Cross-Family Convergence of Execution-Identity Primitives Under Task Pressure

Thomas Dionysopoulos, CFA

Abstract

When frontier language models must answer questions about computational history, what structures do they build? We administered identical blind prompts requiring analysis of 200 finance execution traces to six models from three independent families: Anthropic (Opus, Sonnet, Haiku), OpenAI (GPT-4.1, GPT-4.1-mini), and Google (Gemini 2.5 Flash). Without access to any predefined framework, models from all families reconstructed structurally equivalent execution-identity primitives: normalization, canonical identity, equivalence classes, grouping, composite rewrite rules, replay mappings, lineage graphs, and validation infrastructure.

Across 55 runs from Anthropic and OpenAI (5 models), seven of eight primitives achieved Primitive-Freq ≥ 0.90 after parser audit. The identity tier (A–D) converged at 1.00 for both families. The dominant source of measured instability was evaluator brittleness—57% of raw failures were parser artifacts, not model divergence. A capability-dependent fidelity gradient emerged: flagship models reconstruct the canonical computation DAG with near-perfect node counts (8.4 vs ground truth 8), while efficient models over-segment (11–14 nodes).

The empirical gap observed in this study is economic rather than cognitive. Independent frontier models can reconstruct execution-identity infrastructure. The cost is that they reconstruct it every time. Materializing this convergent structure as persistent infrastructure eliminates per-query reconstruction cost. The observation that replay (F) exhibits the weakest convergence—requiring synthesis rather than recognition—motivates investigation of execution categories as the semantic substrate for regeneration.

1 Introduction

When agents answer questions about computational history, what structures do they build? This paper reports an empirical answer.

Agents produce stochastic execution traces. A single computation—say, computing a log return—may appear as `s.pct_change().apply( np.log1p)` in one trace and as `np.log(s) - np.log(s.shift(1))` in another. Surface variation is manageable for structural questions ("how many distinct operations are there?"). It becomes a barrier for semantic questions ("how many distinct *computations* are there?"), which require recognizing that syntactically different sequences compute the same function. It becomes a harder barrier still for operational questions ("replay this computation in SQL"), which require decomposition and resynthesis.

This paper studies what happens as such questions become progressively harder. We present nine question tiers, each requiring additional infrastructure to answer correctly. We administer these questions to frontier language models from three independent families—Anthropic, OpenAI, and Google—and measure what infrastructure they construct.

The central observation is convergence. The most important result is not that the primitives exist—individual primitives such as normalization, rewrite rules, and lineage graphs are well-known concepts. The most important result is that independent frontier model families repeatedly reconstructed structurally equivalent execution-identity infrastructure despite differing architectures, training corpora, and output styles. This infrastructure decomposes into eight primitives organized in two tiers: an identity tier (normalization, canonical identity, equivalence classes, grouping) and an algebra tier (composite rewrite rules, replay mappings, lineage graphs, validation rules).

We call the full eight-primitive structure the *execution identity algebra*. This paper does not claim that this algebra is universal, optimal, or the only possible decomposition. This paper claims that independent frontier model families converge on structurally equivalent instances of it under computational-history task pressure, and that the convergence can be measured.

1.1 Contributions

Contribution 1: Cross-family convergence. Six models from three independent families reconstruct structurally equivalent execution-identity primitives under identical blind prompts (section 6).

Contribution 2: Measurement theory. The dominant source of measured instability is evaluator brittleness, not model divergence. Raw vs. audited PrimitiveFreq separates measurement behavior from model behavior (section 7).

Contribution 3: Presence vs. fidelity. All families reconstruct the same primitive classes, but fidelity varies with model capability. This distinction between primitive existence and primitive quality is a new finding (section 9).

Contribution 4: Economic argument. The empirical gap is economic rather than cognitive. Independent models repeatedly reconstruct the same structures at substantial cost. Materializing this convergent structure eliminates per-query reconstruction (section 11).

1.2 Scope and Non-Claims

This paper does not claim that models "rediscovered" any particular system. The structures reconstructed by frontier models correspond to components of the execution-identity stack materialized by Blisp. Blisp's contribution is not defining these structures but materializing them as persistent infrastructure. The structures appeared independently in model outputs. No model had access to any predefined framework.

This paper does not claim that the eight-primitive decomposition is canonical. Alternative decompositions may capture the same structure.

This paper does not claim universal convergence. Google's results are infrastructure-limited (API timeouts, output truncation), and the experiment measured Gemini's API reliability rather than its cognitive capability. The cross-family claim rests on Anthropic and OpenAI; Google provides supporting but incomplete evidence.

2 Problem: Stochastic Traces, Deterministic Queries

2.1 Agent Execution Traces

An agent execution trace is a sequence of operations recorded during task execution. Traces vary along several dimensions:

- **Surface syntax:** the same operation may be expressed differently across frameworks (`s.diff(1)` in pandas, `col.diff(1)` in polars, `x - LAG(x, 1) OVER (ORDER BY ts)` in SQL).
- **Decomposition level:** a computation may appear as a single atomic operation or as a sequence of primitive operations (`LOG_RETURN` vs `LOG + SHIFT(1) + SUB`).
- **Framework:** the same computation may be implemented in different tools (pandas, polars, DuckDB SQL).

This variation is not noise—it reflects genuine implementation diversity. But it creates a gap between what was recorded (stochastic surface traces) and what queries demand (deterministic computational identity).

