NEUROFORM: NEURO-SYMBOLIC FORMALIZATION OF LEGAL RULES WITH COUNTEREXAMPLE-GUIDED REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

A generated formal specification can execute and be satisfiable while still misrepresenting its source rule—for example, by omitting an exception or an applicability condition. We study whether such behavioural errors can be detected and used to repair specifications at test time. NeuroFORM treats a language model as an untrusted proposal generator: given a legal clause and an expert-verified structured representation, it generates a candidate specification, compares its solver outcomes with expected outcomes on source-conditioned scenarios, and feeds failures back for bounded repair. The method operates on propositional encodings of obligations, permissions, prohibitions, exceptions, and boundary conditions. We evaluate it on 238 provisions from GDPR, CCPA, DPDP, and PIPL, with 2,322 span-level annotations and six behavioural scenario categories per provision. Across three independent runs, the full system achieves mean held-out behavioural accuracies of 78.43% with an LLM generator and 75.07% with an SLM generator, exceeding the strongest corresponding functional baselines by 14.77 and 20.33 percentage points, respectively. Among cases eventually repaired, 99.5% succeed by the second attempt within the successful repair invocation in both model tiers. These findings show how behavioural verifier failures can provide effective test-time feedback for generated symbolic specifications.

1 INTRODUCTION

Translating natural-language rules into executable formal specifications remains a challenging problem in neuro-symbolic AI. LLMs can translate natural language into formal representations, but a well-formed translation need not preserve the intended meaning (Wu et al., 2022; Pan et al., 2023). In our setting, a candidate may parse, execute, and be satisfiable while omitting an applicability condition or exception that changes its behaviour on a source-conditioned scenario. These failures are particularly consequential in legal domains, where obligations, permissions, prohibitions, and exceptions must preserve the semantics of the underlying provision. Despite advances in autoformalization, code generation, and legal reasoning (Wu et al., 2022; Pan et al., 2023; Blair-Stanek et al., 2023; Joshi et al., 2021), reliably aligning generated executable specifications with source provisions remains an open problem.

Grammar-constrained decoding improves structural validity (Geng et al., 2023), while self-refinement and solver-guided methods revise outputs using feedback (Madaan et al., 2023; Shinn et al., 2023; Pan et al., 2023; Orvalho et al., 2025). Legal solver systems likewise revise formalized rules against case outcomes (Chen et al., 2026; Hsia et al., 2025). We ask whether bounded verifier-guided repair improves agreement on disjoint scenarios and whether source-conditioned diagnostics add to generic failure feedback.

Under the DPDP Act §8(7)(a) (Parliament of India, 2023), erasure follows consent withdrawal or when it is reasonable to assume that the specified purpose is no longer served, unless retention is legally necessary. Abstracting from timing, $\phi_0 = (W \vee P) \Rightarrow E$ omits this exception; for $W = L = 1,\ P = E = 0$, it yields UNSAT, whereas $\phi_1 = ((W \vee P) \wedge \neg L) \Rightarrow E$ gives the expected SAT verdict (Figure 1). The example separates NeuroFORM's roles: the benchmark supplies the target verdict, Z3 computes candidate behaviour, and source-conditioned diagnosis proposes a repair. The solver

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1. Clause and annotation DPDP Act §8(7)(a): erase personal data after consent withdrawal W or when it is reasonable to assume that its specified purpose is no longer served P, unless retention is necessary to comply with law L.

2. Initial candidate $\phi_0 = (W \vee P) \Rightarrow E$. **Missing exception:** L.

3. Failing scenario $x : W = L = 1,\ P = E = 0$. Expected SAT; observed UNSAT under ϕ_0. **E4:** missing source-supported exception.

4. Targeted repair $\phi_1 = \big((W \vee P) \underbrace{\wedge \neg L}_{\text{added guard}}\big) \Rightarrow E$. Rechecking x under ϕ_1 gives SAT, as expected.

5. Acceptance and evaluation Acceptance requires agreement on all repair scenarios and passage through the model-based semantic gate. Disjoint scenarios are evaluated only after synthesis.

Figure 1: Illustrative encoding of the erasure obligation in the DPDP Act §8(7)(a), abstracting from timing; this is not a recorded model repair trace. E denotes erasure and L denotes retention necessary for compliance with law. Each displayed verdict tests $\Gamma(x) \wedge \phi_i$. The added exception guard is highlighted in blue and marked with an underbrace.

does not infer the missing legal condition; applying the framework elsewhere likewise requires an explicit scenario interface and independently justified expected outcomes.

NeuroFORM treats the language model as an untrusted proposal generator: candidates are checked on source-conditioned scenarios, failures drive bounded regeneration or repair, and a separate model-based semantic check gates acceptance. On 238 provisions with 2,322 annotations, three-run mean held-out accuracy is 78.43% (LLM) and 75.07% (SLM), exceeding the strongest corresponding functional baselines by 14.77 and 20.33 points, respectively; these are end-to-end comparisons with unmatched inference budgets.

Contributions. Our main contributions are:

- We formulate legal specification repair through a source-conditioned interface that distinguishes benchmark target verdicts, solver-observed behaviour, and source-based repair diagnoses.
- We introduce a benchmark of 238 provisions across four privacy regulations, with expert-verified annotations and disjoint repair and evaluation scenarios covering six behavioural categories. This protocol measures agreement beyond the scenarios used to guide synthesis.
- We evaluate the complete system and conduct separate controlled studies of repair mechanisms and semantic gating. A matched generic-versus-diagnostic comparison estimates the incremental effect of structured feedback and quantifies its uncertainty.

2 Problem Formulation

We formulate legal formalization as a specification-synthesis problem. Given a statutory clause s and its expert-verified structured representation $\hat{a}$, capturing obligations, prohibitions, permissions, exceptions, and boundary conditions, the synthesis task is to construct an executable specification φ in Z3 (de Moura & Bjørner, 2008) whose behaviour is evaluated for consistency with the source representation under the verification protocol. We denote the synthesis mapping by $(s, \hat{a}) \longmapsto \varphi$. Passing the internal behavioural, diagnostic, and model-based semantic checks determines whether a candidate is eligible to be returned; it does not establish equivalence to the statute. We evaluate behavioural agreement separately on withheld scenarios for the same provision. To distinguish the executable program object from the logical formula asserted to Z3, let F_φ denote the conjunction of the Boolean constraints constructed by candidate φ after successful initialization. All satisfiability expressions below are therefore written over F_φ rather than over the program object φ itself.

A concrete scenario is a total valuation $x : \mathcal{P}_s \to \{0, 1\}$ over the Boolean interface propositions $\mathcal{P}_s$ relevant to the tested rule and is represented by the conjunction $\Gamma(x)$:

$$\Gamma(x) = \left(\bigwedge_{p \in \mathcal{P}_s : x(p)=1} p \right) \wedge \left(\bigwedge_{p \in \mathcal{P}_s : x(p)=0} \neg p \right) \tag{1}$$

The scenario fixes the source-interface propositions used by the tested rule. Candidate-local Boolean symbols must be definitionally constrained rather than left free; the validation rules for such auxiliaries are given in the Appendix A.

The symbolic verifier is therefore defined as

$$\mathcal{V}(\varphi, x) = \begin{cases} \mathrm{Check}_{\mathrm{Z3}}(F_\varphi \wedge \Gamma(x)), & \text{if initialization, instantiation, and execution succeed,} \\ \texttt{error}, & \text{otherwise} \end{cases} \tag{2}$$

where $\mathrm{Check}_{\mathrm{Z3}}$ returns one of $\{\texttt{sat}, \texttt{unsat}, \texttt{unknown}\}$. Only `sat` and `unsat` are definitive verdicts. We use FAIL for synthesis failure and set $\mathcal{V}(\mathsf{FAIL}, x) = \texttt{error}$, so failed synthesis receives zero credit in evaluation. The implementation retains error provenance for diagnosis, as detailed in Appendix A.

