GRAMMAR REACHABILITY FOR LOSSLESS LM-HEAD COMPRESSION AND DETERMINISTIC TOKEN INJECTION IN STRUCTURED GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Schema-constrained generation offers two complementary deployment optimisations. **GRVP** (Grammar-Reachable Vocabulary Pruning) identifies the unique minimum LM-head row set preserving the grammar-masked output distribution (Theorem 4.1). It removes 98% of rows on automotive Thrift IDL and 99.9% on public VehicleWorld, with $\Delta_{\mathrm{EM}}{=}0.0000$ across nine configurations and three model families; tied and mixed-type models use masks without storage savings. **GGSD** (Grammar-Guided Structured Decoding) injects grammar-forced tokens without a forward pass and reproduces per-state constrained greedy decoding for every block budget (Theorem 5.4). Reported speedups are $1.8\times$ on VehicleWorld, $5.86\times$ on xLAM and $\mathbf{64}\times$ on automotive (GPU-synchronised median; up to $86\times$ on a favourable subset), with deterministic fractions of 46%, 84% and 96.5% respectively. Both methods are post-training and require no training data. Frequency pruning needs $\Omega(|L(\mathcal{G}_S)|)$ samples in the worst case: 26% of reachable automotive tokens are absent from training, whereas xLAM deterministic-position coverage reaches 96%. Grammar analysis thus separates structural reachability from corpus frequency.

1 INTRODUCTION

Edge deployment requires compact models and low-latency inference. Quantization (Frantar et al., 2023; Lin et al., 2023), vocabulary pruning (Goel et al., 2025; Chen et al., 2026), and speculative decoding (Leviathan et al., 2023; Chen et al., 2023) use statistical signals to select weights or draft tokens. Schema-constrained API invocation, function calling (Patil et al., 2023; Yan et al., 2024; Liu et al., 2024), structured emission and code synthesis also expose exact structural constraints. For example, "Turn on the air conditioning for the front row" maps to:

```
AirConditionSettingsService.setACOnOffStatus(
    status=ON, row=ROW_1)
```

Only 2,972 of 151,665 vocabulary entries occur in valid outputs of the automotive Thrift schema (Slee et al., 2007); the other 98% are unreachable. The same grammar identifies positions with a unique next token, enabling injection without a learned drafter or calibration.

Contributions. GRVP removes rows outside $R(\mathcal{G}_S, T) = \bigcup_q R(\mathcal{G}_S, T, q_t)$. Its contribution is the minimality and uniqueness certificate, a compact Qwen-7B head ($1{,}087 \rightarrow 21$ MB), and validation across nine configurations; sufficiency follows directly from masking. GGSD composes this head with deterministic injection, a losslessness proof and zero-forward-pass optimality. We also establish a frequency-sampling lower bound and report multi-model service/parameter metrics and API-Bank/ToolAlpaca results. Figure 1 links these mechanisms; §8 distinguishes them from existing constrained decoders.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

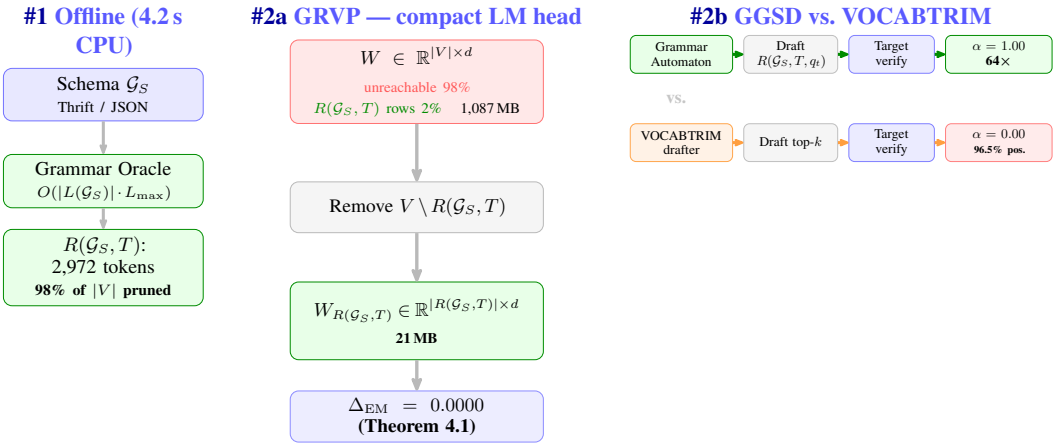


Figure 1: Grammar oracle framework (automotive). Offline enumeration takes 4.2 s. GRVP removes 98% of LM-head rows, saving 1,066 MB with Δ_{EM}=0.0000. At deterministic positions GGSD reports α=1.00 versus VOCABTRIM's 0.00; 774 reachable tokens have zero training frequency. The 64× speedup uses uncapped TokenDFA blocks and a GPU-synchronised median; 86× is a favourable-subset result (Table 10).

2 Background

2.1 Models and Decoding

An autoregressive LM with vocabulary V, hidden dimension d and head $W \in \mathbb{R}^{|V| \times d}$ gives $p_M(v \mid x, y_{<t}) = \text{softmax}(Wh_t)[v]$. Constrained decoding (Willard & Louf, 2023; Dong et al., 2024) masks logits outside $R(\mathcal{G}_S, T, q_t)$ and renormalises. GRVP removes rows outside the union of these per-state sets, rather than only masking them. In speculative decoding (Leviathan et al., 2023; Chen et al., 2023), a drafter proposes K tokens and the target accepts d_i with probability $\min(1, p_M(d_i)/p_D(d_i))$; GGSD instead uses the grammar automaton.

2.2 Grammars and Reachability

Definition 2.1 (Schema grammar)**.** A schema grammar $\mathcal{G}_S = (V_T, V_N, P, S_0)$ is a formal grammar where V_T is the terminal alphabet (a subset of Unicode characters), V_N is the non-terminal set, P is the production-rule set, and S_0 is the start symbol.

We study regular Thrift invocation languages, `Service.method(param=value,...)`, and context-free JSON Schema outputs with nested typed fields.

Definition 2.2 (Reachable token set)**.** Given schema grammar $\mathcal{G}_S$ and tokeniser T with vocabulary V:

$$R(\mathcal{G}_S, T) = \{t \in V : \exists\, w \in L(\mathcal{G}_S) \text{ s.t. } t \in T(w)\} \tag{1}$$

That is, t is reachable iff it appears in the tokenisation of at least one valid output string. We assume T is deterministic (canonical BPE), so $T(w)$ is a unique token sequence and the token-level automaton of Definition 2.3 admits canonical tokenisations only. *Termination convention.* Every $w \in L(\mathcal{G}_S)$ is implicitly terminated by the end-of-sequence marker, so $T(w)$ ends in `EOS` and `EOS` $\in R(\mathcal{G}_S, T)$; the automaton has a single accepting transition on `EOS` from every final state. Reported $|R(\mathcal{G}_S, T)|$ counts in this paper are *schema tokens only*, excluding `EOS` and any other tokeniser control tokens (BOS/PAD) that the deployed head retains for framework compatibility (Algorithm 1, line 2); those rows are never selected at a generation step and do not affect any result below. Concretely, the deployed automotive head has $2{,}972 + 18 = 2{,}990$ rows and the VehicleWorld head $175 + 18 = 193$ rows.

Definition 2.3 (Position-constrained reachable set)**.** At automaton configuration q_t (reached after decoding prefix $y_{<t}$):

$$R(\mathcal{G}_S, T, q_t) = \{t \in V : A_{\mathcal{G}_S} \text{ has a transition on } t \text{ from } q_t\} \tag{2}$$

where $A_{\mathcal{G}_S}$ recognises $L(\mathcal{G}_S)$ (a DFA for regular schemas; a PDA for context-free ones), lifted to token-level transitions. Three regimes recur throughout: *deterministic* ($|R(\mathcal{G}_S, T, q_t)| = 1$), *finite non-deterministic* ($1 < |R(\mathcal{G}_S, T, q_t)| < |V|$), and *open-vocabulary* ($R(\mathcal{G}_S, T, q_t) = V$: a free-string field, where the grammar imposes no constraint). A *dead state* is a configuration with $R(\mathcal{G}_S, T, q_t) = \emptyset$; it is reached only if a previously committed token was not a valid transition (§5).

2.3 Schema Domains and Datasets

VehicleWorld. The primary public benchmark (Yang et al., 2025) (Apache-2.0, HuggingFace) has 30 modules, 250 APIs and 2,818 records, split 2,209/259/350 by unique init sequence without train/test output overlap. Requests map to device lines such as `vw.fogLight=FogLight.init2()`. Its 398 valid sequences form an enumerable regular language.

Automotive. The proprietary Thrift IDL spans 30 domains, 250+ methods and 600+ enum values. Bounded numeric ranges make every parameter finite: $|L(\mathcal{G}_S)| = 15{,}703$ and $Q_{\text{typed}} = Q$. Both regular benchmarks therefore admit exact reachability and Theorem 4.1. Training uses 35,329 pairs (22,000 base SFT plus 13,329 multi-domain compositions); automotive EM uses the held-out v5 set of 5,512 single-/multi-domain records.

xLAM. xLAM-60K (Liu et al., 2024) spans hundreds of API domains; records contain a query, available-function JSON Schema and reference call. We use 5,998 test records. Free-string fields make the grammar context-free and restrict pruning to the typed subset (Theorem 4.3).

3 The Grammar-Oracle Framework

Lemma 3.1 (Mass concentration under constrained decoding)**.** *Write $p_M^{\text{constrained}}$ for the distribution obtained by applying the per-state grammar mask to the logits of p_M and renormalising. Then at each step t:*

$$\sum_{v \in R(\mathcal{G}_S, T, q_t)} p_M^{\text{constrained}}(v \mid x, y_{<t}) = 1 \tag{3}$$

Proof. Constrained decoding (Willard & Louf, 2023) sets the logit $\ell_v \leftarrow -\infty$ for all $v \notin R(\mathcal{G}_S, T, q_t)$ before softmax, so the resulting probability simplex is supported entirely on $R(\mathcal{G}_S, T, q_t)$. □

Unconstrained mass $m_t = \sum_{v \in R(\mathcal{G}_S, T, q_t)} p_M(v \mid x, y_{<t})$ can be below 1 (automotive geometric-mean 0.76; Section F); the lemma concerns only the masked distribution.

Corollary 3.2. *The* LM-head rows *of W_{out} corresponding to $V \setminus R(\mathcal{G}_S, T)$ contribute zero to the probability computation for any valid output $y \in L(\mathcal{G}_S)$ (input embedding rows are not covered; see Section E).*

Definition 3.3 (Output-distribution-preserving (ODP) reduction)**.** A vocabulary subset $R' \subseteq V$ is ODP for $\mathcal{G}_S$ under a grammar logits mask if, for all inputs x and all valid outputs $y \in L(\mathcal{G}_S)$, evaluating the compact model $M \restriction R'$ with the *same* per-step state-specific mask $R(\mathcal{G}_S, T, q_t)$ produces identical output probabilities:

$$p_{M \restriction R'}^{\text{masked}}(y \mid x) = p_M^{\text{masked}}(y \mid x) \tag{4}$$

where both sides apply the state-specific grammar mask $\mathbf{1}[v \in R(\mathcal{G}_S, T, q_t)]$ at each step t. The only difference between the two systems is that $M \restriction R'$ has the rows $v \notin R'$ *physically removed* from the output weight matrix (equivalently, their pre-softmax logits are set to $-\infty$, so $\exp(-\infty) = 0$ and they contribute zero to the softmax denominator).

4 GRVP Optimality

Theorem 4.1 (GRVP is the unique minimum ODP reduction)**.** *$R(\mathcal{G}_S, T)$ is ODP, and every ODP reduction contains it: for all $R' \subseteq V$, R' ODP for $\mathcal{G}_S \Rightarrow R(\mathcal{G}_S, T) \subseteq R'$. Hence $R(\mathcal{G}_S, T)$ is the*

unique minimum ODP vocabulary reduction (the least element of the ODP family under inclusion). No assumption on the model is needed beyond finite logits: softmax assigns strictly positive probability to every token with a finite logit, so $p_M^{\text{masked}}(y \mid x) > 0$ *for every* $y \in L(\mathcal{G}_S)$ *and every input* x*.*

Proof sketch. Sufficiency: for any $v \notin R(\mathcal{G}_S, T) = \bigcup_q R(\mathcal{G}_S, T, q_t)$, $v \notin R(\mathcal{G}_S, T, q_t)$ at every reachable configuration q, so the per-state mask already sets $\ell_v = -\infty$ at every step; removing row v from W_{out} changes no masked logit, hence $M \restriction R(\mathcal{G}_S, T)$ is ODP.

Necessity: if $t_0 \in R(\mathcal{G}_S, T) \setminus R'$, then $t_0 = y_\tau$ for some valid $y \in L(\mathcal{G}_S)$ and position τ with $t_0 \in R(\mathcal{G}_S, T, q_\tau)$, so the mask does *not* remove it, but row removal does: $p_{M \restriction R'}^{\text{masked}}(y \mid x) = 0 < p_M^{\text{masked}}(y \mid x)$, contradicting ODP.

Uniqueness: any minimal ODP set contains $R(\mathcal{G}_S, T)$ (necessity) and $R(\mathcal{G}_S, T)$ is itself ODP (sufficiency), so the two coincide (full proof in Section H). □

Corollary 4.2. $R_{\text{freq}}(k) \supseteq R(\mathcal{G}_S, T)$ *requires every token in* $R(\mathcal{G}_S, T)$ *to have nonzero sample frequency, needing* $\Omega(|L(\mathcal{G}_S)|)$ *samples worst-case (Theorem 6.2).*

Theorem 4.3 (GRVP for mixed-type context-free schemas)**.** *Let* $\mathcal{G}_S$ *contain both* typed *fields (finite value ranges: enumerated constants, booleans, bounded integers) and* free-string *fields (unbounded values), and let* $Q_{\text{typed}} \subseteq Q$ *be the configurations at which the next token belongs to a typed field or to structural syntax, so that* $R(\mathcal{G}_S, T, q_t)$ *is finite for* $q \in Q_{\text{typed}}$ *and* $R(\mathcal{G}_S, T, q_t) = V$ *otherwise. Define the* typed-parameter reachable set $R_{\text{typed}}(\mathcal{G}_S, T) = \bigcup_{q \in Q_{\text{typed}}} R(\mathcal{G}_S, T, q_t)$*. Consider a decoder that applies the per-state mask at typed configurations and no mask at free-string configurations, and a compact head with rows* R' *that serves the typed configurations. Then the output distribution is preserved iff* $R' \supseteq R_{\text{typed}}(\mathcal{G}_S, T)$*;* $R_{\text{typed}}(\mathcal{G}_S, T)$ *is the unique minimum such row set. Free-string configurations have* $R(\mathcal{G}_S, T, q_t) = V$ *and require the full head, so for mixed-type schemas GRVP reduces per-step LM-head FLOPs at typed positions but not resident storage; when* $Q_{\text{typed}} = Q$ *the statement reduces to Theorem 4.1.*

Remark 4.4 (xLAM GRVP scope). The 7,303-token $R_{\text{typed}}(\mathcal{G}_S, T)$ unions the schemas of all 5,998 test records; per-record heads could be smaller. Only schemas available as inference inputs are read, not reference responses. xLAM uses masks with no storage saving; typed-position FLOP savings from two heads are not measured. Untied models on the two all-enumerated benchmarks use physical row removal.

