Computational Identity: Supplementary Evidence and Conservative Index Characterization

Thomas Dionysopoulos, CFA

June 2026 — Working Note

Abstract

Paper 12 defines Computational Identity (CI) and demonstrates it in two domains (DSL, SQL). This note consolidates three additional experiments and a formal characterization of CI as a *conservative index* for execution agreement. The experiments cover: (A) agent tool governance with CI-augmented authorization, (B) cross-version drift detection in dataframe query plans, and (C) behavioral CI extracted by independent language models. Together with Paper 12, they establish CI across five application contexts in three domains. The conservative index theorem formalizes the error structure: CI has zero false positives by construction, and canonicalization monotonically reduces false negatives.

1 Relationship to Paper 12

Paper 12 contributes:

1. The formal definition of CI: $\mathrm{CI}_L(e) = H(\mathrm{ser}(\mathrm{plan}(\mathrm{canon}(e))))$.
2. The identity stack (six existing layers, none covering computation).
3. Implementation in a DSL (BLISP): 97 false hits eliminated, 515 comparisons, 0 mismatches.
4. Implementation in SQL: 22/22 TPC-H equivalence classes collapsed, 56 variants, 0 false equivalences.
5. Five boundary cases showing structural vs. semantic limits.

This note does **not** repeat Paper 12's contributions. It extends the evidence base with three new experiments and provides a formal characterization of CI's error structure that Paper 12 states informally.

2 The Conservative Index Theorem

Definition 1 (Conservative Index)**.** *Let Σ be a set of computation specifications and $\equiv$ a semantic equivalence relation on Σ. An identity function $I : \Sigma \to \mathcal{H}$ is a* conservative index *for $\equiv$ if:*

$$I(s_1) = I(s_2) \implies s_1 \equiv s_2 \qquad \textit{(soundness: no false positives)}$$

but not necessarily:

$$s_1 \equiv s_2 \implies I(s_1) = I(s_2) \qquad \textit{(completeness may fail: false negatives exist)}$$

Definition 2 (Sound Canonicalizer)**.** *A function $C : \Sigma \to \Sigma$ is a* sound canonicalizer *for $\equiv$ if $C(s_1) = C(s_2) \implies s_1 \equiv s_2$.*

Theorem 1 (Canonicalization Monotonicity)**.** *Let $I = H \circ ser$ be a conservative index (hash of serialized form). Let C be a sound canonicalizer. Then:*

1. *$I \circ C$ is also a conservative index (soundness preserved).*
2. *$FN(I \circ C) \subseteq FN(I)$, where $FN(I) = \{(s_1, s_2) : s_1 \equiv s_2 \wedge I(s_1) \neq I(s_2)\}$ (false negatives can only decrease).*
3. *Adding sound canonicalization rules is monotone: if C_2 extends C_1 (collapses a strict superset of equivalence classes), then $FN(I \circ C_2) \subseteq FN(I \circ C_1)$.*

Proof. (1) Suppose $I(C(s_1)) = I(C(s_2))$. Since H is collision-free (modulo negligible probability), $\text{ser}(C(s_1)) = \text{ser}(C(s_2))$, so $C(s_1) = C(s_2)$. By soundness of C, $s_1 \equiv s_2$.

(2) If $(s_1, s_2) \in \text{FN}(I \circ C)$, then $s_1 \equiv s_2$ and $I(C(s_1)) \neq I(C(s_2))$, which implies $C(s_1) \neq C(s_2)$. Since C did not collapse s_1 and s_2 to the same form, and without C they certainly differ (different original forms), $(s_1, s_2) \in \text{FN}(I)$.

(3) If C_2 collapses every class that C_1 collapses plus more, then any pair resolved by C_1 is also resolved by C_2, so $\text{FN}(I \circ C_2) \subseteq \text{FN}(I \circ C_1)$. □

Remark 1. *The theorem is straightforward, nearly tautological. Its value is not mathematical depth but conceptual clarity: it frames CI's quality as a measurable, improvable quantity, and guarantees that improving canonicalization never introduces new errors. The hard work is empirical: demonstrating that practical canonicalization achieves substantial false negative reduction across diverse domains.*

Proposition 1 (Incompleteness is inherent)**.** *For any Turing-complete specification language, semantic equivalence is undecidable (Rice's theorem). Therefore no conservative index can achieve zero false negatives in general. However, for each decidable equivalence class (syntactic aliases, algebraic rewrites, optimizer normalizations), a sound canonicalizer exists that eliminates the corresponding false negatives.*

3 Experiment A: Agent Tool Governance (IronClaw)

Full details: `ironclaw_study.tex`. Summary of results.

3.1 Setup

IronClaw is an agent runtime with deny-by-default authorization, trust tiers, capability leases, and event sourcing. Its security model governs *tool access* ("is this agent allowed to call this tool?") but not *computation content* ("what does this tool call compute?").

