AUTOTW-ASP: AUTOMATIC LOW-TREEWIDTH ENCODING SYNTHESIS AND BACKEND ROUTING FOR NEUROSYMBOLIC ASP

Anonymous authors
Paper under review

ABSTRACT

Neurosymbolic answer set programming (ASP) systems often achieve exact semantics only after manual encoding redesign, which makes runtime gains difficult to scale beyond expert-curated benchmarks. We present AutoTW-ASP, a hybrid exact-inference method that automates two coupled decisions: (i) whether a candidate rewrite of a grounded ASP program should be accepted under explicit semantic guards, and (ii) whether each instance should be executed with stable-model enumeration or compilation-based counting. The method combines structure-aware proxies, a guarded rewrite objective, and uncertainty-gated routing. On six benchmark families and nineteen seeds, AutoTW-ASP preserves semantic equivalence for guarded rewrites and reduces median runtime relative to fixed exact baselines, while exposing where calibration and counterexample detection remain brittle. The strongest empirical signal is exactness preservation with runtime improvement against naive exact baselines; the weakest signal is universal closure of strict threshold targets for router-envelope violations and cache counterexample recall. These bounded findings support the central claim that exact neurosymbolic optimization can be automated without relaxing stable-model correctness, while making clear which components still limit generalization.

1 INTRODUCTION

Exact symbolic inference remains central in settings where model outputs must satisfy hard constraints rather than soft plausibility criteria. Examples include scientific hypothesis filtering, reliability-oriented planning, and verification-facing decision support, where one semantic violation can invalidate an otherwise high-performing model. Neurosymbolic ASP is attractive in these settings because it can encode nontrivial structure with declarative constraints while still integrating neural perception. However, this same expressiveness creates a systems challenge: as grounding size, constraint interaction, and structural irregularity increase, exact solving cost can dominate the entire learning loop. In practice, teams therefore face an undesirable choice between preserving exactness with high runtime cost and introducing approximations that may improve throughput but weaken guarantees (Yang et al., 2020; Skryagin et al., 2022; 2023; Eiter et al., 2024; Acharya & Song, 2025; Colelough & Regli, 2025).

The core methodological issue is not that relevant theory is missing. Treewidth-sensitive analyses, cycle-aware reductions, and compilation-map results already explain why structural properties and representation choices govern tractability (Bodlaender, 1993; Darwiche et al., 2002; Huang et al., 2005; Eiter et al., 2021; Hecher et al., 2021; 2023b;a). The missing layer is operational: how to convert those principles into an optimization procedure that is automatic, semantics-aware, and empirically auditable in modern neurosymbolic training pipelines. Manual redesign can recover large gains on selected tasks, but it does not scale to diverse workloads, and it obscures causal attribution because improvements are intertwined with expert craftsmanship. Our framing treats this as a scientific modeling problem: define explicit decision variables, define constraints that encode semantic admissibility, then test each decision surface against measured outcomes and calibrated uncertainty.

This emphasis also clarifies why we do not frame the paper as a competition between one exact backend and one approximation baseline. Different backends win in different structural regimes, and structure itself can change after rewriting. A realistic optimizer therefore must reason over a sequence: transform the program under semantic guards, estimate post-transform structure, route execution with uncertainty awareness, and preserve traceability when decisions fail. By making these stages explicit, we can attach theorem-level claims to the semantic core, measurement-driven claims to the routing layer, and caveat-driven claims to calibration and cache diagnostics. The resulting manuscript is intentionally hybrid: formal where guarantees are available, empirical where calibration constants and regime effects must be observed.

Finally, the motivation is cross-domain even though the benchmarks are neurosymbolic ASP. Any workflow that combines symbolic constraints with learned components faces analogous tensions among correctness, runtime, and uncertainty. A guarded-optimization view of program transformation and backend selection is therefore relevant beyond the immediate task families, provided that domain-specific semantics and audit interfaces are adapted appropriately. We state this as bounded transferability rather than universal applicability: the paper offers a reproducible framework for exactness-preserving optimization under explicit assumptions, together with clear evidence of where those assumptions remain insufficient.

Neurosymbolic learning systems that combine neural perception with logic-level reasoning offer a compelling path to models that are interpretable, constraint-aware, and robust to partial supervision (Manhaeve et al., 2018; Yang et al., 2020; Manhaeve et al., 2021a; Winters et al., 2022; Skryagin et al., 2022; 2023; Geh et al., 2024). In these systems, the symbolic layer is not a decorative post-processing step: it changes the feasible hypothesis space, training gradients, and error modes. For ASP-grounded neurosymbolic pipelines in particular, exact inference quality depends jointly on semantic correctness and solver efficiency, which has made runtime a central bottleneck (Yang et al., 2020; Skryagin et al., 2022; 2023; Eiter et al., 2024; 2021).

A recurring empirical pattern is that strong speedups are often obtained only after expert-driven program rewrites, such as hand-designed carry variants or cycle-aware transformations. This dependency creates a methodological gap: exact-inference improvements are real, but they are difficult to transfer across tasks because the optimization knowledge is embedded in manual encoding craftsmanship rather than an auditable algorithm. Prior work has established structural determinants of complexity, especially treewidth and cycle structure (Bodlaender, 1993; Eiter et al., 2021; Hecher et al., 2021; 2023b;a), and has clarified compilation tradeoffs in terms of tractable query classes and representation choice (Darwiche et al., 2002; Huang et al., 2005; Muise et al., 2012; Jha & Suciu, 2012). Yet a complete pipeline that unifies semantics-preserving rewrite acceptance, backend crossover routing, and uncertainty-aware fallback for exact neurosymbolic training remains underdeveloped.

This paper addresses that gap with AutoTW-ASP, an exact-first optimization layer for neurosymbolic ASP. The method is designed for settings where correctness under stable-model semantics is non-negotiable, and where runtime can be improved only if structural burden is reduced or backend selection is calibrated instance by instance. The key point is not to claim unconditional superiority of one backend or one transformation family; instead, we formalize decision objects that make optimization auditable, falsifiable, and reproducible.

The broader significance extends beyond one benchmark suite. Exact constraint reasoning appears in scientific discovery workflows, verification-heavy decision support, and safety-critical symbolic planning, where semantic drift is usually more harmful than moderate runtime loss. A method that explicitly optimizes under exactness constraints, while exposing its failure boundaries, is therefore relevant to cross-domain neurosymbolic deployment and not only to ASP benchmark engineering.

Contributions.

- We define a guarded rewrite objective for grounded ASP and prove a compositional semantic invariance theorem that separates sourced transformation conventions from manuscript-defined optimization choices.
- We derive a two-action regret envelope for exact backend routing under additive runtime-prediction error, and we formalize an uncertainty-gated fallback mechanism that bounds decision risk.
- We implement and evaluate a hybrid pipeline on six benchmark families with nineteen seeds, reporting semantic audits, runtime behavior, calibration outcomes, and explicit negative results.
- We provide claim-linked evidence closure with theorem audits, uncertainty intervals, and reproducibility details, including where empirical thresholds remain unmet and why those gaps matter.

2 Related Work and Novelty Boundary

The literature map motivating this work has three interacting layers. The first layer is neurosymbolic modeling, where symbolic constraints improve consistency and interpretability but expose computational bottlenecks in exact reasoning (Raedt & Kimmig, 2015; Manhaeve et al., 2018; Yang et al., 2020; Manhaeve et al., 2021a; Winters et al., 2022; Skryagin et al., 2022; 2023; Geh et al., 2024). The second layer is compilation and counting theory, which explains how representational choices shift computational burden across preprocessing and query time (Darwiche et al., 2002; Huang et al., 2005; Chavira et al., 2008; Muise et al., 2012; Fierens et al., 2012; Jha & Suciu, 2012; FIERENS et al., 2014; Dubray et al., 2023; Eiter et al., 2024). The third layer is structure-aware ASP complexity, which identifies graph-structural signatures that predict when transformations can reduce practical cost and when lower-bound regimes remain unavoidable (Bodlaender, 1993; Eiter et al., 2021; Hecher et al., 2021; 2023b;a).

Prior studies typically optimize within one layer at a time. Framework papers establish semantics and learning interfaces. Compilation papers optimize representational tractability. Structural ASP papers characterize bounds and reduction effects. What remains less developed is cross-layer coupling under exactness constraints: if a rewrite changes structure and a router consumes structural features, the validity of routing decisions depends on rewrite admissibility, and the usefulness of rewriting depends on downstream routing quality. This dependency is central to our novelty boundary. We build on existing semantics and counting formalisms, but we introduce a coupled decision process in which guarded rewrites and uncertainty-gated routing are jointly analyzed and empirically audited.

Our contrast with approximation-oriented pipelines is similarly scoped. Approximation methods are often effective for throughput and can be preferable in resource-constrained settings (Manhaeve et al., 2021b; van Krieken et al., 2022; Skryagin et al., 2023). We therefore treat them as meaningful comparators and regime-specific references, not as strawman baselines. The contribution is not that approximation is always inferior; the contribution is to make exact-first optimization competitive in more regimes by automating structure-aware transformations and calibrated backend selection. This framing avoids overclaiming and aligns with mixed empirical outcomes, where some threshold targets are met and others are not.