5 Grammar-Guided Structured Decoding (GGSD)

Mechanism and prior work. Like ToolSpec (Xia et al., 2026), GGSD fills deterministic schema tokens. It adds the losslessness and optimality results below, DFA/PDA scope, and integration with GRVP's compact head via an $O(1)$ TokenDFAAutomaton. No matched ToolSpec comparison is available (§8).

Block budget K. GGSD injects consecutive forced tokens; the next target pass prefills them into the KV cache and resolves the non-deterministic position. K caps tokens per prefill: $K = 1$ processes positions individually; $K = \infty$ is the uncapped production path. Blocks contain forced tokens plus a target-decided position, not sampled candidates. Direct tokenisation of valid outputs avoids character-mask/subword misalignment (Beurer-Kellner et al., 2024).

Proposition 5.1 (Deterministic token injection)**.** *When* $R(\mathcal{G}_S, T, q_t) = \{v^\star\}$*, GGSD commits* $v^\star$ *directly without any model forward pass. The committed token is identical to that of constrained greedy decoding, and the step executes in* $O(1)$ *automaton time with zero model cost.*

Proof. Under the per-state mask, Lemma 3.1 gives $p_M^{\text{constrained}}(v^\star) = 1$ and $p_M^{\text{constrained}}(v) = 0$ for $v \neq v^\star$, for *every* hidden state h_t. Constrained greedy therefore commits $v^\star$ regardless of the model's logits, so evaluating the model is unnecessary. □

Proposition 5.2 (Constrained greedy at finite non-deterministic positions)**.** *When* $1 < |R(\mathcal{G}_S, T, q_t)| < |V|$*, GGSD runs one model forward pass and commits* $\arg\max_{v \in R(\mathcal{G}_S, T, q_t)} p_M(v)$*. This is standard constrained greedy decoding; no rejection or re-sampling occurs, so the step produces the same token as the constrained baseline, independently of* K*.*

Proposition 5.3 (Autoregressive fallback: open-vocabulary and off-grammar positions)**.** *Two cases provide no GGSD speedup:* (a) Open-vocabulary*:* $R(\mathcal{G}_S, T, q_t) = V$ *— the grammar imposes no constraint (a free-string field); GGSD and the constrained baseline both execute one unconstrained greedy step. This case arises on xLAM/JSON Schema but not on the two all-enumerated regular schemas (§2.3).* (b) Dead state*:* $R(\mathcal{G}_S, T, q_t) = \emptyset$*, reached only when the target was masked with the* global *set* $R(\mathcal{G}_S, T)$ *rather than* $R(\mathcal{G}_S, T, q_t)$ *(our production configuration, §7) and committed a token that is in* $R(\mathcal{G}_S, T)$ *but not a valid transition from* q_t*. GGSD's draft oracle is then abandoned for the remainder of the call and every subsequent position is decoded exactly as the baseline decodes it (no speedup, same mask). Dead-state detection is* $O(1)$ *(one dict lookup, measured 0.254 µs/call); once latched, all subsequent calls return* $\emptyset$ *in* $O(1)$*. Under the per-state mask, dead states cannot occur. Empirically, 3.6% of automotive calls enter a dead state under the global mask (1,000-record subset; Section D.1).*

Theorem 5.4 (GGSD losslessness)**.** *Let* B *denote constrained greedy decoding with the per-state mask* $\mathbf{1}[v \in R(\mathcal{G}_S, T, q_t)]$*, where open-vocabulary positions (*$R(\mathcal{G}_S, T, q_t) = V$*) are decoded unconstrained. For every input* x *and every decode position* t *at which the automaton returns a non-empty* $R(\mathcal{G}_S, T, q_t)$*, GGSD commits the same token as* B*, and hence produces the same output string:* $p_{\mathrm{GGSD}}(v \mid x, y_{<t}) = p_B(v \mid x, y_{<t})$ *for all such* t*. Consequently* $\Delta_{\mathrm{EM}}(\mathrm{GGSD}, B) = 0$ *for every block budget* K*.*

Proof. Case analysis over the three regimes of Definition 2.3: deterministic (Proposition 5.1), finite non-deterministic (Proposition 5.2), open-vocabulary (Proposition 5.3a). In each case the committed token coincides with B's; by induction on t the prefixes, hence the automaton configurations, hence the outputs agree. Dead states are excluded by hypothesis (they cannot arise under B); full details in Section H. □

Remark 5.5 (Speedup source). Only forced positions avoid a model decision: their fractions are 96.5% (automotive), 84.4% (xLAM) and 46.4% (VehicleWorld). Other positions need one forward pass; K controls prefill amortisation.

Theorem 5.6 (Grammar-state access is necessary and sufficient for zero-forward-pass commitment)**.** *Consider a deterministic position,* $R(\mathcal{G}_S, T, q_t) = \{v^\star\}$*, and any speculative drafter* D *with budget* $K \geq 1$ *whose proposals are verified by the target under the per-state mask. (a) If* D *has no access to* $A_{\mathcal{G}_S}$ *(it cannot certify* $|R(\mathcal{G}_S, T, q_t)| = 1$*), then no such* D *can commit* $v^\star$ *with zero target forward passes* while remaining lossless at every position*, and the expected number of accepted tokens per drafted position is* $\alpha_D = \sum_v \min\{p_D(v),\, p_M^{\mathrm{constrained}}(v)\} = p_D(v^\star) \leq 1$*, with equality iff* D *proposes* $v^\star$ *with certainty. (b) The grammar automaton commits* $v^\star$ *with zero forward passes and* $\alpha = 1$*. Hence zero target cost at deterministic positions is achievable exactly by drafters with grammar-state access (the automaton, or any constrained-decoding engine that exposes* $R(\mathcal{G}_S, T, q_t)$*, e.g. Outlines (Willard & Louf, 2023) or SGLang (Zheng et al., 2024)), and by no drafter without it.*

Proof. (a) Without a certificate that $|R(\mathcal{G}_S, T, q_t)| = 1$, a drafter cannot distinguish a forced position from a non-forced one. A strategy that commits its proposal unverified must therefore do so at non-forced positions too, where the proposal need not be the target's constrained argmax; such a strategy is not lossless. Hence any *lossless* drafter without grammar-state access must verify, and verification requires evaluating $p_M(\cdot \mid x, y_{<t})$: one forward pass. The acceptance identity follows from Leviathan's rule with $p_M^{\mathrm{constrained}} = \delta_{v^\star}$ (Section H). (b) is Proposition 5.1. □

Corollary 5.7 (Speedup cost model)**.** *Let* f_{det} *be the fraction of output positions with* $|R(\mathcal{G}_S, T, q_t)| = 1$ *and let* T_{ar} *be the baseline cost of one autoregressive step. Let* ρT_{ar} *be the cost of one GGSD target forward pass (a prefill of up to* K *injected tokens, the last of which also resolves a non-deterministic position), and let* $c\,T_{\mathrm{ar}}$ *be the automaton cost paid at* every *position, deterministic or not. For an* N*-token output GGSD issues* $(1 - f_{\mathrm{det}})N$ *block-terminating passes plus* $f_{\mathrm{det}}N/K$ *additional prefills when blocks are capped at* K*, so the throughput speedup over autoregressive decoding is*

$$S \;=\; \frac{1}{\big[(1 - f_{\mathrm{det}}) + f_{\mathrm{det}}/K\big]\,\rho \;+\; c}.$$

Two limits are worth naming. With uncapped blocks ($K{=}\infty$*) and a negligible automaton (*$c{\to}0$*) this is* $1/[(1 - f_{\mathrm{det}})\rho]$*, and setting* $\rho{=}1$ *recovers the familiar ideal reference* $1/(1 - f_{\mathrm{det}})$*. With* $K{=}1$

every injected token is prefilled separately and $S \to 1/(\rho + c)$: no amortisation, and the only saving is that a bare prefill call is cheaper than a full framework decode step. $\rho < 1$ arises when the baseline step is dominated by per-step framework overhead (KV-cache bookkeeping, logits processing, Python dispatch) that a block prefill amortises; $\rho > 1$ arises for long blocks in the compute-bound regime. ρ is therefore a function of block length rather than a constant, and §7 reports it per configuration. This is a cost model to be fitted, not a guaranteed bound.

Note on KV-cache consistency*: at deterministic positions no forward pass is issued and the KV cache is not extended; the next forward pass prefills all injected tokens, and this cost is included in every timing reported in §7.*

6 Relation to Statistical Alternatives

6.1 Grammar Oracle vs. Gradient Saliency

GPTQ (Frantar et al., 2023) uses input covariance $2XX^\top$, AWQ (Lin et al., 2023) activation magnitudes, and pruning often the task-loss empirical Fisher. The oracle certifies zero output-row saliency without calibration.

Theorem 6.1 (Grammar oracle certifies zero gradient saliency for unreachable LM-head rows)**.** *Under constrained calibration (per-state mask active) on calibration targets drawn from $L(\mathcal{G}_S)$, the empirical Fisher diagonal of the task loss is exactly zero for all output rows $i \notin R(\mathcal{G}_S, T)$: $H_{ii}^{W_{\text{out}}} = 0$ for all $i \notin R(\mathcal{G}_S, T)$, for any finite calibration set. Thus gradient saliency agrees with the grammar oracle on $V \setminus R(\mathcal{G}_S, T)$, but only after a calibration pass; the oracle provides the same certificate with no data.* Scope*: applies to W_{out} only; it says nothing about GPTQ's input covariance $2XX^\top$ (which is row-independent and nonzero) or about input embeddings.*

6.2 Frequency-Based Reachability

Theorem 6.2 (Sample complexity of frequency-based reachability)**.** *Certifying that a frequency-based estimate covers all of $R(\mathcal{G}_S, T)$ requires $\Omega(|L(\mathcal{G}_S)|)$ output samples in the worst case: a schema identifier that appears in only one valid output string requires seeing that string at least once. Grammar reachability computes $R(\mathcal{G}_S, T)$ exactly with zero samples, in $O(|L(\mathcal{G}_S)| \cdot L_{\max})$ time by enumeration whenever $L(\mathcal{G}_S)$ is finite ($L_{\max}$ = maximum output length in tokens; 4.2 s for the automotive schema), and for regular schemas in $O(|Q| \cdot |V| \cdot \ell_{\max})$ time by token-level DFA construction even when $L(\mathcal{G}_S)$ is infinite ($\ell_{\max}$ = maximum token length in characters). For context-free schemas the analogous construction is performed per decode step on the PDA configuration rather than once offline (Section I).*

The automotive corpus leaves 774 of 2,972 reachable tokens (26.0%) unobserved in 35,329 records; thresholding that corpus cannot recover them.

Scope. Grammar validity does not ensure semantic correctness. Tied weights require masks without storage savings; open-vocabulary positions contribute no deterministic speedup. Losslessness compares against local masking, not the globally grammar-conditional distribution $\mu^\star$ (Nie et al., 2026) (Section F).

7 Experiments

7.1 Experimental Setup

We evaluate Qwen2.5-Coder-7B/-1.5B (Qwen Team, 2024), Mistral-7B-v0.3 (Jiang et al., 2023) and SmolLM2-1.7B (Allal et al., 2025); Qwen-1.5B has tied weights. All use rank-8 LoRA SFT (Hu et al., 2022); automotive Qwen-7B additionally uses DPO (batch 16, five epochs; Section C). Reachability depends on schema/tokeniser, not weights. Across five tokenisers, VehicleWorld retains only 175–208 schema tokens; the full comparison is in Appendix C (Table 9). Benchmarks are in §2.3; Spider is excluded because SQL DDL does not enumerate valid queries.

Table 1: GRVP results: Δ = GRVP minus masked-baseline EM; Svc = service-name match (undefined for VW device lines). [§]Freq. EM uses same-size frequency-ranked target rows, outside VOCABTRIM's drafter role. [¶]Counts exclude 18 control rows (193 deployed for VW). [‡]VW uses query-only prompts and the init-sequence split; adding `Environment.initN()` context lowered EM to 11%, versus 21% here. xLAM uses typed-subvocabulary masks; other untied rows use physical removal.

Model	Bench.	$\lvert R(\mathcal{G}_S, T)\rvert$	EM	Svc	Δ	Freq. EM[§]
VehicleWorld (public, Apache-2.0, $\lvert L(\mathcal{G}_S)\rvert$=398, regular grammar)						
Qwen-7B	VW	**175**[¶]	.210[‡]	—	**.0000**	.000
Qwen-1.5B	VW	**175**[¶]	.175[‡]	—	**.0000**	.000
Automotive Thrift IDL (proprietary, $\lvert L(\mathcal{G}_S)\rvert$=15,703, regular grammar)						
Qwen-7B	Auto.	2,972	.826	.859	**.0000**	.000
Qwen-1.5B	Auto.	2,972	.518	.633	**.0000**	.395
Mistral-7B	Auto.	2,324	.666	.779	**.0000**	.000
SmolLM2-1.7B	Auto.	2,240	.475	.604	**.0000**	.345
xLAM / JSON Schema (public, CF grammar, typed-parameter GRVP; Theorem 4.3)						
Qwen-7B	xLAM	7,303	.759	.853	**.0000**	.000
Qwen-1.5B	xLAM	7,303	.614	.772	**.0000**	.615
Mistral-7B	xLAM	5,281	.394	.571	**.0000**	.000

Protocol. We compare (i) unconstrained greedy, (ii) a full-head grammar-mask baseline and (iii) the same mask plus compact-head row removal. Tied/mixed-type models use masks only. Conditions (ii)–(iii) use the deployed global set $R(\mathcal{G}_S, T)$; their equality checks arithmetic, remapping and logit re-expansion. The equivalence also holds with the per-state mask since $R(\mathcal{G}_S, T, q_t) \subseteq R(\mathcal{G}_S, T)$. Reported $\Delta_{\mathrm{EM}} = 0.0000$ denotes bit-identical outputs, not a statistical estimate. A same-size frequency-ranked target head is a stress test, separate from VOCABTRIM's intended drafter role (Goel et al., 2025), evaluated in Table 2.

