

Article

Not peer-reviewed version

Toward Candidate $P = NP$ Routes via Multitime Transport Compilation: Receipt-Gated Canonicalization and Conditional Deciders

[Rogério Figurelli](#) *

Posted Date: 28 January 2026

doi: 10.20944/preprints202601.2234.v1

Keywords: P vs NP; NP-completeness; SAT; reductions; multitime; Temporal State Machines; receipt-gated protocols; auditability; no-reopen; abstain gating; canonicalization; Weisfeiler–Lehman; proof logging (DRAT/LRAT/FRAT); verification budgeting; selective classification; conditional deciders; co-evolution ledger; Circle of Equivalence



Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a [Creative Commons CC BY 4.0 license](#), which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Toward Candidate $P = NP$ Routes via Multitime Transport Compilation: Receipt-Gated Canonicalization and Conditional Deciders

Rogério Figurelli

Independent Researcher, Brazil; figurelli@gmail.com

Abstract

This paper continues the Multitime Barriers program by reframing progress on P vs NP as an operational transport question across reductions: do “reasons” for satisfiable or unsatisfiable outcomes remain stable, checkable, and auditable when transformed under declared regimes? We introduce Multitime Transport Compilation, a receipt-gated protocol that converts admissible transformations into comparable, replayable artifacts under a Temporal State Machine kernel. The kernel treats clocks as transition families and enforces enriched state constraints: admissibility corridors, no-reopen discipline, abstain gating, and recovery-to-okay feasibility. Within this framework, we define a conditional candidate decider as a pipeline — canonicalize $\rightarrow$ key $\rightarrow$ library lookup $\rightarrow$ bounded verification — whose claims are restricted to what can be closed within the current receipts. Empirically, we report transport tests and closure behavior under multiple stages and regimes, highlighting (i) stable measured overlap coexisting with closure deficits (key_hit below gate and abstain above gate) and persistent unresolved ties, and (ii) a counterexample where increased Weisfeiler–Lehman depth correlates with measured overlap loss and a scale-guard violation, motivating rollback discipline. The contribution is not a proof of $P = NP$, but a constructive, auditable research program: make comparability mechanical, classify failure modes by receipts, and publish a protocol that other groups can scale and verify.

Keywords: P vs NP ; NP -completeness; SAT; reductions; multitime; Temporal State Machines; receipt-gated protocols; auditability; no-reopen; abstain gating; canonicalization; Weisfeiler–Lehman; proof logging (DRAT/LRAT/FRAT); verification budgeting; selective classification; conditional deciders; co-evolution ledger; Circle of Equivalence

Contributions

- Multitime Transport Compilation: a receipt-gated compilation protocol that turns admissible transformations into comparable artifacts and bounded verification obligations.
- Mechanical gates that separate measured transport stability (overlap) from closure sufficiency (key_hit, abstain, ties) and scale behavior (bounded library growth).
- Receipt-backed counterexample to monotone refinement assumptions: increasing WL depth (as configured) correlates with measured overlap loss and scale-guard failure, justifying rollback as a governance primitive.
- A paper-first runtime discipline: each run is a single receipt-producing event, and the paper records only receipt-backed empirical claims.
- A multitime epistemic ledger ($P(\tau)+NP(\tau)=1$) and a closure archetype (CoE) used as governance motifs, not as complexity-theoretic theorems.

1. Introduction

The P vs NP question is usually framed as a global boundary in computational feasibility. This paper approaches it as a constructive systems question: can one build a conditional, auditable

decision pipeline that remains comparable under admissible transformations and closes within bounded resources?

Reductions and representation changes are central to NP-completeness, but they also create a transport problem: invariants that appear decisive in one representation may fail to remain stable, checkable, or comparable after transformation.

In [1], we argued that barriers to $P = NP$ can be reinterpreted as multitime barriers: different clocks govern transformation, verification, comparability, and repair, and “reasons” may fail to travel efficiently.

The present paper extends that diagnosis into an executable compilation program: Multitime Transport Compilation converts admissible transformations into receipt-gated comparability, enabling conditional candidate deciders under explicitly declared regimes rather than informal claims.

This work is oriented toward discovering evidence consistent with a constructive route to $P = NP$, but it does not claim $P = NP$. The purpose is to make “progress” legible as a sequence of auditable, falsifiable steps inside an explicit operational envelope: a declared set of admissible transformations and verification budgets, with outcomes recorded as receipts and promotion blocked unless mechanical gates are satisfied [1].

This stance is compatible with the standard NP-completeness framing (where reductions define what must be preserved) [4,5], and with the broader complexity-theory norm that claims must be conditional on clearly stated assumptions and models [7–9]. It also acknowledges that major barriers and non-monotonicities can arise when one attempts to turn informal “structure” into algorithms, motivating the paper’s insistence on gates, rollback, and bounded growth rather than narrative confidence [10,11,23,24].

Empirical statements are therefore restricted to what is receipt-backed by the runs reported here, and all promotion decisions are gated mechanically: overlap and scale constraints serve as comparability controls, while key_hit, abstain, and tie closure serve as closure controls [1]. The explicit inclusion of abstain follows the reject-option and selective classification tradition, where refusal is a legitimate outcome but must be measured and bounded rather than hidden inside error bars or anecdotal exceptions [21,22].

Finally, the verification posture is informed by the SAT community’s emphasis on making solver behavior testable and checkable, including proof-oriented workflows and standards for inspectable artifacts — even though full external proof-log integration remains a limitation in the receipts shown here [12–17].

2. Background and Continuity with the TSM Kernel

2.1. The TSM Kernel as an Operational Constraint System

The Temporal State Machine (TSM) kernel treats clocks as transition families and state as enriched with governance constraints:

- Admissibility corridors: each run declares allowed transformations and canonicalizers.
- Receipts: empirical claims must be tied to runtime receipt hashes.
- No-reopen discipline: once receipts are logged, results are not reinterpreted by introducing hidden conditions.
- Abstain gating: refusal to decide is allowed but measured and mechanically gated.
- Recovery-to-okay feasibility: changes that cannot be repaired within declared budgets are not promotable.

2.2. Continuity with [1]

[1] motivates the transport framing and governance principles. This paper extends [1] by operationalizing “transport” as a compiled pipeline whose outputs are comparable across stages and regimes, and by using receipts to distinguish drift-like failures from closure-like failures.

2.3. Epistemic Accounting and Closure Archetypes ($P+NP=1$ and CoE)

Following [1], we adopt an explicit accounting posture toward claims using the co-evolution ledger $P+NP=1$ as a governance motif [2]. Here the notation is not a complexity-theoretic theorem; it is an epistemic accounting identity: within a declared operational frame, credit is partitioned between what is actually closed under receipts and gates and what remains unclosed.

We also use the Circle of Equivalence (CoE) as a closure archetype: equivalence is not a slogan but a closed operational loop that must satisfy minimal consistency before execution [3]. In this paper, CoE is not imported as a result about P vs NP ; it is used as a compact design motif for no-reopen closure: if a run fails gates (key_hit, abstain, ties), the correct action is abstention and controlled retesting under one changed knob, not retroactive reinterpretation.