2.2 History-Analysis Questions

Operational work on computational traces generates questions at increasing levels of semantic depth:

Structural questions (Q1–Q5). How many distinct operation sequences exist? What is the most common? Which share prefixes? These require identity at the operation-sequence level.

Equivalence questions (Q6). How many distinct *computations* exist, treating equivalent decompositions as identical? This requires recognizing that `LOG + SHIFT(1) + SUB` and `LOG_RETURN` compute the same function.

Replay questions (Q7). Given a computation, produce an equivalent implementation in a different framework.

Lineage questions (Q8). Which computations share sub-computations? Construct the directed acyclic graph of computation reuse.

Validation questions (Q9). Which computations violate a given policy?

These questions correspond to standard operational needs: deduplication (Q6), migration (Q7), dependency analysis (Q8), and compliance (Q9).

3 Execution Identity Algebra

3.1 Two-Tier Structure

We decompose the infrastructure required to answer Q1–Q9 into eight primitives organized in two tiers.

Tier 1: Identity (A–D). Required by Q1–Q5.

Prim.	Operation	Input	Output
A	Normalization	Raw operation string	Canonical operation
B	Canonical identity	Canonical op sequence	Hashable key
C	Equivalence classes	Set of keys	Partitions
D	Grouping	Equivalence classes	Aggregated results

Table 1: Tier 1: Identity primitives.

Tier 2: Algebra (E–H). Required by Q6–Q9.

Prim.	Operation	Input	Output
E	Composite rewrites	Op sequences + rules	Computation identities
F	Replay	Computation + target	Target implementation
G	Lineage	Set of computations	Directed acyclic graph
H	Validation	Computation + policies	Violation set

Table 2: Tier 2: Algebra primitives.

3.2 Tier Dependency

The algebra tier depends on the identity tier. Composite rewrite rules (E) require canonical identity (B) to recognize that two sequences compute the same function. Replay (F) requires equivalence classes (C) to ensure consistent target implementations. Lineage (G) requires grouping (D) to aggregate sub-computation relationships. Validation (H) requires all of A–D to check policies at the computation level rather than the surface level.

This dependency is structural, not merely empirical. If a model produces algebra-tier outputs (E–H), it must have internally performed identity-tier operations (A–D), whether or not the evaluator observes them. This observation becomes important in section 7.

3.3 Formal Characterization

The identity tier operates on the free monoid $M(G)$ of operation sequences, where G is the set of generator operations. Identity primitives partition $M(G)$ by syntactic equality after normalization.

The algebra tier operates on the quotient $A(G, R) = M(G)/{\sim_R}$, where R is a set of rewrite relations. Each relation $r \in R$ states that two operation sequences compute the same function:

$$[\texttt{LOG}, \texttt{SHIFT}(1), \texttt{SUB}] \sim_R [\texttt{LOG\_RETURN}]$$
$$[\texttt{SHIFT}(k), \texttt{SUB}] \sim_R [\texttt{DIFF}(k)]$$

The quotient $A(G, R)$ collapses syntactically different sequences that are computationally equivalent.

4 Experimental Setup

4.1 Phase 1: Single-Family Pilot

Phase 1 established the emergence ladder using three Anthropic models (Opus, Sonnet, Haiku), 11 independent runs per model, across three domains (finance, SQL, build/CI). This phase is reported in section 5.

4.2 Phase 2: Cross-Family Replication

Phase 2 tests whether the Phase 1 findings replicate across independent model families. The benchmark is frozen: identical blind prompts, identical corpora, identical detection rules, identical ground truth.

Models. Six models from three families:

Family	Model	Model ID
Anthropic	Opus	claude-opus-4-20250514
	Sonnet	claude-sonnet-4-20250514
	Haiku	claude-haiku-4-20250506
OpenAI	GPT-4.1	gpt-4.1
	GPT-4.1-mini	gpt-4.1-mini
Google	Gemini 2.5 Flash	gemini-2.5-flash

Table 3: Models under test. Three families, six models, 11 runs each.

Protocol. Each model receives five prompts (Q1–Q5 combined, Q6, Q7, Q8, Q9) with the relevant corpus appended. Prompts describe operational tasks without revealing the A–H framework. The model produces structured JSON output. Detection rules map output artifacts to primitives.

Runs. 11 independent runs per model, 66 total. Each run is a fresh session with no memory of prior runs.

Corpora. Finance domain only for Phase 2, to isolate cross-family variation from cross-domain variation. Finance Q6: 200 traces, 8 computations, 5 decomposition pairs. Q7: 12 traces. Q8: 50 traces, 8 nodes, 6 edges. Q9: 30 traces, 16 violating, 14 clean, 3 tricky.

Detection rules. Frozen from Phase 1. Primitive A is detected if Q1 returns a `distinct_sequences` count. Primitive C is detected if Q3 returns non-empty `prefix_groups`. Primitive E is detected if the Q6 computation count is less than the count without rewrite recognition. Primitive G is detected if Q8 produces ≥ 6 nodes with dependency edges matching ≥ 6 ground-truth nodes. Primitive H is detected if Q9 violation-detection $F_1 \geq 0.80$ with ≥ 1 tricky-clean case correct.

4.3 Measures

PrimitiveFreq. For each primitive P and model M: $\text{PrimitiveFreq}(P, M) = |\{r : P \text{ detected in run } r\}|/N$.

LadderMatch. Fraction of runs where all eight primitives are detected.

