Proposal Collapse and Execution Fibers in Stochastic Program Generation

Thomas Dionysopoulos, CFA

Abstract

When a stochastic code generator (an LLM) proposes executable specifications in response to natural-language prompts, two distinct kinds of variation emerge. *Surface-form variation* produces syntactically different proposals that resolve to the same deterministic execution. *Execution ambiguity* produces proposals that resolve to different executions. The operational distinction between these two phenomena is induced by the deterministic execution system. We formalize this distinction through *execution fibers*—the preimages of execution classes under a deterministic projection—and measure its empirical structure using controlled perturbation experiments.

Across 2,200 proposals (1,200 baseline, 1,000 controlled perturbations), we observe 28 execution classes with non-uniform fiber cardinality (Gini = 0.467). Synonym perturbations produce near-perfect intra-fiber stability ($\bar{\sigma} = 0.992$, $\bar{\rho} = 0.985$): surface rewording is absorbed by canonicalization. Metric and family substitutions produce *zero* same-fiber transition mass ($\bar{\rho} = 0.000$) with *perfect* per-variant stability ($\bar{\sigma} = 1.000$): they create clean transitions to different execution identities, not noisy instability. The execution adjacency graph is sparse (density = 0.095, 10 connected components), and prompt support correlates strongly with fiber cardinality ($r = 0.925$).

These results demonstrate that deterministic execution semantics induces observable, measurable structure on the stochastic proposal space. The structure is discrete, finite, and made observable through the execution system's provenance structure.

1 Introduction

A system that accepts natural-language requests and produces executable programs faces a structural problem: the mapping from prompts to programs is stochastic, but program execution is deterministic. The same prompt, issued repeatedly to an LLM, yields different textual outputs. Some of these differences do not change execution identity—a parameter reordered, a synonym substituted—and some do: a different scoring metric, a different strategy family, a different parameter range.

The standard approach treats all variation uniformly. Either the generator is "correct" (it produced an acceptable program) or it is "wrong" (it did not). This binary view discards structure. The variation is not uniform: it decomposes into a part that is absorbed by deterministic canonicalization and a part that survives.

This paper makes that decomposition explicit. We define *execution fibers* as the preimages of execution classes under a deterministic projection. We design controlled perturbation experiments that isolate the two kinds of variation. We measure the resulting structure across 2,200 LLM-generated proposals in a quantitative finance execution system (BLISP) and find that:

1. Fiber cardinality is non-uniform and heavy-tailed (Gini = 0.467).
2. Synonym perturbations are almost entirely intra-fiber ($\bar{\rho} = 0.985$).
3. Metric and family substitutions create clean inter-fiber transitions ($\bar{\rho} = 0.000$), not diffuse instability.
4. The execution adjacency graph is sparse (density = 0.095).
5. Prompt support correlates strongly with fiber cardinality ($r = 0.925$).

The objects studied here are finite, discrete, and fully grounded in observable execution behavior. We make no claims about the internal structure of the generator. The structure we report is induced by deterministic execution semantics on observable proposal behavior.

2 The Proposal Collapse Problem

2.1 Setting

Consider a system with three components:

1. A *stochastic proposal generator* G (an LLM) that, given a natural-language prompt p, produces executable specifications $e \in \mathcal{E}_R$ where $\mathcal{E}_R$ is the set of all valid specifications expressible in the execution system's representation R.
2. A *deterministic execution system* that evaluates specifications.
3. A *canonicalization function* $\pi : \mathcal{E}_R \to \mathcal{B}_R/\sim_R$ that maps every valid specification to an execution class $[a]_R$.

The canonicalization function is deterministic: given the same specification and the same system state (dictionary, data), it always produces the same execution class. In our implementation, π is realized as a cryptographic hash (SHA-256) over the canonical fields of a Formal Proposal Record (FPR): goal, family, scoring metric, parameters, and dictionary hash. Two specifications belong to the same execution class if and only if they produce the same hash.

The *collapse problem* is that G is stochastic while π is deterministic. Repeated calls $G(p)$ produce a distribution over $\mathcal{E}_R$, and π collapses that distribution onto $\mathcal{B}_R/\sim_R$. The question is: what structure does this collapse exhibit?