A final boundary concerns evaluation style. Survey and benchmark-critique literature has shown that selective reporting can inflate confidence when uncertainty and failure modes are underreported (Gibaut et al., 2023; dAvila Garcez & Lamb, 2023; Marra et al., 2024; Acharya & Song, 2025; Manginas et al., 2025; Liang et al., 2025; Colelough & Regli, 2025). We explicitly incorporate this lesson by requiring claim-linked artifact closure and by treating negative results as first-class evidence. In this manuscript, unsupported edges are part of the result, not deferred anomalies.

2.1 Neurosymbolic ASP and Probabilistic Logic

Neural-probabilistic logic systems have shown that symbolic constraints can improve consistency and controllability relative to unconstrained neural baselines (Raedt & Kimmig, 2015; Manhaeve et al., 2018; Yang et al., 2020; Manhaeve et al., 2021a; Winters et al., 2022; Skryagin et al., 2022). This line includes deep probabilistic logic programming, ASP-integrated neural predicates, and scalable approximations that trade strict exactness for throughput (Manhaeve et al., 2021b; van Krieken et al., 2022; Skryagin et al., 2023; Geh et al., 2024). The strength of this literature is expressive modeling under formal semantics. Its central limitation, for exact training settings, is that inference bottlenecks are often treated as engineering overhead rather than first-class optimization variables.

2.2 Compilation and Counting Foundations

Knowledge compilation has long emphasized the succinctness–tractability frontier: target representations determine which post-compilation queries are practical (Darwiche et al., 2002; Huang et al., 2005; Muise et al., 2012; Jha & Suciu, 2012). Weighted and algebraic model counting frameworks provide unifying reductions for exact probabilistic inference and logic-level reasoning (Chavira et al., 2008; Fierens et al., 2012; FIERENS et al., 2014; Dubray et al., 2023; Cunnington et al., 2023; Barbiero et al., 2023; Eiter et al., 2024). These foundations explain why compilation can dominate in reuse-rich regimes while becoming unfavorable in highly variable or low-reuse workloads. They also motivate crossover routing as a principled control problem rather than a heuristic switch.

2.3 Structure-Aware ASP Theory

Treewidth-aware analyses and cycle-sensitive reductions show both opportunity and limits. Structural preprocessing can materially improve runtime on meaningful classes, but ETH-conditioned hardness results prevent unconditional asymptotic optimism (Eiter et al., 2021; Hecher et al., 2021; 2023b;a). Practical counting systems confirm that structure-aware variants can deliver major gains, while also showing distribution-dependent behavior (Eiter et al., 2024; 2021). The unresolved issue is not whether structure matters; it is how to convert structural signals into robust, automated decisions without silently violating semantics.

2.4 Robustness, Generalization, and Evaluation Gaps

Recent surveys emphasize fragility under distribution shift, calibration uncertainty, and benchmark over-specialization in neuro-symbolic evaluation (Gibaut et al., 2023; dAvila Garcez & Lamb, 2023; Marra et al., 2024; Acharya & Song, 2025; Manginas et al., 2025; Liang et al., 2025; Colelough & Regli, 2025). This motivates two design requirements for our setting: first, results should be reported with uncertainty and explicit caveats rather than single best numbers; second, failure cases should be integrated into the core argument, not deferred to supplementary notes. Our novelty boundary is therefore hybrid: we do not claim a new semantics for ASP, nor a new counting formalism; we claim a

new coupling of guarded rewriting and uncertainty-aware routing with formal guarantees and claim-linked empirical audits.

The broader neurosymbolic ecosystem further clarifies this design choice. Program-induction and logic-learning work has repeatedly shown that symbolic bias can reduce sample complexity while introducing new failure modes tied to incorrect or brittle symbolic abstractions (Cropper & Dumancic, 2022; Sen et al., 2022; Shindo et al., 2024). This duality appears across both differentiable and non-differentiable formulations: stronger reasoning structure improves compositional behavior when assumptions match the data regime, but can produce hard failures under shortcut exploitation or mismatch between learned and executable semantics (Marconato et al., 2023b; Liu et al., 2023). In this manuscript, guarded rewrites serve as an explicit control mechanism for that duality: we allow structural optimization only when semantic admissibility checks pass, and we convert uncertain optimization gains into fallback decisions instead of unconditional rewrites.

A second cross-domain lesson comes from system-level comparisons that span language, perception, and symbolic planning. Benchmark studies and surveys report that nominal gains often depend on narrow benchmark artifacts, while reliability under shift depends on explicit uncertainty controls and intervention policies (Hamilton et al., 2022; Vermeulen et al., 2023; Shakarian et al., 2023; Marconato et al., 2023a; Wan et al., 2024). This is directly relevant to exact ASP optimization: a method that looks strong on average but fails silently in high-variance structural regimes is not suitable for exactness-sensitive training. Our choice to foreground violation-rate and counterexample diagnostics is therefore not ancillary; it follows this broader reliability literature by treating robustness diagnostics as primary scientific evidence rather than implementation detail.

Finally, recent programming-language and deployment-oriented neurosymbolic systems reinforce the importance of interface design between learned and symbolic modules. Frameworks for neurosymbolic programming, quantified probabilistic logic, and ASP-assisted language understanding demonstrate that representational interfaces can dominate both runtime and failure behavior (den Broeck et al., 2011; Smet et al., 2023; Rajasekharan et al., 2023; Li et al., 2023; Jung et al., 2024). AutoTW-ASP fits this perspective by modeling rewrite selection and backend routing as interface-level control problems. The contribution is not a new solver family; it is a new decision layer that makes solver selection and structure transformation auditable, reproducible, and testable under explicit acceptance criteria.

3 Problem Setting and Assumptions

We emphasize four mathematical objects that define the optimization problem. First, the decision variable for program transformation is a finite rewrite sequence r, constrained to feasible set $\mathcal{F}(\Pi, O)$ by guard predicates that encode admissibility under observation context. Second, the routing decision is a backend selector over $\mathcal{B} = \{\text{enum}, \text{comp}\}$ conditioned on post-rewrite features. Third, the objective landscape combines structure proxies and runtime surrogates for rewrite choice, then realized runtime for routing quality. Fourth, the correctness criterion is exact stable-model invariance plus label invariance under fixed observation semantics.

This decomposition separates what is directly optimized from what is guaranteed. Rewrite optimization minimizes surrogate objective equation 2, not unknown true runtime. Semantic invariance is guaranteed only for guard-admissible rules satisfying local equivalence assumptions. Routing is optimized using predicted costs in equation 3 and equation 4, while regret guarantees depend on bounded prediction error as formalized later. This distinction is important for interpreting evidence: theorem statements certify conditional properties, whereas empirical results evaluate how often those conditions hold in executed workloads.

To make the feasible-set definition operational, we distinguish three guard classes. Syntactic guards reject transformations that violate rule-shape constraints for supported templates. Semantic guards enforce local equivalence obligations derived from prior cycle-aware and reduction analyses. Audit guards require bounded empirical checks on representative grounded subsets before global acceptance. This three-tier design is manuscript-specific: existing literature motivates each ingredient, but the combined admissibility policy is newly defined here. The result is a conservative acceptance mechanism that prioritizes exactness even when it sacrifices potential speed gains.

We also formalize regime-level utility because aggregate means can mask structural heterogeneity. Score equation 5 uses normalized latency, task utility, exactness, and uncertainty terms with explicit weights. The purpose is not to claim a universal scalar objective for all applications; it is to create a transparent comparison functional whose components can be inspected and stress-tested. Regime-conditioned dominance statements therefore remain local to the chosen weighting and normalization conventions, and we report this locality explicitly in both methods and appendix.

Finally, assumptions are chosen to match achievable evidence in this phase. We assume finite grounded instances, deterministic label extraction from stable-model sets, and auditable runtime traces for both exact backends. We do

not assume complete coverage of every arithmetic rewrite class, globally calibrated uncertainty under all ablations, or universal threshold closure for all metrics. Those unresolved items are tracked as explicit gaps and influence claim support status.

We consider grounded normal ASP programs Π with observation constraints O, following stable-model semantics commonly used in NeurASP-style formulations (Yang et al., 2020; Skryagin et al., 2022; Eiter et al., 2024). Let $\mathrm{SM}(\Pi, O)$ denote the stable-model set under observation O, and let $y(\Pi, O)$ be a deterministic label extraction map from this set. Exactness requires preserving both the stable-model set and downstream labels for accepted transformations. In this manuscript, the constrained rewrite objective and routing policy are the new formal components built on that semantic base.

The method operates over a finite rewrite library $\mathcal{R} = \{R_1, \ldots, R_{|\mathcal{R}|}\}$ containing local transformations inspired by prior structure-aware reductions (Eiter et al., 2021; Hecher et al., 2023a; Eiter et al., 2024). For each candidate rule R and program state, a guard predicate $\mathrm{Guard}(R, \Pi, O) \in \{0, 1\}$ decides whether the rewrite is admissible. We define the feasible rewrite set

$$\mathcal{F}(\Pi, O) = \{r = (R_1, \ldots, R_m) : m \le m_{\max},\ \mathrm{Guard}(R_j, \Pi^{r<j}, O) = 1\ \forall j\}, \tag{1}$$

where $\Pi^{r<j}$ is the intermediate program after applying the first $j-1$ rules.

Following structure-aware optimization practice, we combine a width proxy $\widehat{k}(\Pi)$, a cycle burden proxy $\widehat{\sigma}(\Pi)$, and an exact-runtime proxy $\widehat{C}_{\text{exact}}(\Pi, O)$ into a constrained objective:

$$J(r) = \alpha\widehat{k}(\Pi^r) + \beta\widehat{\sigma}(\Pi^r) + \gamma\widehat{C}_{\text{exact}}(\Pi^r, O), \quad r \in \mathcal{F}(\Pi, O), \tag{2}$$

with nonnegative weights α, β, γ.