2.4. Multitime Generalization: Clock-Indexed Epistemic Ledger

We generalize the ledger notation from a single index to multitime windows. Let τ denote a declared clock-window (or clock-vector) that indexes the operational regime under which receipts are produced — e.g., canonicalization depth and budget, library growth constraints, verification budget, and tie-resolution policy. We then write $P(\tau)+NP(\tau)=1$ as an epistemic accounting identity: within τ , the protocol's "credit" is partitioned between what is actually closed under receipts and mechanical gates ($P(\tau)$) and what remains unclosed, refused, or unresolved ($NP(\tau)$). This is not a claim about the equality of complexity classes; it is a governance rule that prevents narrative inflation. Credit can shift between $P(\tau)$ and $NP(\tau)$ across runs only when closure receipts improve under a one-knob change, while preserving admissibility corridors and scale guards.

3. Method: Receipt-Gated Miner Protocol

3.1. Receipts and Artifacts

Each run produces receipt-like artifacts that declare configuration and report outcomes under bounded budgets:

- RCR (RunConfigReceipt): configuration, budgets, and regime declaration.
- DSR (Discovery Summary Receipt): discovery coverage, abstain rate, overlap, and no_overlap indices.
- LR (Library Receipt): lib_keys, key_hit rate, candidate statistics, canonicalization time.
- TBR (Tie-Break Receipt): tie_rate, tiebreak_success, unresolved count and samples (when enabled).
- UR (Uplift Receipt): run-level verdict and a printed rationale string.

This paper uses only what appears in these logs, and treats receipts as the boundary of what is claimable for each run. [1]

3.2. Candidate-Decider Compilation Pipeline

A "candidate polynomial decider" in this paper is conditional and receipt-gated: canonicalize $\rightarrow$ key $\rightarrow$ library lookup $\rightarrow$ bounded verification, with abstain allowed but mechanically gated. The point is not to assert a decider, but to make the path to one mechanically checkable: when gates fail, the paper records which failure mode blocks promotion under declared regimes and budgets. [1,4–9]

3.3. Operational Definitions (Overlap, Key_Hit, Abstain, Ties, Growth)

To reduce ambiguity, the paper uses the following operational meanings as reflected in receipts:

- overlap: measured agreement of the transport signature between the compared sets under the run's declared canonicalizer+key semantics. overlap_min summarizes worst-case overlap; no_overlap lists concrete indices where overlap fails. (This is an instrumentation-dependent proxy, not a universal semantic invariant.) [1]

- `key_hit`: fraction of test instances whose derived key is found in the library index produced under the same declared semantics; `key_hit_min` summarizes worst-case key-lookup closure.
- `abstain`: fraction of instances where the pipeline refuses to promote a decision under declared budgets and constraints (e.g., missing key, insufficient verification closure, or unresolved tie conditions); `abstain_max` summarizes worst-case refusal.
- `ties`: `tie_rate` measures how often multiple candidates compete; `tiebreak_success` and `unresolved` report whether the declared tiebreak policy closes contestability. Persistent unresolved ties count as non-closure. [21,22]
- `library growth`: `lib_keys` tracks library size; the scale guard constrains growth across sizes to resist memorization-by-fragmentation within the envelope. [1]

3.4. Mechanical Gates and Paper Taxonomy (Decoupled from Printed Rationales)

The miner prints mechanical gate verdicts and a printed rationale label. For auditability, the paper separates:

- Mechanical gate failure: determined by thresholds on `overlap`, `key_hit`, `abstain`, `ties`, `lib_keys`, and the scale guard.
- Failure taxonomy label (paper): assigned from receipts and must be consistent with metrics. In particular, if `overlap_min` = 100% but gate failure arises from `key_hit`/`abstain`/`ties`, the correct paper label is closure deficit under stable measured overlap (e.g., `KEY_MISS` and/or `TIES_BROKEN`), not `KEY_DRIFT`. [1]

Taxonomy labels used (paper):

- `KEY_DRIFT`
- `KEY_MISS`
- `DISCOVERY_COVERAGE_LIMIT`
- `RECALL_TRUNC`
- `TIES_BROKEN`
- `CANONICALIZER_INSUFFICIENT`
- `VERIFY_COST`

3.5. Lemmas and Theorem-Style Operational Claims (Conditional, Receipt-Scoped)

The statements in this subsection are not complexity-theoretic theorems. They are protocol-level claims about what the instrumentation and gates allow the paper to infer, conditional on the declared semantics and the correctness of the receipts. [1,4–9]

Lemma 3.5.1 (Mechanical promotion only).

A run is “promoted” by this paper if and only if it satisfies all declared gates. No qualitative interpretation can override a failing gate. [1]

Lemma 3.5.2 (Transport-closure separation).

A run may have `overlap_min` = 100% under the declared canonicalizer+key semantics and still fail promotion because closure does not meet gates — e.g., `key_hit` remains below threshold, `abstain` exceeds the allowed bound, and/or `ties` remain unresolved.

Therefore, overlap stability is evidence of invariance under the current measurement proxy, not evidence of a decider (and not a P = NP claim). In the paper’s multitime framing, this corresponds to the transport clock remaining stable while closure clocks (lookup recall, refusal rate, contestability resolution) remain open. [1]

Lemma 3.5.3 (Audit-consistent classification rule).

If the receipts report `overlap_min` = 100% (worst-case across the relevant stages) and `no_overlap` is empty, then the transport proxy is stable under the declared canonicalizer+key semantics. In that situation, any gate failure must be classified as **closure-limited** (e.g., insufficient `key_hit`, excessive `abstain`, failed tie closure, bounded-growth violations), even if a printed runtime label says `KEY_DRIFT`. The point is not to “rename” failures, but to keep the paper’s interpretations consistent with its own instrumentation: when the transport clock (overlap) is receipt-stable, the bottleneck is

located on closure clocks, and knob selection should target closure variables rather than drift mitigation. This separation is a core auditability rule: it prevents post-hoc narrative drift and makes the next admissible experiment mechanically determined by what failed. [1]

Lemma 3.5.4 (Tie closure blocks promotion).

Under the declared tie gate ($\text{tie_rate} \leq 1\%$ OR ($\text{tiebreak_success} \geq 99\%$ and $\text{unresolved} = 0$)), persistent contestability is treated as non-closure: if tie_rate remains high while tiebreak_success stays near zero and unresolved ties persist, the run cannot be promoted — even when overlap is stable and other closure metrics improve. The reason is operational and audit-driven: unresolved ties mean the pipeline encountered competing candidates that it could not mechanically adjudicate within its declared policy and budgets, so any “decision” would be either arbitrary or hidden work. In this protocol, tie handling is therefore part of the decider story (not a cosmetic post-processing step): until ties are closed by a validated, receipt-visible policy, the system has not demonstrated governable closure at scale. This aligns with reject-option/selective-decision principles: abstention/deferral can be admissible, but only when it is explicitly accounted for, mechanically bounded, and progressively reduced rather than used to mask difficulty. [1,21,22]

Lemma 3.5.5 (Bounded growth is necessary under library reuse).

If the library is the vehicle for reuse ($\text{canonicalize} \rightarrow \text{key} \rightarrow \text{lookup} \rightarrow \text{verify}$), then bounding lib_keys and enforcing a scale guard are necessary to resist memorization-by-fragmentation within the tested envelope. These guards do not prove generalization, but without them improved key_hit can be achieved by uncontrolled key fragmentation. [1]

Proposition 3.5.6 (Receipt-backed negative results isolate bottlenecks).