2.2 Collapse Ratio

Prior work (Paper 2 in this series) introduced the *collapse ratio*:

$$\gamma = \frac{\delta_S}{\delta_C}$$

where δ_S is the number of distinct surface-form specifications and δ_C is the number of distinct execution classes, measured over repeated generations from the same prompt at a fixed temperature. When $\gamma > 1$, multiple surface forms map to the same execution class. Paper 2 measured γ across 1,200 generations (30 prompts $\times$ 4 temperatures $\times$ 10 repetitions) and found that γ increases with temperature, confirming that surface variation grows faster than execution variation.

The collapse ratio establishes that many-to-one collapse occurs. It does not characterize the *structure* of that collapse: which specifications collapse together, how stable the collapse is under perturbation, or what the graph of transitions between execution classes looks like. This paper provides those characterizations.

3 Execution Fibers

3.1 Definitions

Definition 1 (Execution Fiber)**.** *For an execution class $[a]_R \in \mathcal{B}_R/\sim_R$, the* execution fiber *is the preimage under the deterministic projection:*

$$F([a]_R) = \pi^{-1}([a]_R) = \{e \in \mathcal{E}_R : \pi(e) = [a]_R\}$$

The fiber $F([a]_R)$ contains all specifications—however differently worded or structured—that resolve to the same execution class. Variation within a fiber is *surface-form variation*: provenance-invisible and absorbed by canonicalization.

Definition 2 (Fiber Cardinality)**.** *Given a sample S of proposals, the* sample fiber cardinality *is:*

$$\phi_S([a]_R) = |F([a]_R) \cap S|$$

Fiber cardinality is sample-dependent. We observe ϕ_S over a corpus; we do not claim to measure $|F([a]_R)|$ in any absolute sense.

Definition 3 (Collapse Stability)**.** *For a prompt p and n independent generations, the* collapse stability *is:*

$$\sigma(p) = \frac{1}{n} \max_{[a]_R} \sum_{i=1}^{n} \mathbb{1}[\pi(e_i) = [a]_R]$$

Collapse stability measures the fraction of proposals from a single prompt that land in the dominant execution class. When $\sigma(p) = 1$, all proposals from p resolve to the same class: the prompt's behavior is fully determined despite stochastic generation.

Definition 4 (Prompt Support)**.** *The* prompt support *of an execution class is the set of distinct prompt variants that reach it:*

$$S([a]_R) = \{p : [a]_R \in C_P(p)\}$$

where $C_P(p) = \{\pi(e) : e \in G(p)\}$ is the set of execution classes reached by prompt p.

3.2 The Central Distinction

The fiber structure induces a partition of all observable proposal variation into exactly two categories:

	Surface-form variation	**Execution ambiguity**
Location	Intra-fiber	Inter-fiber
Provenance	Invisible	Visible
Effect	Absorbed by canonicalization	Changes execution identity
Example	"ranked by" vs. "sorted by"	Sharpe ratio vs. max drawdown

This distinction is a structural consequence of the existence of a deterministic projection π. Any system that (a) accepts stochastic proposals and (b) evaluates them deterministically with a canonical identity function necessarily partitions proposal variation into these two categories.

The deterministic execution system induces the operational distinction between intra-fiber and inter-fiber variation. The canonicalization function π determines the equivalence relation; prompt structure contributes to which classes are reached, but not to which proposals are equivalent.

4 Perturbation and Collapse Profiles

4.1 Perturbation Design

To measure the fiber structure empirically, we need controlled perturbations that isolate the two kinds of variation. We define three perturbation types with decreasing expected intra-fiber mass:

Synonym perturbation (E2). Replace surface-level words while preserving the intended execution identity: "ranked by" ↔ "sorted by", "strategy" ↔ "approach", "returns" ↔ "return". These perturbations should not change which execution class the generator targets. If the execution system's canonicalization is well-designed, synonym perturbations should be almost entirely intra-fiber.

Metric substitution (E3). Hold the strategy family constant and substitute the scoring metric: Sharpe ratio → max drawdown → volatility → risk-adjusted drawdown. These perturbations change a provenance-visible field (the metric is part of the canonical hash), so each metric should produce a different execution class. Per-variant stability should remain high—each metric prompt individually should collapse to a single class—but cross-variant transition mass should be zero.