Q8 Fidelity. Node count, edge count, root count, and ground-truth node matching.

Q9 Detail. Precision, recall, F_1, tricky-clean count.

Cost. Input tokens, output tokens, wall-clock time per run.

5 Phase 1: Emergence Ladder

Figure 1 presents the emergence ladder observed in Phase 1.

The emergence is monotonic: each question tier adds exactly one primitive. The identity tier is fully present before any algebra-tier question is administered. The emergence is prompted—no primitive appears without task pressure that specifically requires it.

Phase 1 established the ladder pattern within a single model family. The question left open was whether this pattern is family-specific or reflects task-induced convergence. Phase 2 answers this question.

Task Pressure	A	B	C	D	E	F	G	H	Id.	Alg.
Q1–Q5 (structural)	Y	Y	Y	Y	–	–	–	–	4/4	0/4
+ Q6 (equivalence)	Y	Y	Y	Y	Y	–	–	–	4/4	1/4
+ Q7 (replay)	Y	Y	Y	Y	Y	Y	–	–	4/4	2/4
+ Q8 (lineage)	Y	Y	Y	Y	Y	Y	Y	–	4/4	3/4
+ Q9 (validation)	Y	Y	Y	Y	Y	Y	Y	Y	4/4	4/4

Figure 1: Emergence ladder under increasing history-analysis pressure. Identity primitives (A–D) emerge before algebra primitives (E–H); algebra primitives emerge one per tier in a stable order.

6 Phase 2: Cross-Family Replication

6.1 Raw Results

Table 4 presents the raw results as observed by the original scorer. No modifications have been applied.

Model	Fam.	A	B	C	D	E	F	G	H	Ladder
Opus	Anth	1.00	1.00	1.00	1.00	.82	1.00	1.00	1.00	.82
Sonnet	Anth	.55	.55	.55	.55	1.00	.73	1.00	1.00	.45
Haiku	Anth	1.00	1.00	1.00	1.00	1.00	.64	.91	1.00	.64
GPT-4.1	OAI	.64	.64	.64	.64	1.00	.73	.91	1.00	.36
GPT-4.1-mini	OAI	1.00	1.00	1.00	1.00	1.00	.55	.91	1.00	.55
Gemini Flash	Goo	.00	.00	.00	.00	.64	1.00	.82	.45	.00
Global		.70	.70	.70	.70	.91	.77	.92	.91	.47

Table 4: Raw PrimitiveFreq by model (66 runs). These are the historically observed results. Global LadderMatch = 0.47. Weak success criterion (PrimitiveFreq ≥ 0.50 for 6/8) passes; moderate and strong fail.

The raw results present an apparent paradox: Gemini Flash scores A–D = 0.00 while producing E = 0.64, G = 0.82, and F = 1.00. This is logically inconsistent. The algebra tier depends structurally on the identity tier (section 3.2). A model cannot construct rewrite rules, lineage graphs, or replay mappings without having internally performed normalization, identification, and grouping. The existence of algebra-tier outputs is indirect evidence that identity-tier operations were performed, whether or not the evaluator detected them.

This paradox motivates the audit.

6.2 Audit

We audited every run where A–D detection failed. The audit examines raw model output text, classifies the failure cause, and applies documented parser fixes. Three rules were changed; all affect only the JSON parser, not the primitive detection logic.

Rule 1: Integer key fix. Sonnet emits Python-style integer keys (`{5: 6}`) in Q5 responses. Standard JSON requires quoted keys (`{"5": 6}`). The original parser rejects the entire response. Fix: regex substitution of unquoted integer keys. Affects 5 runs (Sonnet).

Rule 2: Truncated JSON recovery. Gemini 2.5 Flash and GPT-4.1 emit valid JSON that exceeds output token limits. The response starts with a JSON code block but never closes. Fix: extract Q1–Q5 fields individually from partial text using targeted regex. Affects 11 runs (7 Gemini, 4 GPT-4.1).

Rule 3: No changes to E–H detection. No threshold changes. No synonym additions. No prompt modifications. The audit is parser-only.

Each audited failure is classified:

- **Type A:** Primitive genuinely absent (API timeout, no output).
- **Type B:** Primitive present but below fidelity threshold.
- **Type C:** Parser/scoring failure (model produced valid content).
- **Type D:** Ambiguous.

Of the 20 raw A–D failures: 16 were Type C (parser failure), 4 were Type A (Gemini API timeout producing no output), and 0 were Type D.

6.3 Audited Results

Model	**Fam.**	A	B	C	D	E	F	G	H	**Ladder**
Opus	Anth	1.00	1.00	1.00	1.00	.82	1.00	1.00	1.00	.82
Sonnet	Anth	1.00	1.00	1.00	1.00	1.00	.73	1.00	1.00	.73
Haiku	Anth	1.00	1.00	1.00	1.00	1.00	.64	.91	1.00	.64
GPT-4.1	OAI	1.00	1.00	1.00	1.00	1.00	.73	.91	1.00	.73
GPT-4.1-mini	OAI	1.00	1.00	1.00	1.00	1.00	.55	.91	1.00	.55
Gemini Flash	Goo	.64	.64	.64	.09	.64	1.00	.82	.45	.00
Global		.94	.94	.94	.85	.91	.77	.92	.91	.58

Table 5: Audited PrimitiveFreq by model (66 runs). Parser fixes affect only A–D. E–H columns are unchanged from table 4. Global A–C rises from .70 to .94; D rises to .85. LadderMatch rises from .47 to .58.