If two runs differ by a single declared knob while preserving canonicalizer+key semantics and regimes, and worst-case closure metrics do not improve, then the receipts support the narrow conclusion that this knob is not the primary bottleneck within the tested envelope. This is not a global statement about SAT or NP-completeness; it is a scoped bottleneck isolation statement. [1,4–9]

Proposition 3.5.7 (Constructive route means governed iteration).

A “constructive route” in this paper means: define comparability by canonicalizer+key semantics; compile transport into a bounded library; verify under bounded policy; gate promotion mechanically; iterate admissible knobs while preserving receipts and rollback discipline. The route is constructive as a testable compilation program, not as a completed proof of $P = NP$. [1]

Theorem 3.5.8 (Reporting discipline soundness, conditional).

Assuming receipts are faithfully produced by the declared miner implementation, the protocol ensures:

- (a) comparability across runs is mechanical (same semantics, same gates, same reporting form),
- (b) empirical statements are bounded to the declared operational envelope, and
- (c) failure modes are separable into transport stability (overlap proxy), closure deficits ($\text{key_hit}/\text{abstain}$), contestability deficits (ties), and bounded-growth constraints (lib_keys and scale guard), without requiring narrative interpretation. [1]

3.6. Why These Are Not Solver Benchmarks

A fair critique is that these results resemble solver logs. The intended contribution is different: these are transport and closure receipts under a governance kernel. The program’s key objects are comparability stability (overlap), closure sufficiency (key_hit , abstain), contestability (ties), and bounded growth (lib_keys and scale guard). This instrumentation constrains what can be claimed and makes progress auditable, rather than optimizing performance narratives. [1,7–9,12–17]

3.7. What “Progress” Means in This Paper (Mechanical Promotion Only)

This work is oriented toward discovering evidence consistent with a constructive route to $P = NP$, but it does not claim $P = NP$. Empirical statements are restricted to what is receipt-backed by the runs reported here, and all promotion decisions are gated mechanically. [1,4–9]

Concretely, “progress” is defined as movement along a disciplined ladder of auditable conditions. First, runs must be comparable: the run configuration (stages, regimes, canonicalizer parameters, budgets, and gates) is bound by receipts so that outcomes can be interpreted without post-hoc narrative changes. [1] Second, the protocol must make failure modes classifiable: when $\text{overlap_min} = 100\%$ under the current proxy, failures are treated as closure-limited ($\text{key_hit}/\text{abstain}/\text{ties}$) rather than drift-limited, because this distinction determines which knobs are admissible to turn next. [1] Third, progress requires that closure be demonstrated mechanically: the candidate decider is not “improved” by rhetoric, only by meeting explicit thresholds on key recall, abstention, tie closure, and bounded library growth, including scale guards that resist memorization-by-fragmentation. [1]

This definition makes abstention and reject behavior first-class: the miner is allowed to refuse when it cannot close safely under its declared policy, and such refusals are recorded rather than hidden. This aligns with selective decision systems where abstention can be rational but must be accounted for and mechanically bounded if the goal is eventual promotion to a total decider. [21,22]

3.8. Remark ($W = I \wedge C$ as an Audit Philosophy, Non-Operational)

As a non-operational framing, the protocol aligns with the archetype $W = I \wedge C$, where W is wisdom, I is intelligence, and C is consciousness (intelligence elevated by awareness, contestability, and self-binding).

In this reading, C is not “constraint” itself; rather, C is the capacity to notice when the system is not justified to act, to keep reasons comparable, and to refuse promotion when closure is not mechanically demonstrated. The miner’s receipts, budgets, gates, and rollback discipline are therefore best understood as a computable proxy for C : a consciousness-like operator that forces the system to remain accountable to its own declared limits rather than to narrative confidence. [3]

This “computable C ” is implementable as a runtime layer that sits above the inference engine. Concretely: (i) every run declares a regime and produces receipts (RCR/DSR/LR/TBR/UR) that bind what the system is allowed to claim; (ii) promotion is permitted only when explicit gates are satisfied (closure thresholds, tie closure, bounded growth, scale guard), otherwise the system must abstain; (iii) rollback is admissible when a knob degrades overlap stability or violates scale behavior, preserving audit continuity across multitime clocks (transport stability versus closure capacity). In this sense, consciousness is operationalized as “self-binding under receipts”: the system continuously checks whether it is inside an admissible corridor and refuses to act when it cannot justify closure within its declared budgets. [1]

This remark is motivational: it does not alter metrics or gates. It clarifies why the paper treats abstention, tie closure, and bounded growth as first-class requirements: they are the computational face of “awareness” in a system that would otherwise optimize for output without accountability. [21,22]

4. Multitime Transport Compilation Protocol

Multitime Transport Compilation is an *operational* protocol for turning “solver behavior” into auditable, comparable evidence about whether a reusable *transport signature* (canonicalization $\rightarrow$ key $\rightarrow$ lookup $\rightarrow$ verify) can be made to close under bounded resources.

The protocol treats distinct “clocks” as first-class: (i) a transport-stability clock (measured overlap under fixed semantics), (ii) closure clocks (key_hit , abstain , tie closure), and (iii) a scale/bounded-growth clock (library growth and cross-size guards). Empirical statements in this paper are restricted to what is receipt-backed by declared runs [1], with standard complexity context in [4–9].

4.1. Miner Stages $S1$ – $S5$

A run is a fixed staged pipeline. Each stage is a declared *test point* defined by (i) a size parameter n (e.g., 128/256/512/1024 in the reported runs), (ii) a split into paired sets (denoted in logs as N_A / N_B under a fixed split policy), and (iii) a regime — a declared composition of admissible transformations applied inside the miner.

Operationally, each stage executes a stack of the form:

- $N0_robust \circ WL$ (canonicalization + Weisfeiler–Leman-style refinement), optionally followed by
- $WL_tiebreak$ (a tie-resolution module, when enabled), and then
- library lookup + bounded verification under the stage's budgets.

Stages $S1$ – $S5$ are not “tuning passes”; they are predefined measurement checkpoints meant to expose different failure modes under controlled envelopes. The exact stage→regime schedule is declared by the run configuration and must not be altered post hoc if a run is to remain comparable [1].

The WL component is motivated as a practical canonical-form heuristic (in the orbit of graph canonicalization practice), not as a proof of canonical minimality [18–20].

4.2. Canonicalization and Key Semantics

Canonicalization maps each input instance into a representation intended to be stable under declared admissible transformations. In the reported runs, this pipeline includes (as declared by regime) components such as:

- PLE (pure literal elimination),
- BCE (blocked clause elimination),
- dup / lit-dup (controlled duplication normalization),
- taut-heavy (tautology-heavy simplification),
- sub (bounded subsumption-style reductions),

together with WL rounds to compute a refinement signature [18–20]. These are deliberately *budgeted*: the canonicalizer parameters (e.g., subsumption budgets, iteration caps, node caps) define what the protocol considers an admissible amount of work per instance, which is essential to keep the experiment interpretable as “bounded compilation” rather than unbounded solving [1].