We tested 20 built-in tools across three scenarios, using CI computed over DataFusion logical plans for SQL-dispatching tools.

3.2 Three Scenarios

Scenario 1: Schema Blindness. `read_query` accepts {`query` : `string`}. A benign `SELECT c_name FROM customer WHERE c_custkey=42` and a data exfiltration `SELECT c_acctbal FROM customer WHERE c_acctbal > 9000` both pass IronClaw's authorization. CI produces distinct identities. A CI whitelist blocks the unauthorized query.

Scenario 2: LLM Semantic Drift. Three SQL rewrites of TPC-H Q5 (different aliases, join order, one hallucinated filter). IronClaw: all three pass. CI: Day 1 = Day 2 (same canonical plan), Day 3 ≠ Day 1 (hallucinated filter detected).

Scenario 3: Supply Chain Drift. A certified query template is modified by a dependency update (added filter, added JOIN). IronClaw: all pass (same tool, same schema). CI: each modification produces a different identity, breaking the cryptographic seal.

3.3 Result

Three classes of vulnerabilities that schema-based authorization misses, all caught by CI. Reproducible via `cargo run -example ironclaw_demo`.

4 Experiment B: Polars Version Drift

Full details: `polars_ci_study.tex`. Summary of results.

4.1 Setup

CI applied to Polars, a dataframe library with lazy evaluation:

$$\text{CI}(e) = \text{SHA-256}(\texttt{polars:}v\texttt{:}\text{plan}(e))$$

where plan(e) is from `LazyFrame.explain()`. Two versions tested: **0.20.31** and **1.38.1**.

4.2 Results

Table 1: Polars CI results across three scenarios.

Scenario	**v0.20**	**v1.38**
Syntax equivalence (collapse rate)	2/5	3/5
Cross-version drift (5 identical queries)	5/5 drifted	
Boundary cases (join, node type)	2 consistent	

Key finding. Every query produces a different optimized plan between v0.20 and v1.38. Unversioned CI silently invalidates computation caches. Versioned CI makes drift explicit by construction. The same flaw-detect-fix pattern as the SQL case study in Paper 12.

Canonicalization improvement. v1.38's optimizer gained predicate pushdown through aggregation, collapsing one additional equivalence class (filter-then-group = group-then-filter). CI captured this improvement automatically. This is Theorem 1 in action: the optimizer upgrade is a stronger canonicalizer.

5 Experiment C: Discovery Certificates

Full details: `dc_full_experiment.py`, `dc_full_results.json`.

Table 2: Discovery Certificate dimensions.

Dimension	What it captures	Values
Mutability	State change?	PURE, MUTATES
Action	Operation type	READ, SEARCH, CREATE, UPDATE, DELETE, MERGE, OVERWRI
Output domain	Result category	DATA, CONTENT, STRUCTURE, DIFF, PR, ISSUE, REF, REPO, US

5.1 Setup

45 tools from 4 MCP servers (GitHub, filesystem, git, SQLite). 3 models: Claude Sonnet 4.6, GPT-4o, Gemini 2.5 Flash. Each model received tool schemas (no tool names) and classified each tool into a behavioral certificate with three dimensions:

Behavioral CI = SHA-256(mutability | action | output_domain), truncated to 16 hex.

5.2 Phase A: Convergence and Separation

Collapse (cross-model agreement).

- Strict (3/3 models agree): **26/45** tools (58%)
- Majority (2/3 agree): **39/45** tools (87%)
- Unknown classification rate: 0% (all 135 classifications resolved)

Separation (confusable pairs). 15 confusable pairs tested (tools with identical or near-identical schemas). **10/15** correctly separated by CI. 5 unseparated pairs are genuinely similar tools (e.g., `get_pull_request` vs. `get_pull_request_files`: both PURE|READ, differing only in output granularity).

All 3 CRITICAL-severity pairs separated. All 4 HIGH-severity pairs separated.

Table 3: CI vs. schema hash on confusable pairs.

Method	Separates confusable	Collapses equivalent
Schema hash	9/15	N/A
Behavioral CI (strict)	10/15	26/45 (58%)
Behavioral CI (majority)	10/15	39/45 (87%)

Baseline comparison.

5.3 Phase B: Agent Error Detection

30 prompts presented to 3 models, each choosing from all 45 tools. Prompts designed to be adversarial: same-schema tool groups, ambiguous task descriptions.

Key finding. CI catches every error that schema catches, plus 4 additional errors that schema is blind to (same-schema tool confusions). Schema catches zero errors that CI misses. CI strictly dominates schema-based validation on this benchmark.

Table 4: Phase B error detection results ($n = 90$ agent decisions).

