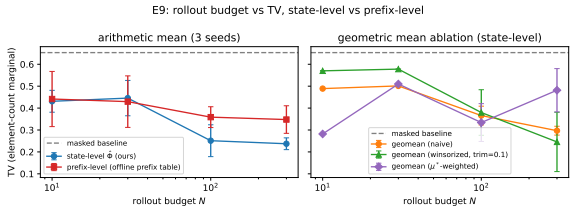

Figure 3: Rollout budget vs. TV: state-level $\hat{\Phi}$ (ours) vs. an offline prefix-level table, estimated from the same N masked rollouts. Left: arithmetic mean with three-seed error bars (half-range). Right: log-domain (geometric-mean) estimator ablation, with three-seed error bars (half-range) at N=100/300.

N	state-level	prefix-level	state table	prefix table
10	0.431 ± 0.050	0.441 ± 0.126	35	106
30	0.446 ± 0.081	0.429 ± 0.118	40	231
100	$\mathbf{0.251} \pm 0.073$	0.359 ± 0.048	43	509
300	$\mathbf{0.237} \pm 0.027$	0.348 ± 0.063	44	1055

Table 2: Element-count marginal TV vs. a 2000-sample IS reference (masked baseline 0.653); real 49k vocabulary, budgeted JSON-array task, SmolLM2-135M. Mean $\pm$ half-range over three seeds; at $N \geq 100$ the state-level table wins in every individual seed. The prefix table is an *offline* frequency table keyed by exact prefix (ASAp's estimand, not its online loop); unseen prefixes fall back to the mean entry. App. G re-runs N=100/300 under a larger protocol (400 eval, 2000 ref): 0.238 ± 0.054 at N=100, same estimator.

Once the rollout budget reaches $N \geq 100$ the state-level table is stably better — it wins in all three seeds at both $N = 100$ and $N = 300$ — with a table that saturates at $\sim$44 buckets, while the prefix table grows linearly with N and degrades to a default on the mostly-unseen evaluation prefixes; at $N \leq 30$ the two are within noise. Decode-time cost of the correction is a lookup — zero additional forward passes.

4.4 FULL-SCALE LATENCY AND QUALITY (E11)

We measure end-to-end cost and downstream quality of the correction on a budgeted JSON task at three scales (budget 48, 200 eval samples; $\hat{\Phi}$ table estimated once from $N{=}100$ masked rollouts; V100, fp16 for 1.7B/8B; all rows are three-seed means; App. K repeats the task at budget 96, so its accuracies differ).

Model	Method	fwd/sample	sec/sample	accuracy	mean $\log p$
SmolLM2-135M	masked	22.06	4.5	0.986	-14.4
	$\hat{\Phi}$	21.96	7.3	0.949	-12.3
SmolLM2-1.7B	masked	22.08	15.8	0.999	-18.6
	$\hat{\Phi}$	21.53	22.8	0.999	$\mathbf{-5.2}$
Qwen3-8B	masked	21.02	13.4	0.997	-41.4
	$\hat{\Phi}$	21.52	19.6	0.996	-44.7

Table 3: E11 (three-seed means): the correction adds essentially zero decode-time forward passes (the sec/sample gap is reference-implementation overhead; the matched-machinery, interleaved timing of App. E puts the increment within a few percent of zero, output token-identical). At 1.7B accuracy is preserved and $\log p$ improves in every seed; at 8B accuracy is preserved with $\log p$ statistically unchanged — the pattern §4.2 predicts as sharper content preferences push this task toward the boundary; at 135M accuracy drops. AWRS-IS/SMC baselines on the same task: App. K.

4.5 THE PREDICTED BOUNDARY, CONFIRMED: CONTENT-DEPENDENT SCHEMAS (E12, E14)

The identity gives an a priori computable criterion: lookup coarsening stops helping exactly when Φ depends on content outside the state features. We test this prediction on nested and long JSON schemas whose validity depends on generated *values* (E12; budget 96, 200 eval samples per case), and on a supported subset of real JSONSchemaBench schemas (E14; 15 schemas per model, at all three scales).

Model	Task	acc (masked/$\hat{\Phi}$)	$\log p$ (masked/$\hat{\Phi}$)	TV (masked/$\hat{\Phi}$)
SmolLM2-135M	nested	0.762 / 0.403	-109 / -211	0.942 / 1.000
	long	0.925 / 0.778	-38 / -114	0.617 / 0.998
SmolLM2-1.7B	nested	0.989 / 0.665	-108 / -219	0.936 / 1.000
	long	0.992 / 0.871	-20 / -84	0.810 / 0.859
Qwen3-8B	nested	0.996 / 0.979	-100 / -172	0.654 / 1.000
	long	0.998 / 0.981	-47 / -71	0.022 / 0.910

Table 4: E12: on content-dependent schemas the lookup $\hat{\Phi}$ degrades at every scale, exactly as the criterion predicts (135M and 1.7B rows are three-seed means; 8B two-seed means, per-seed direction identical). TV is that of the *output-length marginal* vs. a self-normalized IS reference (the 200 masked samples, exact sequence weights); sequence-level TV is intractable here, and 1.000 means disjoint length support. On Qwen3-8B downstream accuracy is essentially unchanged while log-probability and TV degrade sharply: task-level accuracy can mask a shifted sampling law, so distribution-fidelity metrics must be reported separately.

On real JSONSchemaBench schemas (E14) the aggregate token-level TV is $0.640 \to 0.931$ (135M), $0.607 \to 0.757$ (1.7B), and $0.508 \to 0.696$ (Qwen3-8B, three-seed mean) from masked to $\hat{\Phi}$; two further 135M seeds on an expanded subset reproduce the direction ($0.761 \to 0.937$, $0.771 \to 0.969$). The lookup helps on only 3/15, 6/15, and 3–5/15 schemas — those whose validity is (near-)content-free — while at 1.7B the expanded subset splits evenly (13/27 and 12/27; aggregate TV $0.658 \to 0.681$, $0.707 \to 0.641$): applicability on real schemas *widens with scale*, echoing mini-SQL (§4.2), yet half the schemas stay content-bound. This is the criterion working as designed: it partitions schemas *before any decoding* into those where the free lookup suffices and those needing a content-reading $\hat{\Phi}$ (companion paper).

Estimator ablation (geometric mean). §3 prescribes a μ^*-weighted geometric mean as the optimal table entry, but under sampled estimation none of its realizations beats the plain arithmetic mean (at $N{=}100$: TV 0.238 ± 0.054 arithmetic vs 0.334–0.380 for the log-domain variants): at tens of visits per bucket the weighted variants concentrate on a few low-mass rollouts and add more variance than the bias they remove — a bias–variance trade, which is why TAB-Ω and every serving run use the arithmetic mean. Realizing the theory's weighting needs an estimator beyond bucket averaging (regression, i.e. a learned $\hat{\Phi}$ head), developed in the companion paper. Full per-variant, per-seed numbers: App. G.

