LOGIT-PATH TRACING: EXACT GREEDY DECODING PARTITIONS OF LANGUAGE MODELS UNDER AFFINE LOGIT CONTROLS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many language-model decoding methods expose continuous parameters, such as guidance scales and objective weights. We study settings in which token logits are affine in these parameters and ask which greedy outputs occur anywhere in the control domain. Finite sampling can detect observed outputs but cannot certify the absence of unobserved output cells. Under greedy decoding, affine token logits form an upper envelope whose intersections partition the parameter domain into regions with a constant next-token prediction. We introduce *Logit-Path Tracing* (LPT), which recursively constructs these regions and verifies their winners against the full vocabulary. Because pairwise logit gaps are affine, vertex verification certifies each polyhedral cell, yielding the complete greedy decoding partition under exact arithmetic. For finite-precision logits, we extend LPT with *ε-robust certification*, using a pairwise-gap error tolerance to separate certified and unresolved regions. We evaluate three base models on 200 test prompts over one- and two-dimensional control domains, with the pairwise-gap tolerance calibrated on a separate set of 100 prompts. LPT constructs the complete greedy decoding partition, with a mean of 5.23–10.36 output cells per prompt in one dimension and 18.82–68.30 in two dimensions. Using these complete partitions as reference, finite sampling observes only 45.0–87.9% of the output cells under matched execution time and 33.9–63.1% under matched model-evaluation budgets.

1 INTRODUCTION

Continuous parameters are common in language-model decoding. Guidance scales, objective weights, and expert coefficients can continuously adjust model behavior at inference time (Shi et al., 2024; Dekoninck et al., 2024; Liu et al., 2021). Changing these parameters can alter both the next-token prediction and, through autoregressive generation, the complete output. We study methods in which each token logit is affine in the control parameters for every fixed prefix, including linear combinations of endpoint logits, objective-weighted predictions, and expert–anti-expert logit differences.

Evaluation over a continuous range of control values raises a basic question: does *any* value in the range produce one of the target greedy outputs? If we know exactly which output is produced in each part of the range, we can check any desired property of every possible greedy output once per output cell. Finite sampling, in contrast, only reveals the outputs produced at the sampled control values. It can detect a target output when one of those values produces it, but cannot rule out the possibility that the target appears between sampled values. Denser grids reduce these gaps but do not eliminate them.

Affine token logits make this question tractable under greedy decoding. For a fixed prefix and continuous control parameter vector, each token logit is affine over the parameter domain, and the greedy next token is determined by their upper envelope. The greedy winner can change only where two token logits become equal. Because the logit difference between any two tokens, which we call the *pairwise logit gap*, is affine, changes in the greedy winner occur on affine equality sets that induce a partition of the domain into polyhedral regions with a constant next-token prediction. These regions must then be propagated autoregressively: each winning token changes the prefix

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.

and therefore the affine logit functions at the next step. A small region missed at one step can thus remove an entire downstream output branch.

We introduce *Logit-Path Tracing* (LPT), which recursively constructs this partition and verifies every proposed next-token region against the full vocabulary. Because pairwise logit gaps remain affine, their extrema over a polyhedral cell occur at its vertices, so verifying the winner at the vertices certifies the entire cell. LPT propagates all verified winners until EOS or the generation horizon, producing the complete greedy decoding partition under exact arithmetic.

Exact arithmetic is unavailable in practical model inference. Small numerical errors are most consequential near region boundaries, where competing token logits are nearly equal. We therefore extend LPT with ε*-robust certification*: regions are certified only when the winning margin exceeds the prescribed pairwise-gap error bound, while uncertain control parameter values are reported as unresolved. This separates outputs that are stable under the assumed numerical error from those for which no reliable decision can be made.

We evaluate LPT on three base models and 200 test prompts over one- and two-dimensional control domains, using a separate set of 100 prompts to calibrate the numerical error tolerance. LPT recovers complete greedy decoding partitions with mean output-cell counts ranging from 5.23 to 68.30 across models and control dimensions, while dense-grid spot checks and candidate-set sensitivity tests confirm consistent sampled decisions and final partitions. We compare LPT with uniform sampling and adaptive subdivision under matched execution time and matched model-evaluation budgets, and further evaluate sensitivity to sampling budget, robust tolerance, batch size, and numerical precision.

Our contributions are as follows:

1. We formulate evaluation over continuous decoding parameters as an existence problem and show why finite parameter sampling cannot certify the absence of a target output.
2. We introduce LPT, which recursively constructs greedy decoding regions and reduces full-vocabulary certification to finite vertex checks. Under exact arithmetic, this yields the complete greedy decoding partition; under finite precision, it yields ε-robust certified and unresolved regions.
3. We validate LPT across three models and one- and two-dimensional control domains, evaluating partition recovery, numerical robustness, and finite parameter sampling under matched computational resources.

2 Related Work

Controllable and multi-objective decoding. Decoding-time control methods combine models, objectives, or auxiliary signals to vary generation behavior without retraining a single policy. Multi-objective decoding and language model arithmetic combine predictions using user-specified weights or algebraic operators (Shi et al., 2024; Dekoninck et al., 2024), while related methods steer generation with learned scorers, expert–anti-expert combinations, contrastive signals, layer-wise comparisons, or attribute-conditioned guidance (Mudgal et al., 2024; Liu et al., 2021; Li et al., 2023; Chuang et al., 2024; Krause et al., 2021). Model- and adapter-interpolation methods similarly expose continuous trade-offs between behaviors (Rame et al., 2023; Kangaslahti & Alvarez-Melis, 2025). LPT focuses on the resulting decoding problem when token logits are affine in such continuous controls.

Parametric analysis and robust verification. Parametric optimization studies how discrete solutions change as continuous parameters vary. For affine scores, upper envelopes induce piecewise-constant optimal regions, an idea used in minimum-error-rate tuning and related optimization problems (Macherey et al., 2008; Kumar et al., 2009; Galley & Quirk, 2011; Hopkins & May, 2011; Gajjar et al., 2021; Gusfield et al., 1994; Edelsbrunner, 1987). Lattice-based MERT also uses piecewise-linear score envelopes over candidate translations, but those candidates are fixed by the lattice or hypergraph. LPT applies the same geometric principle to autoregressive greedy decoding: each recovered region induces a new prefix and hence a new full-vocabulary set of affine token logits, so the candidate set must be verified and expanded recursively. Continuous language-model interpolation studies how outputs vary along such controls (Kangaslahti & Alvarez-Melis, 2025); LPT focuses on the certification problem for the induced greedy partitions. In practice, finite-precision inference

can vary across numerical formats, batch sizes, and execution conditions (Yuan et al., 2025; He & Thinking Machines Lab, 2025). Related robust parameter analyses distinguish resolved from unresolved regions under uncertainty (Jansen et al., 2022); LPT similarly provides ε-robust certificates for finite-precision greedy decoding partitions.

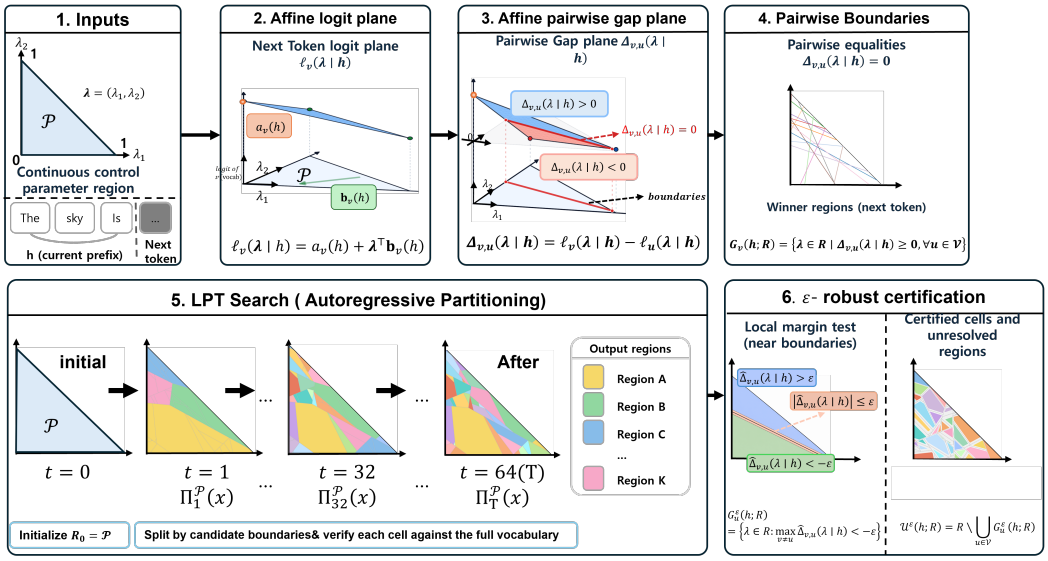

Figure 1: Overview of Logit-Path Tracing (LPT). For a fixed prefix, token logits are affine functions of the control vector $\boldsymbol{\lambda} \in \mathcal{P} \subset \mathbb{R}^d$. Pairwise logit equalities induce a partition of the control domain into candidate polyhedral cells. LPT verifies each candidate cell against the full vocabulary, refines any missed winner, and recursively propagates the verified tokens to new prefixes. The terminal cells form the complete greedy decoding partition, while ε-robust certification separates certified and unresolved regions under finite-precision pairwise-gap error.