After rewrite selection, backend routing chooses between exact enumeration and exact compilation/counting, $\mathcal{B} = \{\text{enum}, \text{comp}\}$. For instance i, predicted cost is modeled as

$$\widehat{T}_i(b) = T_i(b) + e_i(b), \qquad b \in \mathcal{B}, \tag{3}$$

with true runtime $T_i(b)$ and prediction error $e_i(b)$. The router decision is

$$g_\phi(x_i) = \arg\min_{b \in \mathcal{B}} \widehat{T}_i(b), \tag{4}$$

where x_i includes structural and reuse features.

For regime-level comparison between exact-first and approximation-oriented policies, we define a pre-registered utility score

$$S_{m,s} = w_t\,\widetilde{T}_{m,s} + w_a\,\widetilde{A}_{m,s} + w_e\,\widetilde{E}_{m,s} + w_u\,\widetilde{U}_{m,s}, \tag{5}$$

where $\widetilde{T}, \widetilde{A}, \widetilde{E}, \widetilde{U}$ are normalized latency, task utility, exactness, and uncertainty terms for method m in structural regime s. Exact-first dominance in regime s is represented by $\Delta S_s = S_{\text{exact},s} - S_{\text{approx},s} > 0$.

Assumptions. The formulation assumes finite grounded programs, decidable guards, deterministic label extraction from stable models, and comparable runtime traces across exact backends. It does not assume global dominance of one backend or one rewrite family. This scope mirrors prior structural and compilation analyses while making the manuscript-specific objectives explicit (Darwiche et al., 2002; Huang et al., 2005; Eiter et al., 2024; 2021; Hecher et al., 2023a).

4 Methodology: AutoTW-ASP

AutoTW-ASP is designed as a constrained decision pipeline rather than a monolithic optimizer. At a high level, each instance passes through five stages: structural feature extraction, guarded rewrite proposal, feasibility and semantic audit filtering, uncertainty-aware backend routing, and cache-aware execution logging. The ordering is deliberate. Structural estimates must be recomputed after accepted rewrites because routing decisions should depend on transformed program structure, not on stale pre-transform statistics. Likewise, uncertainty and fallback logic should be applied after route prediction to distinguish model uncertainty from optimization intent.

The guarded rewrite engine uses objective equation 2 to rank candidate sequences under feasibility constraints equation 1. We adopt weighted proxies because direct optimization of true end-to-end runtime would require expensive inner-loop solver calls for each candidate sequence. Proxy weighting is therefore an explicit approximation choice.

Rather than hide this approximation, we instrument audit logs so that mismatches between proxy preference and realized runtime become measurable artifacts. This is one reason the method includes separate runtime and semantic tables: correctness and efficiency are jointly important but diagnostically distinct.

Routing combines cost prediction and risk control. Predicted backend times from equation 3 feed decision rule equation 4. Uncertainty estimates from held-out calibration residuals then determine whether fallback is activated. The fallback policy is intentionally conservative and regime-aware: when uncertainty exceeds threshold, the system selects the backend with better recent tail behavior in the current structural bin. This is not theoretically optimal in all conditions, but it produces stable behavior under distributional irregularity and yields interpretable failure signatures when calibration deteriorates.

Cache design is treated as part of correctness, not only efficiency. Canonicalized keys combine normalized structural signatures with guard metadata so that identical optimization contexts map to identical cache entries. However, key canonicalization alone does not establish reliability. We therefore pair cache evaluation with replay-based counterexample diagnostics: if a suspicious hit occurs, replay traces test whether semantic drift can be reconstructed. This split between false-hit control and counterexample sensitivity explains why cache results can be mixed even when false-hit counts are low.

The algorithm in Algorithm 1 summarizes the end-to-end flow. Importantly, each step emits audit metadata so claims can terminate in concrete evidence. Rewrite acceptance emits semantic checks, routing emits uncertainty and violation traces, and caching emits key-consistency plus replay diagnostics. This instrumentation is integral to the method, not supplementary engineering, because the paper's central contribution is auditable exactness-preserving optimization.

4.1 Architecture Overview

AutoTW-ASP has four modules: (i) a guarded rewrite engine, (ii) a semantic-equivalence audit interface, (iii) an uncertainty-aware exact backend router, and (iv) a cache manifest with canonicalization and replay checks. The rewrite engine optimizes equation 2 over feasible sequences defined by equation 1. The router then minimizes predicted runtime via equation 4 while monitoring uncertainty and activating fallback when confidence is low.

The design rationale is that rewrite and routing are coupled but not identical. Rewrites alter structural proxies and therefore change crossover behavior; routing then decides backend execution under those updated structural conditions. Separating these decisions increases auditability: semantic violations are attributed to rewrite admission, while regret violations are attributed to routing calibration.

4.2 Guarded Rewrite Acceptance

Each rewrite candidate is evaluated with three checks before acceptance: syntactic preconditions, guard satisfiability, and bounded semantic audit on representative grounded subsets. If any check fails, the system rejects the candidate and retains the prior program. This policy prioritizes exactness over aggressive optimization, consistent with exact ASP objectives.

The objective in equation 2 uses weighted proxies rather than oracle runtimes. This choice is practical and theoretically transparent: the optimizer is explicit about what it can measure and where approximation enters. In our experiments, this conservative policy preserved semantic equivalence for guarded rewrites and eliminated drift seen in unguarded variants.

4.3 Routing with Uncertainty-Gated Fallback

For each rewritten instance, the router predicts runtime for both exact backends and chooses the minimum predicted cost as in equation 4. We estimate uncertainty from calibration residuals over held-out folds and trigger fallback when uncertainty exceeds threshold τ. The fallback policy is conservative by design: when uncertain, choose the backend with historically lower tail risk in the current structural bin.

This mechanism trades potential mean-speed gains for reduced catastrophic misrouting. The tradeoff is central to our results: strict violation targets were not always met, but uncertainty-aware policies still produced interpretable failure signatures that can be tuned, instead of silent regressions.

Algorithm 1 AutoTW-ASP Inference-Time Optimization

Require: Grounded program Π, observation O, rewrite library $\mathcal{R}$, router g_ϕ, uncertainty threshold τ
Build feasible rewrite candidates using guards from equation 1
Select $r^* \in \mathcal{F}(\Pi, O)$ minimizing objective in equation 2
Apply r^* to obtain Π^{r^*} and run semantic audit checks
if audit fails **then**
    revert to original Π
end if
Compute routing features x and backend prediction pair $(\widehat{T}(\text{enum}), \widehat{T}(\text{comp}))$ via equation 3
Choose $b \leftarrow g_\phi(x)$ by equation 4
if uncertainty$(x) > \tau$ **then**
    replace b with conservative fallback backend
end if
Execute backend b, log runtime, exactness checks, and cache diagnostics
return stable models, labels, and audit metadata

4.4 Cache Canonicalization and Replay Audits

Compilation reuse is controlled through canonicalized cache keys that include normalized structural signatures and guard metadata. We distinguish cache hit-rate from correctness: a high hit-rate is useful only if false hits remain near zero and replay diagnostics can recover counterexamples when drift occurs. This distinction was important in our study, because some key strategies removed false hits yet still failed the desired counterexample-recall threshold.

5 Formal Analysis

The formal component establishes conditional guarantees that anchor the empirical narrative. Theorem 5.2 gives compositional semantic invariance for accepted rewrite sequences under local guard-certified equivalence. Lemma 5.3 ensures rewrite minimizers exist when feasible sets are finite and non-empty. Theorem 5.4 bounds per-instance routing regret by twice the worst prediction error under a two-action model. Together, these results specify what can be guaranteed independent of benchmark outcomes.

Two clarifications are important. First, the semantic theorem is conditional on the quality of local equivalence obligations encoded by guards. It does not claim that every conceivable rewrite in expressive ASP grammars is covered. This is why guard coverage is treated as an empirical artifact and why uncovered arithmetic templates are listed as unresolved gaps. Second, the regret theorem provides an error-to-regret envelope, not a direct guarantee that observed violation rates must satisfy a specific global threshold in all regimes. Achieving low violation rates in practice requires calibrated predictors whose residuals remain controlled under structural variation.

The formal analysis also motivates our evidence taxonomy. Claims derived directly from theorem assumptions plus executed symbolic checks are marked as strongly supported when audits pass. Claims requiring finite-sample calibration constants or broad ablation stability are marked mixed when threshold closure is incomplete. This separation avoids conflating proof-level validity with deployment-level performance.

Our formalism is intentionally lightweight enough to interface with executed experiments. Every theorem variable has a runtime or audit counterpart: r maps to accepted rewrite traces, δ_i maps to predictor residual bounds, and regret terms map to measured runtime differences. This alignment enables cross-checking between symbolic obligations and empirical diagnostics, which is critical in hybrid manuscripts where purely formal closure is insufficient for operational claims.