4.6 Interaction with speculative decoding (E18)

Nie et al. (2026) prove an impossibility: any speculative decoder whose verification target has only *local mask access* samples μ_{proj}, not μ^*, under strict Leviathan rejection — no drafter can help. We sidestep the theorem rather than contradict it: moving the verification target itself to $q(a \mid y) \propto p_V(a \mid y)\mathbf{1}[\text{legal}]\,\hat{\Phi}(\sigma(ya))$ voids the local-mask premise at zero extra forwards, since $\hat{\Phi}$ is the offline table (App. I checks both limits by enumeration: masked target $= \mu^V_{\text{proj}}$, exact-Φ target $= \mu^*$, each to 10^{-15}). The correction thus composes mechanically: a masked drafter (SmolLM2-135M) proposes chunks of 4 tokens that a verifier (SmolLM2-360M) accepts or resamples under its own masked (optionally $\hat{\Phi}$-weighted, or lenience-relaxed) target; full grid on three budgeted JSON tasks (3 cases $\times$ 3 seeds $\times$ 32 eval samples, budget 96).

Task	Sampler	TV vs μ^*	mean log p	score	verifier fwd/token
array	masked	0.653	-34.2	0.769	1.000
	spec-strict	0.622	-32.9	0.769	**0.262**
	spec-$\hat{\Phi}$	0.716	-36.8	0.744	0.289
long	masked	0.593	-29.6	0.983	1.000
	spec-strict	0.625	-29.3	0.977	**0.225**
	spec-$\hat{\Phi}$	0.968	-87.6	0.847	0.277
nested	masked	0.956	-107.8	0.902	1.000
	spec-strict	0.970	-110.6	0.900	**0.263**
	spec-$\hat{\Phi}$	1.000	-185.6	0.711	0.360

Table 5: E18 (32-draw grid): strict speculative acceptance matches the masked sampler's quality with 3.8–4.5$\times$ fewer verifier forwards per token (3.6–4.5$\times$ on the higher-sample E29d grid quoted in the abstract), while the $\hat{\Phi}$-weighted verifier reproduces the content-dependence failure of §4.5 inside the speculative loop — on nested schemas it is also the *slowest* arm (acceptance 0.75 vs 0.90), costing both quality and speed. Aggregates over 9 (case, seed) blocks per task.

The speed-up rows are speculation against the *uncorrected* masked target: spec-strict reproduces μ^V_{proj} (TV vs μ^* within noise of masked), so its gain is speed, not fidelity; the state-lookup $\hat{\Phi}$ target degrades on all three tasks here, array included, so a usable SPEC-Ω target needs a $\hat{\Phi}$ that passes the §4.5 gate (the exact-Φ limit, App. I, is faithful). Higher-precision reruns (128 eval draws, $N{=}256$) and an independent replication at $N{=}48$ reproduce every reading above the 32-draw noise floor, and lenience degrades quality monotonically (App. J).

Deployment scale, with wall clock (E29/E29c). Replicating the grid on a V100 with the deployment-scale pair (135M drafter, *1.7B* verifier, fp16) confirms the forward-count picture is unchanged by moving the verification target, with the corrected target paying the predicted 2–8-point acceptance price (App. I). A first, naive implementation was *slower* than masked end-to-end; after adding KV-cache reuse and precompiled lookup tensors — certified by an equivalence test that the emitted distribution is unchanged (max probability gap $5 \cdot 10^{-12}$) — spec-strict runs $2.0/3.0/1.7\times$ faster than masked. Latency is implementation-dependent; the acceptance-price claim is not (App. J).

Error bars on the speedup (E29d). Because two runs per cell cannot separate a speedup from timing noise, we repeated the optimized grid at $6\times$ the sample size (72 records, `results/e29d_spec_omega_gpu_fast_grid.json`). The speedups survive with

room to spare: masked takes $20.8\pm2.5/49.5\pm3.4/31.2\pm3.3$ s (array/long/nested) against spec-strict $11.1\pm0.4/14.9\pm0.2/17.1\pm0.7$ s ($1.9/3.3/1.8\times$) and spec-$\hat{\Phi}$ $12.3\pm1.4/18.5\pm3.0/20.0\pm4.3$ s ($1.7/2.7/1.6\times$). The forward-count and quality picture is reproduced at the larger sample: verifier forwards per token $0.235\pm0.008/0.223\pm0.006/0.281\pm0.010$ for spec-strict (3.6–$4.5\times$ fewer than one per token) at task score $0.986/0.997/0.995$ against masked's $0.979/1.000/0.984$ — and spec-$\hat{\Phi}$ still degrades on nested JSON (TV 1.000), the same non-lumpable boundary as §4.5, now with seed variance attached.

A second latency tier (sparse legal-set steps, adaptive draft length; E31) is certified distribution-safe but its timing gain is within between-run machine drift, so we report it as unresolved (App. J).

4.7 The production grammar engine, unmodified (E32)

All experiments above drive our own DFA machinery. E32 instead lands the corrector on `xgrammar` — the default grammar backend of vLLM and SGLang — without modifying the engine (`results/e32/`, package `phicorrect/`). The engine does not expose its parser state, but the next-token bitmask it already computes every step is a projection of that state, so hashing the bitmask bytes gives a legitimate coarsening σ in the sense of §3, available in *any* engine at zero extra grammar work; child states for $\hat{\Phi}(ya)$ come from `accept_token`/`rollback` probes (no LM forwards). Steps with > 64 allowed tokens are left *wholly* uncorrected (a per-step constant, harmless by §3); in E32 this cap never fired (0 of 1,120–1,654 corrected steps per seed), while skip-unseen fired on 181–415 steps at 135M and every step at 8B. Bitmask hashing is many-to-one over parser configurations; collisions add within-state variance the identity prices, which we have not measured against true parser states.

On SmolLM2-135M-Instruct with a budgeted JSON-array grammar compiled by `xgrammar` (full 49,152-token vocabulary, budget 24, 80 calibration rollouts, 150 evaluation samples, three seeds; `results/e32/`), every output under both samplers terminates grammatically and the correction moves the output law toward the self-normalized IS reference on two of three seeds (token-length TV $0.531 \to 0.307$, $0.572 \to 0.605$, $0.526 \to 0.377$; element-count $0.664 \to 0.320$, $0.661 \to 0.685$, $0.722 \to 0.195$). A wider prompt$\times$seed grid on the same engine (E37, `results/e37/`; SmolLM2 135M/360M/1.7B, four prompts, three seeds) shows the same picture in aggregate (e.g. 1.7B seed 0 mean TV $0.491 \to 0.305$) with the residual failure the identity predicts: on the one prompt where masking is already nearly unbiased (TV 0.005) the table's finite-sample noise makes the correction worse (TV 0.335) — the MC-variance term of §3 — and the calibration ESS ($\approx$ 4–15 of 80) flags exactly those cells. On Qwen3-8B the protocol gives TV $0.000 \to 0.000$ on all seeds: the model emits one completion essentially deterministically, so $\mu_{\mathrm{proj}} = \mu^*$ (the E32-scale counterpart of §4.2). Serving cost (E36, `results/e36/`): with the per-(state, budget) cache shared across requests, corrected decoding on a V100 reaches 35.6 vs. masking's 36.1 tok/s on SmolLM2-1.7B ($\approx 1\%$ overhead), matcher probes falling $1060 \to 37$ over 30 requests. The corrector thus attaches to vLLM/SGLang through their existing logits-processor hook with zero extra LM forwards.