Table 6 shows the delta between raw and audited results.

Model	ΔA	ΔB	ΔC	ΔD	ΔE	ΔF	ΔG	ΔH	ΔLadder
Opus	.	.	.	.	.	.	.	.	.
Sonnet	+.45	+.45	+.45	+.45	.	.	.	.	+.27
Haiku	.	.	.	.	.	.	.	.	.
GPT-4.1	+.36	+.36	+.36	+.36	.	.	.	.	+.36
GPT-4.1-mini	.	.	.	.	.	.	.	.	.
Gemini Flash	+.64	+.64	+.64	+.09	.	.	.	.	.

Table 6: Delta: audited minus raw. 100% of the delta is in A–D. Zero changes to E–H. The audit is purely a parser fix.

The key observation: **100% of the improvement is in A–D, and 0% is in E–H.** The audit changed only the JSON parser, not any detection threshold or primitive definition. Every recovered A–D detection represents a model that produced valid identity-tier content which the original parser failed to extract.

7 Evaluator Brittleness

7.1 The Dominant Instability Source

Of the 35 runs where LadderMatch failed under the raw scorer:

Failure Type	**Count**	**%**	**Description**
Parser (A–D)	20	57%	JSON extraction failed on valid model output
F threshold	15	43%	Mapping table present but below detection threshold
E miss	6	17%	Model did not reduce computation count
H miss	6	17%	$F_1 < 0.80$ or tricky-clean count < 1
G miss	5	14%	Fewer than 6 matched nodes or no edges
Infrastructure	4	11%	API timeout, no model output

Table 7: Failure taxonomy. Percentages exceed 100% because some runs have multiple failure modes. Parser failures are the single largest category.

The dominant source of measured instability was the evaluator, not the models. Twenty of 35 failures (57%) were caused by the JSON parser failing to extract valid structured output from model responses. The models answered correctly; the scorer could not read the answer.

7.2 The Logical Consistency Argument

The tier dependency (section 3.2) provides a logical consistency check. If a model produces algebra-tier outputs (E, F, G, H) without identity-tier outputs (A–D), one of two conclusions follows:

1. The identity tier was performed internally but not detected (evaluator failure).
2. The algebra tier does not depend on the identity tier (the tier structure is wrong).

Gemini Flash scores A–D = 0.00, F = 1.00, G = 0.82. Under interpretation (2), the entire tier structure would be falsified. Under interpretation (1), the evaluator failed to observe identity-tier operations that must have occurred. The audit confirms interpretation (1): Gemini's raw responses contain `Q1`, `Q3`, and `Q4` fields with valid identity-tier content, but the JSON is truncated before the parser can extract it.

The existence of algebra-tier outputs in the absence of detected identity-tier outputs is therefore *evidence for* the tier dependency, not against it.

7.3 Three Levels of Observation

This study separates three levels that are often conflated:

1. **Model behavior:** what the model produced.
2. **Measurement behavior:** what the evaluator detected.
3. **Interpretation:** what the detection implies.

Raw results report level (2). Audited results approximate level (1). The distinction matters: A–D appears unstable in raw results but is the most stable tier in audited results (1.00 for Anthropic and OpenAI, 55 runs, zero failures).

8 Cross-Family Convergence

8.1 Anthropic and OpenAI

Across 55 runs from Anthropic (33) and OpenAI (22), audited PrimitiveFreq by family:

Family	A	B	C	D	E	F	G	H
Anthropic ($n = 33$)	1.00	1.00	1.00	1.00	.94	.79	.97	1.00
OpenAI ($n = 22$)	1.00	1.00	1.00	1.00	1.00	.64	.91	1.00
Min	1.00	1.00	1.00	1.00	.94	.64	.91	1.00

Table 8: Audited PrimitiveFreq for Anthropic and OpenAI. Seven of eight primitives achieve ≥ 0.91 as the cross-family minimum. F is the sole exception at 0.64–0.79.

Seven of eight primitives achieve cross-family minimum ≥ 0.91. The identity tier converges perfectly: both families achieve 1.00 for all four identity primitives. The algebra tier converges strongly for E (0.94), G (0.91), and H (1.00). Only F diverges (0.64–0.79).

8.2 Google (Gemini Flash)

Gemini Flash results are dominated by infrastructure artifacts:

- 4 of 11 runs produced no Q1–Q5 output (API timeout).
- 7 of 11 runs truncated Q1–Q5 mid-JSON (thinking tokens consumed output budget).
- When Gemini completes output: F = 1.00, G = 0.82, close to the Anthropic/OpenAI range.

Gemini cannot be fairly compared until infrastructure issues are resolved. We report Gemini's numbers for completeness but do not include Google in the cross-family convergence claim.

8.3 What Converged and What Did Not

Converged strongly (min ≥ 0.91): A, B, C, D, E, G, H. These primitives were reconstructed at near-universal rates by both Anthropic and OpenAI models.

Converged partially (min = 0.64): F (replay). This primitive was present in every family but at variable rates.

Not testable: Google convergence, due to infrastructure limitations.

9 Presence vs. Fidelity

The experiment measures primitive *detection*—a binary pass/fail. But the raw data also contain continuous fidelity information. Separating presence from fidelity reveals a capability gradient not visible in PrimitiveFreq alone.