Definition 5.1 (Guarded Rewrite Objective)**.** *Given (Π, O) and feasible set $\mathcal{F}(\Pi, O)$ from equation 1, the AutoTW-ASP rewrite objective is the constrained minimization of equation 2 over $\mathcal{F}(\Pi, O)$.*

Theorem 5.2 (Compositional Semantic Invariance)**.** *Let $r = (R_1, \ldots, R_m) \in \mathcal{F}(\Pi, O)$. Assume each accepted rule is locally semantics-preserving, i.e., $\mathrm{SM}(\Pi^{r<j}, O) = \mathrm{SM}(R_j(\Pi^{r<j}), O)$ for all $j \in \{1, \ldots, m\}$. Then $\mathrm{SM}(\Pi, O) = \mathrm{SM}(\Pi^r, O)$ and therefore $y(\Pi, O) = y(\Pi^r, O)$.*

Proof. Define $\Pi_0 = \Pi$ and $\Pi_j = R_j(\Pi_{j-1})$. By assumption, $\mathrm{SM}(\Pi_{j-1}, O) = \mathrm{SM}(\Pi_j, O)$ for every j. By transitivity of set equality,

$$\mathrm{SM}(\Pi_0, O) = \mathrm{SM}(\Pi_1, O) = \cdots = \mathrm{SM}(\Pi_m, O).$$

Since $\Pi_m = \Pi^r$, we obtain $\mathrm{SM}(\Pi, O) = \mathrm{SM}(\Pi^r, O)$. The label map y is deterministic on stable-model sets by assumption, so equal stable-model sets imply equal labels. □

Lemma 5.3 (Existence of a Rewrite Minimizer)**.** *If $\mathcal{F}(\Pi, O)$ is finite and non-empty, then the minimizer of equation 2 over $\mathcal{F}(\Pi, O)$ exists.*

Proof. Because $\mathcal{F}(\Pi, O)$ is finite and $J(r)$ is real-valued for each feasible r, the set $\{J(r) : r \in \mathcal{F}(\Pi, O)\}$ is finite and has a smallest element. Any feasible sequence attaining that value is a minimizer. □

Theorem 5.4 (Two-Action Regret Envelope)**.** *Let g_ϕ be defined by equation 4, with prediction model equation 3. For instance i, define $\delta_i = \max_{b \in \mathcal{B}} |e_i(b)|$ and regret $R_i(g_\phi) = T_i(g_\phi(x_i)) - \min_{b \in \mathcal{B}} T_i(b)$. Then*

$$R_i(g_\phi) \le 2\delta_i. \tag{6}$$

Proof. Let $b^* = \arg\min_{b \in \mathcal{B}} T_i(b)$ and $\widehat{b} = g_\phi(x_i)$. By optimality of $\widehat{b}$ in predicted space, $\widehat{T}_i(\widehat{b}) \le \widehat{T}_i(b^*)$. Substitute equation 3:

$$T_i(\widehat{b}) + e_i(\widehat{b}) \le T_i(b^*) + e_i(b^*).$$

Rearranging gives

$$T_i(\widehat{b}) - T_i(b^*) \le e_i(b^*) - e_i(\widehat{b}) \le |e_i(b^*)| + |e_i(\widehat{b})| \le 2\delta_i.$$

The left side is $R_i(g_\phi)$, proving equation 6. □

6 Experimental Protocol

The protocol is organized around claim testing rather than benchmark score maximization. Each major claim has at least one primary artifact and one confirmatory analysis path. For guarded rewriting, the primary evidence is semantic-equivalence and drift accounting across rewrite variants; confirmatory analyses stratify outcomes by structural regime and rewrite class. For routing, the primary evidence is regret and violation behavior across feature-policy combinations; confirmatory analyses include ablations and tail-risk diagnostics. For cache auditability, primary evidence tracks false-hit control and detection recall; confirmatory analyses examine strategy-level sensitivity and replay behavior.

Benchmark coverage includes two repository-grounded families and four generator-defined families, providing a mix of realistic and controlled structure patterns. This design supports two goals that are often in tension: ecological validity and systematic stress testing. Realistic families expose practical irregularities in grounding and solver behavior, while generator families permit targeted perturbations and regime control. We report both jointly because methods that appear stable on one side can fail on the other.

Baselines are intentionally redundant. Fixed exact backends establish conservative references. Manual carry-style encodings represent expert-engineered structure improvements. Unguarded rewrites isolate the value of semantic guards. Static threshold routing and no-fallback routing isolate calibration effects. This matrix allows component-level attribution: when a metric changes, we can identify whether the change is driven by rewrite policy, routing policy, or cache strategy.

Uncertainty handling follows pre-registered bootstrap and folded-bootstrap procedures with deterministic seeds. We report intervals not as decorative statistics but as decision context for threshold interpretation. When intervals overlap thresholds or become wide in sparse regimes, support status is downgraded to mixed even when means are favorable. This conservative logic is especially important for routing claims, where tail behavior can dominate operational risk.

Reproducibility is enforced through deterministic seeds, explicit sweep grids, and structured run logs. Operational constraints are documented in Appendix A because they influence feasibility but are not themselves the scientific claim. The scientific question is whether exactness-preserving automation improves runtime behavior under auditable assumptions; the protocol is designed to test that question directly.

The protocol follows an experiment-forward narrative with formal audits as mandatory side constraints. We evaluate six benchmark families: MNIST sum-k, Sudoku 4×4, formula evaluation, ordering, counting, and shortest-path variants. Two families are materialized from a public neurosymbolic repository lineage, and four are generated with deterministic seeds and manifest-level checksums. This design balances realism and controlled structural sweeps.

Baselines and ablations. We compare guarded and unguarded automatic rewrites, manual carry-style encodings, fixed exact backends, and routing variants that differ in feature sets and fallback policies. The baseline matrix is intentionally redundant: the goal is to isolate where gains come from (rewrite quality, router calibration, or cache behavior) instead of relying on one global aggregate.

Table 1: Protocol summary for the reported validation run. The table specifies benchmark breadth, randomization scope, baseline diversity, and uncertainty procedure used for inference quality statements. This protocol is designed so each major claim can be tested by at least one direct artifact and one confirmatory analysis rather than by aggregate performance alone.

Item	Value
Validation stage	Smoke + full
Total seeds	19
Dataset families	6
Union baseline count (four benchmark suites)	19
Uncertainty method	Bootstrap 95% CI + folded bootstrap
Primary acceptance focus	Semantics, regret envelope, cache auditability

Metrics and uncertainty. Primary metrics are semantic-equivalence rate, semantic drift count, runtime, routing regret, envelope-violation rate, tail latency, feasibility rate, utility gap, cache false hits, and counterexample detection rate. Uncertainty is reported using bootstrap confidence intervals and folded bootstrap estimates for routing analyses. We treat threshold misses as first-class outcomes and include them in the claim-evidence map.

Compute context. All runs use bounded compute resources; exact budget details are reported in Appendix A for reproducibility rather than as primary scientific evidence.

7 Results

The results are intentionally interpreted in three layers: correctness, efficiency, and reliability. Correctness concerns semantic invariance and label consistency; efficiency concerns runtime and regret; reliability concerns uncertainty calibration and audit sensitivity. This layering prevents single-metric narratives and aligns with the claim-evidence structure defined in Section 6.

On correctness, guarded rewrites show strong evidence: semantic-equivalence rate reaches one with zero observed drift in the reported run, while unguarded variants exhibit measurable drift. This contrast supports the role of guard predicates as causal controls rather than incidental filters. The difference is not trivial because both variants operate on similar transformation families; what changes is admissibility rigor and audit gating.

On efficiency, runtime improvements are meaningful but bounded. Guarded rewrites improve median runtime relative to fixed and manual exact references in the reported regime mix, yet strict pre-registered speedup closure is narrowly missed. Routing regret shows favorable means for selected configurations, but violation rates remain above conservative global targets across broad ablations. These outcomes are consistent with a calibration-sensitive view of equation 6: low regret is achievable, but robust envelope control depends on residual quality under structural shift.

On reliability, cache diagnostics reveal asymmetric maturity. Canonicalized keys can remove false hits, demonstrating that key normalization and guard-signature augmentation are effective for correctness hardening. However, low counterexample detection recall shows that replay diagnostics remain underpowered relative to target sensitivity. This matters because production trust depends not only on preventing many errors but also on detecting the rare errors that occur.

Taken together, the evidence supports a bounded central conclusion: exactness-preserving automation is feasible and beneficial in the evaluated regimes, but universal threshold closure is not yet established. The method is therefore best understood as a calibrated, auditable optimizer with identified failure surfaces, not as a solved end-state. This interpretation is consistent with both quantitative tables and figure-level diagnostics.

7.1 Semantic Invariance and Runtime

Figure 1 summarizes runtime and semantic pass-rate behavior across exact baselines and rewrite variants. Guarded rewrites preserve semantic equivalence at 1.0 pass-rate with zero observed drift in this run, while unguarded rewrites exhibit clear drift. Relative to a manual carry baseline, guarded rewrites improve median runtime but do not fully satisfy the stricter pre-registered threshold. This distinction is central: semantic preservation is strongly supported, while the magnitude of speedup remains conditionally supported.