Key semantics. The miner then forms a key from the canonicalized representation (conceptually: a hash of the canonical form / signature under the declared semantics). This enables:

- key_hit : fraction of test instances whose keys appear in the library built from the paired set under the same semantics (recall of the key index).
- $overlap$: fraction of instances whose keys are consistent across the compared sets under the same semantics (a measured invariance of the *current* key definition, not a universal semantic notion).
- $abstain$: fraction of instances the pipeline refuses to promote because closure cannot be achieved within declared policies (e.g., key absent, verification failure, budget exhaustion, or unresolved tie).

These definitions are intentionally operational: the “candidate decider” in this paper is precisely the composition $canonicalize \rightarrow key \rightarrow lookup \rightarrow verify \rightarrow promote/refuse$, not an unconstrained SAT solver [1,12,17].

4.3. Library Dynamics, Scale Guard, Bounded Growth

The library is the compilation substrate: it stores bounded information indexed by keys, so that future instances sharing a key can reuse previously compiled structure rather than re-derive it. This is the mechanism by which the protocol probes a *constructive* route (reusable transport) instead of a purely per-instance search [1].

Because a key-indexed library can degenerate into memorization, the protocol treats bounded growth as a hard constraint:

- lib_keys tracks the number of distinct keys held by the library (an index-size proxy).
- cap_per_key bounds how many candidates may be stored or retained per key.

- `max_candidates` bounds how many candidates may be considered (a recall/truncation knob).

To resist “memorization by fragmentation,” the protocol also imposes a scale guard: library growth at larger sizes must remain close to library growth at smaller sizes (as declared in the gate). This forces the route, if it exists, to look like reusable indexing rather than an expanding lookup table [1]. This framing is consistent with why disciplined reuse and bounded growth matter in any attempt to turn NP search into polynomial-time reuse under fixed semantics [4–9].

4.4. Verification Policy and Artifacts

Verification is treated as a bounded runtime step (the run declares a verify budget). Verification checks are internal to the miner’s policy: they determine whether a candidate can be promoted as “closed” under the declared regime and budgets.

Importantly, the receipts shown in this paper are *not yet* accompanied by externally checkable proof artifacts in standard SAT proof-log formats (e.g., DRAT/LRAT/FRAT-style ecosystems). As a result, auditability is currently strongest at the *protocol level* (gates, budgets, receipts, and reproducibility of the pipeline), rather than at the level of third-party replayable proof objects. Integrating standardized, independently checkable proof logs would strengthen reproducibility and align the methodology more closely with modern proof-checking practice in SAT workflows [15], with broader proof-complexity context in [23,24].

4.5. Mechanical Gates (Promotion Rules)

Promotion is mechanical: the paper defines explicit thresholds that must be met to treat a run as supporting a stronger claim. Gates are chosen to separate three distinct outcomes:

1. Transport stability (measured by `overlap_min` under fixed semantics),
2. Closure (high `key_hit`, low `abstain`, tie closure),
3. Bounded reuse at scale (bounded `lib_keys` and scale guard).

Tie handling is a first-class gate because unresolved ties represent contestability failures: multiple competing candidates exist and cannot be mechanically resolved within policy. If `tie_rate` remains high, `tiebreak_success` remains near zero, and unresolved cases persist, promotion is blocked — even when overlap is stable — because the protocol cannot honestly claim closure at scale under its own rules [1]. This is aligned with selective decision systems: abstention can be rational, but only if bounded and policy-consistent [21,22].

Finally, the paper’s failure taxonomy is defined to be *audit-consistent*: when overlap receipts show `overlap_min` = 100%, failures are classified as closure-limited (e.g., `KEY_MISS` / `abstain` / `ties`) rather than “drift,” even if a printed rationale string says otherwise. This is required for interpretability: the transport clock may be stable while closure clocks are not [1].

4.6. Reporting Discipline (Run Ledger)

Runs are reported under a strict ledger discipline:

- Each run is identified by a `RunConfigReceipt` (RCR) binding stages, regimes, budgets, canonicalizer parameters, and gate thresholds.
- Each stage emits receipts (including stage summaries) so that “what happened” is anchored to declared settings rather than post-hoc narrative.
- Each run is appended as its own subsection in Results (“Run gXXXX: supported by receipts”), and no aggregate claim is made unless multiple independent receipts support it under comparable configurations [1].

This reporting discipline is the core methodological contribution: it turns the search for constructive routes into a governed process where progress and failure are visible, comparable, and mechanically classifiable — without pretending that any single notebook-scale experiment settles the P vs NP question [1,4–9].

5. Results

This section reports results under a strict receipt-backed discipline. Each run is treated as a single, auditable episode whose claims are limited to what is explicitly present in its receipts (RunConfigReceipt, Stage Summaries, Gates, and UR Verdicts).

The purpose is not to “tell a success story,” but to make progress and failure mechanically comparable across runs, with knob changes interpreted as controlled interventions rather than post-hoc narrative.

Accordingly, each Run gXXXX subsection is evidence about a specific operational envelope: the staged miner design (S1–S5), declared regimes (R1–R3), canonicalizer+key semantics, and stated budgets. These runs are not presented as general SAT solver benchmarks; they are transport-and-closure receipts that localize bottlenecks under the paper’s candidate-decider pipeline (canonicalize → key → lookup → verify) and its mechanical promotion gates.

A crucial audit rule is enforced throughout: when overlap receipts show overlap_min=100.0%, the paper treats transport stability (under the current metric) as provisionally satisfied, and classifies the dominant failure as closure-limited (e.g., key_hit, abstain, tie closure, bounded growth), even if the miner prints a generic rationale such as KEY_DRIFT.

Conversely, when overlap receipts fall below 100.0% or show explicit no_overlap indices, the run is classified as measured drift under the current instrumentation. This separation implements the multitime ledger idea: different “clocks” can fail independently, so transport can remain stable while closure clocks do not.

In sum, “supported by receipts” means something concrete: every numeric claim in the run subsections (overlap_min, key_hit_min, abstain_max, lib_keys_max, scale-guard status, and tie metrics where reported) is directly taken from that run’s receipts, and the subsection’s failure taxonomy is required to be consistent with those measurements.

Any aggregate statement is deferred unless it is explicitly backed by multiple receipts reported in this paper.

5.0. Run Ledger (Receipt-Backed Reporting Discipline)

Each accepted run is recorded as its own receipt-backed subsection because the protocol treats empirical claims as audit objects, not narrative summaries: a claim is admissible only if it is traceable to a specific RunConfigReceipt and the associated stage receipts (DSR/LR/TBR/UR).

This is the same governance logic introduced in the multitime kernel framing of [1]: comparability is produced by declared regimes, mechanical gates, and no-reopen discipline, so the “unit of evidence” is the closed run, not an averaged storyline.

Accordingly, the paper makes no aggregate claim (e.g., “we improved closure” or “WL helps/hurts”) unless multiple independent run receipts explicitly support it under comparable conditions. This prevents over-interpreting variance that is known to arise in SAT-style pipelines due to heuristic interactions, instance distribution shift, and budget sensitivity — phenomena that have long been emphasized in SAT solver engineering and evaluation practice [12,13,17].

It also mirrors the broader complexity-theory norm that general conclusions require careful control of assumptions and distributions, rather than extrapolation from a small set of experiments [7–9].