Family substitution (E4). Hold the scoring metric constant and substitute the strategy family: momentum → carry, momentum → mean reversion. These perturbations change the most structurally significant provenance field. Like metric substitution, they should produce clean inter-fiber transitions with high per-variant stability.

4.2 Transition Mass

Definition 5 (Same-Fiber Transition Mass)**.** *For two prompt variants p_i, p_j each generating n proposals, the* same-fiber transition mass *is:*

$$\rho(p_i, p_j) = \frac{1}{n^2}\left|\{(e, e') : \pi(e) = \pi(e'),\ e \in G(p_i),\ e' \in G(p_j)\}\right|$$

When $\rho = 1$, every proposal from p_i and every proposal from p_j land in the same fiber: the perturbation is entirely intra-fiber. When $\rho = 0$, no proposal pair shares a fiber: the perturbation produces complete inter-fiber separation.

4.3 Experimental Protocol

All experiments use `claude-sonnet-4-20250514` at temperature 0.7. Canonical hashing uses the BLISP `FPR-HSH` builtin (SHA-256 over canonicalized FPR fields, dictionary-hash dependent).

Experiment	Design	API calls
E1 (baseline)	30 prompts × 4 temperatures × 10 reps	1,200
E2 (synonym)	10 families × 5 variants × 10 reps	500
E3 (metric)	10 families × 4 metrics × 10 reps	400
E4 (family)	5 families × 2 variants × 10 reps	100
	Total	2,200

E1 reuses the 1,200-generation corpus from Paper 2 (no new API calls). E2, E3, E4 are new controlled perturbation experiments (1,000 new API calls). Validation rate across all 2,200 proposals: 100%.

5 Collapse Stability and Transition Mass

5.1 Collapse Stability Results

Table 1 reports collapse stability σ by perturbation type. Each row summarizes σ values across all prompt variants of that type, where σ for a single variant is the fraction of its 10 repetitions that land in the dominant execution class.

Three observations:

Table 1: Collapse stability σ by perturbation type.

Type	n	$\bar{\sigma}$	med σ	min σ	perfect
Baseline (E1)	30	0.983	1.000	0.775	83.3%
Synonym (E2)	50	0.992	1.000	0.900	92.0%
Metric (E3)	40	1.000	1.000	1.000	100.0%
Family (E4)	10	1.000	1.000	1.000	100.0%

Synonym stability is high. Mean $\sigma = 0.992$. Of 50 synonym variants, 46 (92%) achieve perfect collapse ($\sigma = 1.0$). The remaining 4 variants, all from a single perturbation family (SYN05: "momentum ranked by returns"), have $\sigma = 0.9$: 9 of 10 repetitions land in the dominant class. The minority class in SYN05 differs in the family field (`MOM_REV` vs. `MOM_WZS`), reflecting genuine ambiguity in the prompt rather than canonicalization failure.

Metric and family stability is perfect. Every metric variant (40/40) and every family variant (10/10) achieves $\sigma = 1.0$. When the prompt unambiguously specifies a metric or family, the generator produces the same execution class in all 10 repetitions. This is not trivial: at temperature 0.7, the generator's surface text varies across repetitions, but all variation is absorbed by canonicalization.

Baseline stability is slightly lower. The baseline prompts (from Paper 2's uncontrolled corpus) include deliberately ambiguous and adversarial prompts. Their mean $\sigma = 0.983$ with a minimum of 0.775 reflects real execution ambiguity in some prompts, not canonicalization failure.

5.2 Transition Mass Results

Table 2 reports same-fiber transition mass ρ by perturbation type. Each row summarizes ρ values across all variant pairs within perturbation families.

Table 2: Same-fiber transition mass ρ by perturbation type.

Type	pairs	$\bar{\rho}$	min ρ	perfect
Synonym	100	0.985	0.820	90.0%
Metric	60	0.000	0.000	0.0%
Family	5	0.000	0.000	0.0%

The transition mass results are sharp:

Synonym perturbations are intra-fiber. Mean $\rho = 0.985$. Of 100 synonym variant pairs, 90 have $\rho = 1.0$ (every proposal pair shares a fiber). The 10 imperfect pairs all involve SYN05, the single family with a minority alternative class.