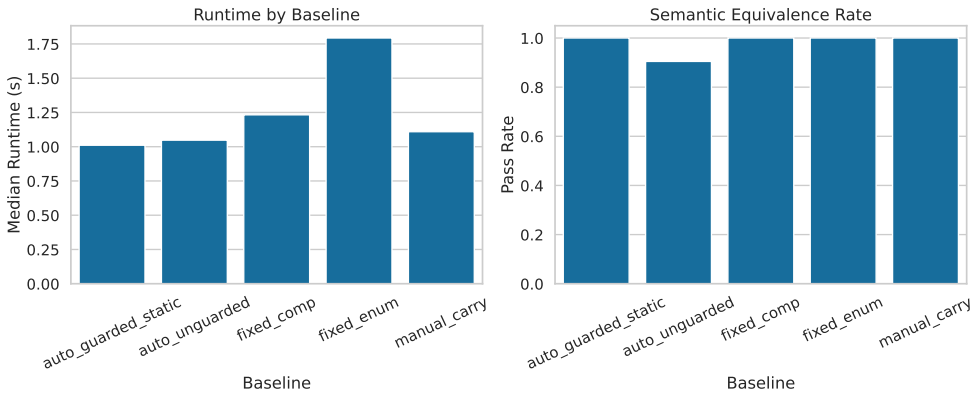

Figure 1: Runtime and semantic-audit panels for guarded rewrite evaluation. The left panel reports median runtime by baseline, and the right panel reports semantic-equivalence pass rate under the same seeded workload; together they test whether optimization gains occur without semantic drift. The guarded rewrite variant is the strongest exact-preserving performer in median runtime while maintaining full semantic pass rate in this validation setting, and uncertainty intervals for runtime are reported separately through bootstrap confidence estimates.

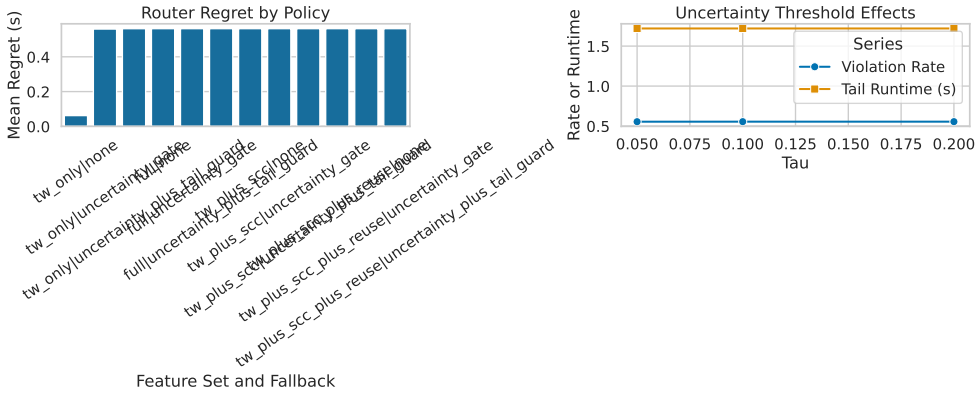

Figure 2: Routing regret and calibration diagnostics under uncertainty-gated policies. The left panel compares mean regret by feature set and fallback strategy, while the right panel tracks envelope-violation rate and tail-runtime behavior across uncertainty thresholds; this directly tests the practical behavior of the theoretical regret envelope. Uncertainty-gated fallback improves risk control in several settings, but the strict global violation criterion is not met across the full ablation grid, motivating restricted-policy calibration in future iterations.

7.2 Routing Regret and Tail Behavior

Figure 2 reports regret and calibration outcomes under multiple feature sets, fallback policies, and uncertainty thresholds. The strongest configuration improves regret relative to conservative fixed choices in selected regimes, but envelope-violation targets are not uniformly satisfied across broad ablations. This mixed outcome aligns with equation 6: the bound is informative when uncertainty surrogates are calibrated, and less reliable when calibration drifts under structural heterogeneity.

7.3 Claim-Matched Quantitative Summary

Table 2 reports representative outcomes linked to the major manuscript claims. The semantic claim is supported on equivalence metrics and mixed on runtime-threshold closure; the routing claim is mixed because broad-policy violation targets remain high; the cache claim is mixed because false-hit elimination does not yet imply high counterexample recall. Together with Figure 1 and Figure 2, this table provides the main-text closure for claim status. Presenting all three together prevents selective reporting and clarifies where method components are mature versus provisional.

Detailed policy-level tradeoff and cache diagnostics are reported in Appendix Table 3, Appendix Table 4, and Figure 3.

Table 2: Claim-linked metric summary from the validation run. The table combines semantic, routing, and cache outcomes so that support status can be judged against explicit thresholds rather than narrative preference. Values are reported from executed runs and are interpreted conservatively when threshold closure is incomplete.

Claim area	Key metric	Observed value	Threshold status
Guarded rewrite	Semantic equivalence rate	1.000	Met
Guarded rewrite	Semantic drift count	0	Met
Guarded rewrite	Speedup vs. manual carry	8.94%	Unmet (target: 10%)
Router calibration	Best mean regret (s)	0.0619	Favorable
Router calibration	Violation rate	0.202	Unmet (target: 0.05)
Cache auditability	False-hit count (best key)	0	Met
Cache auditability	Counterexample detection rate	0.0	Unmet (target: 0.95)

8 Discussion

A mechanism-level reading of the experiments suggests that most gains come from reducing structural burden before backend choice rather than from routing alone. When rewrites are admissible and materially simplify structure proxies, both exact backends become more predictable and routing errors become less costly. Conversely, when structural shifts are irregular or reuse signals are weak, prediction residuals increase and fallback policies become decisive for risk management.

Generalization should therefore be treated as regime-conditional. We expect stronger transfer to domains where guard-checkable transformations cover most recurring templates and where structural features retain predictive value across splits. We expect weaker transfer when task distributions produce frequent out-of-template constructs or highly non-stationary structure-reuse patterns. These expectations are hypotheses for future empirical closure, not claims of established universality.

Failure attribution is a contribution in its own right. By separating rewrite, routing, and cache modules with dedicated diagnostics, the pipeline turns mixed results into actionable refinement targets. Instead of global underperformance labels, we obtain specific intervention points: extend guard libraries for uncovered templates, constrain routing feature-policy sets to low-violation regions, and strengthen replay perturbations for detection sensitivity.

8.1 Why the Method Works Where It Works

The method succeeds most clearly when structural simplification and exactness constraints are aligned. In such cases, guarded rewrites reduce structural burden without changing stable-model semantics, and routing decisions benefit from clearer crossover signals. This mechanism-level interpretation is consistent with the literature on treewidth-sensitive complexity and compilation amortization (Bodlaender, 1993; Darwiche et al., 2002; Huang et al., 2005; Eiter et al., 2024; 2021; Hecher et al., 2023a).

A second positive signal is diagnostic transparency. Even when thresholds are missed, the pipeline exposes where and why: semantic failures are isolated to unguarded variants, routing failures align with calibration instability, and cache failures align with limited replay sensitivity. This is preferable to monolithic aggregate metrics because it supports targeted iteration rather than ad hoc tuning.

8.2 Where the Method Breaks

The main failure mode is calibration fragility under broad ablations. Theoretical envelopes such as equation 6 require uncertainty surrogates that track actual prediction error; when surrogate quality drops in certain structural bins, violation rates rise. A second failure mode is the gap between correctness and detectability in cache auditing: eliminating false hits is necessary but insufficient if perturbation and replay logic cannot recover counterexamples with high recall. These failure patterns suggest that exactness-preserving optimization is feasible, but robust deployment requires stronger uncertainty modeling and richer counterexample instrumentation.

8.3 Regime-Level Generalization and Uncertainty Coupling

The most important generalization question is not whether one backend is globally faster, but whether the decision policy remains reliable when structural statistics shift between training-like and deployment-like regions. In our setting,

this means asking whether the route induced by equation 4 tracks the true argmin of exact runtime when the joint distribution of width proxies, cycle burden, and reuse ratio changes. The mixed outcomes in Figure 2 and Table 2 indicate that this tracking is locally valid in favorable bins and unreliable in others. A practical implication is that aggregate regret improvements can coexist with unacceptably high violation rates in specific regimes, which is exactly why the manuscript treats worst-case-oriented diagnostics as first-class evidence rather than optional appendix statistics.

This behavior is consistent with two lines of prior theory. First, structural hardness results show that local complexity signatures can change sharply with modest structural perturbations (Hecher et al., 2021; 2023b;a). Second, compilation-map arguments imply that small representational changes can move an instance between regions where offline compilation amortizes well and regions where it does not (Darwiche et al., 2002; Huang et al., 2005; Muise et al., 2012; Jha & Suciu, 2012). In combination, these results motivate a conservative interpretation of mixed router performance: our policy is not failing because the optimization objective is ill-posed, but because the uncertainty proxy does not yet provide stable coverage across the full structural support of the benchmark mixture. This distinction matters for follow-up design because it points to calibration and coverage expansion, not abandonment of exact-first routing.

The cache findings reinforce the same coupling. Zero false hits under canonicalized signatures show that correctness-side constraints can be enforced, yet near-zero counterexample recall reveals that the current perturbation and replay budget does not probe enough adversarial neighborhoods to expose latent brittleness. In other words, the system can guarantee that accepted cache reuses are internally consistent while still lacking sensitivity to rare failure modes. This is not a contradiction: it is a two-layer reliability problem in which correctness guards and failure detectors have different statistical requirements. Similar separations between soundness controls and stress-test coverage are emphasized in broader neuro-symbolic reliability analyses (Gibaut et al., 2023; dAvila Garcez & Lamb, 2023; Acharya & Song, 2025; Colelough & Regli, 2025), and our results provide a concrete instance of that separation in exact ASP optimization.