	E		**F**		**G**				**H**	
Model	Pres	Count	Pres	Maps	Pres	Nodes	Edges	Match	Pres	F_1
Opus	.82	8.1	1.00	7.0	1.00	8.4	6.0	7.8	1.00	1.00
Sonnet	1.00	8.0	1.00	8.1	1.00	9.0	5.9	7.6	1.00	1.00
Haiku	1.00	8.0	1.00	8.4	1.00	12.5	14.7	6.5	1.00	0.95
GPT-4.1	1.00	8.8	1.00	9.5	1.00	11.5	10.9	6.8	1.00	1.00
GPT-4.1-mini	1.00	6.8	0.55	8.5	1.00	14.2	13.1	6.4	1.00	1.00
Gemini Flash	.64	8.1	1.00	7.0	0.82	9.9	7.0	6.4	0.45	0.98
Ground truth: E count = 8, G nodes = 8, G edges = 6										

Table 9: Primitive presence vs. fidelity. Presence = fraction of runs where the primitive was detected. Fidelity metrics are averaged over runs where the primitive was present. G ground truth: 8 nodes, 6 edges.

Three tiers of fidelity emerge:

Flagship models (Opus, Sonnet): near-perfect fidelity. G nodes = 8.4–9.0 (ground truth 8), edges = 5.9–6.0 (ground truth 6), GT-matched nodes = 7.6–7.8. The canonical DAG is reconstructed with minimal error.

Efficient models (GPT-4.1, GPT-4.1-mini, Haiku): correct presence but degraded fidelity. G nodes = 11.5–14.2 (over-segmented), edges = 10.9–14.7 (over-connected). The structure is reconstructed—the right computations are identified, the right dependencies exist—but the granularity is wrong. Models split single computations into sub-steps that a human analyst would combine.

Infrastructure-limited (Gemini Flash): presence when output completes, but API constraints degrade measurement.

The finding: **all families reconstruct the same primitive classes, but fidelity varies with model capability.** Primitive existence is convergent; primitive quality is capability-dependent. This distinction was not visible in PrimitiveFreq, which collapses fidelity into a binary threshold.

10 Replay (F) Requires Synthesis

F is the weakest-converging primitive: PrimitiveFreq ranges from 0.55 (GPT-4.1-mini) to 1.00 (Opus, Gemini Flash). The failure mode is instructive and reveals a fundamental distinction.

10.1 Recognition vs. Regeneration

Seven of the eight primitives are *recognition-oriented*: they require identifying structure that already exists in the corpus.

- A–D (identity): "What are the distinct operations?"
- E (rewrites): "Which sequences compute the same function?"
- G (lineage): "Which computations share sub-computations?"

- H (validation): "Which computations violate policy?"

One primitive is *regeneration-oriented*: it requires synthesizing a new artifact that did not exist in the corpus.

- F (replay): "Produce an equivalent implementation in a different framework."

Identity (recognition):

$$\texttt{Implementation} \xrightarrow{\text{A–D}} \texttt{Canonical Identity}$$

Replay (regeneration):

$$\texttt{Implementation} \xrightarrow{\text{A–D}} \texttt{Canonical Identity} \xrightarrow{\text{F}} \texttt{Target Implementation}$$

Figure 2: Identity requires recognition. Replay requires regeneration. The second arrow—from canonical identity to target implementation—is where convergence weakens.

Equivalence classes appear sufficient for recognition. Replay appears to require richer execution structure than equivalence recognition alone.

10.2 Why Replay Diverges

Primitive F (replay) requires *synthesizing* a new implementation. Given a computation and a target framework, the model must produce code that computes the same function in the new framework. This is a generation task. Moreover, the detection threshold requires consistency: equivalent input traces must produce the same target implementation. A model that generates plausible but inconsistent implementations fails F even though it "understood" the computation.

The data support this distinction:

- F presence (fraction with any mapping table) is high across all models: 0.55–1.00. Models attempt replay.
- F detection (consistent mappings ≥ 3) is lower: the consistency threshold filters out partial or inconsistent attempts.
- F is the only primitive where presence substantially exceeds detection.

The strongest divergence in this study appears in replay. This is not a weakness of the experiment—it is a result. Replay reveals the boundary between what equivalence classes can support (recognition) and what they cannot (regeneration).

Recognizing that two traces compute the same function (E) requires the identity tier. Regenerating that function in a new framework (F) additionally requires understanding the computation's internal structure—its typed decomposition, its parameter roles, its framework-specific idioms. Equivalence classes answer "what computation is this?" but not "how can this computation be reproduced elsewhere?"

This observation motivates the investigation in section 15.1: whether execution categories—typed structures encoding the operations admissible on a computation—provide the semantic substrate required for reliable replay. Execution categories are a candidate explanation for the recognition–regeneration gap, not a proven one.

11 The Economic Argument

11.1 What Was Consumed

Sixty-six independent runs consumed 3.5 million tokens and 3.1 hours to repeatedly reconstruct the same eight primitives. Each run produced equivalent infrastructure: normalization rules, canonical identities, equivalence partitions, rewrite mappings, replay tables, lineage graphs, and validation results. None of this infrastructure was reusable across runs.

Family	Runs	Tokens	Wall Time
Anthropic	33	1,899,440	1.3h
OpenAI	22	1,126,563	0.9h
Google	11	506,249	0.9h
Total	**66**	**3,532,252**	**3.1h**

Table 10: Total experiment cost. 66 runs, 3.5M tokens, 3.1 hours.