Metric and family substitutions produce zero same-fiber mass. Every metric pair ($\rho = 0.000$ across all 60 pairs) and every family pair ($\rho = 0.000$ across all 5 pairs) achieves complete inter-fiber separation. No proposal from a "Sharpe" variant ever shares a fiber with a proposal from a "max drawdown" variant of the same prompt.

This result has an important interpretation: metric and family substitutions do not create noisy instability. They create *clean movement* to different execution identities. Per-variant stability is perfect ($\sigma = 1.0$); it is only the cross-variant transition mass that is zero. The

generator reliably produces the execution class corresponding to the specified metric or family; it does not wander unpredictably across classes.

5.3 Separation Principle

These two results jointly establish a separation:

> Surface-level rewording (synonyms) is almost entirely absorbed by the execution system. Provenance-level changes (metric, family) produce clean, stable transitions between execution classes.

The operational distinction between absorbed and non-absorbed variation is not gradual. It is sharp: synonyms produce $\bar{\rho} = 0.985$; metric/family substitutions produce $\bar{\rho} = 0.000$. The deterministic execution system separates these two regimes cleanly.

6 Execution Adjacency

6.1 Adjacency Graph

Definition 6 (Execution Adjacency)**.** *Two execution classes* $[a]_R, [b]_R$ *are* adjacent *if there exist perturbation-related prompt variants* $p_i \sim_P p_j$ *such that* $[a]_R \in C_P(p_i)$ *and* $[b]_R \in C_P(p_j)$*.*

The adjacency graph $\mathcal{G} = (V, E, w)$ has execution classes as vertices and perturbation-witnessed transitions as edges. Edge weight records the number of perturbation witnesses supporting the transition.

Table 3: Execution adjacency graph properties.

Property	Value
Vertices (execution classes)	28
Unique edges	36
Density	0.095
Connected components	10
Largest component	19 vertices
Singleton components	9
Perturbation witnesses	81
SCR-layer witnesses	60
FAM-layer witnesses	21

The graph has 36 unique edges supported by 81 labeled perturbation witnesses: 60 at the scoring metric (SCR) layer and 21 at the family (FAM) layer.

6.2 Graph Structure

Three structural features are notable:

Sparsity. With 28 vertices, the complete graph would have $\binom{28}{2} = 378$ edges. The observed 36 edges give a density of 0.095. Most execution class pairs are never connected by a perturbation witness. Execution classes that differ in both family and metric (e.g., `MOM_WZS/SRP` and `CARRY/MDD`) are rarely adjacent because no single perturbation changes both fields simultaneously.

Component structure. The graph has 10 connected components. The largest component contains 19 of 28 vertices—these are the classes reachable from common prompt families through metric and family substitution. The 9 singleton components are execution classes reached only by specific baseline prompts that are not perturbation-related to any E2/E3/E4 variant.

Divergence layers. The 81 perturbation witnesses decompose into two provenance layers:

- **SCR-layer** (60 witnesses): transitions between classes that share a family but differ in metric. These are induced by E3 (metric substitution) experiments.
- **FAM-layer** (21 witnesses): transitions between classes that differ in family. These include E4 (family substitution) experiments and cross-family transitions observed in baseline prompts.

The strongest single edge (weight 16) connects `MOM_REV/RET` and `MOM_WZS/RET`: a FAM-layer transition produced by the SYN05 minority variant, where the prompt "momentum ranked by returns" occasionally produces mean-reversion instead of z-score momentum.

7 Ambiguity Structure

Not all execution ambiguity is uniform. The fiber structure shows where execution ambiguity concentrates in the observed data and what form it takes.

7.1 Fiber Cardinality Distribution

The 28 observed execution classes have highly non-uniform cardinality. Table 4 shows the 10 largest fibers.

Table 4: Top 10 execution classes by sample fiber cardinality ϕ_S.

Rank	Family / Metric	ϕ_S	Share	Support
1	MOM_WZS / SRP	257	11.7%	14
2	MOM_REV / SRP	250	11.4%	13
3	CARRY / SRP	250	11.4%	13
4	CARRY / MDD	166	7.5%	11
5	MOM_REV / MDD	150	6.8%	9
6	VOL_TGT / VOL	129	5.9%	10
7	CARRY / RET	100	4.5%	7
8	VOL_TGT / SRP	83	3.8%	9
9	CARRY / SRP*	80	3.6%	2
10	CARRY / VOL	71	3.2%	5

*Distinct from rank 3; differs in parameter values.