Candidate specifications are evaluated over six scenario categories, $\mathcal{S} = \{s_{\mathrm{comp}}, s_{\mathrm{viol}}, s_{\mathrm{exc}}, s_{\mathrm{bound}}, s_{\mathrm{multi}}, s_{\mathrm{compound}}\}$ corresponding to Compliance, Violation, Exception, Boundary, Multi-Exception, and Compound-Boundary cases. Let $\kappa(x) \in \mathcal{S}$ denote the category of a concrete scenario x, and let each scenario have an expected verdict $y \in \{\texttt{sat}, \texttt{unsat}\}$. The internal repair harness is

$$h_{\mathrm{rep}}(s) = \{(x_i^{\mathrm{rep}}, y_i^{\mathrm{rep}})\}_{i=1}^{H}, \qquad H = 6 \tag{3}$$

with one repair scenario from each category, i.e., $\{\kappa(x_i^{\mathrm{rep}}) : i = 1, \ldots, 6\} = \mathcal{S}$. Each x_i^{rep} is instantiated through $\Gamma(x_i^{\mathrm{rep}})$ and evaluated by the symbolic verifier. Expected repair-scenario verdicts are assigned from the source clause and expert-verified annotation before the candidate is checked; they are not inferred from the candidate's solver output.

Scenario-associated verdict mismatches and non-definitive outcomes provide behavioural counterexamples; source-based missing-predicate or over-constraint diagnoses with a matching definitive verdict are diagnostic witnesses. Candidate-global initialization or execution failure is represented by the candidate-level diagnostic `E1`, whereas `E4v` denotes the candidate-level all-false structural warning; the complete diagnostic taxonomy is defined below and detailed in Section 3.2.2. The `E4v` warning may trigger bounded structural repair but, because it can also flag a valid inactive conditional, it is not itself a correctness failure. Behavioral agreement on the repair harness is

$$A_{\mathrm{beh}}(\varphi, h_{\mathrm{rep}}) = \frac{1}{H} \sum_{i=1}^{H} \mathbb{I}[\mathcal{V}(\varphi, x_i^{\mathrm{rep}}) = y_i^{\mathrm{rep}}] \tag{4}$$

Because every target verdict is either `sat` or `unsat`, an incorrect definitive verdict, `unknown`, or `error` contributes zero. Let $d(\varphi, s, \hat{a}, h_{\mathrm{rep}})$ be the primary diagnostic. A candidate is diagnostically admissible only after all repair scenarios receive their expected definitive verdicts. The admissible states are `E0` (no detected error) and `E4v` (an all-false structural warning). The latter may trigger bounded repair, but is not proof of vacuity and does not itself prohibit acceptance; the diagnostic states and precedence are detailed in Appendix A and Section 3.2.2. Let $g_{\mathrm{sem}}(\varphi, s, \hat{a}) \in \{0, 1\}$ be a separate, model-based semantic-consistency check. Return eligibility is

$$\mathrm{Accept}(\varphi) = \mathbb{I}[\varphi \neq \mathsf{FAIL} \ \wedge \ A_{\mathrm{beh}}(\varphi, h_{\mathrm{rep}}) = 1 \ \wedge \ d(\varphi, s, \hat{a}, h_{\mathrm{rep}}) \in \{\texttt{E0}, \texttt{E4v}\} \ \wedge \ g_{\mathrm{sem}}(\varphi, s, \hat{a}) = 1]. \tag{5}$$

This predicate specifies eligibility, not the order of regeneration, structural repair, and final checking.

Counterexample-guided repair. For a repair scenario whose observed verdict $\hat{y}_i = \mathcal{V}(\varphi, x_i^{\mathrm{rep}})$ differs from its expected verdict y_i^{rep}, the repair feedback records

$$c_i = (x_i^{\mathrm{rep}}, y_i^{\mathrm{rep}}, \hat{y}_i, d_i), \qquad \hat{y}_i \neq y_i^{\mathrm{rep}}, \tag{6}$$

where d_i is the associated diagnosis. This is a scenario-level behavioural counterexample; source-based diagnoses without a verdict mismatch are instead diagnostic witnesses. Candidate-global errors follow a separate repair path. A candidate that passes the behavioural checks but fails g_{sem} is rejected rather than repaired from a solver counterexample.

Held-out evaluation. For each provision s, the evaluation harness $h_{\text{eval}}(s)$ contains six scenarios, one per category. Its concrete valuations are disjoint from those in $h_{\text{rep}}(s)$ and are unavailable during synthesis. The same provision s and representation $\hat{a}$ remain available; "held-out" therefore refers to scenario instances, not unseen laws. For benchmark provisions $\mathcal{B}$ and synthesis outputs φ_s, overall behavioural accuracy is the scenario-level micro-average

$$A_{\text{overall}} = \frac{1}{6|\mathcal{B}|} \sum_{s \in \mathcal{B}} \sum_{j=1}^{6} \mathbb{I}\big[\mathcal{V}(\varphi_s, x_j^{\text{eval}}) = y_j^{\text{eval}}\big] . \tag{7}$$

A failed synthesis, `unknown`, or `error` contributes zero.

3 METHODOLOGY

NeuroFORM generates executable candidates from $(s, \hat{a})$, evaluates them on source-conditioned repair scenarios, and uses failures for bounded repair. Figure 1 illustrates the process, Algorithm 1 gives the search procedure, and implementation details appear in Appendix B.

3.1 ANNOTATION SCHEMA AND BENCHMARK CONSTRUCTION

We construct a benchmark of 238 legal provisions drawn from GDPR, CCPA, DPDP, and PIPL, comprising 2,322 span-level annotations. These regulations capture diverse legislative styles, jurisdiction-specific terminology, and deontic structures.

Deontic Annotation Schema The expert-verified input marks obligations, permissions, prohibitions, conditions, exemptions, entities, and references using a schema motivated by deontic logic Von Wright (1951). These are span-level annotations with descriptions and subsection structure, not a supplied executable Boolean formula. Candidate synthesis must compose the annotated conditions and exceptions with the operative rule. NeuroFORM evaluates the resulting propositional Z3 constraints through scenario verdicts; this operationalization is not a complete deontic semantics. The label mappings, treatment of permission queries, and handling of unresolved references are specified in Appendix B.

Verified annotation. An expert reviews and corrects the initial span annotations before synthesis. NeuroFORM receives the resulting representation $\hat{a}$, whose stored fields are

$$\hat{a} = \Big(\mathsf{section_id}, \mathsf{subsections}, \big\{(\mathsf{span}_j, \mathsf{label}_j, \mathsf{desc}_j)\big\}_{j=1}^{m}, \mathsf{label_metrics}\Big) \tag{8}$$

where m denotes the number of annotated spans associated with the provision. The field label_metrics contains annotation-level summary information derived from the identified labels and does not contain repair-harness or evaluation-harness outcomes.

The representation preserves subsection structure, labelled spans, and descriptions, but does not generally encode how conditions combine, which consequents they govern, or the scope and polarity of exceptions. NeuroFORM must infer these relationships from the clause, using $\hat{a}$ as structured guidance rather than an executable specification. The benchmark contains 238 provisions and 2,322 annotations from GDPR, CCPA, DPDP, and PIPL. Annotation procedures, the 30-section reliability check, and per-regulation counts appear in Appendix B.

3.2 NEUROFORM: FROM LEGAL CLAUSES TO EXECUTABLE SMT

Given a legal clause s and its expert-verified annotation $\hat{a}$, NeuroFORM performs candidate generation, symbolic verification, diagnostic counterexample-guided repair, semantic alignment, and acceptance. Algorithm 1 summarizes the procedure.

3.2.1 CANDIDATE GENERATION AND VERIFICATION

At each outer iteration, the generator proposes three Python/Z3 candidate programs from $(s, \hat{a})$. Candidates that initialize successfully declare scenario variables and assert constraints for evaluation. Generation settings and the ranking equation appear in Appendix B.

Algorithm 1 NeuroFORM test-time synthesis and repair.