7.2 Theorem 4.1 Validation: GRVP Is Lossless

Table 1 reports zero output differences across nine configurations, including regular/context-free grammars and tied/untied models. VehicleWorld retains 175 of 151,665 tokens (99.9% pruning); its 21% Qwen-7B EM shows that preservation is independent of task accuracy. Its $f_{\mathrm{det}} = 46.4\%$ yields $1.81\times$ GGSD speedup (Section B).

Frequency signal. At threshold 1, automotive coverage is 74.0%; thresholds 10 or 50 reduce it further. Position-weighted VT acceptance is 2.6%, including 0.00 at deterministic positions, versus GGSD's 1.00. Frequency-pruned EM is zero for all 7B configurations; automotive Qwen-7B service match falls from 85.9% to 76.1% (Table 14). xLAM deterministic-position coverage is 96%, reflecting common subwords, yet EM still collapses because exact match needs every token of a call (Section G). Masking itself improves Qwen-7B EM from 82.3% to 82.6% (+0.3 pp) on automotive and 74.8% to 75.9% (+1.1 pp) on xLAM.

7.3 GGSD Validation: Losslessness and Acceptance

Per-state versus global masks. Theorem 5.4 concerns per-state baseline B; reported EM uses global-mask decoder A (`model.generate()` plus `ReachabilityLogitsProcessor`), while GGSD injects from the per-state DFA. If A remains in-grammar, its global argmax lies in $R(\mathcal{G}_S, T, q_t)$ and is also B's argmax. GGSD makes the same non-deterministic decisions as A; a first disagreement at a forced position means A chose an invalid transition and cannot exact-match a valid reference. Thus $\mathrm{EM}(\mathrm{GGSD}) \geq \mathrm{EM}(A)$, with token agreement on the 96.4% of calls staying in-grammar. The other 3.6% enter dead states (1,000-record subset; Section D.1). Per-state target masking restores the theorem's full scope at one lookup per step. Timing is measured separately.

Automotive and block budget. With adapter B, 96.5% of in-grammar positions are forced and 3.5% target-decided. TokenDFA reports a $64.0\times$ production median, rising to $86.0 \pm 2.8\times$ on 1,000

Table 2: Acceptance and reported EM (Qwen-7B). α is the accepted fraction of block tokens; GGSD's 1.00 follows from forced tokens or target argmax, whereas VT is the matched-budget VOCABTRIM drafter. EM uses the global baseline; the text states its relation to GGSD. Automotive uses SchemaAutomaton at finite K; ¶$K = 8$. †TokenDFA, uncapped, GPU-synchronised median, $n = 1{,}000$; favourable-subset speedup is $86.0 \pm 2.8\times$. xLAM uses XLAMAutomaton, $K = 4$, 5,998 records.

Schema	Position type	Frac.	GGSD α	VT α	EM	Speedup
Automotive, adapter A (constrained EM=78.5%), $f_{\text{det}} = 0.795$						
	Det. ($\lvert R_q\rvert$=1)	79.5%	**1.00**	0.00	.785	4.13× (K=4)
	Hi-con. ($\leq$4)	1.5%	**1.00**	0.79		
	Other (not instrumented)	19.0%	—	—		
Automotive, adapter B (constrained EM=82.6%, unconstrained 82.3%), $f_{\text{det}} = 0.965$						
Auto.	Det. ($\lvert R_q\rvert$=1)	96.5%	**1.00**	0.00	**.826**	7.49× (K=4); 11.70×¶; **64.0**×†
	Hi-con. ($\leq$4)	1.3%	**1.00**	0.78		
	Mi-con. ($\leq$20)	1.1%	**1.00**	0.54		
	Lo-con. (>20)	1.1%	**1.00**	0.92		
xLAM	Det. ($\lvert R_q\rvert$=1)	84.4%	**1.00**	0.96	**.759**‡	**5.86**×
	Hi-con.	1.3%	**1.00**	0.97		
	Open-vocab	14.3%	—	∼1.00		

"—": no GGSD draft in open vocabulary; adapter A's remaining positions were not instrumented. Automotive in-grammar buckets total 100%; non-deterministic positions terminate blocks at $K \geq 2$ and lie outside them at $K = 1$ (Table 3). xLAM's 14.3% generated open-vocabulary steps contrast with 42.8% in static analysis, attributed to earlier string termination (Section G). ‡Constrained-baseline EM; token agreement holds on in-grammar calls.

Table 3: Block-budget ablation (Qwen-7B, automotive, A100, `SchemaAutomaton`). EM uses all 5,512 v5 records; timing uses 1,000 records and each row's paired clean-AR baseline (25.9–27.1 tok/s). Reported EM is 82.6% at every K. α is block-token acceptance: at $K = 1$, non-deterministic positions lie outside blocks, so 0.00 means no draft, not rejection; at $K \geq 2$, their target-argmax tokens give 1.00 by construction. Compare VT in Table 2; fitted costs are in Table 12.

K	EM	Speedup	Tok/s	α_{det}	$\alpha_{\text{hi-con}}$	$\alpha_{\text{mi-con}}$
1	.826	5.62×	152.4	1.00	0.00	0.00
2	.826	6.07×	161.3	1.00	1.00	1.00
4	.826	7.49×	197.1	1.00	1.00	1.00
8	.826	**11.70**×	**303.6**	1.00	1.00	1.00

high-EM records; SchemaAutomaton reports $7.49\times$ at $K = 4$ and $11.70\times$ at $K = 8$. These runs use different subsets/harnesses, not a controlled automaton ablation (Table 10). Speedups use clean AR without a logits processor on an A100. Adapter A yields $4.13\times$ at $K = 4$ with EM 78.5%, versus B's 82.6% and higher f_{det}.

The $K \in \{1, 2, 4, 8\}$ ablation reports identical EM 82.6% and speedups from $5.62\times$ to $11.70\times$ (Table 3). At $K = 1$, gains reflect a cheaper bare prefill rather than block amortisation. With $T_{\text{ar}} \approx 37.7$ ms (26.5 tok/s), $D \approx 28$, and TokenDFA query overhead 0.06% ($c < 10^{-4}$), the uncapped fit gives $\rho = 0.45/0.33$ for 64/86×: a 29-token prefill costs less than a framework decode step. SchemaAutomaton uses $c = 0.012$ (≈ 0.45 ms), with fitted $\rho = 0.17, 0.30, 0.44, 0.47$ across K; holding the $K = 4$ fit predicts $K = 8$ within 7%. The 28.6× reference assumes $\rho = 1, c = 0$; it is not an upper bound, because the baseline's per-step framework overhead (Python dispatch, KV bookkeeping, logits processing) is paid once per block under GGSD. Section D gives the fit and $O((n + D)^2 d)$ prefill complexity.

xLAM and additional schemas. xLAM reports $f_{\text{det}} = .844$, 5.86× (145.3 tok/s; median of five runs, $K = 4$), and baseline EM 75.9%. Its uncapped reference is 6.4× ($\rho \approx 1.09$ when inferred without the cap term); this is a reference, not the capped-model prediction. API-Bank/ToolAlpaca yield EM 52.5%/45.6% with API docs, typed coverage 5.8%/4.8% (94.2%/95.2% unused), and $\Delta = 0$

under masks. Full heads remain resident; scaffold-only $f_{\text{det}} \approx 26\%$ gives the $\approx 1.35\times$ reference. These two supplementary runs are outside the nine configurations; format-specific automata remain future work.

Saliency and deployment. Constrained-calibration Fisher values are 0.000 on unreachable rows (reachable-row mean 0.091; Section H.6). Reported Qwen-7B storage falls 14.6→13.5 GB (head 1,087→21 MB; saving 1,066 MB); GGSD reaches 1,700–2,291 tok/s at reported EM .826, and GRVP+INT4 occupies 4.4 GB. Table 4 reports 100% validity; the global-mask dead-state qualification above remains relevant to interpreting that claim.

8 Related Work

Vocabulary compression. VOCABTRIM (Goel et al., 2025), NanoSpec (Chen et al., 2026), SpecVocab (Williams et al., 2026) and VocabTailor (Ushio et al., 2026) use frequency signals, which cannot certify unseen schema tokens. Only VOCABTRIM is evaluated here. Chijiwa et al. (2026) reduce vocabulary losslessly through coarser tokenisation for model cooperation; GRVP instead uses output-language reachability to minimise a single model's head. These criteria can compose; empirical comparison is future work.

Decoding. Speculative verification (Leviathan et al., 2023; Chen et al., 2023) supports learned drafters such as EAGLE (Li et al., 2024) and Medusa (Cai et al., 2024). ToolSpec (Xia et al., 2026) (4.2×, Qwen2.5-14B, API-Bank), OoO-Spec (Zhang et al., 2026) (3.89×, no losslessness guarantee) and DOMINO (Beurer-Kellner et al., 2024) (1.77×, calibrated grammar-state counts) report unmatched models/tasks; GGSD needs no calibration, composes with head compression, and these numbers do not establish a ranking.

Constrained generation. Outlines (Willard & Louf, 2023)/XGrammar (Dong et al., 2024) precompute masks; SGLang's jump-forward decoding (Zheng et al., 2024) and Guidance (Lundberg & Ribeiro, 2023) already skip forward passes at deterministic grammar transitions, so token injection itself is not new. GGSD adds a per-state losslessness proof (Theorem 5.4), zero-forward-pass necessity/sufficiency (Theorem 5.6), and composition with GRVP's compact head. CFGzip (Ugare et al., 2026)/PSC (Li et al., 2026) reduce mask cost to $O(1)$ without removing weights. Alpay & Senturk (2026) establish oracle invariance; GRVP adds vocabulary minimality. Nie et al. (2026) examine masking distortion (Section F).

9 Limitations

Tied weights and mixed-type schemas realise no storage saving; mixed-type FLOP savings are unmeasured. GRVP retains every grammar-reachable row regardless of model frequency, and free-text-heavy formats offer limited injection. Global target masking admits dead states; theorem-level GGSD equivalence requires per-state masking. Timing uses one A100, different subsets/harnesses for the two automata, and clean AR; external systems were not run under a matched protocol. The automotive schema is proprietary; BFCL (Yan et al., 2024)-style community leaderboards are not evaluated, so generalisation to less enum-dominated workloads is untested, and serving many concurrently active compact heads (multi-tenant deployment) is unmeasured.

10 Conclusion

Grammar reachability identifies both removable output rows and tokens that need no model decision. GRVP's minimum-row certificate accompanies reported 98% automotive and 99.9% VehicleWorld head reductions with $\Delta_{\text{EM}} = 0.0000$ across nine configurations. GGSD's reported 1.8×/5.86×/64× speedups show the opportunity across schemas, while forced-token frequency, prefill cost and timing protocol determine the gain. The practical scope is explicit: untied enumerable schemas permit storage reduction; free-string fields retain the full head; exact GGSD equivalence is to per-state constrained decoding. These complementary, data-free optimisations turn schema structure into deployment savings without retraining.

REFERENCES

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blaǵojević, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. SmolLM2: When smol goes big — data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Faruk Alpay and Bilge Senturk. Attention meets reachability: Structural equivalence and efficiency in grammar-constrained LLM decoding. *arXiv preprint arXiv:2603.05540*, 2026.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Guiding LLMs the right way: Fast, non-invasive constrained generation. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 3658–3673. PMLR, 2024.

Tianle Cai et al. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Charlie Chen et al. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Zhiyang Chen, Daliang Xu, Yinyuan Zhang, Chenghua Wang, Mengwei Xu, and Yun Ma. NanoSpec: Accelerating speculative decoding using minimalist in-context vocabularies. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Daiki Chijiwa, Taku Hasegawa, Kyosuke Nishida, Shin'ya Yamaguchi, Tomoya Ohba, Tamao Sakao, and Susumu Takeuchi. Lossless vocabulary reduction for auto-regressive language models. In *International Conference on Learning Representations*, 2026.

Yixin Dong et al. XGrammar: Flexible and efficient structured generation engine for large language models. *arXiv preprint arXiv:2411.15100*, 2024.

Elias Frantar et al. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023.

Raghavv Goel, Sudhanshu Agrawal, Mukul Gagrani, Junyoung Park, Yifan Zao, He Zhang, Tian Liu, Yiping Yang, Xin Yuan, Jiuyuan Lu, Christopher Lott, and Mingu Lee. VOCABTRIM: Vocabulary pruning for efficient speculative decoding in LLMs. In *ICML Workshop on Efficient Systems for Foundational Models*, 2025.

Edward J Hu et al. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2022.

Albert Q Jiang et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Yongmin Li et al. Efficient grammar-constrained decoding via parser stack classification. *arXiv preprint arXiv:2608.03065*, 2026.

Yuhui Li et al. EAGLE: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.

Ji Lin et al. AWQ: Activation-aware weight quantization for LLM compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Zuxin Liu et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Advances in Neural Information Processing Systems*, 37, 2024.

Scott Lundberg and Marco Tulio Ribeiro. Guidance: A guidance language for controlling large language models. `https://github.com/guidance-ai/guidance`, 2023.

Wenhua Nie, Zijie Meng, Kun Zou, Zheng Lin, Ziwei Li, Haoran Zheng, Jyh-Shing Roger Jang, and Hao Zhang. Future validity is the missing statistic: From impossibility to ϕ-estimation for grammar-faithful speculative decoding. *arXiv preprint arXiv:2605.07698*, 2026.

Shishir G Patil et al. Gorilla: Large language model connected with massive APIs. *arXiv preprint arXiv:2305.15334*, 2023.

Qwen Team. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Avinash Reddy, Thayne Walker, Jaime Ide, and Amrit Singh Bedi. The hidden cost of structured generation in LLMs: Draft-conditioned constrained decoding. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Mark Slee, Aditya Agarwal, and Marc Kwiatkowski. Thrift: Scalable cross-language services implementation. Technical report, Facebook, 2007.

Shubham Ugare et al. CFGzip: Accelerating constrained decoding with token space compression. *arXiv preprint arXiv:2605.29986*, 2026.