5.1. Run g0008: Stable Measured Overlap with Closure Deficit and Unresolved Ties

RCR=ed782b6b16643ba386113e428e0d8759a4a14c483546a0b7e10185abca03d57e.
Config: S=12 skeletons, V=10 variants (fixed split 5/5), budgets discovery=280000 and verify=400000, WL rounds=12, cap_per_key=10, max_candidates=18. Canonicalizer params: sub_max_k=6, sub_budget_pairs=15000, bce_budget=8000, ple_max_iters=60, big_node_cap=50000. Regimes: R1=ple•bce, R2=dup•taut-heavy•lit-dup, R3=sub•dup.

Worst-case stage summaries:

- overlap_min=100.0% (S1–S5)
- key_hit_min=88.3% (S2–S4)
- abstain_max=11.7% (S2–S4)
- lib_keys_max=15 (S1)
- ties: tie_rate_max=50.0%, tiebreak_success_min=0.0%, unresolved_any=True
- scale guard: PASS

Mechanical gates fail under thresholds key_hit $\geq$ 99, abstain $\leq$ 1, and ties closure. Printed gate rationale is “KEY_DRIFT,” but since overlap_min=100.0%, the paper classifies the dominant issue as closure deficit under stable measured overlap, plus tie blockage.

Failure taxonomy (paper): KEY_MISS (primary), TIES_BROKEN (blocking).

What this enables: This run isolates a stable-overlap envelope while demonstrating insufficient closure. It supports focusing subsequent tests on closure knobs (coverage/recall/canonicalizer budgets) and tie resolution rather than drift mitigation.

5.2. Run g0009: WL-Depth Increase Correlates with Measured Drift and Scale-Guard Failure

RCR=dfae2ddc04dbc4d0a29341b021f561c7c9940c6e19a303c1f091e015ed0cbea7.

Same as g0008 except WL rounds=16.

Worst-case stage summaries:

- overlap_min=83.3% (S1–S3) and overlap_min=75.0% (S5), with no_overlap indices present
- key_hit_min=68.3% (S5)
- abstain_max=31.7% (S5)
- lib_keys_max=23
- ties: tie_rate_max=50.0%, tiebreak_success_min=0.0%, unresolved_any=True
- scale guard: FAIL (lib_keys(1024)=23 while lib_keys(512)=15, violating the +2 constraint)

Here overlap receipts support measured drift under the current metric, and scale behavior degrades.

Failure taxonomy (paper): KEY_DRIFT (primary) plus DISCOVERY_COVERAGE_LIMIT and/or RECALL_TRUNC symptoms (secondary), and TIES_BROKEN (blocking).

What this enables: This run provides a receipt-backed counterexample to monotone refinement: deeper WL (as configured) can reduce overlap and violate scale guards. It justifies rollback as an admissible governance move.

5.3. Run g0010: Rollback to WL=12 Restores Stable Overlap and Scale Guard; Closure Deficit Remains

RCR=f9ab65fd10c3b71f91820d2973d63d9ac6db5ec87c2b3cd7e35d80eb80374875.

Same as g0008 (WL rounds=12).

Worst-case stage summaries:

- overlap_min=100.0% (S1–S5)
- key_hit_min=88.3% (S2–S4)
- abstain_max=11.7% (S2–S4)
- lib_keys_max=15
- ties: tie_rate_max=50.0%, tiebreak_success_min=0.0%, unresolved_any=True
- scale guard: PASS

Mechanical gates fail for closure and ties. Printed gate rationale is “KEY_DRIFT,” but overlap receipts indicate stable measured overlap; the paper classifies the issue as closure deficit under stable measured overlap, plus tie blockage.

Failure taxonomy (paper): KEY_MISS (primary), TIES_BROKEN (blocking).

What this enables: Together with g0009, this supports treating WL=16 (in this configuration) as destabilizing measured overlap and scale behavior, while WL=12 preserves stable overlap. Remaining obstacles are closure and tie resolution.

5.4. Run g0011: Recall Ablation via Cap-Per-Key Increase Shows No Closure Uplift Under Stable Overlap

RCR=7e4a0539558f0c0475e502ec5d3358e4fad1557a63213be2c928a3861cfc59a0
(ConfigHash12=7e4a0539558f).
Config: S=12 skeletons, V=10 variants (fixed split 5/5), budgets discovery=280000 and verify=400000, WL rounds=12, cap_per_key=20, max_candidates=18. Canonicalizer params: sub_max_k=6, sub_budget_pairs=15000, bce_budget=8000, ple_max_iters=60, big_node_cap=50000. Regimes: R1=ple•bce, R2=dup•taut-heavy•lit-dup, R3=sub•dup.

Worst-case stage summaries:

- overlap_min=100.0% (S1–S5)
- key_hit_min=88.3% (S2–S4)
- abstain_max=11.7% (S2–S4)
- lib_keys_max=15 (S1)
- ties: tie_rate_max=50.0%, tiebreak_success_min=0.0%, unresolved_any=True
- scale guard: PASS

Mechanical gates fail under thresholds key_hit≥99, abstain≤1, and ties closure. Printed gate rationale is “KEY_DRIFT,” but since overlap_min=100.0%, the paper classifies the dominant issue as closure deficit under stable measured overlap, plus tie blockage. Increasing cap_per_key increases candidate load (avg_cand and p90_cand rise, with p90_cand saturating max_candidates=18), yet worst-case closure metrics (key_hit_min, abstain_max) do not improve.

Failure taxonomy (paper): DISCOVERY_COVERAGE_LIMIT (primary), TIES_BROKEN (blocking).

What this enables: This run provides a negative ablation: increasing per-key recall capacity does not move worst-case closure under the current configuration. It motivates shifting the next test knob toward discovery coverage (discovery_steps) or canonicalizer strength, while treating tie closure as a separate promotion blocker.

5.5. Run g0012: Doubled Discovery Budget, Stable Overlap, No Closure Uplift

RCR=ccc10332c0607211ae8903fa1681c8c20839d319d65a8380a4e2ae3c9b6e502e
(ConfigHash12=ccc10332c060). Configuration highlights: S=12 skeletons, V=10 variants (fixed split 5/5). Budgets: discovery=560000, verify=400000. WL rounds=12. cap_per_key=20, max_candidates=18. Canonicalizer parameters: sub_max_k=6, sub_budget_pairs=15000, bce_budget=8000, ple_max_iters=60, big_node_cap=50000. Regimes: R1=ple•bce, R2=dup•taut-heavy•lit-dup, R3=sub•dup.

Worst-case stage summaries (receipts):

- overlap_min=100.0% (S1–S5)
- key_hit_min=88.3% (S2–S4)
- abstain_max=11.7% (S2–S4)
- lib_keys_max=15 (S1)
- ties: tie_rate_max=50.0%, tiebreak_success_min=0.0%, unresolved_any=True (S1–S5, where reported)
- scale guard (1024 vs 512): PASS

Mechanical gates (thresholds: overlap=100%, key_hit≥99%, abstain≤1%, lib_keys≤24, tie closure, scale guard) fail at all stages. The miner prints rationale “KEY_DRIFT” (UR-lite receipt=a6c042a9b81e; UR-full receipt=933b6846d326), but because overlap_min=100.0% throughout, this run is classified in-paper as closure-limited under stable measured overlap: key_hit and abstain do not satisfy gates, and tie closure remains blocked (tie_rate=50% with tiebreak_success=0% and unresolved cases).

What this enables: This run tests whether increasing discovery budget alone improves closure under an otherwise stable-overlap envelope; it does not, within the reported worst-case metrics. It motivates the next experiments to target closure knobs (recall/candidate truncation and tie closure) rather than transport stability, since the transport proxy remains stable while closure gates remain unsatisfied.