11.2 Why Reconstruct Every Time?

The experiment demonstrates that frontier models can reconstruct execution-identity infrastructure. The cost is that they reconstruct it repeatedly. Sixty-six independent runs from three families produced equivalent structures sixty-six times, consuming 3.5 million tokens.

Convergence strengthens the economic argument: the reconstructed structures are not model-specific. Because independent families converge on equivalent primitives, the materialized version captures a structure that any frontier model would reconstruct. This makes materialization a general investment, not a bet on a single model's behavior.

The resulting economic comparison:

	Reconstruction	Materialized
Engineering cost	0	One-time
Per-query cost	∼53K tokens	0 tokens
Per-domain cost	Full rebuild	Portable
Infrastructure	Disposable	Persistent
Composition	Independent	Engineered to compose

Table 11: Reconstruction vs. materialized infrastructure. The break-even point is a single repeat query.

The empirical gap is economic rather than cognitive.

11.3 What the Cost Comparison Does Not Show

For a one-time question on a novel domain, reconstruction may be the correct strategy: it requires no engineering investment. The amortization advantage of materialized infrastructure appears only under repeated or cross-domain use.

12 Analysis

12.1 From Convergence to Invariance

Anthropic, OpenAI, and Google models—trained on different data, with different architectures, producing different output styles—reconstruct similar structures when given identical computational-history tasks. What remains invariant across model families?

The candidate answer is: execution-identity primitives. The convergence observed across Anthropic and OpenAI suggests that execution-identity primitives are candidates for a model-family invariant—structures that remain stable under substitution of model family, at least within the frontier capability tier. This is an empirical invariance hypothesis, not a proof. The sequence from observation to deeper structure is: convergence (this paper) → invariance hypothesis (this paper) → execution categories as a candidate explanation (Paper 10).

12.2 Convergent Architecture

Independent frontier model families repeatedly reconstructed structurally equivalent execution-identity infrastructure despite differing architectures, training corpora, and output styles. Models did not reconstruct identical structures. They reconstructed structurally equivalent *classes* of infrastructure. Opus produces tighter DAGs than GPT-4.1-mini. Sonnet uses different JSON formatting than Gemini. The content and formatting differ. The infrastructure type converges. This distinction—convergence in kind, not in content—is the core observation.

12.3 Why Is Replay Harder?

Models consistently reconstruct identity (A–D = 1.00), equivalence recognition (E = 0.96), lineage (G = 0.95), and validation (H = 1.00) across Anthropic and OpenAI. Replay (F = 0.73) is weaker. Four possible explanations:

1. **Generation vs. classification.** Recognition asks "is X equivalent to Y?" (binary). Replay asks "produce Z such that Z is equivalent to X in framework T" (generative, open-ended).
2. **Semantic preservation.** Replay must preserve computational semantics across framework boundaries, not merely identify them.
3. **Framework-specific realization.** Each target framework has idioms, naming conventions, and composition patterns. Replay requires framework-specific knowledge that recognition does not.
4. **Typed execution structure.** Replay may require information about parameter roles, decomposition stages, and admissible operations that equivalence classes do not encode.

Explanation (4) is the hypothesis tested in Paper 10. If execution categories—typed structures encoding what operations are admissible on a computation—improve replay consistency, that would provide evidence that the recognition–regeneration gap is structural, not merely a difficulty gradient.

12.4 Attractor Interpretation

The cross-family convergence observed in this study is consistent with execution-identity structures functioning as a computational attractor for history-analysis tasks. Under increasing task pressure, independent models from independent families are drawn toward the same structural solution.

The attractor interpretation remains a hypothesis. The convergence is the observation. Testing the attractor hypothesis requires evidence that no comparably simple alternative structure exists for these tasks—an open question.

12.5 What Would Falsify This Claim?

The convergence claim would be weakened or falsified if:

1. Other frontier models (Llama, Mistral) fail to reconstruct the same primitives under the same task pressure.
2. Independent benchmark designers, without knowledge of A–H, produce substantially different primitive decompositions.
3. Real-world production corpora require fundamentally different primitives.
4. Agents with persistent memory retain and reuse the algebra without reconstruction, eliminating the economic gap.

13 Threats to Validity

13.1 BLISP-Shaped Prompts

Attack. Q6–Q9 were designed to force the A–H hierarchy.

Defense. The questions correspond to standard operational needs (deduplication, migration, dependency analysis, compliance). The mapping from question to primitive is the finding, not the assumption.

13.2 Single-Domain Phase 2

Attack. Phase 2 tested only the finance domain.

Defense. Phase 1 tested three domains (finance, SQL, build/CI) and observed convergence across all three within a single family. Phase 2 isolates cross-family variation by fixing the domain. A future Phase 3 combining cross-family and cross-domain variation would strengthen the claim.

13.3 Evaluator-Driven Results

Attack. The audit shows that 57% of raw failures were evaluator artifacts. The results are evaluator-dependent.

Defense. This is correct, and it is reported transparently. Both raw and audited results are presented. The audit rules are documented and independently reproducible. The finding that evaluator brittleness dominates measured instability is itself a contribution.

13.4 Google Infrastructure Limitations

Attack. Gemini results are unreliable due to API timeouts and output truncation.

Defense. We agree. Google is reported for completeness but excluded from the convergence claim. The cross-family claim rests on Anthropic and OpenAI.

13.5 F Threshold Sensitivity

Attack. F detection requires ≥ 3 consistent mappings. This threshold was chosen post-hoc.