Require: Clause s, verified annotation $\hat{a}$, repair harness h_{rep}
Ensure: Accepted specification or FAIL

for $t = 1, \ldots, T_{\max}$ **do**
  $\mathcal{C}_t \leftarrow \mathcal{G}_{\text{cand}}(s, \hat{a}; k)$
  for all $\varphi \in \mathcal{C}_t$ **do**
    $(S_\varphi, A_\varphi, D_\varphi) \leftarrow \textsc{Oracle}(\varphi, s, \hat{a}, h_{\text{rep}})$
  end for
  Stably sort $\mathcal{C}_t$ by decreasing S_φ ▷ Search order
  $\varphi_{\text{rep}} \leftarrow \bot$
  for all $\varphi \in \mathcal{C}_t$ **do**
    if $D_\varphi = \texttt{E1}$ **then** ▷ Candidate-level failure
      continue
    end if
    if $D_\varphi = \texttt{E0}$ **then** ▷ Repair scenarios agree
      if $g_{\text{sem}}(\varphi, s, \hat{a}) = 1$ **then**
        return φ
      end if
      continue
    end if
    if $\varphi_{\text{rep}} = \bot$ **then**
      $\varphi_{\text{rep}} \leftarrow \varphi$
    end if
  end for
  if $\varphi_{\text{rep}} \neq \bot$ **then**
    $\varphi' \leftarrow \textsc{DiagnosticRepair}(\varphi_{\text{rep}}, s, \hat{a}, h_{\text{rep}}, I_{\max})$
    if $\textsc{Accept}(\varphi') = 1$ **then** ▷ Recheck and align
      return φ'
    end if
    if $D_{\varphi_{\text{rep}}} = \texttt{E4v}$ **and** $\textsc{Accept}(\varphi_{\text{rep}}) = 1$ **then**
      return φ_{rep} ▷ Warning alone is non-fatal
    end if
  end if
end for
return FAIL

The verifier checks each candidate on six source-conditioned repair scenarios. For scenario x_i with expected verdict y_i, Z3 returns $\hat{y}_i = \mathcal{V}(\varphi, x_i)$. A definitive mismatch $y_i \neq \hat{y}_i$ exposes an error on that case; initialization failures and non-definitive outcomes are recorded separately. Acceptance requires complete repair-harness agreement and the additional checks below.

3.2.2 Counterexample-Guided Repair

A failed repair scenario supplies its assignment, expected verdict, and observed verdict. The solver establishes the observed verdict; a separate source-conditioned diagnostic stage uses $(s, \hat{a})$ and the candidate to propose a repair target. Figure 1 illustrates a candidate that returns UNSAT despite the legal-retention exception; an E4 diagnosis would target the omitted exception. The repair model then receives the scenario mismatch and source-conditioned localisation, proposes a revision, and the same repair harness is rerun.

Operational failures trigger regeneration. Scenario-level errors trigger bounded repair. An all-false E4v pattern can trigger a bounded structural-repair attempt, but is only a warning: it can also arise from an inactive valid implication and does not by itself prevent acceptance. The full taxonomy, precedence, localization tests, and feedback record are given in Appendix B.4.

A persistent E4v warning remains acceptance-admissible after its bounded repair opportunity, provided all repair scenarios still have the expected verdicts.

3.2.3 Semantic Alignment and Acceptance

After behavioural verification, a separate fixed-decoding model checks the candidate against the clause and verified annotation. This Boolean gate rejects negative, missing, and unparseable re-

sponses. It is a model-based consistency check, not symbolic verification or a proof of legal equivalence. Gate rejection does not produce a solver counterexample; the search continues to another candidate. Parsing rules and failure handling are specified in Appendix B.

Generation and repair use fixed outer and inner budgets. We report observed repair convergence and computation in the experiments; solver-call and working-space bounds are given in Appendix B.

3.3 Evaluation Protocol

For each provision, the returned specification is evaluated on six categories-Compliance, Violation, Exception, Boundary, Multi-Exception, and Compound-Boundary with one repair scenario and one disjoint evaluation scenario per category. Evaluation scenarios are withheld from generation, verification, repair, and semantic alignment, so "held-out" denotes new scenario instances for seen provisions, not unseen laws.

Expected verdicts were derived from the source clause and verified annotation without access to generated specifications or failure traces; complex exception and boundary cases received additional expert review. Because evaluation cases were finalized after initial synthesis, output blinding limits conditioning on observed failures but does not establish a prospectively fixed protocol. We report scenario-level behavioural accuracy A_{overall}, category accuracy, and synthesis acceptance separately; primary results are means and sample standard deviations over three runs.

4 Experiments and Results

We evaluate 238 provisions from GDPR, CCPA, DPDP, and PIPL. Each provision has six evaluation scenarios with assignments disjoint from its repair scenarios, yielding 1,428 held-out cases per run (Section 3.3).

For generation and repair, the SLM and LLM configurations use Qwen/Qwen2.5-Coder-7B-Instruct and Qwen/Qwen2.5-Coder-32B-Instruct, respectively (Hui et al., 2024). Both use microsoft/phi-4 (Abdin et al., 2024) for fixed-decoding semantic alignment. The nominal search generates $k = 3$ candidates per outer iteration, with $T_{\max} = 7$ outer iterations and up to $I_{\max} = 7$ repair steps per repair invocation.

Generation and decoding settings appear in Appendix B.3; budget sweeps appear in Appendix D.5.

The primary benchmark reports means over independent seeds 22, 44, and 76. Statistical validation and controlled component studies were run in separate stochastic blocks. We compare configurations within each block and do not interchange their absolute point estimates. Generation and decoding settings appear in Appendix B.3; repair-budget and candidate-list sweeps appear in Appendix D.5.

4.1 Baselines and Primary Outcome

We compare against seven task-specific functional baselines: direct Zero-Shot generation (B1), Few-Shot generation with three exemplars (B2), CoT-Retry (B3), Grammar-Constrained generation (B4) (Geng et al., 2023), Naive CEGIS with generic verdict feedback but no diagnostic localization (B5), a propositional adaptation inspired by AquaForte (B6) (Lv et al., 2026), and a fine-tuned generator (B8). These are our implementations under the benchmark interface, not scores reported in the cited papers. B7 L4L-Schema and B9 RAG-SMT are best-effort adaptations with low executability under that interface; their low-executability scores appear in Appendix D.1 and are excluded from the functional-baseline ranking. Inference budgets are not fully matched, so these comparisons measure end-to-end system differences rather than the causal effect of an individual NeuroFORM component.

The primary outcome is held-out scenario-level accuracy A_{overall} (Eq. 7). Synthesis acceptance and repair-completion rates are reported separately with their respective denominators.

4.2 Overall Behavioural Fidelity

Table 1 reports overall held-out scenario-level behavioural accuracy across three independent runs on 238 provisions. Per-regulation results appear in the appendix.

Table 1: Held-out scenario-level behavioural accuracy (%), averaged over three runs. All rows are evaluated on the same benchmark; the strongest functional baseline in each tier is underlined.

Method	Key distinction	SLM	LLM
NeuroFORM	Diagnostic repair	**75.07**	**78.43**
B1 Zero-Shot	Direct generation	50.56	57.42
B2 Few-Shot	Three exemplars	46.85	<u>63.66</u>
B3 CoT-Retry	Reasoning and retry	<u>54.74</u>	58.54
B4 Grammar	Constrained decoding	39.43	48.60
B5 Naive CEGIS	Generic verdict feedback	51.82	56.93
B6 AquaForte-inspired	Propositional adaptation	52.24	58.89
B8 Fine-Tuned	Supervised generator	52.59	56.30

NeuroFORM exceeds the strongest functional baseline by 20.33 percentage points with the SLM (75.07 versus 54.74) and 14.77 points with the LLM (78.43 versus 63.66). These are three-run end-to-end comparisons; they do not identify which component causes the gain. The regulation-level breakdown and the B7/B9 reproduction results are reported in the appendix.