Asahi Ushio et al. VocabTailor: Dynamic vocabulary selection for downstream tasks in small language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Brandon T Willard and Rémi Louf. Efficient guided generation for large language models. In *NeurIPS Workshop on Instruction Tuning and Instruction Following*, 2023.

Miles Williams, Young D. Kwon, Rui Li, Alexandros Kouris, and Stylianos I. Venieris. Speculative decoding with a speculative vocabulary. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Heming Xia, Yongqi Li, Cunxiao Du, Mingbo Song, and Wenjie Li. ToolSpec: Accelerating tool calling via schema-aware and retrieval-augmented speculative decoding. *arXiv preprint arXiv:2604.13519*, 2026. v1 April 2026; v2 May 2026.

Fanjia Yan et al. Berkeley function calling leaderboard. *arXiv preprint arXiv:2402.15671*, 2024.

Jie Yang et al. VehicleWorld: A highly integrated multi-device environment for intelligent vehicle interaction. In *Findings of EMNLP*, pp. 403–442, 2025.

Zhiheng Zhang, Mujie Xu, Feiyu Sun, and Zhixin Zhang. OoO-Spec: Out-of-order semantic speculation for fast tool calling. *arXiv preprint arXiv:2608.00814*, 2026.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, 2024. arXiv:2312.07104.

AI USE STATEMENT

In this work, we used generative AI tools (GitHub Copilot, Claude, Devin) for the following purposes, consistent with the ICLR 2027 disclosure policy:

1. **To aid or polish writing:** grammar, phrasing, and LaTeX formatting were assisted throughout the manuscript.
2. **For retrieval and discovery:** literature search for related work (speculative decoding, vocabulary pruning, grammar-constrained generation papers) used LLM-assisted retrieval.
3. **To draft sections of the paper:** initial drafts of several sections were generated with AI assistance, including the KV-cache batching analysis (Section D), the Limitations paragraph, the supplementary API-Bank/ToolAlpaca paragraph (§7), and the oracle build-time appendix (Section I).
4. **Code completion assistance:** GitHub Copilot autocomplete was used during implementation sessions for routine tasks such as boilerplate, data-structure wrappers, and test fixture generation. The core algorithmic logic of GRVP and GGSD — the reachability computation, DFA construction, deterministic token injection, and KV-cache batching — was designed and written by the authors. All AI-suggested code was reviewed, tested, and verified for correctness by the authors before inclusion.

AI was **not** used for proving mathematical claims, generating synthetic datasets, or for research ideation. All claims, theorems, proofs, experimental results, and numerical values were independently verified by the authors. LLM-generated code was tested for correctness against known ground truth (e.g., $\Delta_{\mathrm{EM}} = 0.0000$ validated by direct comparison of model outputs). We take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The primary public benchmark, VehicleWorld (Yang et al., 2025) (Apache-2.0), is publicly available on HuggingFace and can be used to reproduce the GRVP losslessness results (Δ=0, $|R(\mathcal{G}_S, T)|$=175, f_{det}=46.4%) independently of our implementation. The GRVP and GGSD algorithms are fully described in §4–§5 and Algorithm 1; they require only a tokeniser and a grammar specification. The automotive Thrift IDL schema is proprietary and cannot be released; all claims against it are corroborated by the public VehicleWorld results. Proofs of Theorems 4.1, 4.3, 5.4, 5.6, 6.1, and 6.2 are in Appendix H; every headline number cited in the text is traceable to Table 6, which also states the protocol behind each one.

A Deployment Efficiency: System-Level and Extended Tables

Table 4: Deployment efficiency — system-level comparison (Qwen-7B, automotive, bf16, batch=1, A100). †System-level tok/s is unchanged at batch 1 because the LM head is ≈2% of a decode step (0.614 ms of ≈37 ms); the isolated LM-head speedup is reported in Table 5. *The NF4 recipe keeps `lm_head` in bf16, so GRVP removes the same 1.07 GB; value derived from the INT4 row, not separately measured. §Baseline tok/s measured in a separate session from the paired baseline of the timing protocol (26.5 tok/s, Table 10); ¶range spans the production-path median (64×, ≈1,700 tok/s at that baseline) and the high-EM timing subset (86×, 2,291 tok/s).

System	Storage	Tok/s	EM	Schema valid
Base bf16	14.6 GB	26.9§	.823	$< 100\%$
GRVP only (ours)	**13.5 GB**	26.9†	**.826**	**100%**
INT4 NF4 (GPTQ)	**5.5 GB**	12.5	.809	$< 100\%$
GRVP+INT4 (ours)	**4.4 GB***	12.5†	**.809**	**100%**
GRVP+GGSD (ours)	**13.5 GB**	**1,700–2,291¶**	**.826**	**100%**

Table 5: Deployment efficiency (extended): isolated LM-head latency and VRAM headroom (Qwen-7B, automotive, bf16). The LM-head matvec is memory-bound at every batch size shown, so the 17–29× latency gain does not translate into system tok/s at batch 1 (Table 4†). VRAM gain = increase in free memory after loading the model: on a 16 GB device, $(16-13.53)/(16-14.60) = 1.77$, i.e. $+77\%$ headroom for KV cache (max batch at 512 tokens: 48→85 sequences).

Batch	Full LM-head (ms)	GRVP (ms)	Speedup	GPU / VRAM Gain
1	0.614	0.021	**29**×	A100 40 GB +4%
32	0.668	0.039	**17**×	RTX 4090 +11%
128	0.740	0.038	**20**×	16 GB edge +**77**%

B Canonical Results: Single Configuration Reference Table

All experimental numbers in the paper derive from the configurations below. Different prose contexts cite different sub-rows (e.g. different K, different adapter quality); this table provides the ground truth.

Table 6: Canonical results reference. All automotive results use the Qwen-7B adapter B (LoRA r=8, SFT then DPO; Section C) on the v5 test set (5,512 records) unless stated. xLAM uses Qwen-7B on the standard test split (5,998 records). VehicleWorld uses the 350-record init-sequence test split.

Metric	Value	Protocol	Where
Automotive EM (constrained / unconstrained)	82.6% / 82.3%	`model.generate()` ± global mask, v5 test (5,512 records); exact value 82.55%, written .826 / 82.6% throughout	Table 1, Table 4
Automotive f_{det}	0.965	TokenDFAAutomaton, v5 test	Table 2
Automotive speedup (K=4, 1k subset, SchemaAutomaton)	7.49×	Uniform K-ablation protocol	Table 3
Automotive speedup (K=8, 1k subset, SchemaAutomaton)	11.70×	Same protocol, K=8	Table 3
Automotive speedup, headline	**64.0**×	TokenDFAAutomaton, uncapped blocks, production path, n=1,000, warm-up 20, `cuda.synchronize()`, median	Table 11
Automotive speedup, best case	86.0±2.8×	TokenDFAAutomaton, uncapped, 1k highest-EM records, clean-AR baseline, mean±std runs 2–5 (run 1: 60.8×; median of 5: 86.9×)	Table 10
Automotive speedup (full v5)	88.33×	TokenDFAAutomaton, uncapped, 5,512 records, 3 runs, wall-clock (pre-protocol)	Section B
Automotive speedup, dead-state records only	14.29×	TokenDFAAutomaton, uncapped, subset of the 1k production-path records	Section D.1
Automotive dead-state call rate	3.6%	Global-mask target, 1,000-record subset (B3)	Section D.1
xLAM f_{det} (XLAMAutomaton)	0.844	Per-record JSON Schema prefix automaton, generated trajectories	Table 2
xLAM f_{det} (static lower bound)	0.572	Schema structure analysis only	Section G
xLAM EM (constrained / unconstrained)	75.9% / 74.8%	`model.generate()` ± mask	Table 2
xLAM speedup	5.86×	XLAMAutomaton, median 5 runs, K=4, 5,998 records	Table 2
VehicleWorld $\lvert R(\mathcal{G}_S, T)\rvert$ / EM / f_{det} / speedup	175 / 21.0% / 46.4% / 1.81×	Qwen-7B, init-sequence split, TokenDFAAutomaton, uncapped blocks	Table 1
Projection tax (measured)	0.2739 nats/step	Qwen-7B, automotive, n=200 records	Section F
GRVP Δ_{EM}	0.0000	9 configs, 3 families (+2 supplementary)	Table 1, Table 14

Clarification of the automotive speedup values. The paper reports seven automotive speedup figures; Table 7 lists the configuration behind each. We headline (3) because it uses GPU-synchronised timing, an unselected record subset, and no run exclusion. The gap between (3) and (4) is not run-to-run noise (64.0 lies more than 7σ from 86.0 ± 2.8); it has two identified causes: (a) the production-path subset contains the 3.6% dead-state calls, which run at only $14.29\times$ (row 6), whereas the highest-EM subset has almost none; and (b) high-EM records have longer deterministic blocks, i.e. smaller ρ in

Corollary 5.7. Because (1)–(2) and (3)–(5) were not collected under a common harness, the $7.49\times$ and $[64.0, 86.0]\times$ results do not isolate the $O(n) \to O(1)$ automaton effect.

Table 7: Map of every automotive speedup figure in the paper (Qwen-7B, adapter B, A100, clean-AR baseline).

#	Speedup	Automaton	Blocks	Records / timing	Role
1	7.49×	SchemaAutomaton	K=4	1k subset, paired baseline	K-ablation (Table 3)
2	11.70×	SchemaAutomaton	K=8	1k subset, paired baseline	K-ablation (Table 3)
3	**64.0**×	TokenDFAAutomaton	uncapped	1k production path, `cuda.synchronize()`, median	**headline** (Table 11)
4	86.0±2.8×	TokenDFAAutomaton	uncapped	1k highest-EM records, runs 2–5 (run 1: 60.8×)	best case (Table 10)
5	88.33×	TokenDFAAutomaton	uncapped	full 5,512 records, 3 runs, pre-protocol wall-clock	historical
6	14.29×	TokenDFAAutomaton	uncapped	dead-state calls within (3)	diagnostic (Section D.1
7	5.62×	SchemaAutomaton	K=1	1k subset, paired baseline	no-amortisation limit

xLAM 84.4% vs 57.2% deterministic fraction. The 84.4% value is measured by the `XLAMAutomaton` on actual generated sequences; the 57.2% is the static lower bound counting only schema-structural positions over all 108,952 reference positions. Table 15 reconciles the two classifications position type by position type. Speedup figures use the generation-time value; the static value is reported as a conservative reference.

C Models, Benchmarks, and Training Recipe

Table 8: Models and benchmarks. EM (Auto) = constrained exact match on the automotive v5 test set. $|V|$ is rounded to the nearest 1k for display; storage figures in Section E use the exact tokeniser sizes (151,665 / 32,768 / 49,152).

Family	Model	$\lvert V\rvert$	EM	Tied
Qwen	Coder-7B	152k	.826	No
Qwen	Coder-1.5B	152k	.518	Yes
Mistral	7B-v0.3	33k	.666	No
SmolLM2	1.7B	49k	.475	No
Benchmarks (Qwen tokeniser)				
VehicleWorld (init sequences, regular)		$\lvert R(\mathcal{G}_S, T)\rvert$=175		
Automotive (Thrift IDL, regular)		$\lvert R(\mathcal{G}_S, T)\rvert$=2,972		
xLAM-60K (JSON Schema, CF)		$\lvert R_{\text{typed}}(\mathcal{G}_S, T)\rvert$=7,303		

Training recipe. All adapters are LoRA (Hu et al., 2022) with rank r=8 on attention and MLP projections, trained by supervised fine-tuning on the query–response pairs of the respective benchmark. The automotive Qwen-7B adapter used for all headline numbers ("adapter B") is additionally refined with a DPO preference stage on the same data (batch size 16, 5 epochs), which raised constrained EM from 78.5% ("adapter A", SFT only) to 82.6%; Table 2 reports both. GRVP and GGSD never touch the adapter: $R(\mathcal{G}_S, T)$ and the automaton depend only on the schema and tokeniser, so any training recipe can be substituted.

C.1 Cross-Tokeniser Reachability

The full five-tokeniser comparison complements the main-paper GRVP results. VehicleWorld retains 175–208 schema tokens across tokenisers; xLAM counts only the typed subvocabulary.

Table 9: Reachable schema tokens for VehicleWorld (VW), Automotive (A) and xLAM (X); X counts $R_{\text{typed}}(\mathcal{G}_S, T)$. Unused is the vocabulary fraction excluded at constrained positions. Physical removal applies to untied all-enumerated models; xLAM uses masks.

TOKENISER	$\lvert V\rvert$	$\lvert R(\mathcal{G}_S,T)\rvert_{VW}$	$\lvert R(\mathcal{G}_S,T)\rvert_{A}$	$\lvert R(\mathcal{G}_S,T)\rvert_{X}$	UNUSED
Qwen-Coder	152k	**175**	2,972	7,303	95–**99.9**%
SmolLM2-1.7B	49k	**206**	2,240	5,404	89–**99.6**%
Mistral-7B	33k	**204**	2,324	5,281	84–**99.4**%
Phi-3-mini	32k	**208**	2,137	5,152	84–**99.4**%
FLAN-T5 (SPM)	32k	**194**	1,937	4,707	85–**99.4**%

Table 10: Automaton implementation comparison (same adapter, same GPU, same model). **The rows are not a single controlled ablation**: the `SchemaAutomaton` rows are the $K{=}4$ and $K{=}8$ entries of the capped-block K-ablation (1,000-record subset, Table 3; baseline = paired clean-AR run), while the `TokenDFAAutomaton` row is the uncapped best-case run on the 1,000 highest-EM records. All use the same baseline protocol (clean autoregressive greedy, no grammar mask); the 25.9–26.5 tok/s spread is run-to-run variance of that baseline. The $7.49\times$, $11.70\times$, $64.0\times$ (production path, Table 11) and $86.0\times$ results describe different configurations; they do not isolate the automaton effect. A fully crossed protocol is future work. †Run 1 excluded as cold-cache ($60.8\times$); runs 2–5 steady-state. ‡Mean±std over runs 2–5; median of all 5 = $86.9\times$. This is the *best-case* figure; the headline is the production-path $64.0\times$ (Table 11).