5 A unified view: every corrector is a Φ-estimator

The Doob transform makes the design space one-dimensional: every distribution-correcting decoder is characterized by how it estimates (or implicitly reweights by) Φ, and identity (1) prices any explicit $\hat{\Phi}$. Table 6 organizes existing methods on the two induced axes (cost–bias frontier: Figure 4, App. B); this paper sits at its zero-extra-inference end.

The "bias vs. extra inference" trade-off is thus a theorem-shaped budget: the only free lunch is coarsening along directions where $\log \Phi$ is (μ^*-a.s.) constant; non-lumpability (§4.5) is fixed only by extra inference or a content-reading $\hat{\Phi}$.

6 Related work

Grammar-aligned decoding (Park et al., 2024) learns expected grammaticality per exact prefix — our baseline in §4.3 — with follow-up engine work (Park et al., 2025). Nie et al. (2026) prove masking samples μ_{proj} and introduce the Φ view, evaluating estimators on enumerable languages

Method	implicit/explicit $\hat{\Phi}$	extra inference	generalizes	residual bias
masking	$\hat{\Phi} \equiv 1$	none	—	full Jensen gap of $1/\Phi$
GAD/ASAp (Park et al., 2024)	per-prefix MC	repeated regeneration	no	$\to 0$ on visited prefixes
lookahead (Nakshatri et al., 2024)	k-step MC rollout	k rollouts per token	—	MC variance of $\hat{\Phi}$
AWRS (Lipkin et al., 2025) / CARS (Parys et al., 2025)	implicit (adaptive rejection)	extra fwd per reject	—	exact
SMC / P-GCD (Zhao et al., 2024; Dang et al., 2026; Alpay and Kilictas, 2026)	learned twist + particle wts.	K particles	via twist net	$\to 0$ as $K \to \infty$
MCMC (Anaya Gonzalez et al., 2025)	implicit (stationary reweighting)	extra fwd per MC step	—	$\to 0$ as chain mixes
DCCD (Reddy et al., 2026)	implicit (draft cond.)	one full draft pass	—	uncharacterized
state lookup (ours)	$f(\sigma(y))$, offline	none	via states	within-state $\mathrm{Var}[\log \Phi]$
learned head (Özgü et al., 2026); companion	$g_\theta(h(y), \sigma(y))$	none (reuses h)	via features	regression error

Table 6: Distribution-correcting decoders as Φ-estimators. "Generalizes" = whether the estimate transfers to prefixes never visited during estimation. For explicit $\hat{\Phi}$, identity (1) gives the exact KL cost; for implicit reweighting schemes the cost is paid in inference instead.

only. AWRS (Lipkin et al., 2025) and CARS (Parys et al., 2025) (exact adaptive rejection) trade extra forward passes for exactness — complementary Pareto points, as are SMC decoders with tensorized-automaton proposals (Dang et al., 2026) or learned twists (Zhao et al., 2024; Alpay and Kilictas, 2026), of which our state lookup is the zero-particle limit. DCCD (Reddy et al., 2026) conditions on an unconstrained draft (an extra pass); DISC (Ye et al., 2025) is asymptotically unbiased via dynamic importance sampling (extra verification per sample). Concurrently, Özgü et al. (2026) (SHIM) train an offline logit correction on (parser state, lexer state, candidate token) in Syncode — our state-availability observation with a learned head in place of a training-free table, and no error identity or applicability criterion. Such a head is a feature-function $\hat{\Phi}$, so identity (1) prices it unchanged: its residual is the within-feature variance of $\log \Phi$, and content outside the features is as invisible to it as to our table — their own Spider table shows it post hoc: dropping the parser/lexer features costs $24\times$ in KL on a database whose column names diverge only many tokens in. Biagiola et al. (2026) document constrainer-induced distortion in code; Anaya Gonzalez et al. (2025) correct the constrained law with whole-sequence MCMC restarts; Pachet and Roy (2026) show the same bias in music. Lee et al. (2026) find much of the structured-output accuracy loss enters at the *prompt* (we price the decoder-side residual); Parikh (2026) show a JSON register alone collapses answer diversity across 44 models; Ray (2026) quantify the tax for sub-3B on-device models (19.7%→11.0%), where only a zero-extra-forward correction is affordable. Alpay and Senturk (2026) also cast the grammar-conditional sampler as a Doob h-transform, proving sharp distortion results for masking-engine complexity; we price arbitrary approximate h ($\hat{\Phi}$). Lookahead decoding (Nakshatri et al., 2024) estimates Φ by per-token rollouts. TruncProof (Kato and Tarashima, 2026) and GRID (Arjmandi, 2026) provide state-keyed (parser state, budget) infrastructure our correction layers onto; engine work (Li et al., 2026) makes masking cheap, orthogonal to correcting the law.

7 Limitations

External validity. E14's JSONSchemaBench over-represents free-form fields (only 7.9% of object properties are `enum`-typed), so the lookup helps on a minority. We *conjecture* that strict-schema and function-calling traffic (OpenAI, 2024) (`enum`/boolean arguments, tool selection) is mostly applicable, but have not run BFCL, xLAM or Spider. The scale sweep (§4.2) uses Qwen2.5-7B while §4.4–4.5 use Qwen3-8B, so the vanishing scale gain is not replicated within one family.

Empirical scope. Finite/toy languages for exact evaluation; the single §4.3 task is noisy (three-seed error bars); the log-domain estimator is not yet robust at small samples; only AWRS-IS/SMC are re-run on our task (E22); E12 is three-seed at 135M/1.7B, two-seed at 8B. The numerical gate $(\widehat{\mathrm{Var}}, \tau)$ runs end-to-end only in the E24 pilot; for E12/E14 the boundary is predicted from content-dependence and confirmed by ΔTV, not by tabulated gate values.

8 Conclusion

Masking bias is large, real, and does not shrink with scale. Where the automaton state determines future validity, a rollout-estimated state-level $\hat{\Phi}$ table removes most of it at zero per-step forward cost and beats prefix-level tables once $N \geq 100$; where validity depends on content, the identity predicts *in advance* where the lookup stops helping (E12/E14) and prescribes a content-aware $\hat{\Phi}$. This paper makes "when do cheap corrections work?" computable, and solves inside that boundary.

LLM USAGE STATEMENT

In accordance with ICLR 2027's policy on the use of large language models, we disclose the following.

Required disclosure. Generative AI tools (Claude) were used in this work to assist with the implementation and execution of experiments — including monitoring long-running experiment jobs on rented compute infrastructure, and periodically syncing and logging intermediate results — and to assist with the interpretation of results, including identifying and correcting discrepancies between drafted claims and the underlying per-seed data during manuscript preparation. The mathematical identity (§3), its derivation (Appendix A), the state-coarsening method, and the overall experimental design were conceived by the authors. We have not used generative AI tools to generate synthetic data reported as experimental results, to design the core theoretical contribution, or to fabricate citations or experimental numbers.

