Fragment-Sensitive Quantifier Elimination and Safe Table Updates in Tau

Dana Edwards

April 13, 2026

Abstract

We summarize two scoped results from a neuro-symbolic study of Tau Language and TABA-inspired table semantics. First, quantifier elimination in Tau benefits from fragment-sensitive dispatch: on a checked leading-existential propositional fragment, a guarded BDD route with structural preprocessing produced a measured aggregate qelim-time speedup of about 5.15 over Tau's default route on the policy-shaped corpus. Second, a safe table-update fragment has a precise pointwise revision semantics and a feature-gated Tau experiment. Neither result proves full Tau optimization, full TABA tables, NSO lowering, or unrestricted recurrence. The contribution is a scoped method: propose a semantic route, formalize the claim, check or refute it mechanically, benchmark the artifact, and promote only the surviving fragment.

Method. The working loop was

$$\text{Idea} \to \text{FormalStatement} \to \text{Checker} \to \text{Counterexample or Proof} \to \text{Revision} \to \text{ScopedClaim}\,.$$

The symbolic side proposed equivalences, carriers, and compiler routes. The mechanical side used Lean, Tau replay, bounded exhaustive checks, SMT cross-checks, and proof-search agents to reject false statements or confirm scoped ones. This follows the method pattern of contemporary formalized mathematics: human judgment selects the statement and boundary, while formal tools check the exact claim.

Fragment-sensitive qelim. For a Boolean variable, exact compiled forgetting is

$$\exists x.\, f = f[x := \bot] \vee f[x := \top].$$

This is a valid identity for the propositional Boolean function represented by f. It does not imply that arbitrary Tau formulas can be projected by collecting all existentials globally. The experimental dispatcher therefore has the guarded form

$$Q(\varphi) = \begin{cases} \text{abstract}_{\text{BDD}}(\varphi), & \varphi \in \mathcal{F}_{\exists\text{prop}}, \\ Q_0(\varphi), & \varphi \notin \mathcal{F}_{\exists\text{prop}}, \end{cases}$$

where Q_0 is Tau's default qelim route and $\mathcal{F}_{\exists\text{prop}}$ is the supported leading-existential propositional fragment. The guard is part of correctness. The counterexample shape

$$\neg\exists y.\, P(y) \equiv \forall y.\, \neg P(y) \not\equiv \exists y.\, \neg P(y)$$

explains why nested or non-prefix existential formulas must fall back.

Inside the accepted fragment, additional exact rewrites are available. If no quantified variable is shared between distinct components, then

$$\exists X.\, \bigwedge_{i=1}^{k} F_i \equiv \bigwedge_{i=1}^{k} \exists X_i.\, F_i.$$

For a pure zero-test atom $p_x := (x = 0)$,

$$\exists x.\, \Phi(p_x, y) \equiv \Phi(\top, y) \quad \text{if } p_x \text{ occurs only positively,}$$

and

$$\exists x.\, \Phi(p_x, y) \equiv \Phi(\bot, y) \quad \text{if } p_x \text{ occurs only negatively.}$$

For CNF-shaped formulas, Davis-Putnam elimination gives

$$\exists x \left(R \wedge \bigwedge_i (x \vee A_i) \wedge \bigwedge_j (\neg x \vee B_j) \right) \equiv R \wedge \bigwedge_{i,j} (A_i \vee B_j).$$

This route is exact but can create a product of resolvents, so the executable lane uses explicit caps and subsumption-aware minimization before accepting the step.

The current measured candidate is

$$\mathrm{speedup}_{\mathrm{agg}} := \frac{\sum_i t_{\mathrm{default}}(i)}{\sum_i t_{\mathrm{auto}}(i)} = \frac{210.853}{40.940207} \approx 5.15.$$

This is a bounded same-binary benchmark result for `TAU_QELIM_BACKEND=auto`, not a global speed theorem. The measured corpus is policy-shaped, and residual-formula parity was checked semantically over the printed residual fragment.

Safe table update. The safe table fragment treats a table as $T : I \to \alpha$, where I is the key space and α is a Boolean-algebra value carrier. Pointwise revision is

$$\mathrm{Rev}_{G,A}(T)(i) := \big(G(i) \wedge A(i)\big) \vee \big(G(i)' \wedge T(i)\big).$$

For each key i, the revised table uses $A(i)$ inside guard $G(i)$ and the old value $T(i)$ inside $G(i)'$. This is safe in the checked recurrence fragment because G and A are fixed relative to the current recursive state. It is not permission to use same-stratum prime on the current table. The checked laws are monotonicity,

$$T \le U \Longrightarrow \mathrm{Rev}_{G,A}(T) \le \mathrm{Rev}_{G,A}(U),$$

and omega-continuity on increasing chains,

$$\mathrm{Rev}_{G,A}\left(\bigvee_{n<\omega} T_n\right) = \bigvee_{n<\omega} \mathrm{Rev}_{G,A}(T_n).$$

The Tau experiment implements a finite executable surface and symbolic helpers behind a feature flag, and checks table syntax against raw guarded-choice expansions.

Boundary. The results do not prove global replacement of Tau's default qelim route, correctness or speedup on all Tau inputs, unrestricted TABA tables, same-stratum prime in recursive table bodies, current-state-dependent guards without extra monotonicity conditions, full NSO or Guarded Successor lowering, or production integration into upstream Tau. The reusable lesson is

$$\mathrm{Optimization} = \mathrm{SemanticGuard} + \mathrm{ExactRoute} + \mathrm{Fallback} + \mathrm{Evidence}\,.$$

Table and qelim optimization should be planned as fragment dispatch. The fastest route depends on where the exploitable structure lives: syntax, CNF clauses, a compiled carrier, or a table representation.

References. Ohad Asor, *Theories and Applications of Boolean Algebras*, draft v0.25, 2024. R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. Computers*, 35(8), 1986. M. Davis and H. Putnam, "A Computing Procedure for Quantification Theory," *J. ACM*, 7(3), 1960. C. Zengler, A. Kuebler, and W. Kuechlin, "New Approaches to Boolean Quantifier Elimination," 2011. E. Goldberg and P. Manolios, "Quantifier Elimination by Dependency Sequents," *Formal Methods in System Design*, 45, 2014. H. Iwane and H. Anai, "Formula Simplification for Real Quantifier Elimination Using Geometric Invariance," ISSAC, 2017. E. C. Bitye Mvondo, Y. Cherruault, and J.-C. Mazza, "Global Optimization with Alpha-Dense Curves: Resolution of Boolean Equations," *Kybernetes*, 41, 2012. K. G. Papakonstantinou and G. Papakonstantinou, "A Nonlinear Integer Programming Approach for the Minimization of Boolean Expressions," *J. Circuits, Systems, and Computers*, 27(10), 2018.