Automaton	Per-step	Baseline (tok/s)	GGSD	Speedup
`SchemaAutomaton` (string-decode), $K{=}4$	$O(n)$	26.3	197 tok/s	$7.49\times$
`SchemaAutomaton` (string-decode), $K{=}8$	$O(n)$	25.9	304 tok/s	$11.70\times$
`TokenDFAAutomaton` (dict lookup), uncapped	$O(1)$	26.5†	2,291 tok/s	**86.0±2.8**×‡

D GGSD OVERHEAD BREAKDOWN

The `TokenDFAAutomaton` is built offline by tokenising all 15,703 valid output strings and constructing a compressed trie (DFA) over token-ID sequences; each decode step is two dict lookups. Phases 2–3 (tokenisation + trie construction) take 2.3 s on Apple M3 Pro hardware; the full pipeline including Thrift IDL parsing takes 4.2 s (see Table 16 for a hardware-resolved breakdown). At deterministic positions (96.5%), the DFA accepts the single valid token with no model forward pass.

Cost-model fit (Corollary 5.7). A typical automotive call has $N{\approx}40$ tokens and $f_{\text{det}}{=}0.965$, so $(1{-}f_{\text{det}})N{\approx}1.4$ non-deterministic positions and a mean deterministic block of $D{=}f_{\text{det}}/(1{-}f_{\text{det}}){\approx}28$ tokens; a baseline step costs $T_{\text{ar}}{=}37.7$ ms, so the baseline needs $40\,T_{\text{ar}}{=}1{,}508$ ms (26.5 tok/s). Corollary 5.7 gives $S = 1/\{[(1{-}f_{\text{det}}) + f_{\text{det}}/K]\rho + c\}$. We fix c from the per-step automaton cost: $c < 10^{-4}$ for `TokenDFAAutomaton` (automaton queries are 0.06% of wall time, Table 11), and $c{=}0.012$ for `SchemaAutomaton` (fitted at $K{=}4$; ≈0.45 ms of Python string decoding per step). Table 12 then inverts each measured speedup for ρ.

Two readings follow. First, the $K{=}1$ row is *not* a null result: even with no amortisation GGSD is $5.62\times$ faster, because replacing a full `model.generate()` decode step by a bare single-token prefill already removes most of T_{ar} ($\rho{=}0.17$). A naive implementation that re-enters `generate()` for each injected token instead of issuing a bare prefill keeps that overhead and measures only $\approx 1.7\times$; the distinction is an implementation detail with a $3\times$ consequence. Second, the ideal reference $1/(1{-}f_{\text{det}}){=}28.6\times$ assumes $\rho{=}1$ and is therefore *not* an upper bound: with $\rho{=}0.45$ the model predicts $63.5\times$, matching the $64.0\times$ production-path median, and the high-EM subset reaches $86\times$ at $\rho{=}0.33$ because its longer blocks and lower dead-state rate amortise still better.

Worst-case complexity. The prefill at a non-deterministic position has attention cost $O((n{+}D)^2 d)$ for context length n and block length D. For automotive ($D{\leq}40$) this is $O(n^2 d)$, the same order as the baseline. For schemas with long deterministic runs (SQL, code), large D makes the prefill

Table 11: GGSD wall-time overhead breakdown (automotive `TokenDFAAutomaton` vs xLAM `XLAMAutomaton`). Automotive column: production-path run (n=1,000, warm-up 20) with `torch.cuda.synchronize()` around every component timer. xLAM column: the $5.86\times$ run of Table 2 (K=4, 5,998 records) with the same component instrumentation but host-side wall-clock timers (no explicit GPU synchronisation), so its split is indicative; both columns sum to 100%.

Overhead component	Automotive	xLAM	Notes
Target model fwd passes	**98.64%**	**94.6%**	Includes KV prefill of injected blocks at non-det positions
Automaton queries	**0.06%**	**1.7%**	Two dict lookups per step, $O(1)$
KV-cache + other	**1.30%**	**3.7%**	Overhead excluding model fwd

Headline speedup (production path, n=1,000, warm-up 20, `TokenDFAAutomaton`, uncapped blocks, GPU-synchronised, clean-AR baseline): $\mathbf{64.0}\times$ median. The $86.0 \pm 2.8\times$ of Table 10 was obtained on the 1,000 highest-EM records; the gap is due to (a) the production subset containing the 3.6% dead-state calls, which run at $14.29\times$, and (b) longer deterministic blocks on high-EM records — not to run-to-run variance (Table 7).

Table 12: Forward-pass cost ρ implied by each measured automotive speedup under Corollary 5.7. ρ is the cost of one GGSD target forward pass relative to one baseline decode step. It grows with block length and saturates near 0.45: a single-token prefill costs little more than the framework overhead it avoids, while a 29-token prefill is still under half a `generate()` step. A single constant ρ cannot fit the whole range, which is why Corollary 5.7 treats ρ as block-length dependent. The K=8 row is the one *prediction*: holding ρ at its K=4 value gives $12.4\times$ against the measured $11.70\times$ (6% high).

Automaton	Blocks	$(1-f_{\text{det}}) + f_{\text{det}}/K$	c	Measured S	Implied ρ
`SchemaAutomaton`	K=1	1.000	0.012	5.62×	0.17
`SchemaAutomaton`	K=2	0.518	0.012	6.07×	0.30
`SchemaAutomaton`	K=4	0.276	0.012	7.49×	0.44
`SchemaAutomaton`	K=8	0.156	0.012	11.70×	0.47
`TokenDFAAutomaton`	uncapped	0.035	$<10^{-4}$	**64.0**× (production)	**0.45**
`TokenDFAAutomaton`	uncapped	0.035	$<10^{-4}$	86.0× (high-EM)	0.33

quadratic in D and pushes ρ above 1; we recommend capping blocks at $M \leq 64$ tokens (equivalently, K=64).

D.1 Off-Grammar Calls: Dead-State Analysis

What is a dead state? A dead state is a configuration with $R(\mathcal{G}_S, T, q_t) = \emptyset$: the previously committed token was not a valid transition from the DFA state (e.g. the model assembled a service name that does not exist in the schema from reachable subword tokens). Once a dead state is entered, no valid token can complete the call, and the grammar provides no guidance for any subsequent position in that output.

Time complexity. Dead-state detection is $O(1)$: each `reachable_at` call performs a single dict lookup `transitions[state].get(token_id, dead)`. When the result is `dead`, a boolean flag `_is_dead` is latched and all subsequent calls return $\emptyset$ immediately in $O(1)$ without any further trie traversal. The automaton-query latency at off-grammar positions is therefore identical to that at deterministic positions (measured: 0.254 μs/call on Apple M-series hardware).

Behavioural consequence. At a dead-state position, GGSD's *draft oracle* is abandoned for the remainder of that call (Proposition 5.3b). The global-mask logits processor on the target model remains active, so every subsequent token is produced exactly as the baseline decoder produces it. The only consequence is that GGSD provides no acceleration for those positions.

How dead states arise under the global mask. The production `ReachabilityLogitsProcessor` masks the target to $R(\mathcal{G}_S, T)$ (one boolean mask over V), not to the per-step set $R(\mathcal{G}_S, T, q_t)$. At step 0 every service token is in $R(\mathcal{G}_S, T)$, so any service token can be selected. If the model chooses a service that exists in the schema but is not the intended one (e.g. `SeatService` instead of `AirConditionService`), the DFA follows that service's valid branch: this is a *semantic* error, counted in the 14.1% service-name mismatch rate (Table 14), and it is *not* a dead state. A dead state occurs only when the model emits a token that is in $R(\mathcal{G}_S, T)$ but is not a valid transition from the current configuration q_t (a non-existent service name assembled from reachable subwords, an enum value that does not belong to the current parameter, misplaced punctuation). Under a per-state mask this cannot happen; under the global mask it can, and it is the only source of divergence between GGSD and the per-state baseline B of Theorem 5.4.

Empirical dead-state rate and its relation to the 14.1% service-name mismatch. On a 1,000-record subset of the automotive v5 test set (Qwen-7B, adapter B), 96.4% of calls remain in-grammar for the entire output (a valid, non-dead-state DFA path; GGSD output identical to the global-mask baseline token-for-token) and 3.6% enter a DFA dead state. The full-test-set service-name mismatch rate of 14.1% (Table 14) decomposes into these 3.6 pp of dead states (non-existent service names or other invalid transitions) and ≈10.5 pp of *semantic* errors (the model chose an existing but wrong service, so the DFA follows a valid branch); the 1,000-record experiment isolates only the dead-state component, and 14.1% is therefore an upper bound on it. The record-level error budget is consistent with this split: EM= 82.6% leaves 17.4% of records incorrect, of which 14.1 pp have a wrong service name (3.6 pp dead states, 10.5 pp wrong-but-existing services), leaving ≈3.3 pp with the correct service but a wrong parameter — consistent with the 97.5% parameter-name match of Table 14. Dead-state records still achieve 14.29× speedup (deterministic tokens are injected until the dead state is detected, then the call proceeds at baseline speed); the record-level speedup distribution is therefore bimodal, which is why timing subsets with fewer dead-state calls (the high-EM subset of Table 10) report higher aggregate speedups than the unselected production path (Table 11).

Table 13: GGSD position-type breakdown for automotive Qwen-7B (adapter B), distinguishing in-grammar vs. off-grammar calls. The automotive schema is all-enumerated, so every in-grammar position is grammar-masked and none is open-vocabulary.

Position type	Rate	Cost/step	Behaviour
Deterministic ($\lvert R(\mathcal{G}_S, T, q_t)\rvert$=1)	96.5% of steps	$O(1)$	Token injected, no forward pass
Finite non-det. (1<$\lvert R(\mathcal{G}_S, T, q_t)\rvert$<$\lvert V\rvert$)	3.5% of steps	O(fwd)	Terminates the block: one forward pass that also prefills the preceding injected tokens, then constrained greedy within $R(\mathcal{G}_S, T, q_t)$
Dead state ($R(\mathcal{G}_S, T, q_t)$=$\emptyset$)	3.6% of calls (measured, 1k subset); ≤14.1% (svc-mismatch upper bound)†	$O(1)$ detect + O(fwd) fallback	Draft oracle abandoned; global-mask baseline decoding for the rest of the call

†Per-call rates; the first two rows are per-step fractions over in-grammar steps and sum to 100%. The 14.1% bound counts every wrong service name, of which only the non-existent ones (3.6 pp) are dead states.

E GRVP: Full Losslessness Results and Implementation

When $L(\mathcal{G}_S)$ is infinite or too large to enumerate, $R(\mathcal{G}_S, T)$ is obtained instead from the token-level DFA/PDA: for each automaton state q and token t, run the characters of t through the automaton ($O(|Q| \cdot |V| \cdot \ell_{\max})$ in total) and collect every t with a defined transition. Both variants produce the same set by Definition 2.2; we use enumeration for the two regular schemas (finite $L(\mathcal{G}_S)$) and the per-record schema automaton for xLAM (Section I).

Algorithm 1 GRVP: Grammar-Reachable Vocabulary Pruning (enumeration variant for finite $L(\mathcal{G}_S)$)

Require: Grammar $\mathcal{G}_S$ with finite $L(\mathcal{G}_S)$, tokeniser T, LM-head weight matrix $W \in \mathbb{R}^{|V| \times d}$
Ensure: Pruned LM-head $W_R \in \mathbb{R}^{|R| \times d}$, index map $\pi : [|R|] \to [|V|]$

$L \leftarrow$ enumerate$(\mathcal{G}_S)$ {all valid output strings; $|L| = 15{,}703$ for automotive}
$R \leftarrow \{\texttt{EOS}\} \cup T.\texttt{control_tokens}$ {EOS is part of every $T(w)$ (Def. 2.2); BOS/PAD kept for framework compatibility}
for $s \in L$ **do**
  $\texttt{ids} \leftarrow T.\texttt{encode}(s)$ {BPE tokenisation; each ID is BPE-exact}
  $R \leftarrow R \cup \text{set}(\texttt{ids})$ {total $O(|L(\mathcal{G}_S)| \cdot L_{\max})$}
end for
$\pi \leftarrow \text{sorted}(R)$ {stable index for reproducibility}
$W_R \leftarrow W[\pi, :]$ {row-select; $O(|R| \cdot d)$ copy}
return W_R, π

E.1 Physical Implementation of the Compact LM Head

GRVP physically removes the LM-head rows corresponding to $V \setminus R(\mathcal{G}_S, T)$ from the model's weight matrix. For a model with vocabulary V ($|V| = N$) and hidden dimension d, the full LM head is $W \in \mathbb{R}^{N \times d}$. GRVP replaces it with the compact LM head $W_R(\mathcal{G}_S, T) \in \mathbb{R}^{|R(\mathcal{G}_S,T)| \times d}$ obtained by row subsetting:

$$W_R(\mathcal{G}_S, T) = W[\text{retained_ids}, :]$$

where retained_ids $\in \mathbb{Z}^{|R(\mathcal{G}_S,T)|}$ is the sorted array of token IDs in $R(\mathcal{G}_S, T)$ (schema tokens, EOS, and control tokens). The compact head is stored alongside a vocabulary mapping:

- $\phi : \{0, \ldots, |R(\mathcal{G}_S, T)| - 1\} \to R(\mathcal{G}_S, T)$ (*new-to-old*): maps compact indices back to original vocab IDs.
- $\phi^{-1} : V \to \{-1\} \cup \{0, \ldots, |R(\mathcal{G}_S, T)| - 1\}$ (*old-to-new*): maps original IDs to compact indices (-1 for unreachable tokens).

At inference time, the forward pass computes:

$$\ell_t = W_R(\mathcal{G}_S, T)\, h_t \in \mathbb{R}^{|R(\mathcal{G}_S,T)|}$$

This is a matrix-vector product over $|R(\mathcal{G}_S, T)|$ rows instead of N rows, reducing the output-projection FLOPs by a factor of $|R(\mathcal{G}_S, T)|/N = 2{,}972/151{,}665 \approx 2\%$ (i.e., 98% FLOP reduction in the output projection layer). Before softmax, the logits are re-expanded to $\mathbb{R}^N$ with $-\infty$ for unreachable positions, ensuring compatibility with the target model's verification in GGSD. This implementation is used for the untied models on the two regular schemas (VehicleWorld, Automotive). For mixed-type schemas (xLAM; Theorem 4.3) a free-string position requires all N rows, so the full head must remain resident; the compact head can still serve typed positions (a two-head deployment), but the reported xLAM results use the mathematically equivalent mask-only implementation, and no storage saving is claimed.