Recommended disclosure. Generative AI tools were additionally used for drafting and editing portions of the manuscript text, formatting, and language polishing.

Verification. The mathematical identity was verified numerically to machine precision ($\sim 10^{-16}$) across multiple grammars, as reported in §3. All experimental results reported in this paper were generated by code written and executed under the authors' direction; all reported numbers were checked against the underlying raw results (retained in the accompanying repository) before inclusion in the manuscript. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with AI assistance.

REFERENCES

K. Park, J. Wang, T. Berg-Kirkpatrick, N. Polikarpova, L. D'Antoni. Grammar-Aligned Decoding. *NeurIPS*, 2024.

W. Nie et al. Future Validity is the Missing Statistic: From Impossibility to Φ-Estimation for Grammar-Faithful Speculative Decoding. *arXiv:2605.07698*, 2026.

I. Özgü, Y. Wu, G. Van den Broeck, M. Kim. The Parser Already Knows: Lightweight Bias Correction in Constrained Decoding. *arXiv:2608.10137*, 2026.

P. Parys, S. Vaidya, T. Berg-Kirkpatrick, L. D'Antoni. Constrained Adaptive Rejection Sampling. *arXiv:2510.01902*, 2025.

B. Lipkin, B. LeBrun, J. H. Vigly, J. Loula, D. R. MacIver, et al. Fast Controlled Generation from Language Models with Adaptive Weighted Rejection Sampling. *COLM*, 2025.

Z. Liu, F. Wu, B. Summa, Z. Zheng. Adaptive Multilevel Twisted Sequential Monte Carlo for Rare Events Estimation in Language Models. *arXiv:2608.21736*, 2026.

M. Dang, L. Song, H. Zhang, J. Zhao, G. Van den Broeck, S. Ermon. Mitigating Bias in Locally Constrained Decoding via Tractable Proposals. *arXiv:2606.01926*, 2026.

K. Park, T. Zhou, L. D'Antoni. Flexible and Efficient Grammar-Constrained Decoding. *ICML*, 2025.

A. Reddy, T. T. Walker, J. S. Ide, A. S. Bedi. The Hidden Cost of Structured Generation in LLMs: Draft-Conditioned Constrained Decoding. *arXiv:2603.03305*, 2026.

M. Biagiola, J. G. Cesario, L. Di Grazia, G. Zakhour, G. Salvaneschi. The Alignment Problem in Constrained Code Generation. *arXiv:2606.21619*, 2026.

Y. Kato, S. Tarashima. TruncProof: A Guardrail for LLM-based JSON Generation under Token-Length Constraints. *arXiv:2605.13076*, 2026.

M. Arjmandi. GRID: Grammar-Railed Decoding for Enterprise SQL Generation. *arXiv:2607.11951*, 2026.

H. Ye, H. Jain, C. You, A. T. Suresh, et al. Efficient and Asymptotically Unbiased Constrained Decoding for Large Language Models. *AISTATS*, 2025.

L. Li, Y. Dong, G. Wang, Z. Xu, A. Jiang, T. Chen. XGrammar-2: Efficient Dynamic Structured Generation Engine for Agentic LLMs. *arXiv:2601.04426*, 2026.

F. Alpay, B. Kilictas. LatticeBridge: Rare-Event Sequential Inference for Faithful Structured Sequence Synthesis. *arXiv:2606.11203*, 2026.

S. Zhao, R. Brekelmans, A. Makhzani, R. Grosse. Probabilistic Inference in Language Models via Twisted Sequential Monte Carlo. *ICML*, 2024.

N. Nakshatri, S. Roy, R. Das, S. Chaidaroon, L. Boytsov, R. Gangadharaiah. Constrained Decoding with Speculative Lookaheads. *arXiv:2412.10418*, 2024.

F. Alpay, B. Senturk. Attention Meets Reachability: Structural Equivalence and Efficiency in Grammar-Constrained LLM Decoding. *arXiv:2603.05540*, 2026.

E. Anaya Gonzalez, S. Vaidya, K. Park, R. Ji, T. Berg-Kirkpatrick, L. D'Antoni. Constrained Sampling for Language Models Should Be Easy: An MCMC Perspective. *arXiv:2506.05754*, 2025.

F. Pachet, P. Roy. Hidden Biases in Conditioning Autoregressive Models. *arXiv:2604.07855*, 2026.

I. Y. Lee, L. D'Antoni, T. Berg-Kirkpatrick. The Format Tax. *arXiv:2604.03616*, 2026.

T. Parikh. Structured Output Collapses Answer Diversity Across 44 Language Models. *arXiv:2607.18476*, 2026.

F. Li, A. M. Zhang, C. Lv. Constraint Tax in Open-Weight LLMs: An Empirical Study of Tool Calling Suppression Under Structured Output Constraints. *arXiv:2606.25605*, 2026.

J. Ray. The Constraint Tax: Measuring Validity-Correctness Tradeoffs in Structured Outputs for Small Language Models. *arXiv:2605.26128*, 2026.

P. Buchholz. Exact and Ordinary Lumpability in Finite Markov Chains. *Journal of Applied Probability*, 31(1):59–75, 1994.

S. Derisavi, H. Hermanns, W. H. Sanders. Optimal State-Space Lumping in Markov Chains. *Information Processing Letters*, 87(6):309–315, 2003.

M. Ainsworth, J. T. Oden. *A Posteriori Error Estimation in Finite Element Analysis*. Wiley, 2000.

OpenAI. Introducing Structured Outputs in the API. https://openai.com/index/introducing-structured-outputs-in-the-api/, 2024.

A Derivation of the error identity

Write $q(a \mid y) = p(a \mid y)\hat{\Phi}(ya)/Z_q(y)$ with $Z_q(y) = \sum_a p(a \mid y)\hat{\Phi}(ya)$, and recall $\mu^*(a \mid y) = p(a \mid y)\Phi(ya)/\Phi(y)$. By the chain rule for KL divergence over decoding steps,

$$\mathrm{KL}(\mu^* \| q) = \sum_t \mathbb{E}_{y_{<t} \sim \mu^*} \, \mathrm{KL}\big(\mu^*(\cdot \mid y_{<t}) \,\|\, q(\cdot \mid y_{<t})\big).$$

Fix a prefix $y = y_{<t}$ and let $r(ya) = \hat{\Phi}(ya)/\Phi(ya)$. The per-step term is

$$\mathbb{E}_{a \sim \mu^*(\cdot \mid y)} \log \frac{\mu^*(a \mid y)}{q(a \mid y)} = \mathbb{E}_a \log \frac{\Phi(ya)\, Z_q(y)}{\Phi(y)\, \hat{\Phi}(ya)} = \log Z_q(y) - \log \Phi(y) - \mathbb{E}_a \log r(ya).$$