3 Problem Formulation

3.1 Affine token logits

Figure 1 provides an overview of the LPT framework. We begin by formalizing the affine-control decoding setting. Let x be a prompt, $\mathcal{V}$ a finite vocabulary, T a generation horizon, and $\mathcal{P} \subset \mathbb{R}^d$ a bounded polyhedral control parameter domain (e.g., guidance scales or objective weights). We represent an autoregressive branch by a *prefix–region pair* (h, R), where h is a generated prefix and $R \subseteq \mathcal{P}$ contains the control parameter values for which h is a valid greedy prefix. Decoding starts from $(x, \mathcal{P})$.

For every fixed prefix h, let $\ell_v(\boldsymbol{\lambda} \mid h)$ denote the next-token logit assigned to token v under control vector $\boldsymbol{\lambda} \in \mathcal{P}$. We assume that this logit is affine in $\boldsymbol{\lambda}$:

$$\ell_v(\boldsymbol{\lambda} \mid h) = a_v(h) + \boldsymbol{\lambda}^\top \mathbf{b}_v(h), \qquad v \in \mathcal{V}. \tag{1}$$

Here, $a_v(h) \in \mathbb{R}$ is the component of the token logit that does not depend on the control value, while $\mathbf{b}_v(h) \in \mathbb{R}^d$ specifies how the logit changes along each of the d control dimensions. Both may depend arbitrarily on the prefix h; affinity is required only with respect to $\boldsymbol{\lambda}$ after the prefix is fixed.

For example, when $d = 1$, let the scalar control parameter be $s \in [0, 1]$. Equation 1 reduces to

$$\ell_v(s \mid h) = a_v(h) + s\, b_v(h). \tag{2}$$

For interpolation between endpoint logits $z_v^A(h)$ and $z_v^B(h)$, setting $a_v(h) = z_v^A(h)$ and $b_v(h) = z_v^B(h) - z_v^A(h)$ recovers

$$\ell_v(s \mid h) = (1 - s) z_v^A(h) + s z_v^B(h).$$

Thus, endpoint interpolation is the one-dimensional special case of the general affine control model.

3.2 GREEDY REGIONS AND COMPLETE DECODING PARTITION

For a fixed prefix h, let $W(\boldsymbol{\lambda} \mid h)$ denote the set of greedy next-token winners under control vector $\boldsymbol{\lambda}$. For token v, its softmax probability is

$$p_v(\boldsymbol{\lambda} \mid h) = \frac{\exp \ell_v(\boldsymbol{\lambda} \mid h)}{\sum_{u \in \mathcal{V}} \exp \ell_u(\boldsymbol{\lambda} \mid h)}, \qquad \log p_v(\boldsymbol{\lambda} \mid h) = \ell_v(\boldsymbol{\lambda} \mid h) - \log \sum_{u \in \mathcal{V}} \exp \ell_u(\boldsymbol{\lambda} \mid h). \quad (3)$$

Because $\log(\cdot)$ is monotonically increasing, taking the logarithm does not change the maximizing token. Moreover, the second term above is shared by all tokens and therefore does not affect their ordering. Hence,

$$W(\boldsymbol{\lambda} \mid h) = \arg\max_{v \in \mathcal{V}} p_v(\boldsymbol{\lambda} \mid h) = \arg\max_{v \in \mathcal{V}} \log p_v(\boldsymbol{\lambda} \mid h) = \arg\max_{v \in \mathcal{V}} \ell_v(\boldsymbol{\lambda} \mid h). \quad (4)$$

When the maximum is unique, $W(\boldsymbol{\lambda} \mid h)$ contains a single token; at an exact tie, it contains each tied token so that all valid greedy branches are retained.

For tokens $v, u \in \mathcal{V}$, define the pairwise logit gap

$$\Delta_{v,u}(\boldsymbol{\lambda} \mid h) = \ell_v(\boldsymbol{\lambda} \mid h) - \ell_u(\boldsymbol{\lambda} \mid h). \quad (5)$$

By Equation 1, every pairwise gap is affine in $\boldsymbol{\lambda}$. Therefore, unless the two affine logits coincide identically, the control values satisfying $\Delta_{v,u}(\boldsymbol{\lambda} \mid h) = 0$ form a hyperplane when nonempty: a point for $d = 1$, a line for $d = 2$, and a $(d-1)$-dimensional hyperplane in general.

Within a prefix–region pair (h, R), the greedy next-token region of token v is

$$G_v(h; R) = \{\boldsymbol{\lambda} \in R : \Delta_{v,u}(\boldsymbol{\lambda} \mid h) \geq 0 \text{ for all } u \in \mathcal{V}\}. \quad (6)$$

Each condition is an affine halfspace constraint, so $G_v(h; R)$ is a closed polyhedral region. These regions may overlap on tie sets, because multiple tokens can simultaneously maximize the logit at the same control value. Their strict-winner interiors are disjoint, while equality sets are retained as tie cells. In the generic nondegenerate case, these tie sets form lower-dimensional boundaries. Accordingly, greedy next-token regions are intervals for $d = 1$ and polygons for $d = 2$, with shared boundaries whenever ties occur.

Autoregressive decoding recursively propagates these cells: each winning token changes the prefix and therefore the affine logits at the next step. Let $D_T(\boldsymbol{\lambda}; x)$ denote the set of complete greedy outputs obtained at control value $\boldsymbol{\lambda}$ until EOS or horizon T. Because both $\mathcal{V}$ and T are finite, only finitely many output cells can be reached.

Let K denote the number of resulting output cells. For each $k \in \{1, \ldots, K\}$, let $\Omega_k \subseteq \mathcal{P}$ denote a control region on which the complete greedy output set is constant, and let $\mathcal{Y}_k$ denote that output set. The regions Ω_k are disjoint cells obtained by recursively refining strict-winner regions and tie sets. We define the *complete greedy decoding partition* as

$$\Pi_T^{\mathcal{P}}(x) = \{(\Omega_k, \mathcal{Y}_k)\}_{k=1}^{K}, \qquad \mathcal{P} = \bigcup_{k=1}^{K} \Omega_k, \qquad D_T(\boldsymbol{\lambda}; x) = \mathcal{Y}_k \text{ for } \boldsymbol{\lambda} \in \Omega_k, \quad (7)$$

where the cells Ω_k are disjoint by construction. When multiple tokens tie for the maximum logit, the corresponding parameter values belong to a tie cell and all tied greedy branches are preserved in $\mathcal{Y}_k$.

Thus, a greedy next-token region $G_v(h; R)$ describes one decoding step, whereas an output cell Ω_k collects the control values that follow the same complete greedy branch. Recovering the partition reduces evaluation over the continuous domain $\mathcal{P}$ to a finite set of outputs $\mathcal{Y}_1, \ldots, \mathcal{Y}_K$.

4 LOGIT-PATH TRACING

Logit-Path Tracing (LPT) constructs the complete greedy decoding partition $\Pi_T^{\mathcal{P}}(x)$ by recursively processing prefix–region pairs (h, R). At each pair, it constructs a candidate polyhedral partition from a small token set, verifies every candidate cell against the full vocabulary, and propagates each verified winner to the resulting prefix.

4.1 CANDIDATE CONSTRUCTION AND VERIFICATION

For a prefix–region pair (h, R), LPT selects an initial candidate set $S \subseteq \mathcal{V}$ from tokens in $W(\boldsymbol{\lambda} \mid h)$ observed at a small number of control values. The affine token logits in S induce a candidate upper-envelope partition of R: pairwise equality sets induce a decomposition into polyhedral cells on which one candidate token is maximal.

Because S may omit a token that wins inside a proposed cell, every candidate cell is verified against the full vocabulary. Let $Q \subseteq R$ have declared winner u. LPT checks

$$\max_{\boldsymbol{\lambda} \in Q} \Delta_{v,u}(\boldsymbol{\lambda} \mid h) \leq 0 \qquad \text{for every } v \in \mathcal{V}. \tag{8}$$

Since each logit gap is affine and Q is polyhedral, the maximum occurs at a vertex of Q. Full-cell verification therefore reduces to evaluating the gap at finitely many vertices.