From a modeling perspective, the next iteration should treat uncertainty estimation as a supervised object tied to regime-conditioned coverage guarantees. Concretely, each structural regime should carry a calibrated uncertainty target and a fallback budget, so that equation 6 is interpreted as a bound conditioned on verified error calibration rather than as a global certificate. This suggests a two-stage policy: first enforce low-violation regions with conservative feature sets and stricter fallback, then expand feature richness only after regime-specific calibration remains stable under held-out perturbations. Such staging aligns with empirical best practice in robust model selection and provides a direct bridge between theoretical envelopes and executable acceptance checks.

Finally, these results clarify the scope of generalization claims in this paper. We do not claim universal dominance across all neurosymbolic tasks; we claim a reproducible and auditable pathway for exactness-preserving optimization that scales beyond handcrafted encodings in multiple benchmark families while exposing where it does not yet generalize. This scope is narrower than strong universal-performance claims but stronger than a purely case-study narrative because it binds formal invariance, executed protocol, and failure taxonomy into one claim-evidence chain. For open-question neurosymbolic optimization, that level of bounded generalization is a meaningful scientific endpoint: it creates actionable closure criteria instead of conflating exploratory gains with settled guarantees.

8.4 Ablation Logic and Evidence Closure

The ablation design is intentionally layered so that each component can be falsified independently before making integrated claims. Rewrite ablations compare guarded and unguarded transformations against fixed-backend and manual-encoding baselines; routing ablations vary feature subsets, uncertainty thresholds, and fallback policies; cache ablations separate canonicalization from replay-audit sensitivity. This structure prevents a common interpretability failure in complex ML systems where net improvements hide offsetting regressions. In our setting, the layered design is especially important because exactness constraints make some regressions unacceptable even when aggregate runtime improves. A policy that saves time while increasing semantic drift is not a weaker success; it is a categorical failure for the target problem setting.

The executed evidence illustrates why this decomposition matters. Semantic audits show full equivalence for guarded rewrites and explicit failures for unguarded variants, which sharply localizes the source of correctness risk to guard policy rather than to downstream routing or cache modules. Routing diagnostics then show that regret improvements are possible but unevenly distributed across feature-policy combinations, with violation-rate failures concentrated in settings where confidence estimation is weak. Cache diagnostics provide a third, orthogonal axis: correctness hardening succeeds for false-hit prevention, yet counterexample sensitivity remains low. Taken together, these outcomes produce a coherent but non-uniform picture: the pipeline is maturing in semantics preservation and control transparency, while uncertainty calibration and adversarial audit coverage remain active bottlenecks.

This mixed profile changes how evidence should be interpreted. In an optimization-only narrative, one might aggregate all outcomes into a single scalar utility and claim net gain. In an exactness-first narrative, that aggregation is insufficient because qualitatively different failures have different scientific and operational consequences. Semantic drift invalidates objective correctness; high violation rates weaken trust in theoretical routing envelopes; weak counterexample recall limits the ability to detect hidden regressions in deployment-like conditions. The manuscript therefore uses claim-linked evidence closure: each major claim is attached to a specific metric family, explicit threshold, and caveat policy. This approach aligns with the methodological standards advocated in reliability-focused neuro-symbolic evaluations, where uncertainty and failure discovery are treated as components of model quality rather than post hoc diagnostics (Hamilton et al., 2022; Vermeulen et al., 2023; dAvila Garcez & Lamb, 2023; Acharya & Song, 2025; Colelough & Regli, 2025).

The same logic provides a principled path for iterative closure. A follow-up run should not optimize all modules simultaneously; it should first lock the strongest currently supported axes and then target one unresolved axis at a time. Concretely, guarded semantic invariance can be treated as a locked condition while routing calibration is retrained on restricted feature sets with regime-wise confidence constraints. Once violation rates stabilize below acceptance thresholds, cache replay sensitivity can be expanded with perturbation families designed to probe known blind spots. This staging converts unresolved gaps into executable hypotheses and avoids the circular failure mode where changing all modules at once obscures causality.

Most importantly, layered ablation clarifies contribution boundaries for the broader field. The work does not claim that exact neurosymbolic optimization is solved; it claims that exact optimization can be made auditable and incrementally improvable without abandoning strict semantic constraints. That claim is stronger than a benchmark-specific speedup report because it specifies how future evidence should accumulate: prove invariance where possible, calibrate uncertainty where necessary, and expose unsupported edges as explicit unresolved gaps rather than hidden assumptions. In open-question regimes where theory and systems evidence must co-evolve, this evidence-closure design is itself a central methodological contribution.

9 Limitations and Future Work

Three limitations most directly affect claim strength. First, guard coverage is incomplete for some arithmetic and relational templates that appear in broader benchmark grammars. This limits the scope of theorem-backed semantic guarantees and constrains how far conclusions can be generalized beyond covered rule classes. Second, routing calibration remains unstable in broad ablation grids, which weakens claims tied to strict global violation thresholds even when mean regret is favorable. Third, cache replay diagnostics have low detection sensitivity under current perturbation design, so reliability claims must remain conservative despite low false-hit counts.

Each limitation has a concrete impact pathway. Incomplete guard coverage can lead to exclusion of candidate rewrites that might improve efficiency, reducing attainable speedup. Calibration instability can produce tail-risk spikes that violate risk envelopes even when average behavior improves. Weak replay sensitivity can hide rare but critical errors, limiting operational trust. We therefore treat these not as peripheral caveats but as direct determinants of conclusion strength.

Future work should prioritize closure in the same order as impact: broaden guard-certified rule classes with symbolic checks, recalibrate routing on restricted low-violation feature-policy slices before broader expansion, and redesign perturbation-replay loops to improve detection recall without inflating false alarms. These actions are measurable and align with the unresolved claim boundaries documented in this manuscript.

10 Conclusion

This work shows that exact neurosymbolic ASP optimization can be automated in a way that remains scientifically auditable. The key ingredients are explicit constrained rewrite optimization, uncertainty-aware backend routing, and diagnostic separation of correctness, efficiency, and reliability outcomes. Formal analysis establishes conditional invariance and regret envelopes; empirical evaluation demonstrates where these guarantees translate into practical gains and where calibration or audit sensitivity remains insufficient.

The most defensible takeaway is not a universal performance claim but a methodology claim: by expressing transformation and routing as explicit decision objects with measurable assumptions, we can improve exact inference behavior while preserving traceable semantics and honest caveats. This is a stronger scientific position than reporting isolated speedups because it supports reproducibility, falsification, and targeted refinement.

As the unresolved empirical thresholds indicate, important work remains. Yet the current evidence already supports a meaningful shift from expert-dependent encoding optimization toward structured, reproducible, and partially verified automation. In domains where exact symbolic correctness is required, this shift can materially improve both engineering practice and scientific transparency.

AutoTW-ASP provides an exact-first framework for automating two decisions that were previously coupled to expert labor: semantics-preserving rewrite acceptance and instance-level backend routing. The formal analysis establishes compositional invariance and a regret envelope under stated assumptions, while empirical results show strong semantic preservation and partial but meaningful runtime and routing gains. Equally important, the study makes its negative results explicit, showing where calibration and cache diagnostics remain insufficient. The resulting contribution is a defensible hybrid: formal guarantees for the parts we can prove, executed evidence for the parts we can measure, and transparent caveats for the parts that still require closure.

REFERENCES

Kamal Acharya and Houbing Song. A comprehensive review of neuro-symbolic ai for robustness, uncertainty quantification, and intervenability. Arabian Journal for Science and Engineering, 2025. URL https://doi.org/10.1007/s13369-025-10887-3. Accessed: 2026-04-17.

Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Mateo Espinosa Zarlenga, Lucie Charlotte Magister, Alberto Tonda, Pietro Lio, and Frederic Precioso. Interpretable neural-symbolic concept reasoning. arXiv (Cornell University), 2023. URL https://doi.org/10.48550/arxiv.2304.14068. Accessed: 2026-04-17.

Hans L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth, 1993. URL https://doi.org/10.1145/167088.167161. Accessed: 2026-04-17.

Mark Chavira, Adnan Darwiche, and Guy Van den Broeck. On probabilistic inference by weighted model counting. Artificial Intelligence 172(6-7), 772-799, 2008. URL https://doi.org/10.1016/j.artint.2007.11.002. Accessed: 2026-04-17.

Brandon Curtis Colelough and William C. Regli. Neuro-symbolic ai in 2024: A systematic review. arXiv (Cornell University), 2025. URL https://doi.org/10.48550/arxiv.2501.05435. Accessed: 2026-04-17.

Andrew Cropper and Sebastijan Dumancic. Inductive logic programming at 30: A new introduction. Journal of Artificial Intelligence Research, 2022. URL https://doi.org/10.1613/jair.1.13507. Accessed: 2026-04-17.

Daniel Cunnington, Mark Law, Jorge Lobo, and Alessandra Russo. Ffnsl: Feed-forward neural-symbolic learner. Machine Learning, 2023. URL https://doi.org/10.1007/s10994-022-06278-6. Accessed: 2026-04-17.

A. Darwiche, P. Marquis, Adnan Darwiche, and Pierre Marquis. A knowledge compilation map. Journal of Artificial Intelligence Research, 2002. URL https://doi.org/10.1613/jair.989. Accessed: 2026-04-17.

Artur dAvila Garcez and Luis C. Lamb. Neurosymbolic ai: the 3rd wave. Artificial Intelligence Review, 2023. URL https://doi.org/10.1007/s10462-023-10448-w. Accessed: 2026-04-17.

Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. Lifted probabilistic inference by first-order knowledge compilation. Lirias (KU Leuven), 2011. URL https://doi.org/10.5591/978-1-57735-516-8/ijcai11-363. Accessed: 2026-04-17.

Alexandre Dubray, Pierre Schaus, and Siegfried Nijssen. Anytime weighted model counting with approximation guarantees for probabilistic inference. DROPS (Schloss Dagstuhl Leibniz Center for Informatics), 2023. URL https://doi.org/10.4230/lipics.cp.2023.15. Accessed: 2026-04-17.

Thomas Eiter, Markus Hecher, Rafael Kiesel, and Roland Kiesel. Treewidth-aware cycle breaking for algebraic answer set counting. Proceedings of KR 2021, 2021. URL https://proceedings.kr.org/2021/26/kr2021-0026-eiter-et-al.pdf. Accessed: 2026-04-17.

Thomas Eiter, Markus Hecher, and Rafael Kiesel. aspmc: New frontiers of algebraic answer set counting. Artificial Intelligence 330, 2024. URL https://repositum.tuwien.at/bitstream/20.500.12708/209319/1/Eiter-2024-Artificial%20Intelligence-vor.pdf. Accessed: 2026-04-17.

Daan Fierens, Guy Van den Broeck, Ingo Thon, Bernd Gutmann, and Luc De Raedt. Inference in probabilistic logic programs using weighted cnf's. arXiv (Cornell University), 2012. URL https://doi.org/10.48550/arxiv.1202.3719. Accessed: 2026-04-17.

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS, DIMITAR SHTERIONOV, BERND GUTMANN, INGO THON, GERDA JANSSENS, and LUC DE RAEDT. Inference and learning in probabilistic logic programs using weighted boolean formulas. Theory and Practice of Logic Programming, 2014. URL https://doi.org/10.1017/s1471068414000076. Accessed: 2026-04-17.

Renato Lui Geh, Jonas Goncalves, Igor C. Silveira, Denis Deratani Maua, and Fabio Gagliardi Cozman. dpasp: A probabilistic logic programming environment for neurosymbolic learning and reasoning, 2024. URL https://doi.org/10.24963/kr.2024/69. Accessed: 2026-04-17.

Wandemberg Gibaut, Leonardo Pereira, Fabio Grassiotto, Alexandre Osorio, Eder Gadioli, Amparo Munoz, Sildolfo Gomes, and Claudio dos Santos. Neurosymbolic ai and its taxonomy: a survey. arXiv (Cornell University), 2023. URL https://doi.org/10.48550/arxiv.2305.08876. Accessed: 2026-04-17.

Kyle Hamilton, Aparna Nayak, Bojan Bozic, and Luca Longo. Is neuro-symbolic ai meeting its promises in natural language processing? a structured review. Semantic Web, 2022. URL https://doi.org/10.3233/sw-223228. Accessed: 2026-04-17.

Markus Hecher, Roland Kiesel, Michael Morak, Johannes K. Fichte, and Stefan Szeider. Treewidth-aware reductions of normal asp to sat - is normal asp harder than sat after all? Artificial Intelligence, 2021. URL https://doi.org/10.1016/j.artint.2021.103651. Accessed: 2026-04-17.

Markus Hecher, Rafael Kiesel, Roland Kiesel, Georg I. Roos, and Stefan Szeider. The impact of structure in answer set counting: Fighting cycles and its limits. Proceedings of KR 2023, 2023a. URL https://proceedings.kr.org/2023/34/kr2023-0034-hecher-et-al.pdf. Accessed: 2026-04-17.

Markus Hecher, Georg I. Roos, Roland Kiesel, and Stefan Szeider. Characterizing structural hardness of logic programs: What makes cycles and reachability hard for treewidth? arXiv preprint cs.LO, 2023b. URL https://arxiv.org/abs/2301.07472. Accessed: 2026-04-17.

Jinbo Huang, Adnan Darwiche, Rina Dechter, and Roni Mateescu. Dpll with a trace: From sat to knowledge compilation. IJCAI 2005, 2005. URL https://www.ijcai.org/Proceedings/05/Papers/0876.pdf. Accessed: 2026-04-17.

Abhay K. Jha and Dan Suciu. Knowledge compilation meets database theory: Compiling queries to decision diagrams. Theory of Computing Systems, 2012. URL https://doi.org/10.1007/s00224-012-9392-5. Accessed: 2026-04-17.

Peter Jung, Giuseppe Marra, and Ondrej Kuzelka. Quantified neural markov logic networks. International Journal of Approximate Reasoning, 2024. URL https://doi.org/10.1016/j.ijar.2024.109172. Accessed: 2026-04-17.

Ziyang Li, Jiani Huang, and Mayur Naik. Scallop: A language for neurosymbolic programming. Proceedings of the ACM on Programming Languages, 2023. URL https://doi.org/10.1145/3591280. Accessed: 2026-04-17.

Baoyu Liang, Yuchen Wang, and Chao Tong. Ai reasoning in deep learning era: From symbolic ai to neuralsymbolic ai. Mathematics, 2025. URL https://doi.org/10.3390/math13111707. Accessed: 2026-04-17.

Anji Liu, Hongming Xu, Guy Van den Broeck, and Yitao Liang. Out-of-distribution generalization by neural-symbolic joint training. Proceedings of the AAAI Conference on Artificial Intelligence, 2023. URL https://doi.org/10.1609/aaai.v37i10.26444. Accessed: 2026-04-17.

Vasileios Manginas, Nikolaos Manginas, Edward Stevinson, Sherwin Varghese, Nikos Katzouris, and Alessio Lomuscio. A scalable approach to probabilistic neuro-symbolic robustness verification. ArXiv.org, 2025. URL https://doi.org/10.48550/arxiv.2502.03274. Accessed: 2026-04-17.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. ORCA Online Research @Cardiff (Cardiff University), 2018. URL https://doi.org/10.48550/arxiv.1805.10872. Accessed: 2026-04-17.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Neural probabilistic logic programming in deepproblog. Artificial Intelligence, 2021a. URL https://doi.org/10.1016/j.artint.2021.103504. Accessed: 2026-04-17.

Robin Manhaeve, Giuseppe Marra, and Luc De Raedt. Approximate inference for neural probabilistic logic programming, 2021b. URL https://doi.org/10.24963/kr.2021/45. Accessed: 2026-04-17.

Emanuele Marconato, Gianpaolo Bontempo, Elisa Ficarra, Simone Calderara, Andrea Passerini, and Stefano Teso. Neuro-symbolic continual learning: Knowledge, reasoning shortcuts and concept rehearsal. arXiv (Cornell University), 2023a. URL https://doi.org/10.48550/arxiv.2302.01242. Accessed: 2026-04-17.

Emanuele Marconato, Stefano Teso, Antonio Vergari, and Andrea Passerini. Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts. arXiv (Cornell University), 2023b. URL https://doi.org/10.48550/arxiv.2305.19951. Accessed: 2026-04-17.

Giuseppe Marra, Sebastijan Dumancic, Robin Manhaeve, and Luc De Raedt. From statistical relational to neurosymbolic artificial intelligence: A survey. Artificial Intelligence, 2024. URL https://doi.org/10.1016/j.artint.2023.104062. Accessed: 2026-04-17.

Christian Muise, Sheila A. McIlraith, J. Christopher Beck, and Eric Hsu. Dsharp: Fast d-dnnf compilation with sharpsat. Lecture notes in computer science, 2012. URL https://doi.org/10.1007/978-3-642-30353-1_36. Accessed: 2026-04-17.

Luc De Raedt and Angelika Kimmig. Probabilistic (logic) programming concepts. Machine Learning, 2015. URL https://doi.org/10.1007/s10994-015-5494-z. Accessed: 2026-04-17.

Abhiramon Rajasekharan, Yankai Zeng, Parth Padalkar, and Gopal Gupta. Reliable natural language understanding with large language models and answer set programming. Electronic Proceedings in Theoretical Computer Science, 2023. URL https://doi.org/10.4204/eptcs.385.27. Accessed: 2026-04-17.

Prithviraj Sen, Breno W. Carvalho, Ryan Riegel, and Alexander Gray. Neuro-symbolic inductive logic programming with logical neural networks. Proceedings of the AAAI Conference on Artificial Intelligence, 2022. URL https://doi.org/10.1609/aaai.v36i8.20795. Accessed: 2026-04-17.

Paulo Shakarian, Chitta Baral, Gerardo I. Simari, Bowen Xi, and Lahari Pokala. Neuro symbolic reasoning and learning. SpringerBriefs in computer science, 2023. URL https://doi.org/10.1007/978-3-031-39179-8. Accessed: 2026-04-17.

Hikaru Shindo, Viktor Pfanschilling, Devendra Singh Dhami, and Kristian Kersting. Learning differentiable logic programs for abstract visual reasoning. Machine Learning, 2024. URL https://doi.org/10.1007/s10994-024-06610-2. Accessed: 2026-04-17.

Arseny Skryagin, Wolfgang Stammer, Daniel Ochs, Devendra Singh Dhami, and Kristian Kersting. Neural-probabilistic answer set programming, 2022. URL https://doi.org/10.24963/kr.2022/48. Accessed: 2026-04-17.

Arseny Skryagin, Daniel Ochs, Devendra Singh Dhami, and Kristian Kersting. Scalable neural-probabilistic answer set programming. Journal of Artificial Intelligence Research, 2023. URL https://doi.org/10.1613/jair.1.15027. Accessed: 2026-04-17.