4.3 Where Does Behavioural Fidelity Fail?

NeuroFORM performs best on Violation cases ($98.3\%/98.7\%$ for SLM/LLM) and worst on Compound-Boundary cases ($60.1\%/60.5\%$). Boundary cases are also difficult ($60.9\%/64.3\%$). These category results locate residual errors in cases with interacting applicability conditions; they do not by themselves identify which condition was mistranslated. The complete category-by-baseline breakdown is reported in Table 5 in the appendix.

Structural-warning repair. Disabling the complete E4v-guided structural-repair mechanism reduces mean held-out accuracy from 75.07% to 56.20% for the SLM and from 78.43% to 59.94% for the LLM (Table 6, appendix). This ablation removes both the warning and its associated repair opportunity. It establishes the utility of that combined mechanism, not the information value of the E4v warning alone.

Paired benchmark comparison. In a separate paired statistical-validation realization over 238 provisions, NeuroFORM exceeds the strongest functional baseline by 20.53 points for the SLM (95% paired-bootstrap CI [16.24, 24.82]) and 14.78 points for the LLM (CI [10.92, 18.70]). These paired estimates are not the differences between the independently averaged three-run means in Table 1. The acceptance-rate tests, repair-completion definition, and full statistical results are reported in the appendix.

4.4 What Does Test-Time Feedback Contribute?

Repair and convergence. In the repair-budget study, synthesis acceptance is 14.29%/19.33% (SLM/LLM) without repair and 95.80%/93.70% with at most one repair step *per outer iteration*. This measures internal acceptance, not held-out accuracy. Separately, among cases eventually repaired successfully, 203/215 SLM cases (94.4%) and 200/216 LLM cases (92.6%) succeeded on the first repair. By the second repair, the corresponding counts were 214/215 and 215/216 (99.5% in each tier).

These attempt counts are measured within the successful DiagnosticRepair invocation; the counter resets when a new outer iteration invokes repair. The fractions condition on eventual repair success, not on all cases entering repair or on total repair calls across the run.

Generic versus diagnostic feedback. In a separate matched comparison, Generic repair achieves 53.15% held-out accuracy for the SLM versus 56.93% with Diagnostic repair ($\Delta = +3.78$ points; 95% paired-bootstrap CI $[-0.98, +8.61]$). The corresponding LLM values are 55.32% and 56.72% ($\Delta = +1.40$; CI $[-2.52, +5.39]$). Structured diagnostics have positive point-estimate effects in both tiers, but neither interval excludes zero. The comparison therefore does not establish superiority over generic feedback (Figure 2c).

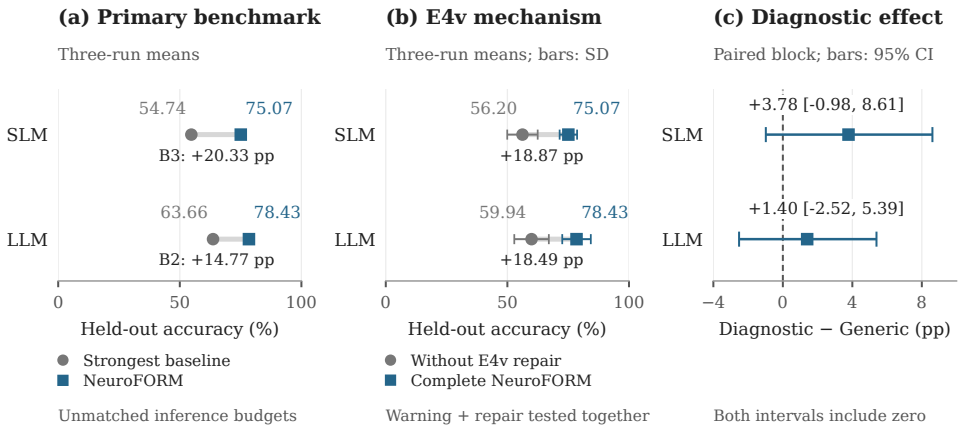

Figure 2: Summary of the primary and component-level results. (a) Held-out behavioural accuracy of NeuroFORM and the strongest functional baseline in each model tier, averaged over three runs. (b) Effect of the complete E4v-guided structural-repair mechanism; error bars show standard deviations across three runs. (c) Matched comparison of diagnostic versus generic repair feedback; error bars show 95% paired-bootstrap confidence intervals. Primary benchmark inference budgets are not fully matched, the E4v ablation tests the warning together with its repair opportunity, and both diagnostic-effect confidence intervals include zero.

Semantic-alignment gate. A separate paired ON/OFF block finds positive held-out accuracy differences when the model-based gate is enforced: +22.74 points for the SLM and +4.12 for the LLM. Gate rejection can change later candidate selection, so this is an end-to-end gate ablation, not a post-hoc filter test or a proof of semantic correctness. The block's absolute values are not compared with the three-run headline means; full results appear in Table 10 in the appendix.

The remaining errors warrant trace-level analysis of compound boundaries and cases in which the supplied local clause lacks necessary context. Our conclusions concern scenario-level behavioural agreement for the evaluated provisions, not complete legal equivalence or transfer to unseen laws.

What verifier feedback establishes. Failed source-conditioned scenarios provide useful test-time repair signals; among eventually successful cases, most succeed by the second attempt within their successful DIAGNOSTICREPAIR invocation. In the matched comparison, structured diagnostics have positive point-estimate gains over generic feedback, but the confidence intervals include zero. The evidence therefore supports verifier-guided repair more strongly than it supports the incremental value of the diagnostic taxonomy. An executable or internally accepted specification may still disagree with a disjoint scenario: the solver determines the verdict for a tested assignment, not equivalence to the source provision.

5 RELATED WORK

Formal generation and symbolic checking. Grammar-constrained generation improves output well-formedness (Geng et al., 2023), while Synchromesh constrains decoding with target-language syntax, scope, typing, and contextual rules (Poesia et al., 2022). Logic-LM translates natural language into symbolic formulations and uses solver error messages for refinement (Pan et al., 2023). Logic-LM++ extends this line of work with multi-step refinement and comparison of proposed symbolic formulations (Kirtania et al., 2024). CLOVER addresses complex natural-language-to-logic translation through compositional generation and solver-assisted comparison of candidate formulas (Ryu et al., 2025). LIPS illustrates a complementary division of labour between neural proposals and symbolic reasoning in proof search (Li et al., 2025). These works use constraints and symbolic reasoning at different stages, including decoding, candidate comparison, proof search, and refinement.

Counterexample-guided repair. Self-refinement methods revise model outputs using feedback (Madaan et al., 2023; Shinn et al., 2023), and counterexample-guided inductive synthesis (CEGIS) already uses failed checks to refine candidate programs (Solar-Lezama, 2008). Orvalho et al. combine MaxSAT fault localisation, LLM completion, and test-suite counterexamples in an iterative repair loop (Orvalho et al., 2025). Thus, neither repairing an executable program after a failed test nor localising a program fault is new in itself. NeuroFORM studies a specific feedback interface for generated legal specifications: source-conditioned scenarios supply assignments and expected SAT/UNSAT verdicts, the solver supplies observed verdicts, and repair uses either generic failures or source-conditioned diagnoses. We evaluate the incremental value of those diagnostics separately.

Legal formalisation and legal solver systems. Catala supports the construction of executable legal programs, and DAPRECO represents GDPR provisions as formal deontic rules (Merigoux et al., 2021; Robaldo et al., 2020). L4L autoformalizes statutory rules, tests them against cases, and describes revising constraints when case outcomes or solver checks expose a problem (Chen et al., 2026). Hsia et al. regenerate legal SMT constraints when a case returns an unexpected solver verdict; a subsequent optimization stage identifies factual changes for compliance (Hsia et al., 2025). Thus, correcting a legal encoding after a case-level verdict mismatch has precedents. NeuroFORM evaluates generated propositional specifications on a fixed source-conditioned scenario protocol, separates repair instances from evaluation instances, and compares generic with structured diagnostic feedback under matched conditions. Agreement on these scenarios is a bounded behavioural measure, not proof of equivalence to the source law.