If full-vocabulary verification finds an omitted token $v \notin S$ such that

$$\max_{\boldsymbol{\lambda} \in Q} \Delta_{v,u}(\boldsymbol{\lambda} \mid h) > 0, \tag{9}$$

then v strictly exceeds the declared winner u somewhere in the candidate cell Q. We add v to S and rebuild the partition. If the maximum is exactly zero, the omitted token ties the declared winner on an equality set, which is retained as a tie cell. Each failed verification therefore enlarges S, and because the vocabulary is finite, this refinement must terminate. Once no omitted competitor strictly exceeds the declared winner, the resulting decomposition recovers the full-vocabulary greedy next-token regions $G_v(h; R)$ defined in Equation 6, including their tie sets.

4.2 RECURSIVE PROPAGATION AND COMPLETENESS

LPT appends each verified maximizing token v to the current prefix and continues recursively on its corresponding greedy next-token region $G_v(h; R)$, retaining all maximizing branches at ties. The new prefix induces a new set of affine token logits, so candidate construction and verification are repeated recursively. A branch becomes terminal when it reaches EOS or horizon T, and the resulting terminal cells form $\Pi_T^{\mathcal{P}}(x)$.

Theorem 1 (Completeness of LPT)**.** *Assume that $\mathcal{V}$ and T are finite, $\mathcal{P} \subset \mathbb{R}^d$ is compact and polyhedral, token logits satisfy Equation 1, and all region operations use exact arithmetic. If LPT verifies every reachable prefix–region pair against the full vocabulary and retains all maximizing branches at ties, then it terminates and returns the complete greedy decoding partition $\Pi_T^{\mathcal{P}}(x)$.*

Proof sketch. Equation 8 ensures that the returned greedy next-token regions $G_v(h; R)$ contain all full-vocabulary maximizers over their respective cells. Recursively propagating these maximizers therefore preserves both coverage of $\mathcal{P}$ and correctness of every reachable greedy branch. Since $\mathcal{V}$ is finite and the recursion depth is bounded by T, all branches terminate and their terminal cells form $\Pi_T^{\mathcal{P}}(x)$. The full proof is given in Appendix B. □

LPT is output-sensitive. If the affine logits are assembled from M source logit evaluations and N distinct prefixes are reached, prefix caching requires MN full-vocabulary source evaluations. The number and geometric complexity of reachable cells can grow rapidly with both sequence length and control dimension.

5 ε-ROBUST CERTIFICATION

The completeness guarantee in Theorem 1 assumes exact logits, whereas practical model evaluations use finite precision. Let $\widehat{\ell}_v(\boldsymbol{\lambda} \mid h)$ denote the computed logit and define the corresponding computed pairwise gap as

$$\widehat{\Delta}_{v,u}(\boldsymbol{\lambda} \mid h) = \widehat{\ell}_v(\boldsymbol{\lambda} \mid h) - \widehat{\ell}_u(\boldsymbol{\lambda} \mid h).$$

We assume that $\varepsilon \geq 0$ bounds the error in every pairwise gap:

$$\left| \widehat{\Delta}_{v,u}(\boldsymbol{\lambda} \mid h) - \Delta_{v,u}(\boldsymbol{\lambda} \mid h) \right| \leq \varepsilon. \tag{10}$$

For a prefix–region pair (h, R), we define the ε-robust greedy next-token region of token u as

$$G_u^{\varepsilon}(h; R) = \left\{ \boldsymbol{\lambda} \in R : \max_{v \in \mathcal{V} \setminus \{u\}} \widehat{\Delta}_{v,u}(\boldsymbol{\lambda} \mid h) < -\varepsilon \right\}. \tag{11}$$

Thus, $G_u^{\varepsilon}(h; R)$ contains the control vectors for which the computed margin of token u exceeds the pairwise-gap error bound. Because the computed gaps are affine, certification over a polyhedral cell again reduces to checking its vertices. Thus the same verification principle applies in any control dimension.

Unlike the greedy regions $G_u(h; R)$, the ε-robust regions $G_u^{\varepsilon}(h; R)$ need not cover all of R. Control vectors that belong to no robust greedy region form the unresolved region

$$\mathcal{U}^{\varepsilon}(h; R) = R \setminus \bigcup_{u \in \mathcal{V}} G_u^{\varepsilon}(h; R). \tag{12}$$

LPT recursively propagates the robust greedy regions $G_u^{\varepsilon}(h; R)$ as before, while any control vector entering $\mathcal{U}^{\varepsilon}(h; R)$ is reported as unresolved for the complete output. In the generic nondegenerate case, the error bound expands each pairwise tie boundary into a surrounding uncertainty region. This becomes an interval around a breakpoint in one dimension and a band around a boundary line in two dimensions, with the same construction extending to higher-dimensional hyperplanes.

6 Experiments

We evaluate LPT in both one- and two-dimensional control spaces, asking whether it recovers the predicted greedy decoding partition, remains reliable under finite-precision inference, and reveals output structure that finite parameter sampling can miss.

6.1 Experimental setup

We evaluate Qwen2.5-3B-Instruct, Qwen2.5-1.5B-Instruct, and SmolLM2-1.7B-Instruct under control dimensions $d \in \{1, 2\}$. For $d = 1$, we interpolate between better- and safer-oriented LoRA adapters trained with DPO on PKU-SafeRLHF-10K using the corresponding response preferences. For $d = 2$, we interpolate logits from three system-prompt-conditioned sources of the same base model—factual, helpful, and safe—over a two-dimensional simplex. Experiments use 200 test prompts with greedy decoding, EOS stopping, horizon $T = 64$, and LPT candidate size $k = 64$. A set of 100 calibration prompts is used to estimate the pairwise-gap tolerance $\hat{\varepsilon}$ for robust certification. Safety prompts are drawn from a held-out PKU-SafeRLHF prompt pool disjoint from the DPO adapter-training prompts. Experiments were conducted on four NVIDIA GeForce RTX 4090 GPUs. Full training, control-construction, and implementation details are provided in Appendix D.

6.2 Main comparison

We compare LPT with two finite-sampling baselines in Table 1. A *uniform sampling* baseline evaluates control parameter values using a deterministic space-filling order, while *adaptive subdivision* refines intervals or simplex cells using a breadth-first largest-cell subdivision order. For the resource-matched comparisons, each baseline evaluates parameter values in a fixed order and stops when the available time or model-evaluation budget is exhausted. The evaluation order is chosen so that even an early stop still covers the domain broadly: the uniform baseline uses a low-discrepancy ordering, while adaptive subdivision proceeds breadth-first. The reported finite-sampling implementations decode sampled parameter values as point queries and do not share source-logit evaluations across different sample points through a prefix trie. Consequently, Panel B of Table 1 should be read as a matched model-evaluation budget for this standard point-sampling implementation rather than as a lower bound against all possible prefix-cached samplers.

We evaluate the methods under two resource controls. **Panel A** uses a fixed time budget: for each model and control dimension, Uniform sampling and Adaptive subdivision run for approximately the same per-prompt time as LPT. **Panel B** uses a fixed model-evaluation budget: the two finite-sampling methods are allowed up to the same number of model evaluations used by LPT for the corresponding model and control dimension. These two comparisons separate the effect of execution time from the effect of the number of model evaluations.

Table 1: Main comparison over one- and two-dimensional parameter domains. LPT denotes the construction with $\varepsilon = 0$ and defines the reference partition for output-cell recall and domain coverage. **Panel A** compares methods under a fixed per-prompt time budget. **Panel B** compares methods under a fixed model-evaluation budget.