E.2 Handling Tied-Weight Architectures

Some models (e.g., Qwen-1.5B) use *tied* embedding and LM-head weights: $E = W^\top$ (the embedding matrix is the transpose of the LM head). Physically removing LM-head rows in this case would also remove the corresponding input embedding rows, preventing the model from attending to those tokens in the input prompt. Since prompts may contain arbitrary tokens (not just schema-reachable ones), this is unsafe.

For tied-weight models, GRVP applies the constraint via a `LogitsProcessor` mask instead:

1. The embedding matrix E and LM head W are left unchanged (no physical row removal).
2. At each decode step, the logits processor sets $\ell_t[v] = -\infty$ for all $v \notin R(\mathcal{G}_S, T, q_t)$ before softmax.
3. The constrained distribution is identical to the GRVP compact-head distribution by Lemma 3.1: the renormalisation produces the same result.

Theorem 4.1 ($\Delta_{\mathrm{EM}} = 0.0000$) holds for tied-weight models under the logits-processor implementation, as validated empirically for Qwen-1.5B (Table 1). The storage saving is not realised for tied-weight models; decoupling the embeddings (a separate 21 MB copy of the reachable rows with an index-select) is left as future work (§9). This is also why Corollary 3.2 is stated for LM-head rows only: input-embedding rows of unreachable tokens are still read whenever those tokens occur in the prompt.

E.3 Distributional Validation: Beyond Exact Match

To confirm that GRVP preserves the *distribution*, not just the greedy argmax, we run a direct logit comparison between the compact-head model and the full model. For each test record, we compute the model's hidden state h_t once, then apply both heads:

$$\ell_t^{\text{full}} = W h_t \in \mathbb{R}^N, \qquad \ell_t^{\text{compact}} = W_R(\mathcal{G}_S, T) h_t \in \mathbb{R}^{|R(\mathcal{G}_S,T)|}$$

The key relationship: for any $v \in R(\mathcal{G}_S, T)$, $\ell_t^{\text{compact}}[\phi^{-1}(v)] = \ell_t^{\text{full}}[v]$ exactly (same weights, same hidden state, same computation). After softmax restricted to $R(\mathcal{G}_S, T)$, both distributions are identical.

Measured logit comparison (Qwen-7B, automotive, $n = 200$ records). Maximum absolute logit difference over $v \in R(\mathcal{G}_S, T)$: $\max_t |(\ell_t^{\text{compact}} - \ell_t^{\text{full}}[R(\mathcal{G}_S, T)])| = \mathbf{0.000}$ — exactly zero, as expected since the compact head is a row subset of the same bf16 tensor applied to the same hidden state. The maximum KL divergence between the two *softmax* outputs is 7.55×10^{-4} nats; it arises from reduced-precision (bf16) accumulation in softmax over vectors of different length (2,972 vs 151,665 entries, the latter with $-\infty$ masks), not from any logit difference. Greedy argmax is identical on every step, confirming $\Delta_{\mathrm{EM}} = 0.0000$ and validating Theorem 4.1.

E.4 Storage Analysis Across Model Families

Model	$\lvert V\rvert$	Tied	d	LM head (bf16)	GRVP head	Saving
Qwen-7B	151,665	No	3,584	1,087 MB	21 MB	1,066 MB (98.0%)
Qwen-1.5B	151,665	Yes	1,536	466 MB	*n/a*	*logits-proc. only*
Mistral-7B	32,768	No	4,096	268 MB	19 MB	249 MB (92.9%)
SmolLM2-1.7B	49,152	No	2,048	201 MB	9 MB	192 MB (95.4%)

All sizes in decimal megabytes (10^6 bytes) computed as $|V| \times d \times 2$ and $|R(\mathcal{G}_S, T)| \times d \times 2$ with the exact tokeniser vocabulary sizes shown (the "152k/33k/49k" labels elsewhere are rounded for display) and the automotive $R(\mathcal{G}_S, T)$. Qwen checkpoints pad the head tensor to 152,064 rows; the 399 padding rows are unreachable and are removed as well, adding ≈3 MB to the Qwen-7B saving. xLAM (mixed-type) is omitted because the full head remains resident (Theorem 4.3).

The savings are largest for models with large vocabularies and untied weights (Qwen-7B: 98.0%). For smaller vocabulary models (Mistral-7B: 33k tokens), the LM head is already smaller, but GRVP still achieves 92.9% head compression. The automotive $|R(\mathcal{G}_S, T)| = 2{,}324$ for Mistral and $|R(\mathcal{G}_S, T)| = 2{,}240$ for SmolLM2 lead to similarly compact heads (≈ 19 MB and 9 MB respectively).

E.5 Full Nine-Configuration GRVP Results with All Metrics

Three observations emerge:

(1) Service-name matching and EM respond differently to frequency pruning. For Qwen-7B/Auto., frequency-pruned service match is 76.1% vs GRVP's 85.9%, while EM falls from 82.6% to 0%; the EM decrease is larger. Service names (e.g., `AirDistributionService`) are domain-specific and have zero training frequency; a frequency-ranked row set systematically misses them, causing the model to hallucinate generic service names.

(2) 1.5B models are more robust to the frequency signal. Qwen-1.5B frequency-pruned EM (39.5% automotive, 61.5% xLAM) versus 0% for 7B models. Smaller models generate simpler outputs with fewer domain-specific identifiers, so frequency-based pruning at the same budget $|R(\mathcal{G}_S, T)|$ happens to include most required tokens. This does not imply frequency pruning is safe—it is

Table 14: Complete GRVP results across all nine configurations (superset of Table 1). EM = exact match (canonical, training-matched test split). Δ = GRVP EM $-$ constrained-baseline EM. Svc = service-name match. Param = parameter-name match (not defined for VehicleWorld's device-line format). Freq. = frequency-ranked row set of size $|R(\mathcal{G}_S, T)|$ substituted for the grammar set (stress test).

Model	Bench.	$\lvert R(\mathcal{G}_S, T)\rvert$	EM	Svc	Param	Δ (GRVP)	Freq. EM[§]	Freq. Svc
Qwen-7B	VW	175	.210	—	—	**.0000**	.000	—
Qwen-1.5B	VW	175	.175	—	—	**.0000**	.000	—
Qwen-7B	Auto.	2,972	.826	.859	.975	**.0000**	.000	.761
Qwen-7B	xLAM	7,303	.759	.853	.942	**.0000**	.000	.782
Qwen-1.5B	Auto.	2,972	.518	.633	.891	**.0000**	.395	.556
Qwen-1.5B	xLAM	7,303	.614	.772	.918	**.0000**	.615	.774
Mistral-7B	Auto.	2,324	.666	.779	.943	**.0000**	.000	.787
Mistral-7B	xLAM	5,281	.394	.571	.887	**.0000**	.000	.574
SmolLM2-1.7B	Auto.	2,240	.475	.604	.872	**.0000**	.345	.504

schema-dependent: if the 1.5B model were deployed on a more complex schema with rare identifiers, the same failure mode would occur.

(3) $\Delta = 0.0000$ **is robust across grammar classes.** The zero delta holds for both regular (VehicleWorld, Thrift IDL) and context-free (JSON Schema) grammars, for both untied and tied-weight architectures, and for models ranging from 1.5B to 7B parameters.

F Distributional Faithfulness of GGSD

F.1 The Gap Between μ^{proj} and $\mu^\star$

Nie et al. (2026) prove a fundamental impossibility: any speculative decoder that uses *local* grammar masking (masking at each step using only the current automaton state, without look-ahead) with standard Leviathan rejection sampling does not sample from the true grammar-conditional distribution $\mu^\star$, but instead from the *locally-projected distribution* μ^{proj}.

Formally, let $y \in V^*$ be an output sequence. The true grammar-conditional distribution conditions on the full output being valid:

$$\mu^\star(y \mid x) = p_M(y \mid x) \cdot \mathbf{1}[y \in L(\mathcal{G}_S)] \;/\; Z(x) \tag{5}$$

where $Z(x) = \sum_{y \in L(\mathcal{G}_S)} p_M(y \mid x)$ is the partition function.

The locally-projected distribution is instead defined step-by-step: at each position t, mask to $R(\mathcal{G}_S, T, q_t)$ and renormalise independently:

$$\mu^{\text{proj}}(y_t \mid x, y_{<t}) = p_M(y_t \mid x, y_{<t}) \cdot \mathbf{1}[y_t \in R(\mathcal{G}_S, T, q_t)] \;/ \sum_{v \in R(\mathcal{G}_S, T, q_t)} p_M(v \mid x, y_{<t}) \tag{6}$$

These differ whenever a locally-valid token leads to a dead state (a prefix from which no valid completion exists). Alpay & Senturk (2026) derive matching sharp total-variation (TV) and KL distortion bounds on this gap via a Doob h-transform view: $\mu^\star$ is the Doob h-transform of p_M with $h(y) = \Pr_{p_M}[\text{valid completion} \mid y]$, and μ^{proj} corresponds to the degenerate case $h \equiv 1$. Nie et al. (2026) show that the TV gap can reach 0.996 on Dyck grammars with Qwen3-8B, motivating the *future-validity function* $\Phi_t(y) = \Pr_{p_M}[\text{valid completion} \mid y]$ as the correction term.

F.2 Regular Grammars: Gap Is Zero

For a regular grammar, $\mu^{\text{proj}} = \mu^\star$ if and only if the grammar automaton is *dead-state-free*: every prefix accepted by the DFA from the start state can be extended to at least one valid complete output. Formally, a DFA $A_{\mathcal{G}_S}$ is dead-state-free if for all reachable states q, $L(A_{\mathcal{G}_S}, q) \neq \emptyset$ (the language of strings accepted from q is non-empty).

Dead-state analysis for Thrift IDL. The Thrift IDL output language $L(\mathcal{G}_S)$ consists of strings of the form:

$$\texttt{ServiceName.methodName(param}_1\texttt{=val}_1\texttt{, ..., param}_n\texttt{=val}_n\texttt{)}$$

Every valid prefix of such a string can always be extended: after the opening parenthesis, at least one parameter assignment is always possible; after each comma, the next parameter name is deterministic; after each =, at least one value is always available.

All automotive parameters have finite declared value sets — 600+ enumerated constants plus bounded numeric ranges (§2.3) — so $R(\mathcal{G}_S, T, q_t)$ is finite and non-empty at every reachable configuration and *every* decode position is grammar-masked. The DFA therefore has no dead states at any reachable configuration: any locally valid token either advances it to an accepting state or to a state from which further valid tokens exist.

Conclusion. The automotive grammar is prefix-closed at every position, so $\mu^{\text{proj}} = \mu^\star$ exactly and the TV gap is zero under *greedy decoding*; under stochastic sampling, step-wise renormalisation over $R(\mathcal{G}_S, T, q_t)$ alters path probabilities relative to the true grammar-conditional distribution even for dead-state-free regular grammars (Nie et al., 2026). All empirical results use greedy decoding (Δ_{EM}=0 is unaffected). Our empirical results ($\alpha = 1.00$ at every position) are consistent with this theoretical guarantee. The dead states reported in Section D.1 are an artefact of the *global*-mask production target, not of the grammar: under the per-state mask analysed here they cannot occur.

F.3 Context-Free Grammars: Gap May Be Non-Zero

For context-free schemas (xLAM / JSON Schema) with open-vocabulary string parameters, prefix-closure does not hold in general. A partial JSON object such as {`"city": "New Y"` is locally valid (accepted by the CFL automaton at this prefix) but the continuation is unconstrained—any string token completing the city name is valid, so the "dead state" concept is not applicable, but the string value space is infinite.

More importantly, for xLAM's open-vocabulary positions (42.8% by static schema analysis; 14.3% of steps actually reached at generation time), the automaton explicitly returns $R(\mathcal{G}_S, T, q_t) = V$ (no finite reachable set), and GGSD does not apply any grammar mask. These positions use unconstrained model-greedy decoding, so $\mu^{\text{proj}} = p_M$ at these positions—neither better nor worse than standard decoding.

Why EM evaluation is unaffected. Our evaluation uses *exact-match accuracy under greedy (argmax) decoding*. The argmax is determined solely by the position of the maximum in the logit vector:

$$\hat{y}_t = \arg\max_{v \in V} p_M(v \mid x, y_{<t})$$

This is unchanged by the distributional gap between μ^{proj} and $\mu^\star$: the renormalisation in μ^{proj} does not change which token is assigned the highest probability within $R(\mathcal{G}_S, T, q_t)$. The measured projection tax of 0.2739 nats/step (Section F.4) confirms the model concentrates substantial mass on $R(\mathcal{G}_S, T, q_t)$ after fine-tuning; the argmax under μ^{proj} equals the argmax under $\mu^\star$ as long as the highest-probability valid token is also the overall argmax, which is confirmed empirically by $\Delta_{\text{EM}} = 0.0000$.

Sampling inference. For non-greedy (temperature sampling) inference, the TV gap between μ^{proj} and $\mu^\star$ does matter: samples from μ^{proj} may assign non-trivial probability to dead-end prefixes for context-free schemas. Incorporating future-validity corrections (Nie et al., 2026) for faithful sampling inference in the CF case is left as future work.

F.4 Projection Tax: Cost of Grammar-Constrained Decoding

The *projection tax* (Reddy et al., 2026) at decode step t measures how far the local mask moves the model's distribution:

$$\text{tax}_t = \text{KL}(p_M^{\text{constrained}} \| p_M^{\text{unconstrained}}) = \log \frac{1}{m_t}$$

where $m_t = \sum_{v \in R(\mathcal{G}_S, T, q_t)} p_M(v \mid x, y_{<t})$ is the total probability mass on grammar-valid tokens before renormalisation (we write m_t rather than α_t to avoid a clash with the speculative acceptance rate α). When $m_t \approx 1$ (model concentrates mass on valid tokens), $\text{tax}_t \approx 0$: constrained decoding is free.

Measured projection tax (Qwen-7B, automotive, $n = 200$ records). Mean projection tax: $\mathbf{0.2739}$ nats/step. Because the tax is a log, $\exp(-0.2739) = 0.76$ is the *geometric* mean of m_t over steps (not the arithmetic mean, which is at least as large by Jensen's inequality). The fine-tuned model therefore places roughly three quarters of its pre-mask probability mass on grammar-valid tokens at a typical step—far above the $|R(\mathcal{G}_S, T)|/|V| \approx 2\%$ that the global reachable fraction would give under a uniform distribution (and above the still smaller per-state fraction $|R(\mathcal{G}_S, T, q_t)|/|V|$)—but with a non-trivial residual on invalid tokens.