5.6. Run g0013: Supported by Receipts

RCR = 507a1c3ebec44d30ab9d840df25e251820011ac5a6cd096d7deafe6b03e266fe.
Same as g0012 except max_candidates=60 (WL=12, cap_per_key=20, discovery=560000, verify=400000).

Worst-case stage summaries:

- overlap_min=100.0% (S1–S5)
- key_hit_min=88.3% (S2–S4)
- abstain_max=11.7% (S2–S4)
- lib_keys_max=15
- ties: tie_rate_max=50.0%, tiebreak_success_min=0.0%, unresolved_any=True
- scale guard: PASS

Gate outcome: increasing max_candidates (a recall/truncation knob) does not improve worst-case closure metrics under the current regime/canonicalizer. Since overlap_min=100.0%, the paper classifies this as closure-limited rather than drift, even if a printed rationale reports KEY_DRIFT. This is an explicit receipt-backed negative result: the closure deficit observed in g0008/g0010/g0011/g0012 is not primarily explained by max_candidates truncation (within this envelope).

Failure taxonomy (paper): KEY_MISS (primary), TIES_BROKEN (blocking).

What this enables: rules out a plausible “easy” explanation (recall truncation) and tightens the paper’s bottleneck diagnosis: remaining gains likely require (i) canonicalizer+key semantics changes, and/or (ii) a validated tie-closure policy.

5.7. Run g0014: Supported by Receipts

RCR = f29e8b32588ed38fe7fef18c20951079e9cf5eaed93195f95a6b974607573ac4.
Same as g0011 (WL=12, cap_per_key=20, discovery=280000, verify=400000) except canonicalizer parameter sub_budget_pairs=60000 (increased from 15000). All other canonicalizer params unchanged (sub_max_k=6, bce_budget=8000, ple_max_iters=60, big_node_cap=50000). Regimes: R1=ple•bce, R2=dup•taut-heavy•lit-dup, R3=sub•dup.

Worst-case stage summaries

- overlap_min=100.0% (S1–S5)
- key_hit_min=88.3% (S2–S4)
- abstain_max=11.7% (S2–S4)
- lib_keys_max=15
- ties: tie_rate_max=50.0%, tiebreak_success_min=0.0%, unresolved_any=True
- scale guard: PASS

Gate outcome (what changed, what did not)

Increasing sub_budget_pairs (a canonicalizer-strength knob intended to collapse more near-equivalences and improve key reuse) does not improve worst-case closure metrics under the current regime/key semantics. Overlap remains maximally stable, but closure remains limited: key_hit stays below the ≥99 threshold, abstain remains above the ≤1 threshold, and tie closure remains blocking.

This is a receipt-backed negative result: within this operational envelope, the closure deficit observed in g0008/g0010/g0011/g0012 is not primarily explained by subsumption-pair budget being too small (at least up to 60000 under the current sub_max_k and key design).

Failure taxonomy (paper)

KEY_MISS (primary), TIES_BROKEN (blocking).

What this enables (why it’s relevant)

Together with g0013 (which rules out max_candidates as the dominant “easy” recall explanation), g0014 rules out a second plausible “easy” explanation (canonicalizer budget). That tightens the paper’s bottleneck diagnosis: further progress likely requires a change in key semantics / canonical form definition and/or a validated tie-closure policy (not merely more recall or more canonicalizer budget inside the current semantics).

6. Discussion: Defense Thesis and Scope

6.1. Beyond Solver Logs: Receipt-Gated Transport and Closure ($W = I^C$)

A fair critique is that the reported runs resemble solver logs. Here, however, each run is treated as a transport-and-closure receipt produced under an explicit governance kernel — evidence about comparability and closure within a declared operational envelope, not an anecdotal performance report [1].

In this protocol, what “counts” is constrained by declared regimes, budgets, and mechanical gates, with empirical statements traceable to run receipts rather than post-hoc interpretation [1]. This extends [1] by making promotion decisions mechanically auditable: comparability and closure are not rhetorical outcomes, they are gated runtime objects [1].

We summarize the stance as $W = I^C$, using the terms as operational mnemonics rather than philosophical claims. I (Intelligence) denotes the ability to produce stable, reusable invariants under the declared canonicalizer+key semantics and admissible transports (proxied by overlap and scale guards). C (Consciousness) denotes the ability to remain governable and self-auditable at runtime: the system must either close decisions under bounded budgets (key_hit, bounded verification) or explicitly refuse (abstain) with receipt-backed justification, while unresolved ties are treated as failed closure rather than acceptable ambiguity [1]. The exponent emphasizes amplification: intelligence that is not governable (low C) does not scale into usable progress, while increasing C (through tighter closure discipline and auditability) is intended to make the same I operationally effective under declared regimes [1].

This discipline is not cosmetic. SAT-style pipelines are sensitive to distribution, heuristic interactions, and budget effects; without strict controls, summaries can easily become misleading narratives [12,13,17].

By treating abstain and tie closure as first-class gates (instead of silently absorbing them into timeouts or informal uncertainty), the protocol aligns with the reject-option principle: refusal is admissible only when measurable, bounded, and systematically reduced rather than used to mask difficulty at scale [21,22].

In short, the novelty is not claiming a solver breakthrough; it is a receipt-gated methodology that separates invariance capacity (I) from governable closure (C), and counts progress (W) only when intelligence remains effective under audit and closure constraints [1].

6.2. Why These Examples Matter for a Constructive Route

The receipts demonstrate two practically crucial separations for any constructive route:

- Stability does not imply closure: overlap can be perfect while key_hit and abstain remain far from decider gates.
- Refinement is not monotone: deeper canonicalization ($WL=16$ here) can induce measured drift and break scale constraints.

These separations are directly actionable: they define what should be tested next and prevent attributing failure to the wrong cause.

6.3. Tie Behavior as a Blocking Bottleneck

Across the reported runs, tie behavior is consistently adverse: tie_rate reaches 50.0%, tiebreak_success remains 0.0%, and unresolved cases persist. Under the declared tie gate (tie_rate $\leq$ 1% OR (tiebreak_success $\geq$ 99% and unresolved=0)), this blocks promotion even when overlap is stable, by design.

The value of this outcome is methodological: the protocol makes incompleteness explicit and receipt-visible, rather than allowing it to be “smoothed over” by post-hoc interpretation or hidden inside aggregate success rates.

In other words, it prevents narrative drift: a run cannot be described as progress toward a conditional decider if its own receipts show that the system cannot resolve contests between candidates within the declared policy and budgets.

6.4. Gate Rationale Consistency

When overlap receipts show $\text{overlap_min}=100.0\%$, the paper classifies the run's dominant failure mode as closure-limited rather than drift, even if the miner's printed gate rationale says KEY_DRIFT. This consistency rule is required for auditability: failure labels must follow the measured receipts, not a potentially over-broad diagnostic string.

It also aligns with the multitime ledger perspective in [1]: distinct clocks can diverge, so the "transport clock" (measured stability of the current canonicalizer+key semantics) may be ok while the "closure clocks" (key_hit, abstain, ties, bounded verification) are not. Under this discipline, stable overlap is treated as evidence of comparability, but it never substitutes for closure; promotion remains gated by the closure receipts.