6 Discussion and Conclusion

What the component studies establish. NeuroFORM separates solver-observed behaviour from source-conditioned repair guidance and model-based semantic acceptance. The experiments support these design choices at different levels. Removing the complete E4v-guided structural-repair mechanism reduces mean held-out accuracy by 18.87 and 18.49 percentage points for the SLM and LLM, respectively (Table 6). This tests the warning together with its repair opportunity. Separately, the matched feedback comparison yields positive diagnostic point estimates but confidence intervals that include zero (Table 9). Thus, the evidence supports the utility of the combined structural-repair mechanism under the tested settings, while leaving the incremental advantage of structured diagnoses over generic feedback unresolved.

Limitations. Our specifications use a propositional scenario interface and do not capture complete deontic, temporal, or cross-provision legal semantics. Evaluation uses finite, disjoint scenarios for provisions seen during synthesis; it cannot establish legal equivalence or generalization to unseen provisions. Expected verdicts requiring context absent from both the supplied clause and annotation may not be recoverable from the synthesis input. The semantic-alignment gate is model-based, not a formal guarantee. NeuroFORM also receives an expert-verified annotation and uses a finite repair budget, limiting what these results establish about formalization directly from raw statutory text. A restricted deterministic annotation-to-Z3 compiler produced specifications for 19 of 238 provisions; this measures the coverage of its fixed mapping rules, not a limit on what a more capable deterministic compiler could recover (Appendix B.2).

Conclusion. An executable specification can still misrepresent its source rule. NeuroFORM uses source-conditioned scenario verdicts to expose such errors and guide bounded test-time repair. The results support verifier-guided repair, while the matched comparison does not establish an additional held-out advantage for structured diagnostics over generic feedback. The resulting fidelity is bounded by the scenarios checked. The approach may extend to access-control policies, eligibility rules, and software requirements with explicit scenario interfaces and independently justified target outcomes. Its applicability is limited where outcomes depend on unresolved context or discretionary judgments; transfer beyond the evaluated legal provisions remains untested.

AI USE STATEMENT

We used generative AI tools to critique the research framing, methodology, experimental comparisons, interpretation, figures, and manuscript drafts, and to assist with language editing and reference checking. As part of the research, the OpenAI API generated preliminary structured annotations; a human reviewer with legal working experience reviewed and corrected them. Language models are also components of NeuroFORM's candidate generation and auxiliary semantic-alignment procedures, as described in the Method section. The authors reviewed AI-assisted material and take responsibility for the manuscript, annotations, and reported results.

REPRODUCIBILITY STATEMENT

Benchmark construction and the scenario protocol are described in Section 3.1 and Appendix C. The synthesis, verification, repair, semantic-alignment, and acceptance procedures are described in Section 3.2 and Appendix B; detailed results and matched comparisons appear in Appendix D. Supplementary materials document the prompts, model and solver configurations, evaluation instances, and analysis procedures used in the reported experiments.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, et al. Phi-4 technical report. Technical Report MSR-TR-2024-57, Microsoft Research, December 2024. URL https://www.microsoft.com/en-us/research/publication/phi-4-technical-report/.

Alexander Blair-Stanek, Nils Holzenberger, and Benjamin Van Durme. Can GPT-3 perform statutory reasoning? In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, pp. 22–31, 2023.

California Legislature. California Civil Code, Section 1798.105, 2018. URL https://leginfo.legislature.ca.gov/faces/codes_displaySection.xhtml?lawCode=CIV&sectionNum=1798.105. California Legislative Information.

Linze Chen, Yufan Cai, Zhe Hou, and Jin Song Dong. Towards trustworthy legal AI through LLM agents and formal reasoning, 2026. URL https://arxiv.org/abs/2511.21033v2. Version 2, revised 5 March 2026.

Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340. Springer, 2008. doi: 10.1007/978-3-540-78800-3_24.

European Parliament and Council of the European Union. Regulation (EU) 2016/679 (General Data Protection Regulation), 2016. URL https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10932–10952. Association for Computational Linguistics, 2023.

Yung-Shen Hsia, Fang Yu, and Jie-Hong Roland Jiang. Neuro-symbolic compliance: Integrating LLMs and SMT for automated financial legal analysis. In *2025 2nd IEEE/ACM International Conference on AI-Powered Software (AIware)*, pp. 1–10. IEEE, 2025. doi: 10.1109/AIware69974.2025.00010.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

Vivek Joshi, P. R. Anish, and S. Ghaisas. Domain adaptation for an automated classification of deontic modalities in software engineering contracts. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1275–1280, 2021.

Shashank Kirtania, Priyanshu Gupta, and Arjun Radhakrishna. LOGIC-LM++: Multi-step refinement for symbolic formulations. In *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations*, pp. 56–63. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.nlrse-1.6/.

Zenan Li, Zhaoyu Li, Wen Tang, Xian Zhang, Yuan Yao, Xujie Si, Fan Yang, Kaiyu Yang, and Xiaoxing Ma. Proving Olympiad Inequalities by Synergizing LLMs and Symbolic Reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/06b35bf96174d97bae899d915a8b7210-Abstract-Conference.html.

Kunhang Lv, Yuhang Dong, Rui Han, Fuqi Jia, Feifei Ma, and Jian Zhang. LLM-guided quantified SMT solving over uninterpreted functions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(17):14304–14312, 2026. doi: 10.1609/aaai.v40i17.38445.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023.

Denis Merigoux, Nicolas Chataing, and Jonathan Protzenko. Catala: A programming language for the law. *Proceedings of the ACM on Programming Languages*, 5(ICFP):1–29, 2021.

Pedro Orvalho, Mikoláš Janota, and Vasco M. Manquinho. Counterexample guided program repair using zero-shot learning and MaxSAT-based fault localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 649–657, 2025.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.248.

Parliament of India. The Digital Personal Data Protection Act, 2023 (Act No. 22 of 2023), 2023. URL https://www.indiacode.nic.in/bitstream/123456789/22037/2/a2023-22.pdf.

Gabriel Poesia, Alex Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable Code Generation from Pre-Trained Language Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KmtVD97J43e.

Livio Robaldo, Cesare Bartolini, Monica Palmirani, Arianna Rossi, Michele Martoni, and Gabriele Lenzini. Formalizing GDPR provisions in reified I/O logic: The DAPRECO knowledge base. *Journal of Logic, Language and Information*, 29(4):401–449, 2020.

Hyun Ryu, Gyeongman Kim, Hyemin S. Lee, and Eunho Yang. Divide and Translate: Compositional First-Order Logic Translation and Verification for Complex Logical Reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3e592c571de69a43d7a870ea89c7e33a-Abstract-Conference.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023.

Armando Solar-Lezama. *Program Synthesis by Sketching*. PhD thesis, University of California, Berkeley, 2008. URL https://www2.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-177.html.

Standing Committee of the National People's Congress. Personal Information Protection Law of the People's Republic of China, 2021. URL https://www.npc.gov.cn/npc/c2597/c5854/bfflywwb/202311/t20231117_433007.html. English text published by the National People's Congress.

Georg Henrik Von Wright. Deontic logic. *Mind*, 60(237):1–15, 1951.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-2344.

A Scenario Semantics and Diagnostic Implementation

Candidate symbols and validation. For a provision s, let $\mathcal{P}_s$ be the declared Boolean source interface. A concrete scenario $x : \mathcal{P}_s \to \{0,1\}$ fixes every interface proposition through $\Gamma(x)$, defined in Eq. 1. Let $\mathcal{Q}_\varphi$ be the Boolean symbols occurring in the constraints F_φ asserted by an executable candidate. We decompose them as

$$\mathcal{Q}_\varphi = (\mathcal{Q}_\varphi \cap \mathcal{P}_s) \dot\cup \mathcal{A}_\varphi, \qquad \mathcal{A}_\varphi := \mathcal{Q}_\varphi \setminus \mathcal{P}_s.$$

Every symbol in $\mathcal{Q}_\varphi \cap \mathcal{P}_s$ must use the declared source-interface identity. An auxiliary $q \in \mathcal{A}_\varphi$ is permitted only when an asserted, acyclic defining equivalence $q \leftrightarrow f_q$ defines it in terms of interface propositions or previously defined auxiliaries. Each auxiliary must have a non-conflicting definition. Under these conditions, a complete valuation of $\mathcal{P}_s$ uniquely determines the auxiliaries. Undeclared, cyclically defined, multiply conflicting, or unconstrained symbols outside the interface are validation errors. These syntactic and definitional checks do not establish that the constraints express the statutory clause.