			Panel A: Matched time					**Panel B: Matched model-evaluation budget**				
d	Model	Method	Output cells	Model evals. ↓	Recall (%) ↑	Coverage (%) ↑	Time (s) ↓	Method	Output cells	Model evals. ↓	Recall (%) ↑	Coverage (%) ↑	Time (s) ↓
1	Qwen2.5-3B	**LPT**	5.2	388.8	*100.0*	100.0	71.2	**LPT**	5.2	388.8	*100.0*	100.0	71.2
		Uniform sampling	4.2	1515.7	87.3	96.7	75.8	Uniform sampling	2.5	367.5	63.1	73.6	18.3
		Adaptive subdivision	4.3	1518.3	87.9	97.1	75.7	Adaptive subdivision	2.5	367.5	63.1	73.6	18.3
	Qwen2.5-1.5B	**LPT**	9.0	648.5	*100.0*	100.0	25.7	**LPT**	9.0	648.5	*100.0*	100.0	25.7
		Uniform sampling	4.8	750.3	63.8	79.6	29.0	Uniform sampling	4.4	612.8	59.7	75.4	24.1
		Adaptive subdivision	4.8	749.9	63.7	79.5	28.9	Adaptive subdivision	4.4	613.4	59.7	75.6	24.0
	SmolLM2-1.7B	**LPT**	10.4	772.4	*100.0*	100.0	38.9	**LPT**	10.4	772.4	*100.0*	100.0	38.9
		Uniform sampling	6.3	1296.3	69.4	87.7	39.9	Uniform sampling	4.8	754.5	56.8	74.2	23.1
		Adaptive subdivision	6.3	1296.1	69.7	87.8	39.7	Adaptive subdivision	4.8	754.3	56.7	74.1	22.9
2	Qwen2.5-3B	**LPT**	68.3	4920.3	*100.0*	100.0	171.2	**LPT**	68.3	4920.3	*100.0*	100.0	171.2
		Uniform sampling	23.9	10104.9	45.0	86.1	172.7	Uniform sampling	16.1	4808.3	33.9	73.4	83.1
		Adaptive subdivision	26.3	10184.5	49.1	85.8	172.9	Adaptive subdivision	17.4	4812.4	36.6	70.2	82.8
	Qwen2.5-1.5B	**LPT**	18.8	1580.5	*100.0*	100.0	70.8	**LPT**	18.8	1580.5	*100.0*	100.0	70.8
		Uniform sampling	10.2	8076.9	71.0	95.4	71.5	Uniform sampling	5.1	1488.5	50.2	75.2	14.6
		Adaptive subdivision	10.7	8119.4	73.9	94.5	71.4	Adaptive subdivision	5.5	1486.5	52.6	74.4	14.7
	SmolLM2-1.7B	**LPT**	26.7	2402.5	*100.0*	100.0	44.4	**LPT**	26.7	2402.5	*100.0*	100.0	44.4
		Uniform sampling	10.6	4631.8	54.0	87.3	45.3	Uniform sampling	7.7	2294.5	43.9	76.7	22.8
		Adaptive subdivision	11.3	4666.1	56.9	86.3	45.3	Adaptive subdivision	8.1	2294.4	45.8	72.8	22.6

We report the mean number of recovered output cells, model evaluations per prompt, output-cell recall, domain coverage, and time per prompt. LPT defines the reference partition. Let $\mathcal{S}$ denote the set of control parameter values actually evaluated by a sampling method. Let $\Omega_1, \ldots, \Omega_{K_+}$ denote the positive-measure output cells in the LPT reference partition. Lower-dimensional tie cells are retained by LPT but not counted in K_+, since they occupy no length or area. For each prompt, we compute

$$N_{\text{pts}} = |\mathcal{S}|, \qquad N_{\text{reg}} = \sum_{k=1}^{K_+} \mathbf{1}[\mathcal{S} \cap \Omega_k \neq \emptyset],$$
$$\text{Recall} = \frac{N_{\text{reg}}}{K_+}, \qquad \text{Coverage} = \frac{\sum_{k:\, \mathcal{S}\cap\Omega_k \neq \emptyset} \mu(\Omega_k)}{\mu(\mathcal{P})}. \tag{13}$$

where μ denotes interval length for $d = 1$ and area for $d = 2$. Model evaluations count source-logit forward calls, and are reported separately from the number of sampled control parameter values. All reported quantities are averaged over prompts. Table 1 reports the comparison under both resource controls across all models and control dimensions; Appendix Table 5 reports the corresponding reference cell counts and sampled parameter values.

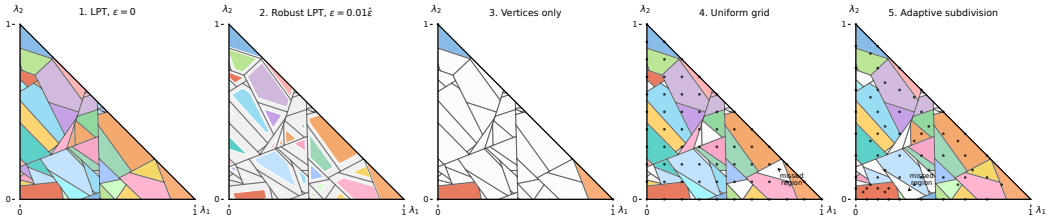

Figure 2: Representative $d = 2$ decoding partition. LPT recovers the complete partition, while robust LPT marks unresolved subregions. Vertices only evaluates the three simplex vertices; the Uniform-grid panel uses a completed grid, while Adaptive subdivision can miss small interior output cells. Colors denote distinct outputs; black points denote sampled control parameter values.

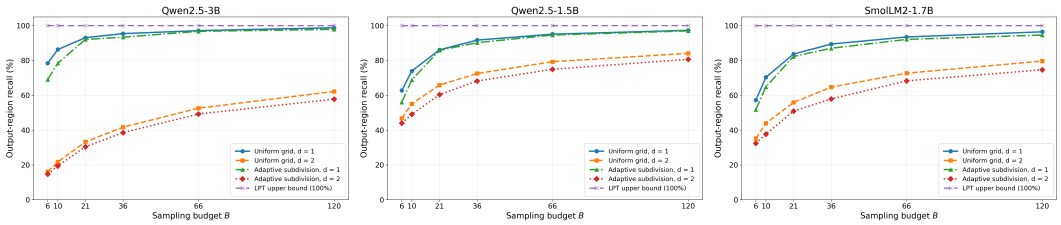

Figure 3: Sensitivity of output-cell recall to sampling budget. Uniform grid and Adaptive subdivision are evaluated over increasing budgets for $d \in \{1, 2\}$, with LPT shown as the 100% reference.

Finite sampling observes only finitely many control parameter values and can therefore miss output cells that contain no evaluated point. As shown in Table 1, the two resource-matched comparisons measure this gap under either comparable execution time or comparable model evaluations. The fixed-time comparison asks how much of the reference partition can be recovered within the execution time required by LPT, while the fixed model-evaluation comparison asks how much can be recovered from the same number of model evaluations. Because LPT alternates polyhedral refinement with sequential full-vocabulary verification, while finite samplers decode predeclared parameter values, the end-to-end time per model evaluation can differ across methods and models. We therefore report both matched-time and matched-evaluation views and treat the latter as a budget comparison. Figure 2 illustrates the underlying failure mode in $d = 2$: small interior output cells can remain unobserved even when the surrounding sampled control parameter values produce the same output. Appendix E.7 further evaluates the corresponding existence question for a fixed deterministic output property.

6.3 Sensitivity to sampling budget

In Figure 3, we examine how finite-sampling methods improve as the number of evaluated control parameter values increases. We vary the sampling budget over $B \in \{6, 10, 21, 36, 66, 120\}$ for Uniform grid and Adaptive subdivision and measure output-cell recall relative to the LPT reference partition. Results are reported separately for $d \in \{1, 2\}$ across all three models. The 100% reference line corresponds to the complete partition recovered by LPT. Output-cell recall increases monotonically with the sampling budget across the measured settings. In one dimension, finite sampling approaches the LPT reference rapidly. At $B = 120$, recall ranges from 94.6% to 98.8% across models and methods. The gap remains substantially larger in two dimensions, where recall at the same budget ranges from 57.8% to 84.1%. Uniform grid is consistently slightly stronger than Adaptive subdivision throughout the sweep, but neither finite-sampling strategy reaches the LPT reference in the measured $d = 2$ settings. The dense-grid spot checks in Appendix E.4 extend this same trend to a much larger 1326-point simplex grid: recall rises to 89.3–95.3%, but still remains below the LPT reference despite requiring tens to hundreds of times more model evaluations on the checked prompts. These results show that increasing the number of sampled control parameter values improves empirical partition recovery, while small output cells can remain unobserved under finite sampling, particularly as the control dimension increases.

6.4 ε-Robust LPT

We examine how the robust tolerance ε affects the partition returned by LPT. At $\varepsilon = 0$, LPT returns the greedy decoding partition, whereas larger values require a larger winner margin before a control subregion can be certified. As ε increases, certified output subregions therefore shrink and control values near decision boundaries become unresolved. We evaluate 23 tolerance levels of $\varepsilon/\hat{\varepsilon}$, with denser sampling near zero where the transition is most pronounced. Appendix Table 6 reports selected operating points; Figure 4 shows the full tolerance sweep.

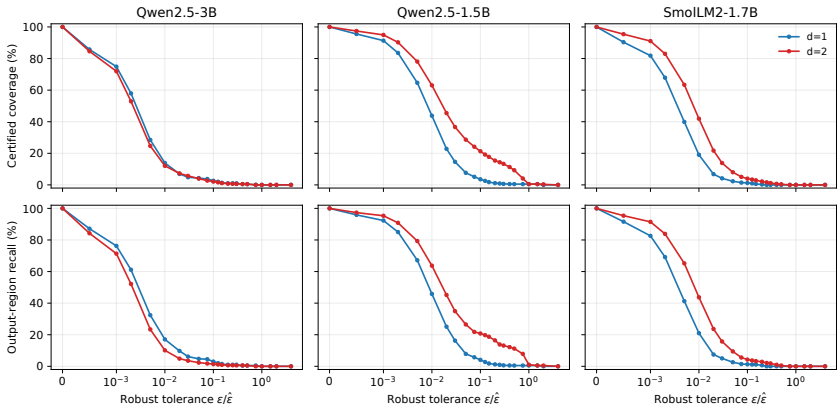

Figure 4: Dense robust-tolerance sweep. The upper row reports certified domain coverage, and the lower row reports output-cell recall. Both are recomputed by intersecting each recovered LPT cell with prefix-wise robust halfspaces requiring the generated token margin to exceed $\varepsilon = \alpha\hat{\varepsilon}$. The transition is concentrated near $\alpha = 0$.