The normalizer collapses onto a μ^*-expectation of r:

$$Z_q(y) = \sum_a p(a \mid y)\, \Phi(ya)\, r(ya) = \Phi(y) \sum_a \mu^*(a \mid y)\, r(ya) = \Phi(y)\, \mathbb{E}_{a \sim \mu^*(\cdot \mid y)}[r(ya)],$$

so the per-step term equals $\log \mathbb{E}_{a\sim\mu^*}[r] - \mathbb{E}_{a\sim\mu^*}[\log r]$ — the Jensen gap of r under $\mu^*(\cdot \mid y)$ — which is identity (1). It is nonnegative by Jensen's inequality, zero iff r is μ^*-a.s. constant on the step (whence consequence (a): per-step constant rescalings of $\hat{\Phi}$ are free), and blind to branches with $\mu^*(a \mid y) = 0$ (consequence (b)). A second-order Taylor expansion of the Jensen gap in $\log r$ around its mean gives $\frac{1}{2}\mathrm{Var}_{\mu^*}[\log r]$ per step; for a state-lookup $\hat{\Phi}(y) = f(\sigma(y))$, minimizing over f per bucket yields $\log f(s) = \mathbb{E}_{\mu^*}[\log \Phi \mid \sigma = s]$ (the μ^*-weighted geometric mean) with residual equal to the μ^*-weighted within-state variance of $\log \Phi$ (consequence (c)).

B THE COST–BIAS FRONTIER

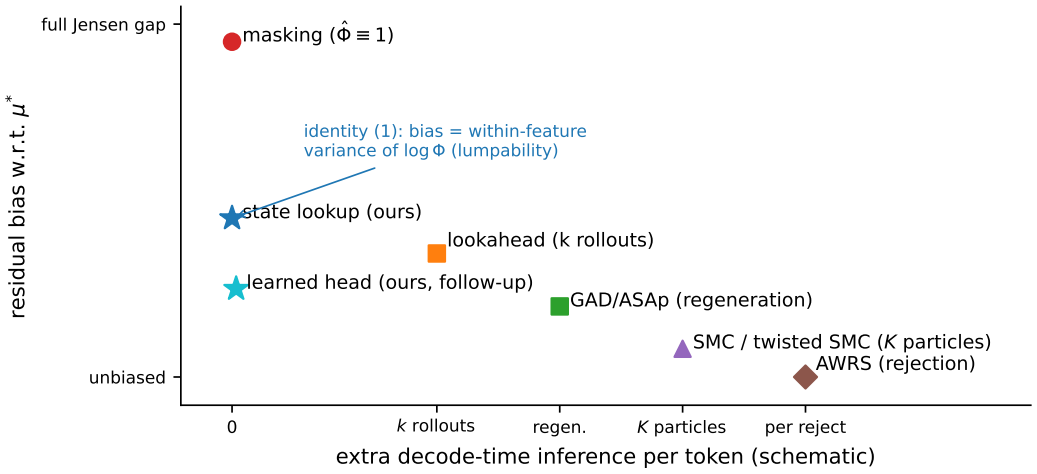

Figure 4: The design space of distribution-correcting decoders (schematic placement, not same-task measurements): extra decode-time inference per token vs. residual bias w.r.t. μ^*. Methods trading forward passes for bias move right and down; the zero-cost column pays instead the Jensen gap of its $\hat{\Phi}$, which identity (1) prices and lumpability (Fig. 1) predicts.

C SCOPE AGAINST PRIOR Φ WORK, AND CITATION GUIDANCE

Nie et al. (2026) analyzed the two endpoints of our identity ($\hat{\Phi} = \Phi$ and $\hat{\Phi} \equiv 1$) on enumerable languages; the approximate $\hat{\Phi}$ strictly between — the object of this paper — is priced only here. Nie et al. (2026) named the Φ viewpoint — masked decoding samples μ_{proj}, Φ is the corrective statistic — and analyzed its two endpoints (exact Φ, and none) on enumerable languages. The works are complementary: theirs the exact-Φ regime on enumerable languages, ours the approximation gap at real vocabularies and model scales. Cite Nie et al. (2026) for the Φ viewpoint, the μ_{proj} characterization, or exact-Φ/enumerable-language guarantees; cite this paper for the cost of an *approximate* $\hat{\Phi}$, zero-extra-forward state coarsening, or a computable applicability boundary.

D IN WHAT SENSE THE CRITERION IS AVAILABLE IN ADVANCE

The criterion of §3 is the quantity $\mathbb{E}\,\mathrm{Var}_{\mu^*}[\log \Phi \mid \sigma]$, and it vanishes exactly when the state map is lumpable for $\log \Phi$. On a finite chain given explicitly, that is a decidable property, checkable in $O(m \log n)$ time in the number of transitions (Buchholz, 1994; Derisavi et al., 2003). Here the kernel is the language model, so the chain is never available explicitly: we estimate the variance from the same small batch of masked rollouts that builds the table and compare it against a threshold. What one holds before decoding is therefore a computable *estimate* carrying Monte-Carlo error — in status closer to an a posteriori error indicator driving adaptive refinement (Ainsworth and Oden, 2000) than to a decision procedure. The distinction we draw against heuristic guidance survives this

weakening: the indicator is a functional of the constraint and the model, evaluated on the schema at hand rather than inherited from past failures.

E DECODE-TIME INCREMENT ON MATCHED MACHINERY (E19K)

The *forward-pass* claim of §4.4 is exact; the wall-clock claim is implementation-dependent, and the reference decoder rebuilds the allowed-token dictionary and the candidate feature rows in Python at every step, which the sec/sample column charges to the correction. E19k removes that artefact and makes the comparison fair: the lookup arm and the masked baseline read one set of per-state candidate tensors precompiled once from the static DFA; $\hat{\Phi}$ becomes a dense $[\text{states} \times (\text{remaining budget}{+}1)]$ tensor read by a single gather (max. deviation 1.4×10^{-8} against the dictionary it replaces); the arms are timed interleaved sample by sample with rotating order, so no arm is charged for a cold cache; and time is normalized per generated token, since the arms emit sequences of different lengths. Output is token-identical to the reference decoder in every seed.

Over three seeds per configuration (eight samples each) the lookup increment over the masked baseline is $+3.1 \pm 3.1\%$ (array JSON, 135M, CPU), $-2.7 \pm 7.6\%$ (nested, 135M, CPU), $+7.7 \pm 12.5\%$ (long, 135M, CPU), $+1.2 \pm 3.9\%$ (nested, 360M, CPU) and $+4.4 \pm 8.2\%$ (nested, Qwen3-8B, V100). The spread is comparable to the mean, and on the shared rented GPU the absolute per-token time varies by up to $2\times$ across seeds (the 1.7B row, $-18.3 \pm 6.4\%$, is not interpretable as a speedup), so the defensible reading is that this protocol resolves no stable increment beyond a few percent — not that the increment is zero, and not a substitute for a load test with continuous batching, where tail latency rather than mean per-token time is what a serving system pays. A fused implementation could plausibly do better: a grammar backend already computes the successor state while building the mask and could emit it for the $\hat{\Phi}$ gather instead of having it recomputed. We have not built that; these numbers are from our PyTorch decoder at batch size one, and they change no distributional claim in this paper.