Verifier outcomes and error provenance. The executable program φ constructs F_φ; the solver checks $F_\varphi \wedge \Gamma(x)$, as defined in Eq. 2. Only `sat` and `unsat` are definitive verdicts. An `unknown` solver result and an `error` in initialization, instantiation, or execution do not establish a logical verdict. The implementation retains provenance beyond the four-valued result of $\mathcal{V}$: candidate-level initialization or diagnostic failures are distinguished from scenario-local solver or API failures when assigning the E1/E2 classes. This classification cannot be recovered from `error` alone. The benchmark supplies each expected scenario verdict $y \in \{\texttt{sat}, \texttt{unsat}\}$; Z3 computes an observed verdict for the candidate and does not infer y from the statutory text.

Repair scenarios and diagnostic states. The internal harness $h_{\text{rep}}(s)$ in Eq. 3 contains exactly one concrete scenario per category in $\mathcal{S}$. Its behavioural agreement is A_{beh} from Eq. 4. The diagnostic classes and their roles are

$$\mathcal{D}_{\text{cls}} = \{\texttt{E0}, \texttt{E1}, \texttt{E2}, \texttt{E3}, \texttt{E4}, \texttt{E4v}, \texttt{E5}\}. \tag{9}$$

$$\begin{aligned} \mathcal{D}_{\text{err}} &= \{\texttt{E1}, \texttt{E2}, \texttt{E3}, \texttt{E4}, \texttt{E5}\}, \\ \mathcal{D}_{\text{warn}} &= \{\texttt{E4v}\}, \\ \mathcal{D}_{\text{acc}} &= \{\texttt{E0}, \texttt{E4v}\}. \end{aligned} \tag{10}$$

The warning set is a subset of the acceptance-admissible set; these sets describe roles rather than a partition. The primary diagnostic $d(\varphi, s, \hat{a}, h_{\text{rep}})$ follows the precedence rules in Section 3.2.2. In particular, E4v is considered only after all repair scenarios have their expected definitive verdicts. Accordingly,

$$d(\varphi, s, \hat{a}, h_{\text{rep}}) \in \mathcal{D}_{\text{acc}} \implies A_{\text{beh}}(\varphi, h_{\text{rep}}) = 1. \tag{11}$$

Equivalently, $g_{\text{diag}}(\varphi, s, \hat{a}, h_{\text{rep}}) := \mathbb{I}[d(\varphi, s, \hat{a}, h_{\text{rep}}) \in \mathcal{D}_{\text{acc}}]$ implies full repair-harness agreement. The E4v state reports an all-false structural pattern. It can also arise from a valid inactive implication and is not proof of vacuity. It may initiate a bounded structural-repair attempt without by itself prohibiting eventual acceptance.

Repair selection and feedback. Let $\mathcal{D}_{\text{scn}}$ denote diagnostic classes associated with affected scenarios under Section 3.2.2, and let $\mathcal{D}_{\text{cand}}$ denote candidate-global diagnostics. For a scenario-associated class d, let $\mathcal{I}_d(\varphi)$ be the nonempty set of affected repair-harness indices specified by the class-specific rules in Appendix B.4. With the fixed harness order, the implementation selects $i^\star = \min \mathcal{I}_d(\varphi)$. On a verdict mismatch or non-definitive scenario outcome, the feedback records the selected scenario, its expected and observed outcomes, and the diagnosis as in Eq. 6. A source-based diagnosis with a matching definitive verdict is a diagnostic witness, rather than a SAT/UNSAT counterexample. Candidate-global failures need not have a unique scenario index. Repaired candidates are checked again on the same repair harness.

The implementation records a numerical score $\sigma_{\text{impl}}(\varphi, h_{\text{rep}})$ for ranking, regeneration, and repair tracking; its operational definition is given in Section 3.2.1. The score is not an independent correctness criterion. If the behavioural and diagnostic checks pass but the model-based semantic gate fails, the candidate is rejected and the next generated candidate is considered; that gate failure is not an execution-derived counterexample. The acceptance predicate in Eq. 5 determines return eligibility, not candidate-selection order. A candidate with only an `E4v` warning can receive a bounded structural-repair opportunity before final semantic checking; a persistent warning remains eligible afterward.

Evaluation scenarios and denominators. For each provision s, the held-out evaluation harness contains exactly six labelled scenarios, one from each category:

$$h_{\text{eval}}(s) = \{(x_j^{\text{eval}}, y_j^{\text{eval}})\}_{j=1}^{6}, \qquad \{\kappa(x_j^{\text{eval}}) : j = 1, \ldots, 6\} = \mathcal{S}. \tag{12}$$

Repair and evaluation scenarios cover the same categories but use disjoint concrete valuations. Writing $X_{\text{rep}}(s) = \{x : (x, y) \in h_{\text{rep}}(s)\}$ and $X_{\text{eval}}(s) = \{x : (x, y) \in h_{\text{eval}}(s)\}$, the stated separation is

$$X_{\text{rep}}(s) \cap X_{\text{eval}}(s) = \varnothing. \tag{13}$$

The clause s and expert-verified representation $\hat{a}$ are available during synthesis. Evaluation scenarios are withheld from the candidate generator, repair procedure, symbolic checks, and semantic gate until final evaluation. Thus, "held-out" means unseen concrete scenarios for previously seen provisions, not held-out legal provisions or necessarily unseen scenario templates. For synthesis output φ_s, the per-provision metric is

$$A_{\text{eval}}(\varphi_s, h_{\text{eval}}(s)) = \frac{1}{6} \sum_{j=1}^{6} \mathbb{I}\big[\mathcal{V}(\varphi_s, x_j^{\text{eval}}) = y_j^{\text{eval}}\big]. \tag{14}$$

For benchmark provision set $\mathcal{B}$, the denominator of the overall scenario-level micro-average in Eq. 7 is $N_{\text{eval}}^{\text{tot}} = 6|\mathcal{B}|$. With $\varphi_s = \mathsf{FAIL}$ for failed synthesis and $\mathcal{V}(\mathsf{FAIL}, x) = \texttt{error}$, each failure contributes zero. Likewise, an `unknown` or `error` outcome contributes zero because every expected evaluation verdict is definitive.

B NeuroFORM: Annotation, Verification, and Repair Details

B.1 Operational annotation mappings

The stored representation $\hat{a}$ is defined in Eq. 8. The following mappings define task-specific propositional scenario checks; they do not establish a general semantics of legal obligation, permission, prohibition, or statutory exceptions.

Motivated by classical deontic logic Von Wright (1951), the annotation schema contains seven core labels together with auxiliary labels. The core schema includes the deontic modalities shown in Table 2, together with condition, exemption, entity, and reference labels. Auxiliary annotations provide additional contextual and descriptive information associated with the identified spans.