Certification decreases sharply even for tolerances that are small fractions of the calibrated bound. At $0.01\hat{\varepsilon}$, certified coverage ranges from 13.9–43.8% in one dimension and 12.1–63.0% in two dimensions, while at $\varepsilon = \hat{\varepsilon}$ it falls to 0–0.5% across all settings. We therefore interpret the bf16–fp32 calibration as a conservative numerical-stability result: LPT at $\varepsilon = 0$ recovers the exact computed partition, but few cells have margins large enough to certify under this broad empirical pairwise-gap bound. Figure 5 shows the same effect geometrically, with unresolved regions expanding around low-margin decision boundaries.

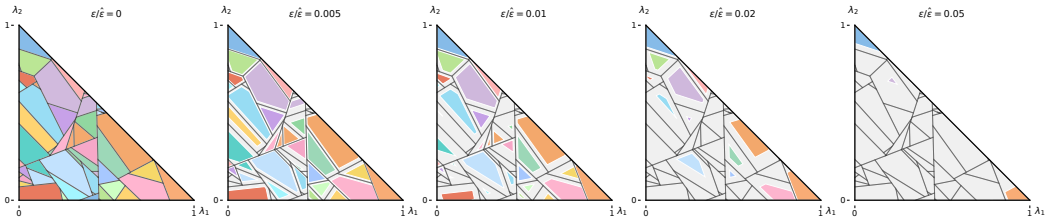

Figure 5: Representative $d = 2$ robust LPT partitions as ε increases on the same prompt as Figure 2. From left to right, $\varepsilon/\hat{\varepsilon} \in \{0, 0.005, 0.01, 0.02, 0.05\}$. Colored portions are certified output subregions, while pale portions are unresolved after clipping by the robust halfspaces.

7 Limitations

LPT is limited to affine logit controls, greedy decoding, and a finite horizon, and its cost can grow with reachable prefix–cell states and control dimension. Our empirical $\hat{\varepsilon}$ is calibrated to specific prompts and execution conditions rather than providing a hardware-independent guarantee, and the conservative bf16–fp32 bound leaves most cells unresolved. We also evaluate models up to 3B with $T = 64$. Finally, semantic safety or usefulness certification would require a task-specific deterministic evaluator over all recovered outputs.

8 Conclusion

We formulated affine-control greedy decoding as a partition-recovery problem over a continuous parameter domain. Logit-Path Tracing recovers the induced partition with full-vocabulary vertex checks and propagates verified cells through autoregressive decoding. The ε-robust extension explicitly reports unresolved regions, providing certificates beyond finite parameter sampling under finite-precision inference.

Ethics Statement

This work develops an exact-output partitioning method for logit interpolation settings that can support downstream audits of harmful responses when paired with a deterministic property evaluator. Experiments minimize reproduction of harmful text and report aggregate labels or redacted summaries. The method could also identify narrow or small control regions associated with harmful outputs; release decisions should weigh reproducibility against misuse risk.

Reproducibility Statement

Assumptions, partition semantics, the verification condition, and the completeness proof are in §3–§4 and Appendix B. The ε-robust certificate and its numerical calibration are described in §5 and Appendix C; §E.5 reports sensitivity to batch size and numerical precision. Appendix D gives the model, prompt, control-construction, hardware, and software configuration.

AI Use Statement

Generative AI tools were used only for writing assistance and code debugging, with all scientific content, implementations, experiments, and reported results verified by the authors.

References

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. DoLa: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Jasper Dekoninck, Marc Fischer, Luca Beurer-Kellner, and Martin Vechev. Controlled text generation via language model arithmetic. In *The Twelfth International Conference on Learning Representations*, 2024.

Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer, 1987.

Kshitij Gajjar, Girish Varma, Prerona Chatterjee, and Jaikumar Radhakrishnan. Generalized parametric path problems. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 536–546. PMLR, 2021.

Michel Galley and Chris Quirk. Optimal search for minimum error rate training. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 38–49. Association for Computational Linguistics, 2011.

Dan Gusfield, K. Balasubramanian, and Dalit Naor. Parametric optimization of sequence alignment. *Algorithmica*, 12(4–5):312–326, 1994.

Horace He and Thinking Machines Lab. Defeating nondeterminism in LLM inference. `https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/`, 2025. Thinking Machines Lab blog post.

Mark Hopkins and Jonathan May. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 1352–1362. Association for Computational Linguistics, 2011.

Nils Jansen, Sebastian Junges, and Joost-Pieter Katoen. Parameter synthesis in Markov models: A gentle survey. In *Principles of Systems Design: Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, volume 13660 of *Lecture Notes in Computer Science*, pp. 407–437. Springer, 2022.

Sara Kangaslahti and David Alvarez-Melis. Continuous language model interpolation yields dynamic and controllable text generation. *Transactions on Machine Learning Research*, 2025.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. GeDi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. Efficient minimum error rate training and minimum Bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 163–171. Association for Computational Linguistics, 2009.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori B. Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021.

Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pp. 725–734. Association for Computational Linguistics, 2008.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, Jilin Chen, Alex Beutel, and Ahmad Beirami. Controlled decoding from language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Alexandre Rame, Guillaume Couairon, Mustafa Shukor, Corentin Dancette, Jean-Baptiste Gaya, Laure Soulier, and Matthieu Cord. Rewarded soups: Towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. *Advances in Neural Information Processing Systems*, 2023.

Ruizhe Shi, Yifang Chen, Yushi Hu, Alisa Liu, Hannaneh Hajishirzi, Noah A. Smith, and Simon S. Du. Decoding-time language model alignment with multiple objectives. *Advances in Neural Information Processing Systems*, 2024. arXiv:2406.18853.

Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Understanding and mitigating numerical sources of nondeterminism in LLM inference. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2506.09501.

A Algorithm Details

Algorithm 1 summarizes the LPT procedure.

Algorithm 1 Logit-Path Tracing with ε-Robust Certification

Require: prompt x; domain $\mathcal{P}$; vocabulary $\mathcal{V}$; horizon T; candidate size k; tolerance $\varepsilon \geq 0$
Queue $\leftarrow \{(x, \mathcal{P})\}$; Terminal $\leftarrow \emptyset$; $\mathcal{U} \leftarrow \emptyset$
while Queue $\neq \emptyset$ **do**
  remove (h, R) from Queue
  if h ends with EOS or reaches T tokens **then**
    add (h, R) to Terminal; **continue**
  end if
  compute affine logits for h and initialize S with top-k candidates
  repeat
    partition R by the upper envelope of S
    verify all candidate cells against $\mathcal{V}$ at their vertices
    add any omitted strict winners to S and retain tied maximizers
  until no token is added
  let $\{(Q_j, W_j)\}_j$ be the verified greedy next-token cells
  if $\varepsilon = 0$ **then**
    for each (Q_j, W_j) and $u \in W_j$ **do**
      insert $(h \circ u, Q_j)$ into Queue
    end for
  else
    for each u with nonempty $G^{\varepsilon}_u(h; R)$ **do**
      insert $(h \circ u, G^{\varepsilon}_u(h; R))$ into Queue
    end for
    add $R \setminus \bigcup_u G^{\varepsilon}_u(h; R)$ to $\mathcal{U}$
  end if
end while
form terminal output cells from Terminal
if $\varepsilon = 0$ **then**
  return $\Pi^{\mathcal{P}}_T(x)$
else
  return certified partition and $\mathcal{U}$
end if

B Proofs

Proof of Theorem 1. Consider any reachable prefix–region pair (h, R) and a candidate cell $Q \subseteq R$ with declared winner u. Verification requires

$$\max_{\boldsymbol{\lambda} \in Q} \Delta_{v,u}(\boldsymbol{\lambda} \mid h) \leq 0 \qquad \forall v \in \mathcal{V}. \tag{14}$$

Since each $\Delta_{v,u}$ is affine and Q is polyhedral, let $\mathbf{q}_1, \ldots, \mathbf{q}_m$ denote the vertices of Q. Then

$$\max_{\boldsymbol{\lambda} \in Q} \Delta_{v,u}(\boldsymbol{\lambda} \mid h) = \max_{i \in \{1,\ldots,m\}} \Delta_{v,u}(\mathbf{q}_i \mid h). \tag{15}$$

Thus, vertex verification is sufficient to establish that u is a full-vocabulary maximizer throughout Q.