F MINI-SQL ACROSS SCALES

Numbers quoted in §4.2, in full.

Model	mask TV	syntactic state	+semantic (col. set)
SmolLM2-135M	0.668	0.642	0.567
SmolLM2-1.7B	0.663	0.414	0.074
Qwen2.5-7B	0.998	0.397	**0.009**

Table 7: Mini-SQL (WHERE $\subseteq$ SELECT) across scales: the semantic abstraction's gain grows monotonically with scale.

G ESTIMATOR ABLATION: PER-SEED NUMBERS

Full per-seed detail behind the summary in §4.5.

§3 predicts the optimal table entry is a μ^*-weighted geometric mean. A naive *unweighted* log-domain average did not beat the arithmetic mean under sampled estimation: over three seeds, element-marginal TV at $N{=}100$ is 0.366 ± 0.047 for the unweighted geometric mean versus 0.238 ± 0.054 for the arithmetic mean (per-seed $0.335/0.343/0.420$ vs $0.191/0.296/0.226$), so the ordering holds in every seed. A winsorized variant (trim 0.1) is the least stable of the estimators, and three seeds locate the instability precisely: its seed-to-seed spread (0.380 ± 0.111 at $N{=}100$, 0.246 ± 0.137 at $N{=}300$) is two to three times that of any other variant ($\text{sd} \approx 0.05$), large enough that the apparent *degradation* with budget visible in a single seed (0.253 at $N{=}100$ rising to 0.394 at $N{=}300$) reverses under averaging. Trimming discards exactly the low-mass rollouts that carry the information, so what it buys in outlier robustness it loses in variance. Even the theory-prescribed μ^*-weighting, implemented via self-normalized trajectory importance weights within each bucket, does not recover the gap (three seeds: 0.334 ± 0.087 at $N{=}100$ and 0.481 ± 0.106 at $N{=}300$, both

above the arithmetic mean's 0.238 and 0.246, and unlike the arithmetic mean it gets *worse* with budget): with tens of visits per bucket the importance weights degenerate onto a few rollouts, inflating variance faster than the weighting removes bias. A further consequence, visible only once these variants have several seeds each: the state table's advantage over the prefix table (§4.3) is a property of the arithmetic estimator, not of state coarsening alone. With the arithmetic mean the state table wins in every seed we have; across the log-domain variants (unweighted, winsorized, μ^*-weighted) it wins in only 4 of 10 seed $\times$ budget points, because the estimator noise that these variants add to each bucket is larger than the coarsening error it is meant to remove. Small-sample log-domain averaging is fragile to single low-mass rollouts; the theory's weighting matters, but realizing it requires estimators beyond bucket averaging (e.g. regression — a learned $\hat{\Phi}$ head), developed in the companion paper.

H TAB-Ω: THE DEPLOYABLE PIPELINE, AND THE EXACT OPTIMUM OF THE STATE TABLE (E24)

The body of the paper prices the state-level lookup; here we (i) package it as a serving algorithm whose offline phase is a by-product of the masked sampler itself, and (ii) replace the second-order prescription of §3(c) — the weighted geometric mean — by the *exact* optimum, which turns out to be a convex program.

The algorithm (TAB-Ω). *Offline:* sample N pilot rollouts from the masked sampler μ_{proj}. Each rollout records the legal mass $Z_t = \sum_{a \text{ legal}} p(a \mid y_{<t})$ at every step, so a single backward pass yields, at every visited step, $R_t = \prod_{s>t} Z_s$ — an unbiased estimate of $\Phi(y_{\le t})$ under μ_{proj} (the same recursion as the exact backward DP, sampled). Aggregate by automaton state: the table entry is the (optionally self-normalized $w(y) = \prod_t Z_t$-reweighted, to approximate the μ^*-weighting the identity asks for) within-state mean of R. *Gate:* certify a state only if its pilot within-state $\mathrm{Var}[\log R]$ is below a threshold τ; for uncertified states report the residual bound $\frac{1}{2}\mathrm{Var}$ and do not claim the correction — the residual is the budget line of the content-aware learned $\hat{\Phi}$ of the companion paper. The gate statistic is conservative: by the law of total variance, $\mathrm{Var}[\log R \mid s] = \mathbb{E}[\mathrm{Var}(\log R \mid y) \mid s] + \mathrm{Var}[\log \Phi \mid s]$, i.e. the single-rollout Monte Carlo variance of R plus the coarsening variance the identity prices. The gate therefore never certifies a non-lumpable state by accident, but it can refuse a lumpable one whose R is noisy or rarely visited; on Dyck, where the exact geometric-mean table is near-exact, the $7/20$ uncertified states in Table 8 are of this kind. Separating the two terms (repeated rollouts per prefix, or the exact backward DP where available) is left open. *Serve:* $\mathrm{logit}(a) \mathrel{+}= \log \hat{\Phi}(\sigma(ya))$ over the masked support. Because the automaton transition $\sigma(ya)$ is computed by the masker anyway, the correction is token-dependent for free and costs zero extra forward passes.

The exact optimum is a convex program. Parametrize the table in log space, $\lambda_s = \log f(s)$, and let $q_\lambda(a \mid y) \propto p(a \mid y)\mathbf{1}[\text{legal}]e^{\lambda_{\sigma(ya)}}$.

Theorem 1. *Let $N_s(y)$ count the transitions of y into state s. Then*

$$\mathrm{KL}(\mu^* \| q_\lambda) = \text{const} - \sum_s \lambda_s \, \mathbb{E}_{\mu^*}[N_s] + \sum_t \mathbb{E}_{y_{<t} \sim \mu^*} \log \sum_{a \text{ legal}} p(a \mid y_{<t}) \, e^{\lambda_{\sigma(y_{<t} a)}},$$

which is jointly convex in λ (each log term is a log-sum-exp of affine functions). The first-order condition is moment matching: at the optimum, the decoder's expected state-transition counts on μ^* prefixes equal μ^*'s own, $\sum_t \mathbb{E}_{y_{<t} \sim \mu^*} q_\lambda(\sigma(y_{<t} a) = s \mid y_{<t}) = \mathbb{E}_{\mu^*}[N_s]$ for every s. The minimizer is unique modulo the per-step gauge (adding a constant to all states reachable at a fixed step leaves q_λ unchanged — the inertness of §3(a)), and the weighted geometric mean of §3(c) is its second-order approximation.