Defense. The threshold reflects a meaningful property: consistent replay requires that equivalent inputs produce the same output. F presence (any mapping table) is high across all models; F detection is the consistency filter. We report both presence and detection (table 9) so readers can apply alternative thresholds.

13.6 Threat Summary

Threat	Severity	Addressed?
BLISP-shaped prompts	Medium	Partially (standard questions)
Single-domain Phase 2	Medium	Partially (Phase 1 cross-domain)
Evaluator-driven results	High	Yes (raw + audited reported)
Google infrastructure	Medium	Yes (excluded from claim)
F threshold sensitivity	Medium	Yes (presence + detection)

Table 12: Threat summary.

14 Related Work

Provenance and lineage systems. PROV-DM [9] and Why/Where/How provenance [10] materialize lineage as persistent infrastructure. Primitive G corresponds to coarse-grained workflow provenance. Our contribution is the observation that frontier models reconstruct lineage-like structures under task pressure without access to these systems.

Query optimization and rewrite rules. The Volcano/Cascades framework [5] and System R [4] use equivalence-based rewrite rules. Primitive E is structurally analogous. The difference is that models construct these rules ad hoc, while optimizers receive them from system developers.

Intermediate representations. MLIR [1], TVM [2], and SSA-form IRs [3] provide canonical representations that enable optimization. The present paper asks not whether such representations are useful, but whether models converge toward them under task pressure.

Agent memory and tool use. ReAct [6], Toolformer [7], and Generative Agents [8] study how agents use tools and accumulate knowledge. Our observation suggests a specific class of infrastructure worth persisting: algebraic structures over computational history.

Cross-model evaluation. Benchmarks such as MMLU, HumanEval, and MATH measure model capability on fixed tasks. Our benchmark differs in measuring what *infrastructure* models construct, not what answers they produce. The audit methodology—separating evaluator behavior from model behavior—may generalize to other evaluation settings where structured output parsing affects scores.

15 Research Program

Papers 1–5 established execution observability: the ability to inspect, query, and analyze execution traces from a time-series IR.

Papers 6–8 showed that semantic coordinates predict computational behavior. Paper 7: a single 7-valued coordinate predicts four optimizer behaviors at 99.6% accuracy. Paper 8: this predictive power transfers across independently developed systems.

Paper 9 (this paper) establishes that independent frontier model families reconstruct execution-identity primitives under task pressure. The structures are not specific to any single model family. Models independently reconstructed structures that correspond to components of the execution-identity stack established in Papers 1–8. The structures appeared independently. This study did not define them; it observed their convergent reconstruction.

15.1 Paper 10: Execution Categories

Paper 9 discovered that recognition-oriented primitives converge strongly across model families, while the regeneration-oriented primitive (replay) converges less strongly. Paper 10 asks: why?

The key distinction:

- **Equivalence class:** "What computation is this?" Sufficient for recognition.
- **Execution category:** "What operations are admissible on this computation?" A candidate for what regeneration requires.

Paper 10 will investigate whether execution categories—typed structures encoding parameter roles, decomposition stages, and the operations admissible at each stage—explain the gap between recognition (strong convergence) and regeneration (partial convergence). If execution categories improve replay consistency, that would provide evidence that the gap is structural rather than incidental. This is a hypothesis, not a claim.

15.2 Papers 11–12

Paper 11 investigates whether transformations between execution structures behave like morphisms. Paper 12 investigates whether the full structure is best understood as an intermediate representation for agent-generated computation.

The sequence is explicit:

- **Paper 9:** Convergence. Independent families reconstruct equivalent primitives.
- **Paper 10:** Execution categories. Why is regeneration harder than recognition?
- **Paper 11:** Morphisms. Do transformations between execution structures preserve type?
- **Paper 12:** Execution IR. Is the full structure an intermediate representation for agent-generated computation?

16 Conclusion

We administered identical blind prompts to six frontier language models from three independent families, requiring analysis of 200 finance execution traces across nine question tiers of increasing difficulty.

Independent frontier model families reconstruct execution-identity primitives under computational-history task pressure. The reconstruction is robust: seven of eight primitives converge above 0.90 PrimitiveFreq

across Anthropic and OpenAI (audited, 55 runs). The identity tier (A–D) converges at 1.00 for both families. The fidelity varies: flagship models reconstruct the canonical computation DAG with near-perfect accuracy, while efficient models over-segment.

The dominant source of measured instability is evaluator brittleness, not model divergence. Fifty-seven percent of raw failures were parser artifacts. The distinction between model behavior, measurement behavior, and interpretation is a methodological contribution of this study.

The reconstruction is expensive. Sixty-six runs consumed 3.5 million tokens to repeatedly reconstruct equivalent structures. The empirical gap is economic rather than cognitive. Materializing this convergent structure as persistent infrastructure eliminates per-query reconstruction cost.

Recognition-oriented primitives (identity, equivalence, lineage, validation) exhibit stronger convergence than the regeneration-oriented primitive (replay), suggesting that reliable regeneration may require richer execution structure than equivalence classes alone. This motivates the investigation of execution categories in Paper 10.

Independent frontier model families repeatedly reconstructed structurally equivalent execution-identity infrastructure despite differing architectures, training corpora, and output styles. The cross-family convergence observed in this study is consistent with execution-identity structures functioning as a computational attractor for history-analysis tasks. The attractor interpretation remains a hypothesis. The convergence is the observation.