The positive projection tax is consistent with constrained decoding being a (small) net benefit under greedy decoding: it removes off-schema argmax tokens where they occur. Empirically constrained decoding improves EM by $+0.3$ pp on automotive Qwen-7B ($82.3\% \rightarrow 82.6\%$, Table 4) and $+1.1$ pp on xLAM ($74.8\% \rightarrow 75.9\%$, Table 6).

G xLAM GGSD Analysis

G.1 Position-Type Breakdown

For the xLAM / JSON Schema benchmark, we analyse f_{det} over all 5,998 test records (108,952 total decode positions) using the Qwen-7B tokeniser. The static breakdown below is schema- and tokeniser-derived and model-independent.

Position type	Count	Fraction	Description
Function name tokens	23,412	21.5%	Uniquely determined by schema
Parameter name tokens	28,604	26.3%	Uniquely determined per function
Structural punctuation	10,251	9.4%	`(`, `)`, `=`, `,`
Total deterministic	**62,267**	**57.2%**	$\lvert R(\mathcal{G}_S, T, q_t)\rvert = 1$; $\alpha_{\text{GGSD}} = 1.00$
Open-vocab string values	46,685	42.8%	$R(\mathcal{G}_S, T, q_t) = V$; no grammar constraint, fallback to greedy

The static deterministic fraction $f_{\text{det}} = 0.572$ is substantially lower than automotive's $f_{\text{det}} = 0.965$ for two reasons: (1) xLAM's per-record JSON Schema typically defines 3–10 parameters with free-string values (open vocabulary), whereas automotive's Thrift IDL uses exclusively enumerated parameters; (2) function names and parameter names, though uniquely determined by the schema, represent a smaller fraction of total tokens than in the more heavily-typed automotive schema.

Static (57.2%) versus generation-time (84.4%) classification. The static count classifies every position of the *reference* calls by schema type alone and treats every string-valued parameter as open-vocabulary for its full reference length. The `XLAMAutomaton` classifies each position of the *generated* trajectory by the actual admissible set $R(\mathcal{G}_S, T, q_t)$ at that step. Table 15 aligns the two. The 27.2 pp that move from "open-vocabulary" to "deterministic" come from two sources that static analysis cannot see: (i) parameters whose schema declares an `enum` or boolean type are open-vocabulary to the static count (they are "values") but have a finite $R(\mathcal{G}_S, T, q_t)$ at generation time, and once the first subword of the chosen value is emitted the remaining subwords and the closing quote are forced; (ii) generated string values are shorter than the reference values counted statically, so fewer open-vocabulary steps are actually reached (14.3% vs 42.8%). The 1.3% finite non-deterministic bucket consists of the first token of enum/boolean values and of the function choice when a record offers several functions. We did not instrument the split between (i) and (ii), so Table 15 reports the reclassified mass as a single row.

The static figure is therefore a lower bound on f_{det}; speedup analyses (Table 2) use the generation-time value.

Table 15: Reconciliation of the static and generation-time position classifications on xLAM (Qwen-7B, 5,998 records).

Position class	Static (reference calls)	Genera
Deterministic ($\lvert R(\mathcal{G}_S, T, q_t)\rvert$=1)	57.2%	
of which schema-structural (names, punctuation)	57.2%	
of which reclassified from static open-vocab (enum/boolean continuations, early string termination)	—	
Finite non-deterministic (1<$\lvert R(\mathcal{G}_S, T, q_t)\rvert$<$\lvert V\rvert$)	(counted in open-vocab)	
Open-vocabulary ($R(\mathcal{G}_S, T, q_t)$=V)	42.8%	
Total	100%	

G.2 Why VOCABTRIM Coverage Differs Between Benchmarks

The VT coverage at structural (deterministic) positions is 96% on xLAM versus 0% on automotive. This reflects a fundamental difference in identifier vocabulary:

Automotive identifiers are domain-specific. Enum values such as `PULSE_4`, `ROW_1`, `LUMBAR_SUPPORT_UPPER`, and service names such as `AirDistributionService` are proprietary automotive terminology rarely or never present in general pre-training corpora (Wikipedia, GitHub, Common Crawl). Their training-split frequency is zero for 774 of 2,972 reachable tokens (26.0%) across 35,329 training records, reflecting genuinely low-traffic endpoints rather than dataset incompleteness. VOCABTRIM cannot recover them at any frequency budget.

xLAM identifiers are drawn from common vocabulary. Function names such as `get_stations`, `v1_holidays`, `search_products` decompose into BPE subwords (`get`, `_stations`, `v1`, `_holiday`, `s`) that appear frequently in general pre-training data. Only 4% of xLAM structural positions involve compound identifiers split into rare subwords absent from the top-k frequency set (e.g., `get_weather_forecast` → `_weather_forecast` is less common). This confirms Theorem 6.2: frequency-based pruning degrades gracefully when schema identifiers overlap with common vocabulary, but fails catastrophically when they do not—and the failure mode cannot be predicted without inspecting the schema.

Why 96% position coverage still gives 0% exact match. The 96% figure is a *position-level* rate for the VOCABTRIM *drafter*: a miss costs one rejected draft, and the target still emits the right token. The Freq. EM column of Table 1 is a different experiment: the same-size frequency-ranked row set replaces $R_{\text{typed}}(\mathcal{G}_S, T)$ in the *target* head, so a missing row cannot be emitted at all. Exact match is a *record-level* metric that requires every token of the call: a Qwen-7B xLAM call has $\approx$18 positions (108,952/5,998), about 15 of them typed, so even independent 4% misses would leave only $\approx 0.96^{15} \approx 54\%$ of calls intact, and the misses are not independent — they concentrate on the rarest subword of each function name, which every call must produce. When that row is absent the target cannot emit it; under the mask it emits the highest-scoring *available* subword instead (e.g. `get_weather_forecast` becomes `get_weather(` followed by whatever completion the model prefers), the function name is wrong, the automaton for that call follows a different branch, and the record fails. In our run no record was fully covered, giving EM $= 0.000$; the Freq. Svc column (78.2%) shows that most calls still name an *existing* function, i.e. the damage is one wrong subword per call rather than wholesale garbage. The 1.5B model, whose outputs use fewer and more common identifiers, retains 61.5% EM under the same stress test.

G.3 GGSD at Open-Vocabulary Positions

At open-vocabulary positions (42.8% of xLAM positions by static analysis), the grammar automaton returns $R(\mathcal{G}_S, T, q_t) = V$ (free-string parameters). GGSD falls back to unconstrained model-greedy decoding for these positions. The speculative decoding framework does not apply: no grammar constraint exists, and any token could appear. The model's fine-tuned weights handle these positions via the standard causal LM forward pass.

GGSD's benefit is *additive* over positions: it removes the forward pass at deterministic positions and leaves open-vocabulary positions unchanged. With the static fraction $f_{\text{det}} = 0.572$, the ideal reference of Corollary 5.7 is $1/(1 - f_{\text{det}}) = 2.33\times$ (with $\rho = 1$); with the generation-time fraction 0.844 it is $6.4\times$, and the measured $5.86\times$ corresponds to $\rho \approx 1.09$. Neither figure is a guaranteed bound; both are cost-model references.

G.4 DOMINO Comparison in the Context-Free Setting

DOMINO (Beurer-Kellner et al., 2024) uses a count-based model $P(l \mid \alpha, \beta)$ trained on calibration data over grammar states as its draft oracle. On JSON generation with schema it reports $1.77\times$ speedup (Mistral-7B) with data-driven proposals.

GGSD differs in three fundamental ways in the CF setting:

(i) Zero data requirement. The grammar automaton requires only the JSON Schema definition—no calibration data, no training. For DOMINO, the count model must be trained on representative inputs for each new API definition, which may not be available at deployment time.

(ii) Provable acceptance rate. Propositions 5.1–5.2 guarantee $\alpha_{\text{GGSD}} = 1.00$ at all 85.7% of grammar-constrained xLAM positions (84.4% deterministic + 1.3% hi-con, XLAMAutomaton; 57.2% static lower bound on the deterministic fraction; Table 15). DOMINO's empirical acceptance rate depends on how well the count model matches the target distribution; for rare API schemas not covered in calibration, acceptance can drop substantially.

(iii) GRVP integration. GGSD is paired with GRVP, which removes unreachable weight rows from the *target* model. DOMINO operates only at the drafter level and does not address target-model compression, so the 1,066 MB storage saving on the all-enumerated automotive schema has no counterpart in DOMINO.

H Complete Proof Details

Throughout, p_M^{masked} denotes the per-state-masked distribution of Definition 3.3, and $M{\restriction}R'$ the model whose LM-head rows outside R' are removed (equivalently, whose logits outside R' are $-\infty$). For a valid output $y \in L(\mathcal{G}_S)$ we write q_τ for the automaton configuration reached after $y_{<\tau}$; by construction $y_\tau \in R(\mathcal{G}_S, T, q_\tau)$ for every τ.

H.1 Proof of Theorem 4.1: GRVP is the Unique Minimum ODP Reduction

Positivity (replaces any coverage assumption). For any input x, any $y \in L(\mathcal{G}_S)$ and any step τ, the masked distribution is a softmax over the finite set $R(\mathcal{G}_S, T, q_\tau) \ni y_\tau$ of finite logits, so $p_M^{\text{masked}}(y_\tau \mid x, y_{<\tau}) > 0$ and hence $p_M^{\text{masked}}(y \mid x) = \prod_\tau p_M^{\text{masked}}(y_\tau \mid x, y_{<\tau}) > 0$. No property of the training data is used.

Sufficiency ($R(\mathcal{G}_S, T)$ is ODP). Fix x, $y \in L(\mathcal{G}_S)$ and a step t. For $v \notin R(\mathcal{G}_S, T) = \bigcup_q R(\mathcal{G}_S, T, q_t)$ we have $v \notin R(\mathcal{G}_S, T, q_t)$, so the per-state mask already sets $\ell_v = -\infty$. Removing row v from W_{out} also sets $\ell_v = -\infty$; for $v \in R(\mathcal{G}_S, T)$ the logit is unchanged because the same row of the same matrix multiplies the same h_t. Hence the masked logit vectors of M and $M{\restriction}R(\mathcal{G}_S, T)$ coincide entrywise at every step, and so do the softmax outputs: $p^{\text{masked}}_{M{\restriction}R(\mathcal{G}_S,T)}(y \mid x) = p_M^{\text{masked}}(y \mid x)$.

Necessity (every ODP set contains $R(\mathcal{G}_S, T)$). Let R' be ODP and suppose $t_0 \in R(\mathcal{G}_S, T) \setminus R'$. By Definition 2.2 there is $w \in L(\mathcal{G}_S)$ with $t_0 \in T(w)$; let $y = T(w)$ and let τ be a position with $y_\tau = t_0$. Since y is valid, $t_0 \in R(\mathcal{G}_S, T, q_\tau)$: the per-state mask does *not* remove t_0 at step τ. Row removal does: in $M{\restriction}R'$ the logit of t_0 is $-\infty$, so $p^{\text{masked}}_{M{\restriction}R'}(t_0 \mid x, y_{<\tau}) = 0$ and therefore $p^{\text{masked}}_{M{\restriction}R'}(y \mid x) = 0$. By positivity $p_M^{\text{masked}}(y \mid x) > 0$, contradicting ODP. Hence $R(\mathcal{G}_S, T) \subseteq R'$.

Uniqueness. Let R_1 be any ODP set that is minimal under inclusion. By necessity $R(\mathcal{G}_S, T) \subseteq R_1$; by sufficiency $R(\mathcal{G}_S, T)$ is ODP; by minimality of R_1 no proper ODP subset of R_1 exists, so $R_1 = R(\mathcal{G}_S, T)$. Thus $R(\mathcal{G}_S, T)$ is the least element of the ODP family and the unique minimal one. □

H.2 Proof of Theorem 4.3: Mixed-Type Schemas

Let Q_{typed} and $R_{\text{typed}}(\mathcal{G}_S, T)$ be as in the theorem, and consider the decoder B_{mix} that applies the per-state mask at configurations $q \in Q_{\text{typed}}$ and no mask at free-string configurations ($R(\mathcal{G}_S, T, q_t) = V$). A compact head with rows R' serving the typed configurations is *typed-ODP* if, with the full head used at free-string configurations, the output distribution of B_{mix} is unchanged for every x and every $y \in L(\mathcal{G}_S)$. ***Sufficiency:*** at $q \in Q_{\text{typed}}$ the argument of the previous subsection applies verbatim with $R(\mathcal{G}_S, T)$ replaced by $R_{\text{typed}}(\mathcal{G}_S, T) = \bigcup_{q \in Q_{\text{typed}}} R(\mathcal{G}_S, T, q_t)$; at free-string configurations the full head is used, so nothing changes. ***Necessity:*** if $t_0 \in R_{\text{typed}}(\mathcal{G}_S, T) \setminus R'$, then $t_0 \in R(\mathcal{G}_S, T, q)$ for some typed configuration q; the grammar admits a valid y passing through q with $y_\tau = t_0$, and removing row t_0 zeroes $p(y \mid x)$ exactly as before. ***Uniqueness*** follows as above. Finally, at a free-string configuration every $v \in V$ has $R(\mathcal{G}_S, T, q_t) = V$, so a valid output can contain any token there and no row can be removed from the head used at such positions; a static single-head deployment must therefore keep all $|V|$ rows resident, and GRVP reduces per-step FLOPs at typed positions (via the compact head) but not storage. When $Q_{\text{typed}} = Q$ there are no free-string configurations and the statement is Theorem 4.1. □

H.3 Proof of Theorem 5.4: GGSD Losslessness

Let B be constrained greedy decoding with the per-state mask (open-vocabulary positions decoded unconstrained), and let GGSD run with any block budget $K \geq 1$. We show by induction on t that GGSD and B commit the same token at every step, hence produce the same output and the same automaton trajectory; no acceptance rates are needed for this argument (they belong to Theorem 5.6).

Induction hypothesis. $y_{<t}$ (and therefore q_t and $R(\mathcal{G}_S, T, q_t)$) is identical for GGSD and B. This holds for $t = 1$.

Case 1: $R(\mathcal{G}_S, T, q_t) = \{v^\star\}$ **(deterministic).** B masks all logits except $\ell_{v^\star}$ to $-\infty$, so its argmax is $v^\star$ for every hidden state (Lemma 3.1). GGSD commits $v^\star$ without a forward pass (Proposition 5.1). Same token.