For executable scenario checking, the core annotations are mapped to propositional Z3 constraints. These mappings are task-specific operational encodings used for executable scenario evaluation and are not intended to constitute a complete formal semantics of deontic or statutory reasoning. In particular, a conditional obligation $O(\phi \mid \psi)$ is encoded as $\text{Enc}_{\text{Z3}}(O(\phi \mid \psi)) = \text{Implies}(\psi, \phi)$,

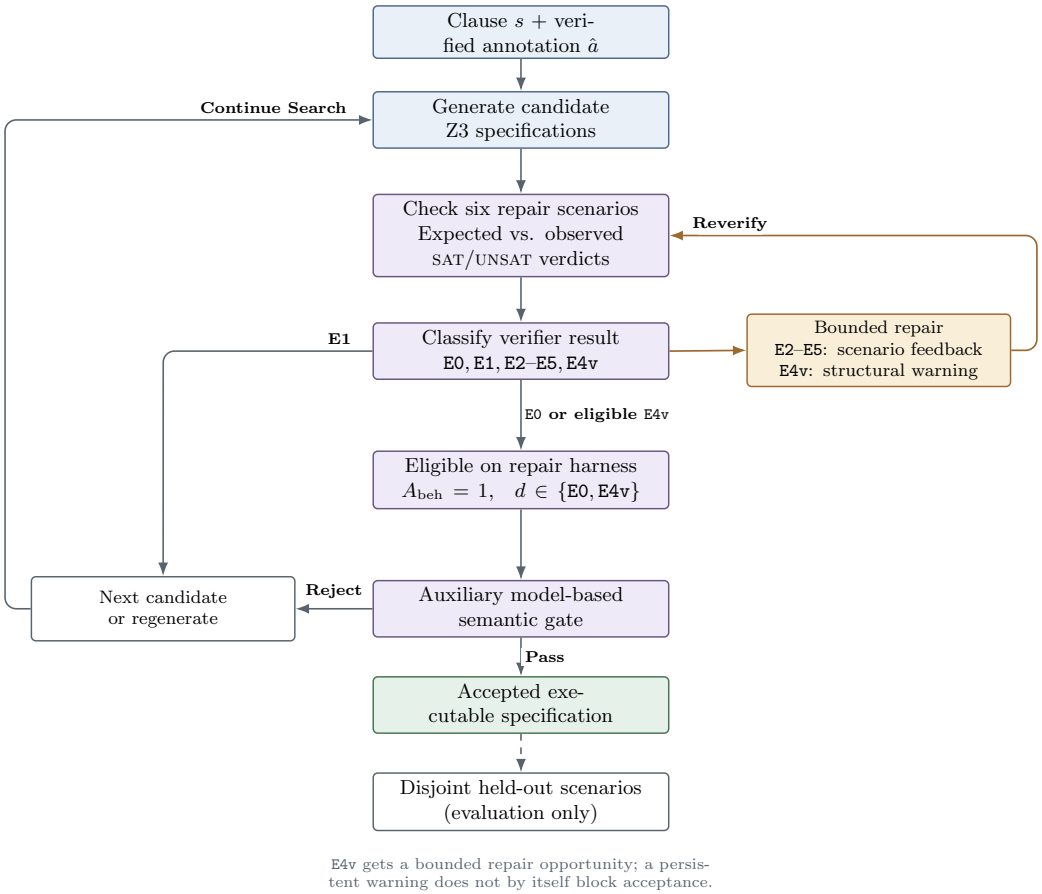

Figure 3: Overview of NeuroFORM's bounded synthesis and repair procedure. Candidates are checked on the repair harness; diagnostic failures guide repair and re-verification. The model-based semantic gate is applied before acceptance, and disjoint scenarios are used only for final evaluation. An `E4v` warning is nonfatal.

where ψ denotes the activating condition and ϕ denotes the required consequent. An unconditional prohibition is represented using `Not`; a conditional prohibition $F(\phi \mid \psi)$ is represented as $\mathrm{Implies}(\psi, \mathrm{Not}(\phi))$. Explicitly permitted alternatives are represented using `Or` in permission-query scenarios.

Table 2: Operational mapping from core annotation labels to executable Z3 constraints.

Label	Operational role	Operational Z3 form
`#obligation`	$O(\phi \mid \psi)$ under condition ψ	`Implies(`ψ`,`ϕ`)`
`#permission`	explicitly permitted alternatives	`Or(`ϕ_1`, . . . ,`ϕ_m`)`
`#prohibition`	prohibition of ϕ	`Not(`ϕ`)`
`#condition`	activating guard ψ	antecedent of `Implies`
`#exemption`	exemption guard E	`Or(`E`, Implies(`ψ`,`ϕ`))`
`#entity`	named actor or legal/data predicate	proposition / predicate variable
`#reference`	resolved cross-provision dependency	resolved external constraint

For an obligation subject to an exemption E, the operational encoding is

$$\mathrm{Enc}_{\mathrm{Z3}}(O(\phi \mid \psi), E) = \mathrm{Or}(E, \mathrm{Implies}(\psi, \phi)) \tag{15}$$

Thus, $E = \mathrm{true}$ satisfies the exemption branch and deactivates the conditional obligation, whereas $E = \mathrm{false}$ requires $\mathrm{Implies}(\psi, \phi)$ to hold. Propositionally, this encoding is equivalent to $\mathrm{Implies}(\psi \wedge \neg E, \phi)$, which makes explicit that the obligation is active exactly when the activating condition holds and the exemption does not.

Conditional permissions are treated through a task-specific permission-query operationalization and are not claimed to provide a general deontic semantics for permission or to distinguish weak from strong permission. For a permission-query scenario with activating condition ψ, we use $\mathrm{Enc}_{\mathrm{Z3}}(P(\phi_1, \ldots, \phi_m \mid \psi)) = \mathrm{Implies}(\psi, \mathrm{Or}(\phi_1, \ldots, \phi_m))$. The unconditional query is obtained by setting $\psi = \mathrm{true}$. Each ϕ_j denotes the truth value of an explicitly permitted alternative in the concrete evaluation assignment. Under Eq. 1, the scenario fixes ψ and every ϕ_j. When ψ is true, the constraint is satisfiable exactly when at least one explicitly permitted alternative is true; when ψ is false, the permission condition is inactive. This encoding is therefore a task-specific query about the listed alternatives. It does not represent general permission: an active assignment with every alternative false is unsatisfiable under this formula, even though permission alone would not require an action. The evaluation protocol must interpret these scenarios as queries about selected alternatives rather than as a general test of permissible inaction.

A `#reference` annotation identifies a cross-provision dependency. An external constraint is incorporated only when the referenced context is explicitly available and resolved. NeuroFORM does not perform unrestricted cross-provision retrieval or reasoning; unresolved external dependencies can therefore contribute to the context-insufficiency errors analyzed later.

B.2 Annotation procedure and corpus composition

Each legal provision s is first processed using an in-context prompted language model to obtain an initial annotation, $\tilde{a} = \mathcal{M}(s; \mathcal{E}, \mathcal{T}, \mathcal{G})$ where $\mathcal{E}$ denotes jurisdiction-specific exemplars, $\mathcal{T}$ denotes the annotation taxonomy, and $\mathcal{G}$ denotes the graph-of-thought prompting template. Here, $\tilde{a}$ denotes the raw language-model annotation prior to expert verification.

The generated annotation is subsequently reviewed and corrected by an expert to obtain the verified structured deontic representation $\hat{a} = \mathcal{H}_{\mathrm{ann}}(s, \tilde{a})$, where $\mathcal{H}_{\mathrm{ann}}$ denotes the expert verification and correction procedure. The resulting expert-verified representation $\hat{a}$, rather than the raw annotation $\tilde{a}$, is supplied to NeuroFORM during specification synthesis.

The verified record $\hat{a}$ and its stored fields appear in Eq. 8 of the main paper.

Deterministic compilation probe. After inspecting the expert-annotated dataset structure, we fixed a rule-based annotation-to-Z3 compiler before comparing its outputs with scenario verdicts. It emitted formulas for 19 of 238 provisions (7.98%). Among the 219 provisions without output,

138 had unresolved logical structure and 81 encountered annotation/schema or compiler-coverage limitations. These counts describe this compiler's coverage; they do not establish that all 219 provisions are intrinsically impossible to compile deterministically.

The annotation workflow comprises structural parsing, model-based span/entity and deontic annotation, followed by expert verification and correction. We distinguish the automatically generated representation $\tilde{a}$ from the expert-verified representation $\hat{a}$ used by NeuroFORM. For reliability assessment, an additional annotator independently re-annotated 30 sections without access to the verified labels. Agreement required matching annotation classes and span boundaries within ± 10 tokens. The subset achieved 81.3% span agreement and 0.82 token-level macro F1 across the seven core classes. These values characterize only the re-annotated subset, not corpus-wide agreement. The benchmark spans four complementary privacy regulations: GDPR, CCPA, DPDP, and PIPL (Table 3). The statutory source texts were the EUR-Lex GDPR, California Civil Code, India Code DPDP Act, and National People's Congress PIPL English text (European Parliament and Council of the European Union, 2016; California Legislature, 2018; Parliament of India, 2023; Standing Committee of the National People's Congress, 2021).