The top 3 classes hold 34.4% of all proposals. The top 5 hold 48.8%. The Gini coefficient is 0.467, indicating moderate concentration: fibers are non-uniform but not monopolistic.

The cardinality range spans two orders of magnitude: the largest fiber contains 257 proposals, the smallest contains 2. The median cardinality is 56.

7.2 Ambiguity Sources

The non-uniform distribution reflects the interaction between prompt diversity and the execution system's canonical categories. Several patterns are visible:

Metric-default bias. The three largest fibers all use Sharpe ratio (SRP) as the scoring metric. When a prompt does not strongly specify a metric, the generator defaults to Sharpe ratio, concentrating proposals in SRP-scored classes.

Family concentration. The 28 classes span 4 families (MOM_WZS, MOM_REV, CARRY, VOL_TGT) and 6 metrics (SRP, MDD, MDD_RADJ, RET, VOL, TRN), producing 21 distinct family/metric combinations. Several combinations have multiple classes (e.g., 3 distinct CARRY/MDD classes, 2 distinct MOM_WZS/SRP classes) that differ in parameter values.

Parameter-level splitting. The 7 classes that share a family/metric combination with another class differ only in parameter values (lookback windows, z-score periods). These are the finest-grained provenance distinctions: the execution system distinguishes parameter values that many prompts leave unspecified.

8 Empirical Results

8.1 Prompt Support

Table 5 reports the prompt support distribution.

Table 5: Prompt support statistics.

Property	Value
Execution classes	28
Max support $\lvert S([a]_R)\rvert$	14
Min support	1
Mean support	4.96
Support Gini	0.456
Support–fiber correlation	0.925

The strong correlation between prompt support and fiber cardinality ($r = 0.925$) is expected but worth documenting. Execution classes reached by many distinct prompts accumulate more proposals. The relationship is nearly linear: an execution class with twice the prompt support has approximately twice the fiber cardinality.

8.2 Hypothesis Assessment

We now assess the five hypotheses stated in the experimental design.

Hypothesis 1 (Non-uniform fiber cardinality)**.** *The distribution of ϕ_S across execution classes is non-uniform.*

Supported. Gini = 0.467. The top 3 of 28 classes (10.7%) contain 34.4% of proposals. Cardinality ranges from 2 to 257. The distribution is non-uniform.

Hypothesis 2 (Synonym perturbation stability)**.** *Synonym perturbations produce high same-fiber transition mass: $\bar{\rho}$ near 1.*

Strongly supported. $\bar{\sigma} = 0.992$, $\bar{\rho} = 0.985$. Of 100 synonym variant pairs, 90 achieve $\rho = 1.0$. All imperfect pairs involve a single perturbation family (SYN05) where the prompt contains genuine family-level ambiguity.

Hypothesis 3 (Inter-fiber transitions under metric and family substitution)**.** *Metric and family substitutions produce stable transitions between distinct execution classes: per-variant* $\sigma = 1.0$ *with cross-variant* $\rho = 0.0$*.*

Supported. Per-variant $\bar{\sigma} = 1.000$ (100% perfect collapse) for both metric and family substitutions. Cross-variant $\bar{\rho} = 0.000$ (complete inter-fiber separation) for both. These substitutions do not create noisy instability; they create clean movement to different execution identities.

Hypothesis 4 (Sparse adjacency)**.** *The execution adjacency graph is sparse: density well below 1, with multiple connected components.*

Supported. Density = 0.095 (36 of 378 possible edges). 10 connected components. The graph is sparse: most execution class pairs are never connected by a perturbation witness.

Hypothesis 5 (Non-uniform support correlated with fiber cardinality)**.** *Prompt support* $|S([a]_R)|$ *is non-uniform and correlates positively with* ϕ_S*.*

Strongly supported. Support Gini = 0.456. Support–fiber correlation $r = 0.925$. Execution classes with larger prompt support have proportionally larger fibers.