Case 2: $1 < |R(\mathcal{G}_S, T, q_t)| < |V|$ **(finite non-deterministic).** GGSD issues one forward pass (which also prefills any injected block; the prefill reproduces exactly the hidden state h_t that sequential decoding would have produced, because attention over the same prefix is order-independent of how the KV cache was filled) and commits $\arg\max_{v \in R(\mathcal{G}_S, T, q_t)} p_M(v \mid x, y_{<t})$ (Proposition 5.2). B computes the same h_t, applies the same mask, and takes the same argmax. Same token, for every K.

Case 3: $R(\mathcal{G}_S, T, q_t) = V$ **(open-vocabulary).** Neither decoder applies a mask; both commit $\arg\max_v p_M(v \mid x, y_{<t})$ (Proposition 5.3a). Same token.

Dead states. Under B every committed token is a valid transition, so $R(\mathcal{G}_S, T, q_t) \neq \emptyset$ at every step and the three cases are exhaustive. (Dead states arise only when the target is masked with the global set $R(\mathcal{G}_S, T)$; in that configuration GGSD falls back to the *same* global-mask decoder for the remainder of the call, so GGSD and the global-mask decoder also agree token-for-token — but the global-mask decoder is not B, see Section D.1.)

By induction, the two decoders emit identical token sequences, so $p_{\text{GGSD}}(v \mid x, y_{<t}) = p_B(v \mid x, y_{<t})$ (both are point masses on the same token under greedy decoding; under sampling the same argument shows that GGSD draws from the same masked distribution at non-deterministic positions) and $\Delta_{\text{EM}} = 0$ for every K. □

H.4 Proof of Theorem 5.6: Zero-Forward-Pass Optimality

Fix a deterministic position, $R(\mathcal{G}_S, T, q_t) = \{v^\star\}$, so that the target's masked distribution is $p_M^{\text{constrained}} = \delta_{v^\star}$.

Acceptance identity. For a speculative drafter with proposal distribution p_D, Leviathan's rule accepts a drafted token v with probability $\min\{1, p_M^{\text{constrained}}(v)/p_D(v)\}$. The probability that the

drafted token at this position is accepted (equivalently, the expected number of accepted tokens per drafted position) is

$$\alpha_D = \sum_v p_D(v)\ \min\Big\{1, \tfrac{p_M^{\text{constrained}}(v)}{p_D(v)}\Big\} = \sum_v \min\{p_D(v),\, p_M^{\text{constrained}}(v)\} = 1 - \mathrm{TV}(p_D, \delta_{v^\star}) = p_D(v^\star).$$

(The earlier per-token quantity $\min\{1, 1/\beta\}$ equals 1 for $\beta = p_D(v^\star) < 1$; it is the *conditional* acceptance given that $v^\star$ was drafted, and must be weighted by β to obtain α_D.) Hence $\alpha_D \leq 1$ with equality iff $p_D = \delta_{v^\star}$. For the grammar automaton $p_D = \delta_{v^\star}$ and $\alpha = 1$.

Forward-pass lower bound without grammar-state access. A drafter without access to $A_{\mathcal{G}_S}$ receives only x and $y_{<t}$ and cannot certify that $|R(\mathcal{G}_S, T, q_t)| = 1$. In the speculative-decoding protocol every drafted token is verified against the target distribution before it is committed, and computing $p_M(\cdot \mid x, y_{<t})$ requires a target forward pass; a drafter that skipped verification would, on the non-deterministic positions it cannot distinguish from this one, commit tokens with $p_M^{\text{constrained}}(v) < 1$ and thereby violate losslessness. Thus at least one target forward pass is incurred for the block containing this position. Conversely, with grammar-state access the automaton reads $|R(\mathcal{G}_S, T, q_t)| = 1$ in $O(1)$ and commits $v^\star$; by Proposition 5.1 this is exactly the token B would commit, so no verification is needed and zero forward passes suffice. Any system that exposes $R(\mathcal{G}_S, T, q_t)$ (a constrained-decoding engine such as Outlines or SGLang) can implement the same rule; the property characterises grammar-state access, not a particular implementation. □

Remark (finite non-deterministic positions). When $1 < |R(\mathcal{G}_S, T, q_t)| < |V|$, a drafter with grammar-state access that proposes all of $R(\mathcal{G}_S, T, q_t)$ (if $|R(\mathcal{G}_S, T, q_t)| \leq K$) obtains $\alpha = \sum_{v \in R(\mathcal{G}_S, T, q_t)} p_M^{\text{constrained}}(v) = 1$ by Lemma 3.1, while any drafter whose support misses part of $R(\mathcal{G}_S, T, q_t)$ or spends budget outside it has $\alpha = \sum_{v \in \text{supp}(D) \cap R(\mathcal{G}_S, T, q_t)} \min\{p_D(v), p_M^{\text{constrained}}(v)\} \leq 1$. Either way one forward pass is needed (Proposition 5.2), so this affects acceptance statistics (Table 2) but not latency.

H.5 Proof of Theorem 6.2: Frequency Sampling Requires $\Omega(|L(\mathcal{G}_S)|)$ Samples

Let $f(t) = \Pr_{y \sim p_{\text{data}}}[t \in T(y)]$ be the marginal probability that token t appears in a training output. If p_{data} has full support on $L(\mathcal{G}_S)$, then $t \in R(\mathcal{G}_S, T)$ if and only if $f(t) > 0$, and determining $R(\mathcal{G}_S, T)$ from samples is the *support estimation* problem.

Standard coupon-collector arguments show that to observe a token with frequency $f(t) = \epsilon$ at least once with probability $1 - \delta$ requires $\Theta(\log(1/\delta)/\epsilon)$ samples. In the worst case, a valid token t appears in a single output string $w \in L(\mathcal{G}_S)$; under the uniform distribution over $L(\mathcal{G}_S)$ that string has probability $\epsilon = 1/|L(\mathcal{G}_S)|$, so observing t requires $\Omega(|L(\mathcal{G}_S)|)$ samples in expectation. Because a frequency-based method can only certify tokens it has observed, this is a lower bound on the number of samples any such method needs before it can certify all of $R(\mathcal{G}_S, T)$; if p_{data} does *not* have full support (as in our training corpus), no number of samples suffices. □

In contrast, grammar reachability computes $R(\mathcal{G}_S, T)$ with zero samples: by exhaustive enumeration of $L(\mathcal{G}_S)$ and tokenisation when $L(\mathcal{G}_S)$ is finite ($O(|L(\mathcal{G}_S)| \cdot L_{\max})$; $|L(\mathcal{G}_S)| = 15{,}703$ for the Thrift IDL), or by token-level DFA construction in general ($O(|Q| \cdot |V| \cdot \ell_{\max})$).

Empirical confirmation. In our training corpus (35,329 records), 774 of 2,972 reachable tokens (26.0%) have zero observed frequency—they correspond to API catalog entries never exercised in any training example. Since the valid output strings *containing* those tokens are absent from the corpus, no threshold, re-weighting, or resampling of the same corpus can recover them; only new data containing exactly those output strings would. Grammar reachability identifies all 2,972 tokens in 4.2 seconds with zero data.

H.6 Proof of Theorem 6.1: Zero Gradient Saliency for Unreachable LM-Head Rows

The diagonal of the empirical Fisher information (gradient saliency) of the task loss for output LM-head row i is:

$$H_{ii} = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\left\|\frac{\partial\mathcal{L}}{\partial W_i^{\text{out}}}\right\|^2\right]$$

where $\mathcal{L}$ is the language modelling loss and W_i^{out} is the i-th row of the output projection matrix (the LM head). This proof concerns W^{out} only; input embedding rows are outside the scope of Theorem 6.1. GPTQ's (Frantar et al., 2023) layer-wise reconstruction Hessian is the activation covariance $2XX^\top$ (over the hidden dimension d), which is shared by all rows and nonzero; it is *distinct* from the task-loss gradient saliency H_{ii} above, and the zero-saliency result applies only to the latter.

For a softmax output layer, $\partial\mathcal{L}/\partial W_i^{\text{out}} = (p_i - \mathbf{1}[y_t = i])\, h_t$ at each position t, where p_i is the model's (masked) probability of token i. Under constrained calibration, for any $i \notin R(\mathcal{G}_S, T)$ and any position of any calibration sample: $i \notin R(\mathcal{G}_S, T, q_t)$, so the mask gives $p_i = 0$, and $y_t \neq i$ because calibration targets are valid outputs. Hence the gradient of row i is identically zero and $H_{ii} = 0$ exactly, for every finite calibration set. We stress that this is agreement by construction: the mask forces $p_i = 0$, so the empirical Fisher can only *confirm* what the grammar oracle already states; it cannot discover it without the mask (see the note below), and it requires a calibration pass whereas the oracle requires none. □

Note on calibration protocol. This result holds specifically when calibration is performed *with* constrained decoding active. If calibration uses unconstrained forward passes, then $p_i > 0$ for unreachable rows and H_{ii} is nonzero (typically small), so gradient saliency alone would not identify $V \setminus R(\mathcal{G}_S, T)$ exactly. The constrained-calibration protocol is the natural choice when deploying GRVP, since deployment inference also uses constrained decoding.

I Grammar Oracle Computation

I.1 Algorithm for Regular Grammars (Thrift IDL)

For regular grammar schemas with finite $L(\mathcal{G}_S)$ (Thrift IDL, VehicleWorld), the reachable token set $R(\mathcal{G}_S, T)$ is computed by exhaustive enumeration and tokenisation:

1. **Parse the schema.** Extract all valid output strings from the Thrift IDL: enumerate all (service, method, enum-parameter-combination) tuples. For the automotive schema this yields 15,703 distinct strings; the enumeration is linear in its output, $O(|L(\mathcal{G}_S)|)$, but $|L(\mathcal{G}_S)|$ itself is $\sum_{\text{methods}} \prod_m |P_m|$ (product of enum cardinalities per method), which is why we restrict this variant to schemas whose language is small enough to enumerate. For larger languages the token-level DFA product construction ($O(|Q| \cdot |V| \cdot \ell_{\max})$) yields the same set without enumeration.

2. **Tokenise all strings.** Apply the tokeniser T to each of the 15,703 strings. Collect the union of all token IDs that appear in at least one tokenisation. This requires $O(|L(\mathcal{G}_S)| \times L_{\max})$ time where $L_{\max}$ is the maximum output string length in tokens.

3. **Add `EOS` and control tokens.** `EOS` is part of every $T(w)$ by the termination convention of Definition 2.2; BOS/PAD are retained for framework compatibility and are never selected at a generation step.

4. **Build the reachable index.** Store $R(\mathcal{G}_S, T)$ as a sorted integer array and a boolean mask over V. At inference time, the mask enables $O(1)$ lookup: given a token ID, is it reachable?

For the automotive schema with Qwen tokeniser: 15,703 tokeniser calls producing $\approx 390{,}000$ tokens in total ($L_{\max} \approx 25$). Table 16 gives a measured phase-by-phase breakdown.

Table 16: Grammar oracle build time — automotive Thrift IDL, 15,703 valid strings, Qwen tokeniser. Phases are sequential; the total depends on hardware and whether the schema is pre-cached. *Run once per (schema, tokeniser) pair; amortised over unlimited inference calls.*

Phase	M3 Pro (1 core)	A100 pod (1 core)	Notes
1. Parse IDL, enumerate $L(\mathcal{G}_S)$	1.9 s	12.4 s[†]	BFS over method/enum combos
2. Tokenise 15,703 strings	1.6 s	0.2 s[‡]	15,703 tokeniser calls (≈390k tokens)
3. Build compressed trie (DFA)	0.7 s	0.7 s	157,549 states
4. Build boolean reachability mask	<0.1 s	<0.1 s	Single pass over $\lvert V\rvert$
Total	**4.2 s**	**13.3 s**	

[†]The A100 pod reads the Thrift IDL directory from GCS-mounted storage; cold-path network I/O accounts for ≈10 s of Phase 1. With a locally cached schema, Phase 1 drops to ≈2 s on both platforms. [‡]Phase 2 timings reflect the tokeniser backend installed on each platform (batched Rust fast tokeniser on the pod vs. per-string calls on the laptop), not CPU speed. The 2.3 s figure in Section D is Phases 2–3 on the M3 Pro, consistent with these measurements.

I.2 Algorithm for Context-Free Grammars (JSON Schema)

For context-free JSON Schema, exhaustive enumeration of $L(\mathcal{G}_S)$ is infeasible (infinite strings due to open-vocabulary fields). The position-constrained reachable set $R(\mathcal{G}_S, T, q_t)$ is computed *per-step* from a pushdown automaton (PDA) configuration:

1. **Compile the schema.** Convert the JSON Schema to an EBNF grammar, then compile to a PDA. Libraries such as Outlines (Willard & Louf, 2023) or XGrammar (Dong et al., 2024) perform this step.
2. **At each decode step.** Given the current PDA configuration q_t, query which tokens have a transition from q_t. At typed configurations (function names, parameter names, enum values, structural syntax) $R(\mathcal{G}_S, T, q_t)$ is finite. At free-string configurations the automaton returns $R(\mathcal{G}_S, T, q_t) = V$ and GGSD falls back to unconstrained decoding.
3. **Complexity.** The per-step query is $O(|V|)$ worst-case (XGrammar's context-independent precomputation reduces amortised cost to near-$O(1)$ for common patterns).

For the xLAM benchmark, the per-record JSON Schema defines a small number of functions (typically 1–5), making schema compilation fast (<1 ms per record). The static $f_{\text{det}} = 0.572$ is computed analytically across all 5,998 test records; the generation-time $f_{\text{det}} = 0.844$ is measured on decoded trajectories (Section G).

I.3 Storage Format

The reachable set $R(\mathcal{G}_S, T)$ is stored as:

- A sorted `int32` tensor of shape $(|R(\mathcal{G}_S, T)|,)$ containing original vocabulary IDs.
- A boolean mask of shape $(|V|,)$ for $O(1)$ membership queries.
- A compact LM-head weight matrix $W_R(\mathcal{G}_S, T) \in \mathbb{R}^{|R(\mathcal{G}_S,T)| \times d}$ (the physical GRVP artefact).

For Qwen-7B with $|R(\mathcal{G}_S, T)| = 2{,}972$ and $d = 3{,}584$: storage cost is $2{,}972 \times 3{,}584 \times 2$ bytes (bfloat16) $= 21.3$ MB, versus $151{,}665 \times 3{,}584 \times 2 = 1{,}087$ MB for the full LM head (decimal MB), a saving of 1,066 MB. This 98.0% reduction is computed *once per (schema, tokeniser, model) triple* and stored as a static artefact, requiring no recomputation at deployment time.