6.5. What This Does Not Support

The current receipts do not support a candidate decider that meets the paper's mechanical gates ($\text{key_hit} \geq 99\%$, $\text{abstain} \leq 1\%$, tie closure, and bounded growth under the scale guard).

The contribution is therefore not a claimed breakthrough, but a disciplined, testable program: a receipt-gated transport compilation protocol that makes comparability and closure explicit, and early run receipts that *localize* bottlenecks in a way that is mechanically actionable. Across the reported envelopes, receipts repeatedly separate (i) closure deficits under stable measured overlap, (ii) non-monotone refinement effects where a "stronger" knob setting can degrade overlap or violate scale guards, and (iii) persistent tie blockage that prevents promotion by design.

The result is a concrete iteration agenda: change one admissible knob at a time, re-run under the same declared semantics, and accept "progress" only when the gates improve on receipts rather than by narrative reinterpretation. [1,12,13,17]

7. Author Intent, Compute Constraints, and Collaboration Note

This work is protocol-forward and idea-forward. It does not attempt to claim the Clay Prize or assert a proof of $P = NP$. The goal is to publish a constructive route and a governance method: define admissible regimes, compile transformations into comparable artifacts, and publish receipts that make failure modes reproducible.

The experiments reported here were run on limited personal compute. Large-scale validation, broader regime coverage, and high-volume stress tests are expected to require substantially more resources.

Accordingly, the paper is written to be scalable by others: it specifies knobs, gates, receipts, and rollback logic so that independent groups — academic labs or industrial research teams — can replicate and extend the protocol under stronger compute.

The author's stance is explicit: the program is motivated by belief that a constructive route to $P = NP$ may exist, but the present work focuses on publishing a disciplined framework and early receipts, and invites collaboration and continuation by groups with greater resources.

8. Limitations and Threats to Validity

8.1. Scope and Distribution

The empirical evidence reported here is conditional on the miner's staged design (S1–S5), the declared regimes (R1–R3), and the generated instance families bound to each RunConfigReceipt (RCR). In other words, the protocol is not "testing SAT in general"; it is testing a declared operational envelope: admissible transport operators, canonicalization budgets, and verification policies, all

governed by the multitime receipt discipline of [1]. Accordingly, the failure modes observed are receipts about behavior under this envelope and cannot be promoted into general claims about SAT or NP-completeness in the reduction-theoretic sense [4,5], as synthesized in standard references [7–9]. Any broader inference requires replication across alternative generators, admissible τ -classes, and size ranges, while preserving the same receipt and gate semantics [1].

8.2. *Overlap Is a Proxy, Not “Reason Transport” Itself*

The `overlap_min` metric should be read strictly as an instrumentation result: it reports invariance under the run’s declared canonicalizer+key semantics, not invariance of SAT or NP-completeness itself. Concretely, it answers: “given this canonical form, this key construction, and this admissibility corridor, does the transport signature remain consistent across the compared sets?” This is valuable because it makes transport stability mechanically auditable inside the protocol [1], but it is not a universal definition of “reason transport.” A different canonicalizer, a different key semantics, or a different admissibility corridor could change overlap behavior. For this reason, `overlap_min` is treated throughout as a proxy tied to declared instrumentation, not as a semantic invariant of the underlying decision problem. [1]

8.3. *Tie-Breaking Remains Incomplete and Blocks Promotion*

Across receipt-backed runs reported so far, `tie_rate` remains high while `tiebreak_success` is zero and unresolved ties persist. Under the mechanical gates, this blocks promotion even in runs where overlap is stable. Conceptually, ties are contestability failures: multiple competing candidates cannot be resolved by the declared policy within budget, so closure cannot be asserted. This is not just an engineering nuisance; it affects the credibility of a “conditional decider” because persistent unresolved ties behave like unbounded abstention at scale. Until tie closure is both effective and validated (i.e., demonstrably non-cheating and not hiding exponential work), promotion is appropriately blocked by design [1].

8.4. *Library Dynamics and Overfitting by Fragmentation*

The library is the mechanism by which transport compilation becomes a candidate route: canonicalize to a key, then reuse previously compiled structure. The same mechanism creates a risk of memorization by fragmentation: apparent improvements in `key_hit` can be achieved by exploding the key space so that instances become nearly unique, sacrificing generalization and potentially hiding complexity in library growth. This is why `lib_keys` bounds and the scale guard are treated as necessary constraints: they aim to keep the library behaving like a reusable index rather than an uncontrolled lookup table. As experiments scale, stricter controls may be required to ensure that closure improvements reflect reusable structure rather than uncontrolled key proliferation [1,9].

8.5. *Abstain Gating Is Principled but Can Mask Difficulty if Not Tightly Governed*

Allowing abstain is not a weakness by itself; it is a safety feature aligned with reject-option and selective classification principles: when confidence or closure is insufficient, refusal can reduce error at the cost of coverage [21,22]. However, in a decider-oriented setting, persistent abstain at scale is effectively non-decision, and can mask hard instances unless aggressively gated (abstain thresholds, no-reopen discipline, and explicit recovery-to-okay requirements) as enforced by the kernel methodology [1]. This paper therefore treats abstain as admissible only when it is measurable, bounded, and systematically reduced through controlled knob iteration [1,21,22].

8.6. *Verification Artifacts Are Bounded but Not Yet Externally Checkable Proof Logs*

The protocol includes bounded verification as a runtime step, and the SAT community has a well-established tradition of making solver outputs checkable through proof-oriented artifacts and proof checking workflows [12–17]. However, the receipts shown here do not yet include externally

checkable proof logs (DRAT/LRAT/FRAT-style) that would allow independent third parties to replay and validate correctness of critical sub-claims, especially for UNSAT-related evidence [15,16]. Integrating standardized proof logs would strengthen reproducibility, separate “the miner asserted closure” from “closure is certified by an external checker,” and make the audit story match the strongest norms of SAT engineering and proof checking [12–17].

9. Conclusion

This paper advances the program of multitime barriers in [1] from diagnosis to an operational compilation protocol: Multitime Transport Compilation. Rather than treating reductions and invariants as purely conceptual objects, we treat them as governed runtime transformations whose comparability must be made mechanical and auditable.

The miner implements a conditional candidate-decider pipeline — canonicalize → key → library lookup → bounded verification — under a Temporal State Machine discipline (admissibility corridors, receipts, no-reopen, abstain gating, and recovery-to-okay feasibility).

This operational stance is motivated by the standard NP-completeness and reduction framework initiated by Cook and Karp [4,5] and consolidated in core complexity references [7–9], while explicitly acknowledging known barrier patterns such as natural proofs and proof-complexity constraints [10,11,23,24]. The objective is not to claim a theorem, but to publish a protocol whose claims are constrained by receipts and therefore remain contestable and reproducible.

Receipt-backed runs reported here establish two concrete, falsifiable separations. First, transport stability under the current metric can hold while closure still fails: with WL rounds set to 12, overlap is measured as stable (overlap_min=100%), yet key_hit remains far below the mechanical gate and abstain remains far above it, and unresolved ties persist. This shows that the “transport clock” (as proxied by overlap under the chosen canonicalizer+key semantics) can be OK while the “closure clocks” are not, turning the problem into a closure deficit rather than measured drift. Second, refinement is not monotone: increasing WL depth (WL=16 in the reported configuration) correlates with measured overlap loss and a scale-guard violation, providing a receipt-backed counterexample to the assumption that stronger canonicalization necessarily improves comparability. This supports rollback as a governance operator rather than an ad hoc retreat, consistent with the kernel logic developed in [1].