If verification fails, at least one previously omitted token is added to the candidate set. Hence the candidate sets form a strictly increasing sequence

$$S_0 \subsetneq S_1 \subsetneq \cdots \subseteq \mathcal{V},$$

which must terminate because $\mathcal{V}$ is finite. At termination, no omitted token can strictly exceed the declared winner. Equality-only cases are assigned to tie cells, so the resulting decomposition exactly recovers the full-vocabulary greedy regions $G_v(h; R)$, including their tie sets. Concretely, for any equality-only competitor pair (u, v) on a verified cell Q, LPT retains the equality set

$$F_{u,v}(Q) = Q \cap \{\boldsymbol{\lambda} : \Delta_{v,u}(\boldsymbol{\lambda} \mid h) = 0\}.$$

When multiple tokens tie simultaneously, the induced equality-set arrangement is refined by the corresponding equalities, and every tied maximizing branch is propagated from the resulting tie cell.

LPT then propagates every maximizing token on its corresponding region, including all branches at exact ties. Each recursion therefore preserves both coverage of the current region and all valid greedy branches. Since the recursion depth is at most T and each step contains finitely many cells and tokens, all branches terminate after finitely many expansions. Their terminal cells cover $\mathcal{P}$ and form the complete greedy decoding partition $\Pi_T^{\mathcal{P}}(x)$.

Robust certification. Suppose Equation 10 holds and $\boldsymbol{\lambda} \in G_u^{\varepsilon}(h; R)$. Then, for every $v \neq u$,

$$\widehat{\Delta}_{v,u}(\boldsymbol{\lambda} \mid h) < -\varepsilon.$$

Using the pairwise-gap error bound,

$$\Delta_{v,u}(\boldsymbol{\lambda} \mid h) \leq \widehat{\Delta}_{v,u}(\boldsymbol{\lambda} \mid h) + \varepsilon < 0.$$

Therefore $\ell_u(\boldsymbol{\lambda} \mid h) > \ell_v(\boldsymbol{\lambda} \mid h)$ for every $v \neq u$, so u is the unique exact greedy winner throughout its certified region.

Moreover, if $\varepsilon_1 \leq \varepsilon_2$, then

$$G_u^{\varepsilon_2}(h; R) \subseteq G_u^{\varepsilon_1}(h; R),$$

and consequently $\mathcal{U}^{\varepsilon_1}(h; R) \subseteq \mathcal{U}^{\varepsilon_2}(h; R)$. At $\varepsilon = 0$, the robust construction agrees with LPT away from exact ties.

C Numerical Certification Details

Practical model inference can introduce small numerical changes in token logits and therefore in pairwise logit gaps. We separate these effects from the numerical precision used to construct the polyhedral regions. Geometric operations are performed at higher precision, while implementation-specific tolerances and solver settings are reported in Appendix D.

To account for variation in the model logits, we estimate a robust tolerance $\hat{\varepsilon}$ on a separate set of calibration prompts. For each calibration example, we compare pairwise logit gaps obtained under two numerical execution conditions and record their discrepancy. We set $\hat{\varepsilon}$ to the largest observed discrepancy. Each nominal LPT cell is then intersected with the prefix-wise affine halfspaces whose computed winner margins exceed $\hat{\varepsilon}$, as required by Equation 11; control values outside these certified subregions are reported as unresolved. This calibration is empirical: it supports claims under the measured prompt set and execution pair, but it is not a formal uniform guarantee over all hardware, kernels, or future prompts. In the reported experiments, $\hat{\varepsilon}$ is calibrated from bfloat16–fp32 differences and is used as a conservative stress test of finite-precision certification. The near-zero coverage at $\varepsilon = \hat{\varepsilon}$ should therefore be read as a numerical-margin diagnostic, not as a failure of nominal LPT partition recovery. Table 2 reports the resulting calibrated pairwise-gap tolerances.

Table 2: Calibrated pairwise-gap tolerance $\hat{\varepsilon}$ used for ε-robust certification.

Model	$d = 1$ $\hat{\varepsilon}$	$d = 2$ $\hat{\varepsilon}$
Qwen2.5-3B	14.5311	16.7943
Qwen2.5-1.5B	3.9308	2.7036
SmolLM2-1.7B	6.9483	4.0013

D Experimental Details

D.1 Experimental configuration

Table 3 summarizes the common experimental configuration, including the base models, control sources, prompt sets, decoding settings, numerical configuration, implementation, and hardware. The experiments use balanced prompt sets from Alpaca and PKU-SafeRLHF. The test set contains

100 prompts from each source, while a separate calibration set contains 50 prompts from each source for estimating $\hat{\varepsilon}$. PKU-SafeRLHF prompts are drawn from a held-out pool disjoint from the DPO adapter-training prompts.

Table 3: Reproducibility record for the reported experiments.

Configuration	Value
Models and controls	
Base models	Qwen2.5-3B; Qwen2.5-1.5B; SmolLM2-1.7B
$d = 1$ control	Better/Safer LoRA; PKU-SafeRLHF-10K
$d = 2$ control	Factual/Helpful/Safe
Control domains	$d = 1$: $[0, 1]$; $d = 2$: simplex
Evaluation	
Test prompts	200 (100 Alpaca; 100 PKU-SafeRLHF)
Calibration prompts	100 (50 Alpaca; 50 PKU-SafeRLHF)
Decoding	Greedy; $T = 64$
Candidate size	$k = 64$
Sampling budgets	$B \in \{6, 10, 21, 36, 66, 120\}$
Pruning / node limit	None; 20,000
Numerical configuration	
Model inference	fp32; batch size 1
Geometry	float64; gap/vertex tolerance 10^{-10}; area tolerance 10^{-12}
Certification	$\hat{\varepsilon}$; bfloat16–fp32
Implementation	
Software	Python 3.13.11; PyTorch 2.10.0; Transformers 4.57.6
Libraries	PEFT 0.19.1; NumPy 2.5.0
Hardware	4× NVIDIA GeForce RTX 4090

Tie handling. The reported experimental implementation resolves exact ties deterministically in the one-dimensional kernel, while the two-dimensional kernel does not separately export lower-dimensional zero-area tie atoms. Lower-dimensional tie cells are excluded from K_+ when computing output-cell recall and have zero measure, so they do not contribute to domain coverage. The supplementary all-ties implementation additionally preserves every maximizing branch, matching the set-valued tie semantics of Sections 3–4 and Theorem 1.

D.2 Control construction

One-dimensional control. For $d = 1$, each base model is paired with two LoRA adapters trained with DPO on PKU-SafeRLHF-10K, one using the better-response preference and the other using the safer-response preference. Let $z_v^{\text{better}}(h)$ and $z_v^{\text{safer}}(h)$ denote their logits for token v at prefix h. We define the controlled logit for $s \in [0, 1]$ as

$$\ell_v(s \mid h) = (1 - s)z_v^{\text{better}}(h) + sz_v^{\text{safer}}(h). \tag{16}$$

Thus, $s = 0$ corresponds to the better-oriented source and $s = 1$ to the safer-oriented source.

Two-dimensional control. For $d = 2$, no additional adapters are trained. Instead, the same base model is evaluated under three author-defined system prompts corresponding to factual, helpful, and safe behavior:

> **Helpful:** *You are a helpful assistant. Give the user the most useful direct answer you can while following the request.*
>
> **Safe:** *You are a safety-focused assistant. Refuse unsafe requests and redirect toward benign, legal, and constructive alternatives.*
>
> **Factual:** *You are a factual assistant. Be precise, calibrated, and explicit about uncertainty. Avoid speculation.*

Let $z_v^{\text{factual}}(h)$, $z_v^{\text{helpful}}(h)$, and $z_v^{\text{safe}}(h)$ denote the resulting source logits. Over the simplex $\lambda_1 \geq 0$, $\lambda_2 \geq 0$, and $\lambda_1 + \lambda_2 \leq 1$, we define

$$\ell_v(\lambda_1, \lambda_2 \mid h) = (1 - \lambda_1 - \lambda_2)z_v^{\text{factual}}(h) + \lambda_1 z_v^{\text{helpful}}(h) + \lambda_2 z_v^{\text{safe}}(h). \tag{17}$$

The simplex vertices $(0, 0)$, $(1, 0)$, and $(0, 1)$ therefore correspond to the factual, helpful, and safe sources, respectively.

D.3 DPO ADAPTER TRAINING DETAILS

Table 4 summarizes the adapter-training configuration used to construct the one-dimensional better–safer endpoint logits. The better adapter treats the dataset's better response as chosen, whereas the safer adapter treats the safer response as chosen. The 500-prompt held-out PKU pool used for safety evaluation is disjoint from the adapter training prompts.

Table 4: DPO LoRA training details for the $d = 1$ better and safer adapters.