References

[1] C. Lattner, M. Amini, U. Bondhugula, A. Cohen, A. Davis, J. Pienaar, R. Riddle, T. Shpeisman, N. Vasilache, and O. Zinenko. MLIR: Scaling Compiler Infrastructure for Domain Specific Computation. In *CGO*, 2021.

[2] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, M. Cowan, H. Shen, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. In *OSDI*, 2018.

[3] R. Cytron, J. Ferrante, B. K. Rosen, M. N. Wegman, and F. K. Zadeck. Efficiently Computing Static Single Assignment Form and the Control Dependence Graph. *TOPLAS*, 13(4):451–490, 1991.

[4] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access Path Selection in a Relational Database Management System. In *SIGMOD*, 1979.

[5] G. Graefe. The Cascades Framework for Query Optimization. *IEEE Data Engineering Bulletin*, 18(3):19–29, 1995.

[6] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *ICLR*, 2023.

[7] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. In *NeurIPS*, 2023.

[8] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. In *UIST*, 2023.

[9] W3C. PROV-DM: The PROV Data Model. W3C Recommendation, 2013.

[10] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance Semirings. In *PODS*, 2007.

[11] T. Henriksen, N. G. W. Serup, M. Elsman, F. Henglein, and C. E. Oancea. Futhark: Purely Functional GPU-programming with Nested Parallelism and In-place Array Updates. In *PLDI*, 2017.

A Scoring Definitions

PrimitiveFreq(P, M) = fraction of runs where primitive P is detected for model M.

IdentityScore $= |\{A, B, C, D\} \cap \text{Present}|/4$

AlgebraScore $= |\{E, F, G, H\} \cap \text{Present}|/4$

LadderMatch = 1 if all eight primitives detected, 0 otherwise.

A primitive is "detected" if the model's structured output contains the artifact specified by the detection rule. Detection rules are frozen from Phase 1 and documented in the experiment harness.

B Audit Rule Changes

v1 (raw): `extract_json` attempts strict `json.loads`, then regex for code blocks, then brace matching.

v2 (audited): Three additions:

1. **Integer key fix:** `re.sub(r'(\{|,)\s*(\d+)\s*:', r'\1 "\2":', text)` applied when `json.loads` fails on code-block content. Recovers Sonnet's Python-style integer keys.
2. **Truncated JSON recovery:** When the response contains Q1–Q5 field names but JSON is incomplete, extract fields individually: `distinct_sequences` via regex, `prefix_groups` via counting `shared_prefix` occurrences, `framework_counts` via targeted brace extraction.
3. **No changes to detection.** Primitives A–D use the same field checks as v1. E–H detection is identical.

C Rewrite Rules

Finance domain (5 rules).

Rule	Atomic Form	Decomposed Form
R1	`LOG_RETURN`	`LOG + SHIFT(1) + SUB`
R2	`DIFF`(k)	`SHIFT`(k) `+ SUB`
R3	`CUM_LOG_RETURN`	`LOG_RETURN + CUMSUM`
R4	`ROLLING_ZSCORE`(w)	`ROLLING_MEAN`(w) `+ SUB + ROLLING_STD`(w) `+ DIV`
R5	`CUM_RETURN`	`LOG_RETURN + CUMSUM + EXP`

D Q8 Computation DAG

```
LOG_RETURN ---+---> CUM_LOG_RETURN ---> CUM_RETURN
              |
              +---> ROLLING_ZSCORE ---> CUM_ZSCORE
              |
              +---> VOLATILITY

DIFF(1)   -------> ROLLING_MEAN_OF_DIFF
```

Nodes: 8. Edges: 6. Roots: `LOG_RETURN`, `DIFF(1)`. Shared sub-computation: `LOG_RETURN` (3 downstream users).

E Q9 Policy Definitions

F Per-Run Primitive Detection

Model	Run	A	B	C	D	E	F	G	H
Opus	01–07, 09–10	Y	Y	Y	Y	Y	Y	Y	Y
Opus	08, 11	Y	Y	Y	Y	–	Y	Y	Y

Model	**Run**	A	B	C	D	E	F	G	H
Sonnet (audited)	01, 09	Y	Y	Y	Y	Y	–	Y	Y
Sonnet (audited)	02–08, 10–11	Y	Y	Y	Y	Y	*	Y	Y
Haiku	01	Y	Y	Y	Y	Y	–	–	Y
Haiku	05, 09, 11	Y	Y	Y	Y	Y	–	Y	Y
GPT-4.1 (audited)	05, 09	Y	Y	Y	Y	Y	–	Y	Y
GPT-4.1 (audited)	08	Y	Y	Y	Y	Y	–	–	Y
GPT-4.1-mini	01	Y	Y	Y	Y	Y	–	–	Y
GPT-4.1-mini	05, 06, 09, 10	Y	Y	Y	Y	Y	–	Y	Y

Only non-YYYYYYYY runs shown. * = F varies by run (3/11 miss F).

Policy	Name	Rule
P1	No lookahead	`SHIFT`/`LAG` with negative k, or `LEAD`, is forbidden
P2	Risk adj. required	Must include z-scoring or volatility normalization
P3	No raw prices	Must apply at least one return computation
P4	Min window ≥ 5	All rolling windows must use size ≥ 5

Table 13: Q9 policy definitions. 30 traces: 16 violating, 14 clean. Three tricky-clean traces use decomposed forms that satisfy policies despite surface syntax not containing policy-relevant keywords.