Agent correct	82
Agent wrong	8
Both CI and schema catch error	3
CI catches, schema misses	**4**
Schema catches, CI misses	**0**
Neither catches	1
CI catch rate (of 8 errors)	7/8 (88%)
Schema catch rate (of 8 errors)	3/8 (38%)

5.4 Canonicalization in Discovery Certificates

The output domain taxonomy evolved from 18 fine-grained categories to 7 macro categories. This is canonicalization applied to behavioral classification:

Table 5: Effect of taxonomy coarsening (canonicalization) on collapse.

Taxonomy	**Strict collapse**	**Majority collapse**
Fine (18 categories)	23/45 (51%)	35/45 (78%)
Macro (7 categories)	26/45 (58%)	39/45 (87%)
False negatives eliminated	3	4
Soundness violations	0	0

This is Theorem 1 applied to a third domain: coarsening the taxonomy is a sound canonicalizer (merging genuinely equivalent categories), and it monotonically reduces false negatives while preserving soundness.

6 Cross-Domain Summary

Table 6: Canonicalization across all domains.

Domain	**Canonicalization type**	**Before**	**After**	**FN eliminated**
BLISP (DSL)	Normalize aliases	284 names	237 canonical	47 (17%)
SQL (TPC-H)	Optimizer + rules	56 variants	22 classes	34 (61%)
Polars	Optimizer upgrade (v0.20→v1.38)	2/5 collapse	3/5 collapse	1 class
Agent tools	Taxonomy coarsening (18→7)	51% strict	58% strict	3 tools

Soundness across all domains. Zero false positives in every experiment:

- BLISP: 0/515 comparisons with matching CI but different behavior
- SQL: 0/22 equivalence classes with false collapse
- Polars: 0 false collapses across both versions
- Agent tools: 0 separation failures on CRITICAL/HIGH pairs

The pattern. Every domain shows the same structure: (1) surface variation exists, (2) canonicalization collapses some of it, (3) soundness is preserved, (4) residual false negatives are characterized. The canonicalization *mechanism* differs (alias lookup, algebraic rewriting, optimizer normalization, taxonomy coarsening) but the *theorem* is the same.

7 What This Note Does Not Contain

- **Protocol design.** When CI disagrees, what must two systems exchange to determine execution agreement? This is a separate research question requiring its own theorem (specification sufficiency). This note stops at: "when canonicalization fails, richer specification exchange becomes necessary."
- **Formal semantics.** Soundness is demonstrated empirically, not proved from a formal semantics of each domain. A formal proof would require axiomatizing the equivalence relation $\equiv$ for each domain, which is out of scope.
- **Completeness bounds.** We measure false negative rates but do not prove tight bounds. The residual false negatives depend on the domain's equivalence structure and the canonicalizer's coverage, both of which are open-ended.

8 Conclusion

CI is a conservative index for execution agreement: matching identity guarantees matching execution (zero false positives), but non-matching identity does not guarantee different execution (false negatives exist).

Canonicalization monotonically reduces false negatives while preserving soundness. This is demonstrated across four application contexts in three domains, with three qualitatively different canonicalization mechanisms.

When canonicalization succeeds, execution agreement reduces to identity comparison. When canonicalization fails, richer specification exchange becomes necessary. Characterizing the minimum exchange required to resolve such disagreements is the natural next question.

9 Artifact Inventory

All artifacts are in the BLISP repository under `artifacts/`.

Data files. Phase A raw certificates: `dc_full_phase_a_raw.json` (135 classifications, cached). Phase B raw proposals: `dc_full_phase_b_raw.json` (90 agent decisions). Polars plans: `polars_ci_v0_20_31.jsor` `polars_ci_v1_38_1.json`. SQL pilot results: `sql_ci_pilot_v2.py` (self-contained, executable).

Table 7: Artifact locations and status.

Experiment	**Directory**	**Key files**	**Status**
Paper 12 (main)	`paper_ci/`	`computational_identity.tex`	Submitted
IronClaw (Exp. A)	`paper_ci/`	`ironclaw_study.tex`	Complete
Polars (Exp. B)	`paper_ci_polars/`	`polars_ci_study.tex, .py`	Complete
Discovery Cert. (Exp. C)	`paper13_protocol/`	`dc_full_experiment.py`	Complete
		`dc_full_results.json`	
		`dc_full_phase_a_raw.json`	
		`dc_full_phase_b_raw.json`	
DC Pilot	`paper13_protocol/`	`dc_pilot_report.md`	Complete
DC Pilot OOS	`paper13_protocol/`	`dc_pilot_oos_results.json`	Complete
SQL canonicalization	`paper12_ironclaw_drift/`	`sql_ci_pilot_v2.py`	Complete
Protocol proposal	`paper13_protocol/`	`protocol_proposal.md`	Draft
This note	`paper_ci/`	`ci_supplementary_evidence.tex`	This file