Field	Value
Dataset	PKU-Alignment/PKU-SafeRLHF-10K train split
Adapters	Two per base model: better-chosen and safer-chosen DPO objectives
Training prompts	5,429 unique prompts; 9,000 training rows used
Validation rows	500 rows from the validation pool
Held-out prompts	500 prompts; overlap with training prompts: 0
LoRA	rank 16; alpha 32; dropout 0.05; bias none
Target modules	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj
DPO / optimization	$\beta = 0.1$; learning rate 10^{-4}; weight decay 0; max grad norm 1.0; AdamW fused; cosine schedule; warmup ratio 0.05
Batching	per-device batch size 2; gradient accumulation 16
Training length	max steps 320; max sequence length 512; seed 20260815

D.4 FINITE-SAMPLING BASELINES

Uniform sampling uses a deterministic low-discrepancy order. In one dimension, the order begins with the two endpoints and then follows the base-2 radical inverse sequence. In two dimensions, it begins with the simplex vertices and centroid and then maps a deterministic low-discrepancy sequence into the simplex. This ordering is used only for resource-matched anytime runs; the budget sweep in Figure 3 uses completed fixed-budget grids.

Adaptive subdivision is also evaluated as an anytime sampler. In one dimension, it starts from the endpoints and midpoint, then repeatedly splits the widest interval by adding the quarter points. In two dimensions, it starts from the simplex vertices and centroid, then repeatedly subdivides the largest triangle by adding its edge midpoints and centroid. Children are processed breadth-first with a largest-cell priority, and previously evaluated vertices are reused within the subdivision state. Because our comparisons use fixed sampling budgets, subdivision continues refining the largest available cells even when the current evaluated labels agree, so that every method spends the same declared sample budget.

E ADDITIONAL RESULTS

E.1 RESOURCE-MATCHED SAMPLE COUNTS

Table 5 shows that the number of sampled control values varies substantially across models and resource constraints. Matching LPT's model-evaluation budget generally permits fewer sampled values than matching its execution time, particularly in the two-dimensional setting.

Table 5: Reference output-cell counts and finite-sampling budgets for Table 1. N_{pts} is the number of evaluated control values per prompt; entries report means over prompts. Panel A matches time; Panel B matches model evaluations.

d	Model	LPT cells per prompt	Panel A Uniform N_{pts}	Panel A Adaptive N_{pts}	Panel B Uniform N_{pts}	Panel B Adaptive N_{pts}
1	Qwen2.5-3B	5.23	16.44	16.50	3.76	3.76
1	Qwen2.5-1.5B	8.98	7.30	7.27	5.94	5.94
1	SmolLM2-1.7B	10.36	11.60	11.60	6.67	6.66
2	Qwen2.5-3B	68.30	77.35	74.60	34.94	34.22
2	Qwen2.5-1.5B	18.82	94.99	94.42	15.61	15.36
2	SmolLM2-1.7B	26.71	35.57	35.57	16.59	16.39

All six nominal LPT settings in Table 5 completed on all 200 test prompts without reaching the 20,000-node limit.

E.2 Robust partition sweep

Table 6 shows that certified coverage decreases rapidly as the tolerance increases. At the calibrated bound $\hat{\varepsilon}$, only 0–0.5% of the domain remains certified across all model and control-dimension settings. The rate of decline varies across models, with Qwen2.5-1.5B retaining substantially more certified coverage at small intermediate tolerances than the other two models. This pattern is consistent with model-dependent differences in winner margins across the recovered partitions.

Table 6: Certified domain coverage under increasing robust tolerance ε. Uncovered control values are unresolved; the rightmost column uses the calibrated pairwise-gap bound $\hat{\varepsilon}$.

d	Model	$\varepsilon = 0$	$0.005\hat{\varepsilon}$	$0.01\hat{\varepsilon}$	$0.05\hat{\varepsilon}$	$0.25\hat{\varepsilon}$	$\hat{\varepsilon}$
1	Qwen2.5-3B	100.0	28.5	13.9	4.3	1.0	0.0
	Qwen2.5-1.5B	100.0	64.6	43.8	7.6	0.9	0.5
	SmolLM2-1.7B	100.0	39.9	19.1	2.3	0.0	0.0
2	Qwen2.5-3B	100.0	24.6	12.1	4.0	0.7	0.0
	Qwen2.5-1.5B	100.0	78.1	63.0	28.6	14.3	0.5
	SmolLM2-1.7B	100.0	63.4	41.9	8.0	1.6	0.0

E.3 Higher-Dimensional Affine Controls

Table 7 reports a synthetic affine-logit stress test for control dimensions $d \in \{1, 2, 3, 4\}$. The main experiments evaluate LPT for $d \in \{1, 2\}$, including the two-dimensional simplex setting. To examine how the geometric construction scales beyond these settings, we additionally evaluate synthetic affine-logit instances for $d \in \{1, 2, 3, 4\}$, using both generic random instances and near-tie stress cases. These experiments isolate the polyhedral component of LPT from autoregressive model inference and measure how the number of active argmax cells grows with control dimension. Across 50 trials with vocabulary size 256, generic random instances recover 3.72, 8.20, 14.60, and 22.80 argmax cells on average for $d = 1, 2, 3, 4$, respectively. The near-tie stress cases recover 3.56, 8.02, 14.08, and 25.04 cells on average, showing that the same polyhedral construction remains applicable in the presence of constructed near ties.

Table 7: Synthetic affine-logit stress test for higher-dimensional controls. Values are mean recovered argmax cells over 50 trials with vocabulary size 256.

Instance family	$d = 1$	$d = 2$	$d = 3$	$d = 4$
Generic random	3.72	8.20	14.60	22.80
near-tie stress	3.56	8.02	14.08	25.04

E.4 Dense-grid spot checks

Table 8 reports dense finite-grid spot checks against the LPT reference partition for completed $d = 2$ runs. These checks are not exhaustive certificates: they only evaluate 1326 simplex grid points per prompt. They nevertheless test whether densely sampled greedy decisions are contained in the LPT partition. Across all three completed model settings, every sampled dense-grid decision falls inside the LPT cell covering that parameter value, yielding zero sample-level mismatches and 100% decision coverage. The output-cell recall remains below 100% for some models because even a dense finite grid can miss small positive-measure cells. On the same 30-prompt subsets, these dense grids require 55.1–$148.3\times$ more source-logit forward calls than LPT while still recovering only 89.3–95.3% of output cells. This complements the budget sweep in Figure 3. Increasing the sampling budget improves recall, but finite sampling remains an empirical coverage strategy rather than a certificate of complete partition recovery.

Table 8: Dense-grid spot checks against the LPT reference partition for $d = 2$. Each run uses 30 prompts and 1326 simplex grid points per prompt. LPT evaluations are recomputed on the same 30 prompts.

Model	Prompts	Grid points per prompt	Dense evals.	LPT evals.	Eval ratio	Sample mismatches	Decision coverage (%)	output-cell recall (%)
Qwen2.5-3B	30	1326	218262	3959	55.1×	0	100.0	89.3
Qwen2.5-1.5B	30	1326	221090	1491	148.3×	0	100.0	94.9
SmolLM2-1.7B	30	1326	225679	1739	129.8×	0	100.0	95.3

E.5 Sensitivity to batch size and numerical precision

Table 9 reports the sensitivity of the recovered LPT partition to batch size and numerical precision. For each model and control dimension, we vary inference precision over {fp32, bfloat16} and batch size over $\{1, 4, 8\}$. The fp32, batch-size-1, $\varepsilon = 0$ partition serves as the reference. Across all tested configurations, the recovered $\varepsilon = 0$ partitions retain 100% output-cell recall and 100% domain coverage relative to this reference. Coverage under the calibrated tolerance $\hat{\varepsilon}$ remains between 0 and 0.5%, consistent with the conservative robust-certification results in the main experiments.

Table 9: Sensitivity to batch size and numerical precision. Each setting is compared with the fp32, batch-size-1, $\varepsilon = 0$ reference for the same model and control dimension.

Model	d	Min output-cell recall (%)	Min domain coverage (%)	Coverage at $\hat{\varepsilon}$ (%)
Qwen2.5-3B	1	100.0	100.0	0.0
Qwen2.5-3B	2	100.0	100.0	0.0
Qwen2.5-1.5B	1	100.0	100.0	0.5
Qwen2.5-1.5B	2	100.0	100.0	0.5
SmolLM2-1.7B	1	100.0	100.0	0.0
SmolLM2-1.7B	2	100.0	100.0	0.0

E.6 Sensitivity to candidate-set size

Table 10 reports partition invariance across initial candidate-set sizes. We vary the initial candidate size over $k \in \{1, 4, 16, 64, 256\}$ and compare the resulting partitions with the $k = 64$ fp32 reference. The completed candidate-size runs recover identical output-cell recall and domain coverage across every model and control dimension.

Table 10: Candidate-set invariance across initial candidate sizes. Each setting is compared with the corresponding $k = 64$ fp32 reference partition.