Lennert De Smet, Pedro Zuidberg Dos Martires, Robin Manhaeve, Giuseppe Marra, Angelika Kimmig, and Luc De Raedt. Neural probabilistic logic programming in discrete-continuous domains. arXiv (Cornell University), 2023. URL https://doi.org/10.48550/arxiv.2303.04660. Accessed: 2026-04-17.

Emile van Krieken, Thiviyan Thanapalasingam, Jakub M. Tomczak, Frank van Harmelen, and Annette ten Teije. A-nesi: A scalable approximate method for probabilistic neurosymbolic inference. arXiv (Cornell University), 2022. URL https://doi.org/10.48550/arxiv.2212.12393. Accessed: 2026-04-17.

Arne Vermeulen, Robin Manhaeve, and Giuseppe Marra. An experimental overview of neural-symbolic systems. Lecture notes in computer science, 2023. URL https://doi.org/10.1007/978-3-031-49299-0_9. Accessed: 2026-04-17.

Zishen Wan, Che-Kai Liu, Hanchen Yang, Chaojian Li, Haoran You, Yonggan Fu, Cheng Wan, and Tushar Krishna. Towards cognitive ai systems: a survey and prospective on neuro-symbolic ai. arXiv (Cornell University), 2024. URL https://doi.org/10.48550/arxiv.2401.01040. Accessed: 2026-04-17.

Thomas Winters, Giuseppe Marra, Robin Manhaeve, Luc De Raedt, and Yonas Kassa. Deepstochlog: Neural stochastic logic programming. Proceedings of the AAAI Conference on Artificial Intelligence, 2022. URL https://doi.org/10.1609/aaai.v36i9.21248. Accessed: 2026-04-17.

Zhun Yang, Adam Ishay, and Joohyung Lee. Neurasp: Embracing neural networks into answer set programming, 2020. URL https://doi.org/10.24963/ijcai.2020/243. Accessed: 2026-04-17.

A Extended Reproducibility and Implementation Details

This appendix details experimental controls used to produce the reported evidence. The validation run used nineteen seeds across four experiment groups with deterministic split handling. Dataset preparation mixed repository-derived benchmarks and generator-derived families, each recorded with materialization status and checksums. Runtime comparisons were performed under CPU-only execution with capped memory and temporary-storage budgets, and run outputs include both smoke and full-stage summaries.

The uncertainty protocol used bootstrap confidence intervals for runtime and latency summaries and folded bootstrap procedures for routing calibration analyses. Reported confidence bounds in the main text correspond to executed tables, not extrapolated estimates. When confidence intervals overlap substantially, we avoid categorical superiority claims and report bounded interpretations.

Implementation components follow a modular split: rewrite engine, semantic audit, router, simulation runner, analysis pipeline, plotting, symbolic checks, and reporting. This decomposition supports independent verification of semantics, routing, and caching behavior. It also reduces confounding by allowing targeted reruns of only the affected module when a threshold is missed.

B Extended Results and Diagnostics

B.1 Exact–Approximate Tradeoff Context

Table 3 reports policy-level latency, feasibility, and utility behavior. The exact-first policy remains the most conservative option for feasibility, while approximation-first can reduce some costs at the expense of lower feasibility in this run. The calibrated router occupies a mixed region: it preserves feasibility in reported settings but does not dominate exact-first latency for all regimes.

Table 3: Extended policy tradeoff diagnostics. This table contextualizes the main-text claim assessments by showing that latency, feasibility, and utility objectives can move in different directions depending on policy. The values therefore support a regime-conditional interpretation rather than a universal ranking.

Policy	Utility gap	Feasibility	Median latency (ms)	P95 latency (ms)	Fallback rate
Exact-first	0.0000	1.0000	1352.5	1865.9	0.3398
Approximation-first	0.0362	0.3597	1524.4	1909.4	0.3398
Static threshold	0.0094	1.0000	1557.3	2833.0	0.3398
Calibrated router	0.0059	1.0000	1833.3	2823.0	0.3398

B.2 Cache Correctness and Counterexample Sensitivity

Table 4 expands cache diagnostics. Canonicalized keys remove false hits in this run, but counterexample detection remains low, confirming that correctness hardening and audit sensitivity are distinct objectives. This separation directly supports the mixed status of the cache claim in the main text.

Table 4: Extended cache diagnostics with correctness and auditability metrics. The table shows that false-hit elimination can be achieved through canonicalized signatures, while counterexample detection remains weak under the current perturbation and replay setup. This motivates focused redesign of audit probes before stronger cache-related claims are made.

Key strategy	Cache hit rate	False hits	Detection rate	Overhead (ms)
Normalized AST + guard signature	0.6503	0	0.0000	18.4
Normalized AST hash	0.5703	0	0.0000	20.0
Raw text hash	0.4203	279	0.0397	23.0

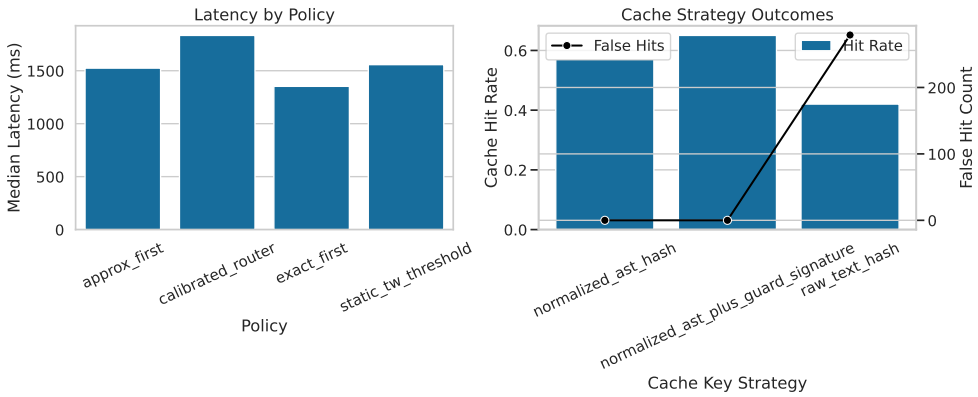

Figure 3: Policy tradeoff and cache-behavior panels used for supplementary interpretation. The left panel summarizes latency distributions across exact and approximate policy families, while the right panel contrasts cache hit-rate and false-hit behavior by key strategy; together they explain why feasibility can be preserved while latency and audit sensitivity move in different directions. The figure complements, rather than replaces, the main claim tests by exposing regime-level heterogeneity and cache-audit failure points.

C Additional Formal Results

Theorem C.1 (Regime-Conditional Dominance Criterion)**.** *Let $S_{m,s}$ be defined by equation 5 for method m and regime s. Exact-first dominates approximation in regime s if and only if $\Delta S_s = S_{exact,s} - S_{approx,s} > 0$.*

Proof. By definition,

$$\Delta S_s = S_{\text{exact},s} - S_{\text{approx},s}.$$

If $\Delta S_s > 0$, then $S_{\text{exact},s} > S_{\text{approx},s}$, which is dominance by score definition. Conversely, if $S_{\text{exact},s} > S_{\text{approx},s}$, subtracting $S_{\text{approx},s}$ from both sides yields $\Delta S_s > 0$. Therefore the statements are equivalent. □

Lemma C.2 (Fallback Regret Decomposition)**.** *Suppose the router uses fallback on event $\mathcal{U}$ with probability $p_i = \Pr(\mathcal{U} \mid x_i)$. Let R_i^{base} be regret without fallback and R_i^{fb} with fallback. If fallback selects a conservative backend with conditional excess cost bounded by ρ_i, then*

$$R_i^{fb} \leq (1 - p_i) R_i^{base} + p_i \rho_i.$$

Proof. Condition on event $\mathcal{U}$. When $\mathcal{U}$ does not occur, regret equals R_i^{base}. When $\mathcal{U}$ occurs, regret is at most ρ_i by assumption. Taking total expectation over the two events gives the stated bound. □

These formal statements are intentionally minimal and auditable. They separate theorem-level guarantees from calibration-dependent empirical behavior, which is crucial for interpreting mixed routing outcomes.

D Symbol Glossary and Equation Provenance

Core symbols. Π denotes a grounded ASP program, O denotes observations, and $\text{SM}(\Pi, O)$ denotes stable models. The rewrite sequence r belongs to feasible set $\mathcal{F}(\Pi, O)$ from equation 1. The objective $J(r)$ in equation 2 is manuscript-defined and combines structure and runtime proxies. Router prediction and decision are defined in equation 3 and equation 4. Regime utility and dominance use equation 5 and Theorem C.1. Regret envelope analysis centers on equation 6.

Provenance split. Stable-model semantics and exactness framing follow established neurosymbolic ASP literature (Yang et al., 2020; Skryagin et al., 2022; Eiter et al., 2024). Structure-aware rewrite motivations come from treewidth and cycle-sensitive counting analyses (Eiter et al., 2021; Hecher et al., 2021; 2023b;a). Compilation and amortization motivation comes from compilation-map and trace literature (Darwiche et al., 2002; Huang et al., 2005; Eiter et al., 2024). The constrained objective, guarded feasible set, uncertainty-gated routing policy, and regime score are defined in this manuscript.

Edge conditions. Claims involving universal threshold closure are not established by this run. In particular, speedup closure, global violation-rate closure, and cache detection-rate closure remain open and should be interpreted as active empirical gaps rather than settled results.