Proof. Expand $\mathrm{KL}(\mu^* \| q_\lambda) = \sum_t \mathbb{E}_{\mu^*}[\log \mu^*(a_t \mid y_{<t}) - \log q_\lambda(a_t \mid y_{<t})]$ and substitute $\log q_\lambda(a \mid y) = \log p(a \mid y) + \lambda_{\sigma(ya)} - \log Z_q(y; \lambda)$ with $Z_q(y; \lambda) = \sum_{a \text{ legal}} p(a \mid y) e^{\lambda_{\sigma(ya)}}$: the λ-dependent part is exactly the displayed expression. Each $\log Z_q$ is a log-sum-exp of the affine maps $\lambda \mapsto \lambda_{\sigma(ya)} + \log p(a \mid y)$, hence convex; the linear term does not affect convexity. Differentiating in λ_s gives $\partial_s \mathrm{KL} = \sum_t \mathbb{E}_{\mu^*}[q_\lambda(\sigma(y_{<t} a) = s \mid y_{<t}) - \mathbf{1}\{\sigma(y_{\le t}) = s\}]$, which vanishes iff the counts

match. Strict convexity holds transverse to the gauge directions because the log-sum-exp Hessian is positive definite on the span of the non-constant exponent differences; along a per-step constant shift both terms change by the same amount, so KL is invariant and q_λ unchanged. A second-order expansion of the Jensen gap in identity (1) around the optimum recovers $\log f(s) = \mathbb{E}_{\mu^*}[\log \Phi \mid \sigma = s]$. □

Numerical verification (E24, exact enumeration). On two toy grammars with a content-dependent model (stdlib-only, exact prefix trees): Dyck ($L \le 10$, 64 sequences, lumpable) and a required-fields schema (56 sequences, content-dependent). All checks assert-pass: convexity by midpoint tests on random table pairs; gauge invariance to 10^{-15}; gradient norm $\sim 10^{-6}$ at the solved optimum (moment matching); $\mathrm{KL}(\mathrm{opt}) \le \mathrm{KL}(\mathrm{geo}) \le \mathrm{KL}(\mu_{\mathrm{proj}})$; exact Φ reaches μ^* to 10^{-16}.

Grammar	state	$\mathrm{KL}(\mu_{\mathrm{proj}})$	opt	geo(μ^*)	pilot N=2048	certified
Dyck	(d, ℓ)	0.387	2.5×10^{-5}	5.7×10^{-5}	0.015	13/20
req-fields	syntactic	1.233	1.168	1.192	1.177	3/12
req-fields	+key set	1.233	0.348	0.373	0.563	3/21

Table 8: E24 (5 seeds, worst-seed pilot KL in nats). On the lumpable grammar the pilot table removes 96% of the masking KL and the gate certifies the visited-mass states; on the content-dependent grammar the syntactic gate refuses (few certified states, $< 5\%$ reduction — exactly the E12/E14 boundary, now executed as a gate rather than read off a table) and the semantic refinement recovers most of the headroom. The gap between the exact convex optimum and the geometric mean is small (3×10^{-5}–2.6×10^{-2} nats), quantifying how good the second-order prescription is.

I SPEC-Ω: GRAMMAR-FAITHFUL SPECULATIVE VERIFICATION, EXACTLY (E25)

Speculative decoding's exactness guarantee holds only w.r.t. the *verification target*: under a grammar mask the standard target is the verifier's masked local law μ^V_{proj}, so even strict Leviathan rejection reproduces a biased distribution and no drafter can help (the impossibility of Nie et al. (2026)). SPEC-Ω moves *only* the accept/reject target to $q(a \mid y) \propto p_V(a \mid y)\mathbf{1}[\mathrm{legal}]\,\hat{\Phi}(\sigma(ya))$, with $\hat{\Phi}$ the TAB-Ω pilot table of App. H — the offline pilot and gate phases are reused verbatim (the table is estimated once from masked *verifier* rollouts), serving adds one lookup per candidate token inside the acceptance test, and the draft mechanism (hence the speedup machinery) is untouched.

On the toy grammars of E24 the induced sequence law of lenient speculative decoding is computed *exactly*: per step, a draft a from the masked drafter d is accepted w.p. $\min(1, Lq(a)/d(a))$ and rejections resample from the residual $\propto \max(0, q - \min(d, Lq))$, so the induced next-token law is $\pi_L(a) = \min(d(a), Lq(a)) + \beta\, r(a)$, which equals q at L=1 and interpolates toward the drafter as L grows; composing π_L over the prefix tree gives exact KL and acceptance numbers. All checks assert-pass: strict masked-target speculation equals μ^V_{proj} to 10^{-15} (regardless of the drafter); strict exact-Φ-target speculation equals μ^* to 10^{-15}; strict pilot-table-target speculation equals the table's ancestral law to 10^{-9} — speculation neither helps nor hurts the distribution, it only buys speed, so the whole E24 error budget (optimum, geometric gap, gate residual) transfers unchanged.

The lenience Pareto frontier, exactly. Sweeping $L \in \{1, 1.2, 1.5, 2, 3, 5\}$ prices the knob practice sets by feel: on Dyck the $\hat{\Phi}$-target curve runs from $(\mathrm{acc}\,0.824, \mathrm{KL}\,0.015)$ to $(0.965, 0.513)$ while the masked-target curve runs from $(0.951, 0.387)$ to $(0.999, 0.542)$; at every lenience the $\hat{\Phi}$ target dominates the masked target on the KL-to-μ^* axis (asserted), and acceptance is monotone in L for both. Lenience buys acceptance by leaking drafter bias into the output — the exact curve makes the exchange rate explicit, so a deployment can pick the operating point with its eyes open.

Grammar	strict target (L=1)	KL vs μ^*	acceptance
Dyck	masked	0.387	0.951
	exact Φ	10^{-15}	0.811
	pilot table (N=2048)	0.015	0.824
req-fields	masked	1.233	0.930
(+key set)	exact Φ	10^{-16}	0.805
	pilot table	0.569	0.847

Table 9: E25 strict speculation, exact numbers. Moving the verification target from masked to the pilot table costs 8–13 acceptance points (the corrected target is farther from the masked drafter — the honest price of faithfulness), while cutting KL to μ^* by 96% (Dyck) and 54% (req-fields with the semantic state).

J SPECULATIVE-DECODING REPLICATIONS AND LATENCY TIERS (E18/E29/E31)

Higher-precision reruns of E18. All three tasks rerun at 128 eval draws and N=256 $\hat{\Phi}$-rollouts per block (same 9 blocks) confirm the body's picture above the 32-draw noise floor. On array, spec-strict matches the masked sampler (TV 0.776 vs 0.757, $\log p$ -33.6 vs -33.6, score 0.766 vs 0.771) at 0.26 verifier forwards per token, while lenience degrades quality monotonically (TV $0.836 \to 0.877$, $\log p$ $-36.4 \to -42.4$ for L=2 $\to$ 8). On long, spec-strict again matches masked (score 0.981 vs 0.981, $\log p$ -29.0 vs -29.2) at 0.23 verifier forwards per token, and on nested likewise (score 0.906 vs 0.907, $\log p$ -108.8 vs -109.3) at 0.26 — while the state-lookup-$\hat{\Phi}$-weighted verifier collapses exactly where the criterion says it must (long: score $0.981 \to 0.733$, $\log p$ $-29 \to -114$; nested: score $0.907 \to 0.603$, $\log p$ $-109 \to -203$), and lenience degrades scores monotonically ($0.979 \to 0.966$ long, $0.893 \to 0.824$ nested for L=2 $\to$ 8).