Model	d	Candidate sizes k	Runs	Min recall (%)	Min coverage (%)
Qwen2.5-3B	1	1, 4, 16, 64, 256	5	100.0	100.0
Qwen2.5-3B	2	1, 4, 16, 64, 256	5	100.0	100.0
Qwen2.5-1.5B	1	1, 4, 16, 64, 256	5	100.0	100.0
Qwen2.5-1.5B	2	1, 4, 16, 64, 256	5	100.0	100.0
SmolLM2-1.7B	1	1, 4, 16, 64, 256	5	100.0	100.0
SmolLM2-1.7B	2	1, 4, 16, 64, 256	5	100.0	100.0

E.7 Property-level existence check

Table 11 evaluates the motivating existence question with a fixed deterministic property. For the 100 safety-family prompts, the property is positive when a decoded output contains at least one of the lexical terms *safe*, *safety*, *legal*, *ethical*, *harmful*, *dangerous*, *illegal*, *benign*, *constructive*, or *instead*. This is not a semantic safety evaluator; it is a deterministic output property used to test whether finite sampling detects a property that exists somewhere in the LPT partition. The table reports the matched model-evaluation budget setting from Table 1.

Table 11: Existence recall for a deterministic safety-term property on safety prompts under the matched model-evaluation budget.

d	Model	LPT-positive prompts	Uniform detected	Uniform recall (%)	Adaptive detected	Adaptive recall (%)
1	Qwen2.5-3B	68	62	91.2	62	91.2
1	Qwen2.5-1.5B	68	56	82.4	56	82.4
1	SmolLM2-1.7B	55	46	83.6	46	83.6
2	Qwen2.5-3B	94	89	94.7	90	95.7
2	Qwen2.5-1.5B	62	51	82.3	51	82.3
2	SmolLM2-1.7B	74	66	89.2	65	87.8

Because sampled outputs are a subset of the LPT partition, false positives are zero relative to this fixed property definition. The misses show that finite sampling can fail to detect that a deterministic output property exists in the continuous domain, even when it recovers a large fraction of the domain by measure.

E.8 Decoded region example

Table 12 shows the decoded continuations associated with a representative recovered partition. To illustrate the decoded outputs associated with the recovered partition, we select a compact SmolLM2-1.7B, $d = 2$ example with exactly 20 LPT cells. Each row corresponds to an entire polyhedral region over which the greedy output is constant, rather than to a single sampled control value. The example also illustrates the distinction between exact-output recovery and semantic property recovery: several missed cells are close textual variants of larger observed regions, so behavioral claims should be made through an explicit deterministic property evaluator such as the one in Appendix E.7.

Table 12: All decoded outputs for a representative $d = 2$ partition with 20 cells using SmolLM2-1.7B. Area is reported as a percentage of the simplex. Margin is the minimum computed winner gap inside the corresponding LPT cell. "Seen" indicates the methods that observe the output associated with each cell. For finite-sampling methods, this means that at least one evaluated control value lies within the cell. "Missed by" indicates the methods that do not. L, V, U, and A denote LPT, vertices-only, the 66-point uniform grid, and adaptive subdivision, respectively.

Cell	Area	Margin	Seen	Missed by	Greedy continuation
0	0.002	0.0011	L	V,U,A	Name: Luna "Luna" Thompson Age: 15 years old Appearance: Luna is a bright and curious 15-year-old with a mop of curly brown hair and bright green eyes. She has a collection of colorful tattoos on her arms and a pet rabbit named Mr
1	0.006	0.0009	L	V,U,A	Name: Luna "Luna" Thompson Age: 15 Appearance: Luna has long, curly brown hair and bright green eyes. She is a bit on the taller side, standing at 5'8" with a slender yet athletic build. She often dresses in comfortable, casual clothing,
2	0.028	0.0037	L	V,U,A	Name: Luna "Luna" Thompson Age: 15 years old Appearance: Luna is a curious and adventurous 15-year-old with a mop of curly brown hair and bright green eyes. She has a collection of colorful tattoos on her arms and a pet rabbit named Mr
3	0.037	0.0035	L	V,U,A	Character Name: Luna "Luna" Thompson Age: 15 Appearance: Luna has long, curly brown hair and bright green eyes. She is petite, standing at 4'8" with a slender build. She often wears casual, comfortable clothing and has a collection of colorful scar
4	0.041	0.0019	L	V,U,A	Name: Luna Nightingale Age: 15 Appearance: Luna has long, curly brown hair and bright green eyes. She is petite, standing at 4'8" with a slender build. She often wears a pair of oversized sunglasses and a faded band t-shirt. She
5	0.111	0.0052	L	V,U,A	Name: Luna "Luna" Thompson Age: 15 Appearance: Luna has long, curly brown hair and bright green eyes. She is a bit on the petite side, but her curiosity and adventurous spirit make her a force to be reckoned with. She often dresses in comfortable clothing
6	0.241	0.0053	L,U	V,A	Name: Luna "Luna" Thompson Age: 15 years old Appearance: Luna has long, curly brown hair and bright green eyes. She is petite and has a collection of colorful tattoos on her arms. She often wears a pair of oversized sunglasses and a faded band t-
7	0.404	0.0160	L	V,U,A	Name: Luna "Luna" Thompson Age: 15 years old Appearance: Luna has long, curly brown hair and bright green eyes. She is a bit on the petite side, but her curiosity and adventurous spirit make her a force to be reckoned with. She often dresses in

Cell	Area	Margin	Seen	Missed by	Greedy continuation
8	0.494	0.0077	L,U,A	V	Character Name: Luna ”Luna” Thompson Age: 15 years old Physical Description: Luna is a curious and adventurous 15-year-old with a pet rabbit named Rosie. She has long, curly brown hair and bright green eyes. She is usually dressed in comfortable clothing,
9	0.667	0.0102	L	V,U,A	Name: Luna ”Luna” Thompson Age: 15 years old Appearance: Luna has long, curly brown hair and bright green eyes. She is petite and has a collection of colorful tattoos on her arms. She often wears a pair of worn-out jeans and a faded band t
10	0.691	0.0191	L,U,A	V	Name: Luna ”Luna” Thompson Age: 15 Appearance: Luna has long, curly brown hair and bright green eyes. She is petite and has a collection of colorful tattoos on her arms. She often wears a pair of oversized sunglasses and a faded band t-shirt.
11	1.16	0.0574	L,V,U,A	–	Character Name: Elara ”Lara” Thompson Age: 15 years old Physical Description: Elara is a curious and adventurous 15-year-old girl with short, curly brown hair and bright green eyes. She has a small tattoo of a rabbit on her left wrist, symbolizing
12	1.55	0.0273	L,V,U,A	–	Character Name: Luna ”Luna” Thompson Age: 15 Appearance: Luna is a bright and curious 15-year-old with a mop of curly brown hair and bright green eyes. She has a collection of colorful tattoos on her arms and a pet rabbit named Mr.
13	1.81	0.0064	L,U,A	V	Character Name: Luna ”Luna” Thompson Age: 15 Appearance: Luna is a curious and adventurous 15-year-old with a mop of curly brown hair and bright green eyes. She has a collection of colorful tattoos on her arms and a pet rabbit named Mr.
14	2.87	0.0108	L,U,A	V	Name: Luna Nightingale Age: 15 Appearance: Luna has long, curly brown hair and bright green eyes. She is petite, but her curiosity and adventurous spirit make her appear more daring than her size. She often wears a pair of worn-out jeans and a faded band t
15	3.79	0.0279	L,U,A	V	Character Name: Luna ”Luna” Thompson Age: 15 Appearance: Luna is a curious and adventurous 15-year-old with an ethereal aura. She has long, curly brown hair and bright green eyes that sparkle with curiosity. She is petite, but
16	5.18	0.0632	L,U,A	V	Name: Luna Nightingale Age: 15 years old Appearance: Luna has long, curly brown hair and bright green eyes. She is petite and has a collection of colorful tattoos on her arms. She often wears a pair of worn-out jeans and a faded band t-shirt.
17	13.17	0.0321	L,U,A	V	Name: Luna Nightingale Age: 15 Appearance: Luna has long, curly brown hair and bright green eyes. She is petite and has a collection of colorful tattoos on her arms. She often wears a pair of worn-out jeans and a faded band t-shirt. She has
18	30.52	0.1303	L,V,U,A	–	Name: Luna Nightingale Age: 15 Appearance: Luna has long, curly brown hair and bright green eyes. She is petite and has a collection of colorful tattoos on her arms. She often wears a pair of worn-out jeans and a faded band t-shirt.
19	37.21	0.1137	L,U,A	V	Character Name: Luna ”Luna” Thompson Age: 15 years old Physical Description: Luna is a curious and adventurous 15-year-old with an ethereal aura. She has long, curly brown hair and bright green eyes that sparkle with curiosity. She is petite