9 Relationship to Papers 1–4

This paper extends the BLISP research program as follows:

Paper 1: Grounding gate and admissibility [1]. Paper 1 introduced the grounding gate and admissibility framework for executable specifications. The 100% validation rate across 2,200 proposals reflects Paper 1's admissibility criteria: every LLM-generated specification passes structural validation before reaching the execution pipeline.

Paper 2: Canonical execution semantics, 8-layer hashing, and collapse ratio [2]. Paper 2 established the deterministic canonicalization and execution pipeline (parse $\rightarrow$ normalize $\rightarrow$ canonicalize $\rightarrow$ plan $\rightarrow$ optimize $\rightarrow$ execute) on which the projection π depends, introduced the 8-layer provenance record that gives execution identity its content, defined the collapse ratio $\gamma = \delta_S/\delta_C$, and showed that surface variation grows faster than execution variation with temperature. This paper uses Paper 2's 1,200-generation corpus as the E1 baseline and extends the analysis from aggregate ratios to per-class fiber structure, perturbation profiles, and adjacency.

Paper 3: Execution categories, quotient identity, and fibers [3]. Paper 3 defined execution categories $\mathcal{B}_R/\!\sim_R$, the quotient projection $\pi : \mathcal{E}_R \rightarrow \mathcal{B}_R/\!\sim_R$, and the fiber construction $F([a]_R) = \pi^{-1}([a]_R)$. This paper takes those definitions from theory to measurement: the controlled perturbation experiments quantify fiber cardinality, collapse stability, transition mass, and adjacency over 2,200 proposals. The divergence layer analysis in Section 6 uses the provenance layers to classify adjacency edges: 60 witnesses diverge at the SCR (scoring metric) layer, 21 at the FAM (family) layer.

Paper 4: Provenance and replay semantics, divergence localization [4]. Paper 4 introduced provenance and replay semantics, establishing that execution identity carries a reproducible audit trail, and provided divergence localization: the ability to identify the exact provenance layer at which two executions diverge. The replay-stable identity that makes fiber membership well-defined—the same specification produces the same FPR_HSH across independent evaluations—depends on the deterministic replay properties established in Paper 4.

The present paper adds the empirical fiber perspective: instead of asking whether individual proposals are admissible (Paper 1) or how many distinct executions they produce (Paper 2), it asks what *structure* the set of all proposals has when viewed through the fiber construction of Paper 3 and the provenance semantics of Paper 4.

10 Limitations and Non-Claims

Sample dependence. All cardinalities, support sizes, and distributions reported here are sample-dependent. With different prompts, different perturbation families, or more repetitions, the specific numbers would change. The structural properties—non-uniformity, separation between synonym and metric/family perturbations, sparsity of adjacency—are more robust, but we report them as observed properties of this sample, not universal laws.

Model dependence. All experiments use a single model (`claude-sonnet-4-20250514`) at a single temperature (0.7, except E1 which includes 0.0, 0.3, and 1.0). Different models or temperatures may produce different fiber structures. The *existence* of fiber structure is a consequence of the execution system, not the model; but the specific cardinalities and transition masses depend on the model's behavior.

Domain specificity. The execution system is BLISP, a quantitative finance specification language. The 4 families and 6 metrics observed span a specific domain. We do not claim these results generalize to arbitrary code generation or to execution systems with different canonical forms.

Graph sparsity is protocol-dependent. The observed sparsity of the adjacency graph depends on the chosen perturbation protocol and sampling regime, and should not be interpreted as an intrinsic invariant of the proposal space independent of experimental design. A different perturbation protocol—for example, one that simultaneously varies family and metric—would produce different edges and potentially different density.

No claims about generator internals. We do not claim that the fiber structure reflects the LLM's internal representations, latent geometry, attention patterns, or cognitive processes. The structure we observe is *induced by the execution system on observable proposal behavior*. The generator is a black box that produces text; the execution system is the structure-imposing component. Any generator—LLM or otherwise—that produces valid specifications for this execution system would exhibit fiber structure. The specific structure depends on the generator's behavior, but the existence of fibers is a property of the deterministic projection.

No topological or geometric claims. The objects in this paper are finite sets with discrete structure. The adjacency graph is a finite graph, not a manifold. Fiber cardinality is a count, not a measure. We use no continuous topology, no latent spaces, and no geometric language.