Low-budget replication. A third, independent replication of the full grid at a *lower* $\hat{\Phi}$ budget (N=48 rollouts per block, 32 eval draws, freshly sampled 9 blocks per task) reproduces every reading, so the conclusions are insensitive to the $\hat{\Phi}$ rollout budget across $N \in \{48, 64, 256\}$: spec-strict again tracks the masked sampler (array TV 0.638 vs 0.660, $\log p$ -32.9 vs -34.2; nested TV 0.970 vs 0.974, $\log p$ -111.2 vs -107.9; long $\log p$ -29.4 vs -29.4) at 0.22–0.26 verifier forwards per token; the state-lookup $\hat{\Phi}$ verifier again collapses on exactly the content-dependent tasks (nested TV 1.000 ± 0.000, $\log p$ -172.0; long TV 0.970 vs 0.579, $\log p$ -84.6 vs -29.4) and is again the slowest speculative arm there (acceptance 0.752 vs 0.900 strict, 0.361 vs 0.261 forwards per token); and lenience again degrades quality monotonically (nested $\log p$ $-117.2 \to -134.9$, score $0.897 \to 0.826$ for L=2 $\to$ 8).

E29: the first deployment-scale run. On a V100-32GB with the deployment-scale pair (135M drafter, 1.7B verifier, fp16, chunk 4, budget 96, 2 blocks $\times$ 8 eval draws, N=32 $\hat{\Phi}$ rollouts; `results/e29_spec_omega_gpu.json`), verifier forwards per token are $0.237/0.225/0.274$ (array/long/nested) for spec-strict and $0.269/0.231/0.302$ for spec-$\hat{\Phi}$, i.e. 3.3–4.4$\times$ fewer verifier passes in both cases, with the corrected target paying the predicted acceptance price (0.903 vs 0.954 array, 0.952 vs 0.970 long, 0.807 vs 0.887 nested). In this first run every speculative arm was *slower* end-to-end than the non-speculative masked sampler (spec-strict $27.1/47.0/31.3$ s vs masked $23.1/45.3/27.4$ s; spec-$\hat{\Phi}$ $44.7/73.2/46.3$ s): the drafter's chunk was generated by 4 sequential single-token Python forwards without KV-cache reuse — while the masked baseline caches — and the $\hat{\Phi}$ arm reassembled its lookup in Python inside every acceptance test.

E29c: the same grid after fixing the implementation. We then gave drafter and verifier a `DynamicCache` (cropped back to the accepted prefix on rejection, so only the uncached suffix is forwarded) and precompiled the DFA's legal set and $\hat{\Phi}$ factors into index tensors keyed by (state, remaining budget). This is a pure implementation change: `experiments/e29b_spec_cache_equiv.py` checks that the sampler emits token-for-token identical sequences with identical acceptance rates and forward counts

(max probability gap $5 \cdot 10^{-12}$, max weight gap 0). Rerunning the grid unchanged (`results/e29c_spec_omega_gpu_fast.json`) reproduces the forward-count and acceptance numbers and reverses the latency verdict: against masked $22.6/44.5/27.1$ s, spec-strict now takes $11.1/14.7/16.2$ s ($2.0/3.0/1.7\times$ faster) and spec-$\hat{\Phi}$ $12.6/15.7/18.6$ s ($1.8/2.8/1.5\times$). The ordering of the two runs is the point: the absence of a speedup was a property of our code, and the only reason the sampler could be optimized at all is that the equivalence test makes the change risk-free with respect to the distribution being measured.

E31: a second latency tier, and its limits. The remaining implementation costs are the full-vocabulary weight construction at each verified step and the fixed draft length. E31 replaces the former with a sparse legal-set step (the normalized legal distribution is invariant to the vocabulary-wide softmax constant, so only |legal| floats leave the device) and makes the draft length adaptive (grow after a fully accepted round, shrink to the accepted prefix after a rejection). An equivalence test (`e31b`) pins what may and may not change: the sparse target matches the dense one to 1.8×10^{-7}, the first-token law is within Monte Carlo resolution, and at fixed chunk the acceptance rate, forwards per token and length match the reference sampler — while the adaptive schedule is allowed to move forwards per token, since that is what it trades against discarded draft compute. Re-timing the grid (`results/e31_spec_fast_grid_partial.json`; the machine expired after 2 of 3 tasks, so this is a partial grid) gives a median wall-clock ratio of 0.95 on the speculative arms — but the *masked* arm, whose code is identical across the two runs, moves by $0.98/0.89$ on the same tasks. This tier is therefore not resolved by our timing protocol: it is at the scale of between-run machine drift, and we report it as such rather than as an additional speedup. The machine-independent quantities are unchanged.

K SAME-TASK AWRS-IS/SMC BASELINES (E22)

We additionally re-implement the *weighting* mechanism of AWRS (Lipkin et al., 2025) and SMC-style decoders (Dang et al., 2026) on the same task and hardware as §4.4, at the 135M and 1.7B scales: "AWRS-IS" is trajectory-level self-normalized importance weighting on the exact allowed mass (the lower-variance best case of AWRS's unbiased weight estimate) — a sequential-IS ablation, *not* AWRS's exact Las-Vegas rejection sampler, whose samples are unweighted and whose cost is paid in extra proposals instead of ESS — and 8-particle SMC with systematic resampling on the same weights (budget 96, 200 eval samples, three-seed means; TV of the length marginal against a shared 2000-sample IS reference).

Model	Method	TV vs ref	accuracy	mean $\log p$	fwd/sample
SmolLM2-135M	masked	0.800	0.974	-11.1	22.4
	state $\hat{\Phi}$	**0.013**	0.980	-3.3	**21.5**
	AWRS-IS	0.037	0.970	-10.6	22.2 (ESS 43/200)
	SMC (K=8)	0.202	0.975	-5.5	170.4
SmolLM2-1.7B	masked	0.937	0.986	-14.6	22.0
	state $\hat{\Phi}$	**0.021**	0.992	-4.3	**21.5**
	AWRS-IS	0.053	0.966	-14.0	22.0 (ESS 6/200)
	SMC (K=8)	0.775	0.992	-12.1	174.2

Table 10: E22 (three-seed means): on this criterion-satisfying task the state lookup dominates — best distribution fidelity at the lowest forward cost. AWRS-IS matches its fidelity in aggregate but its effective sample size collapses (43/200 at 135M, 6/200 at 1.7B): the weights concentrate on few trajectories, so per-sample quality is unreliable (a property of the IS ablation; native AWRS would instead pay more rejected proposals). 8-particle SMC pays $\sim8\times$ the forwards and still trails at this particle budget (asymptotically it is exact; the gap is the finite-K price — a K=2 sweep confirms the monotone trend: TV 0.645/0.910 at 135M/1.7B for $\sim2\times$ the forwards, barely better than masked). This is the tradeoff of §5 measured on one task: extra inference buys asymptotic unbiasedness, the lookup buys accuracy-per-forward inside the criterion's boundary.