The paper does not claim $P = NP$. Instead, it proposes a constructive and auditable research route: make comparability mechanical (overlap and scale guards), classify failure modes using receipt-backed metrics rather than narrative labels, and iterate one admissible knob at a time toward closure (higher key_hit, lower abstain, resolved ties) under declared regimes.

The broader intellectual posture is aligned with the selective classification and reject-option literature — abstain is allowed but must be measured and gated [21,22] — and with the SAT engineering tradition where solver behavior must be made inspectable and checkable via artifacts and proof-oriented tooling [12–17]. The multitime epistemic ledger framing $P(\tau) + NP(\tau) = 1$ is used only as a governance identity for credit assignment under a declared window τ , extending the co-evolution ledger perspective in [2] into a multitime setting while remaining consistent with the closure archetype emphasized by the Circle of Equivalence motif in [3].

Finally, the paper’s canonicalization/key strategy is conceptually adjacent to graph canonicalization and refinement heuristics (including WL-style refinement), while not claiming equivalence to graph isomorphism results; the references provide context for why canonical forms and refinement can be powerful yet nontrivial [18–20].

License and Declarations: License. © 2025 Rogério Figurelli. This preprint is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You are free to share (copy and redistribute the material in any medium or format) and adapt (remix, transform, and build upon the material) for any purpose, even commercially, provided that appropriate credit is given to the author.

A link to the license and an indication of any changes made must be included when reusing the material. The full license text is available at: <https://creativecommons.org/licenses/by/4.0/>.

Author Contributions. This work was conceived, structured, and written in full by Rogério Figurelli. No other authors qualify for authorship under the ICMJE recommendations. Contributions from institutions, collaborators, or organizations that supported background research, data organization, or conceptual framing are acknowledged separately in related project documentation.

Acknowledgments. The author(s) acknowledge prior foundational work and intellectual contributions that have shaped the conceptual and methodological background of this manuscript. Relevant influences include advances in causal reasoning and modeling, developments in artificial intelligence safety and governance, insights from decision sciences and behavioral theories, reinforcement learning frameworks, and broader discussions in information ethics. The author(s) also recognize the value of open-innovation initiatives, collaborative research environments, and public knowledge projects that provide resources, context, and inspiration for ongoing inquiry. Visualizations, figures, and supporting materials were prepared using openly available tools and may have benefited from institutional or laboratory collaborations aimed at promoting reproducibility and transparency.

Data Availability. No proprietary or sensitive datasets were used in the preparation of this manuscript. All conceptual models, equations, and figures were generated as part of the author(s)' own research process. Any supplementary materials, such as simulation data, analytical scripts, or graphical code used to produce figures, are available upon request and may be deposited in an open-access repository in accordance with FAIR data principles. Authors encourage reuse and adaptation of such resources, provided appropriate credit is given.

Ethics Statement. This research does not involve experiments with humans, animals, or plants and therefore did not require ethics committee approval. The work is conceptual, methodological, and computational in scope. Where references to decision-making domains such as healthcare, finance, and retail are made, they serve as illustrative vignettes rather than analyses of proprietary or sensitive datasets.

Conflicts of Interest. The author declares no conflicts of interest. There are no financial, professional, or personal relationships that could inappropriately influence or bias the work presented.

Use of AI Tools. AI-assisted technology was used for drafting, editing, and structuring sections of this manuscript, including the generation of visual prototypes and narrative expansions. In accordance with Preprints.org policy, such tools are not considered co-authors and are acknowledged here as part of the methodology. All conceptual contributions, final responsibility, and authorship remain with the author, Rogério Figurelli.

Withdrawal Policy. The author understands that preprints posted to Preprints.org cannot be completely removed once a DOI is registered. Updates and revised versions will be submitted as appropriate to correct or expand the work in response to community feedback.

References

1.

R. Figurelli, "Multitime Barriers for P vs NP: Why Some Reasons May Not Travel in Polynomial Time," Preprints, 2026, Preprints ID 195943, doi:10.20944/preprints202601.2011.v1.

2.

R. Figurelli, "What if P + NP = 1? A Multilayer Co-Evolutionary Hypothesis for the P vs NP Millennium Problem," Preprints, 2025, doi:10.20944/preprints202507.0640.v1.

3.

R. Figurelli, "Like Archimedes in the Sand: Visual Reasoning and Epistemic Forms in Cub ∞ Machines," Preprints, 2025, doi:10.20944/preprints202507.0937.v1.

4.

S. A. Cook, "The Complexity of Theorem-Proving Procedures," in Proc. STOC, 1971, pp. 151–158.

5.

R. M. Karp, "Reducibility Among Combinatorial Problems," in Complexity of Computer Computations, 1972, pp. 85–103.

6. L. A. Levin, "Universal Sequential Search Problems," *Problemy Peredachi Informatsii*, vol. 9, no. 3, 1973.
7. M. R. Garey and D. S. Johnson, *Computers and Intractability*. W. H. Freeman, 1979.
8. C. H. Papadimitriou, *Computational Complexity*. Addison-Wesley, 1994.
9. S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge Univ. Press, 2009.
10. A. A. Razborov and S. Rudich, "Natural proofs," *JCSS*, vol. 55, no. 1, pp. 24–35, 1997.
11. E. Ben-Sasson and A. Wigderson, "Short proofs are narrow — Resolution made simple," *JACM*, vol. 48, no. 2, pp. 149–169, 2001.
12. N. Eén and N. Sörensson, "An extensible SAT-solver," in *SAT*, 2003.
13. M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver," in *DAC*, 2001, pp. 530–535.
14. A. Biere, "Bounded model checking," in *TACAS*, 2003.
15. M. J. H. Heule, W. A. Hunt Jr., and N. Wetzler, "Trimming while checking clausal proofs," in *FMCAD*, 2013.
16. M. J. H. Heule and O. Kullmann, "The science of brute force," *Commun. ACM*, vol. 60, no. 8, pp. 70–79, 2017.
17. L. Zhang and S. Malik, "The quest for efficient Boolean satisfiability solvers," in *CAV*, 2002.
18. B. Weisfeiler and A. A. Leman, "The reduction of a graph to canonical form and the algebra which appears therein," *NTI*, 1968.
19. L. Babai, "Graph isomorphism in quasipolynomial time," in *STOC*, 2016, pp. 684–697.
20. B. D. McKay and A. Piperno, "Practical graph isomorphism, II," *J. Symb. Comput.*, vol. 60, pp. 94–112, 2014.
21. C. K. Chow, "On optimum recognition error and reject tradeoff," *IEEE Trans. Inf. Theory*, vol. 16, no. 1, pp. 41–46, 1970.
22. Y. Geifman and R. El-Yaniv, "Selective classification for deep neural networks," in *NeurIPS*, 2017.
23. J. Krajíček, *Proof Complexity*. Cambridge Univ. Press, 1995.
24. S. Jukna, *Boolean Function Complexity*. Springer, 2012.
25. J. Håstad, "Some optimal inapproximability results," *JACM*, vol. 48, no. 4, pp. 798–859, 2001.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.