No causal claims. The perturbation experiments establish correlations between prompt changes and execution class transitions. We do not claim that a synonym change *causes* intra-fiber stability in any causal-inference sense. We observe that synonym perturbations *produce* intra-fiber transitions at the measured rates.

11 Conclusion

When a stochastic generator proposes executable specifications, the deterministic execution system induces a partition of all proposal variation into surface-form variation (intra-fiber, provenance-invisible, absorbed by canonicalization) and execution ambiguity (inter-fiber, provenance-visible, execution-identity-changing).

The controlled perturbation experiments in this paper demonstrate that this partition has empirically measurable structure:

1. **Fiber cardinality is non-uniform** (Gini = 0.467): some execution classes attract many more proposals than others.
2. **Synonym perturbations are absorbed** ($\bar{\rho} = 0.985$): surface rewording stays within fibers.
3. **Metric and family substitutions create clean transitions** ($\bar{\rho} = 0.000$, $\bar{\sigma} = 1.000$): changing a metric or family moves proposals to a different execution class reliably, not noisily.
4. **The adjacency graph is sparse** (density = 0.095): most execution class pairs are never connected.
5. **Prompt support predicts fiber size** ($r = 0.925$): widely reachable classes are also large classes.

The key finding is point 3. It would be natural to assume that any perturbation to a prompt introduces instability—that changing "Sharpe" to "max drawdown" would produce a mixture of execution classes rather than a clean transition. Instead, we observe that metric and family substitutions produce *perfect* per-variant stability with *complete* inter-fiber separation. The generator reliably commits to the execution identity corresponding to the specified field. Variation within a prompt variant is purely surface-level; variation across variants is purely structural.

This separation is not a property of the generator alone. It is a property of the coupled system: a stochastic generator paired with a deterministic execution layer that defines canonical identity. The deterministic execution system operationally distinguishes between variation that changes execution identity and variation that does not.

Reproducibility. All experimental code, raw data (2,200 JSONL records), analysis scripts, and output files are available in the BLISP repository under `experiments/paper5_fibers.py` and `experiments/results/paper5_fibers/`.

References

[1] T. Dionysopoulos. The grounding gate: Admissibility and replay guarantees for AI-driven research. *Zenodo*, 2026. doi:10.5281/zenodo.20456984.

[2] T. Dionysopoulos. Canonical execution semantics for stochastic program generators. *Zenodo*, 2026. doi:10.5281/zenodo.20457255.

[3] T. Dionysopoulos. Execution categories for stochastic program generators. *Zenodo*, 2026. doi:10.5281/zenodo.20457403.

[4] T. Dionysopoulos. Provenance algebra for deterministic AI execution. *Zenodo*, 2026. doi:10.5281/zenodo.20457667.

[5] S. Yao et al. ReAct: Synergizing reasoning and acting in language models. *ICLR*, 2023.

[6] T. Schick et al. Toolformer: Language models can teach themselves to use tools. *NeurIPS*, 2023.

[7] S. G. Patil et al. Gorilla: Large language model connected with massive APIs. *arXiv:2305.15334*, 2023.

[8] B. T. Willard and R. Louf. Efficient guided generation for large language models. *arXiv:2307.09702*, 2023.

[9] LangChain, Inc. LangGraph: Build resilient language agents as graphs. `https://github.com/langchain-ai/langgraph`, 2024.

[10] OpenAI. OpenAI Agents SDK. `https://github.com/openai/openai-agents-python`, 2025.

[11] Anthropic. Model Context Protocol (MCP). `https://modelcontextprotocol.io`, 2024.

[12] O. Khattab et al. DSPy: Compiling declarative language model calls into self-improving pipelines. *arXiv:2310.03714*, 2023.

[13] M. Zaharia et al. Accelerating the machine learning lifecycle with MLflow. *IEEE Data Eng. Bull.*, 41(4):39–45, 2018.

[14] L. Biewald. Experiment tracking with Weights & Biases. `https://www.wandb.com`, 2020.

[15] E. Dolstra, M. de Jonge, and E. Visser. Nix: A safe and policy-free system for software deployment. *LISA*, 2004.

[16] W3C. PROV-DM: The PROV data model. W3C Recommendation, 2013.

[17] P. Buneman, S. Khanna, and W.-C. Tan. Why and where: A characterization of data provenance. *ICDT*, 2001.