Table 3: Benchmark corpus.

Regulation	**Year**	**Provisions**	**Annotations**
GDPR (EU)	2018	99	1,225
CCPA (US-CA)	2018	45	421
DPDP (India)	2023	44	388
PIPL (China)	2021	50	288
Total		**238**	**2,322**

B.3 Candidate generation and search ranking

The reported implementation uses $k = 3$ proposals per outer iteration, $T_{\text{gen}} = 0.7$, and fixed-decoding repair and alignment calls with $T_{\text{repair}} = T_{\text{align}} = 0$. The outer and per-repair budgets are $T_{\max} = I_{\max} = 7$. These are search settings, not claims about the number of calls needed for successful cases.

At each outer iteration, NeuroFORM samples an ordered list of $k = 3$ candidate Z3 specifications:

$$\mathcal{C}_t = (\varphi_{t,1}, \ldots, \varphi_{t,k}) := \mathcal{G}_{\text{cand}}(s, \hat{a}; k, T_{\text{gen}}), \qquad k = 3, \quad T_{\text{gen}} = 0.7 \tag{16}$$

Each candidate is intended to be executable and contains variable declarations, logical constraints, and a `verify_data` entry point of section/article encoded as a Python function using Z3. The ordered-list notation preserves generation order for stable score ties and permits duplicate proposals to remain distinct candidates. After successful initialization, the conjunction of the asserted logical constraints is denoted F_φ as in Section 2 and candidates are ranked using the implementation-level verification score.

Each candidate is evaluated against the six-scenario repair harness $h_{\text{rep}}(s) = \{(x_i^{\text{rep}}, y_i^{\text{rep}})\}_{i=1}^{H}$, $H = 6$, $y_i^{\text{rep}} \in \{\texttt{sat}, \texttt{unsat}\}$. The implementation-level ranking score is defined by

$$\sigma_{\text{impl}}(\varphi, h_{\text{rep}}) = 0.5 e_{\text{init}} + 0.2 e_{\text{solver}} + 0.2 e_{\text{verdict}} + 0.1 e_{\text{diag}} \tag{17}$$

where $e_{\text{init}} \in \{0, 1\}$ indicates successful candidate import and initialization, $e_{\text{solver}} \in [0, 1]$ is the fraction of repair-harness scenarios producing definitive `sat`/`unsat` outcomes, $e_{\text{verdict}} = A_{\text{beh}}(\varphi, h_{\text{rep}})$, and $e_{\text{diag}} \in \{0, 1\}$ indicates whether the verifier produces a well-formed diagnostic record, including `E0` when no failure is detected. Operationally, $e_{\text{diag}} = 1$ when the diagnostic stage returns a valid, parseable diagnostic record, and $e_{\text{diag}} = 0$ when diagnostic generation, serialization, or parsing fails. This component therefore measures availability of diagnostic feedback rather than logical or semantic correctness. The coefficients in σ_{impl} are fixed implementation heuristics rather than learned or calibrated quantities; the score is not interpreted as a probability of correctness.

The score is used for candidate ranking and repair tracking. The implementation retains $\theta_{\text{impl}} = 0.80$ as an implementation-level score guard, while candidate-level operational or diagnostic failure

E1 triggers regeneration explicitly. This score guard is not an independent final-acceptance criterion. Because every expected repair-harness verdict is definitive, $A_{\text{beh}} = 1$ implies $e_{\text{init}} = e_{\text{solver}} = e_{\text{verdict}} = 1$, and hence

$$A_{\text{beh}} = 1 \Longrightarrow \sigma_{\text{impl}} = 0.9 + 0.1 e_{\text{diag}} \geq 0.90 \tag{18}$$

Thus, complete repair-harness agreement alone implies $\sigma_{\text{impl}} \geq 0.90$. Moreover, for an acceptance-admissible diagnostic state $d \in \mathcal{D}_{\text{acc}} = \{\texttt{E0}, \texttt{E4v}\}$, the diagnostic record is well formed and hence $e_{\text{diag}} = 1$. Therefore,

$$A_{\text{beh}} = 1 \wedge d \in \mathcal{D}_{\text{acc}} \Longrightarrow \sigma_{\text{impl}} = 1 \tag{19}$$

Consequently, the implementation threshold $\theta_{\text{impl}} = 0.80$ is redundant for final acceptance and is retained only as an implementation-level screening and ranking quantity. In Algorithm 1, we use the shorthand $S_\varphi := \sigma_{\text{impl}}(\varphi, h_{\text{rep}})$, $A_\varphi := A_{\text{beh}}(\varphi, h_{\text{rep}})$, and $D_\varphi := d(\varphi, s, \hat{a}, h_{\text{rep}})$.

B.4 Diagnostic classification and failure records

The diagnostic classifier uses $\mathcal{D}_{\text{cls}} = \{\texttt{E0}, \texttt{E1}, \texttt{E2}, \texttt{E3}, \texttt{E4}, \texttt{E4v}, \texttt{E5}\}$, with $\mathcal{D}_{\text{err}} = \{\texttt{E1}, \texttt{E2}, \texttt{E3}, \texttt{E4}, \texttt{E5}\}$, $\mathcal{D}_{\text{warn}} = \{\texttt{E4v}\}$, $\mathcal{D}_{\text{acc}} = \{\texttt{E0}, \texttt{E4v}\}$ The scenario-level and candidate-level diagnostic classes are $\mathcal{D}_{\text{scn}} = \{\texttt{E2}, \texttt{E3}, \texttt{E4}, \texttt{E5}\}$, $\mathcal{D}_{\text{cand}} = \{\texttt{E1}, \texttt{E4v}\}$. The first set contains errors represented with an affected repair-harness scenario; the second contains candidate-level diagnostics without a required unique x_i. During diagnostic classification, the verifier may also construct a structured semantic diagnostic record $\Delta_{\text{diag}}(\varphi, s, \hat{a})$ containing localization metadata such as missing required predicates and over-restrictive candidate constraints. This record is used only to localize repairs; it is distinct from the final Boolean semantic-alignment gate g_{sem}. Operationally, the localization information used by E4 and E5 is represented in Δ_{diag}. In particular, $\text{Cov}(F_\varphi)$ denotes the set of source-interface predicates for which the diagnostic localization stage identifies a corresponding constraint in the candidate specification, whereas $\text{Unsupported}(r; s, \hat{a}) = 1$ denotes a candidate constraint r for which the localization stage identifies no support in either the source clause s or the verified representation $\hat{a}$. These quantities are used only for diagnostic localization and repair; they are not additional acceptance criteria and are distinct from the final semantic-alignment decision g_{sem}.

For each repair scenario define $\hat{y}_i = \mathcal{V}(\varphi, x_i^{\text{rep}})$. The classifier returns one primary diagnostic under the following mutually exclusive precedence. E1 denotes a candidate-global operational failure that prevents reliable candidate evaluation and triggers regeneration. E2 is reserved for a scenario-local non-definitive solver/API outcome after candidate initialization has succeeded. For definitive verdict mismatches, E4 is assigned when comparison with $(s, \hat{a})$ localizes a required obligation/condition/exemption predicate that is not represented by a corresponding candidate constraint; E5 is assigned when $y_i^{\text{rep}} = \texttt{sat}$, $\hat{y}_i = \texttt{unsat}$, and a source-unsupported, over-restrictive candidate constraint is localized; and E3 is the remaining definitive polarity/operator mismatch after E4 and E5 have been excluded. If no scenario-level error remains, the all-false structural test may emit E4v; otherwise the class is E0. Because E4v is only a warning, it may trigger bounded structural repair but does not by itself block final acceptance. The total precedence is $\texttt{E1} \succ \texttt{E2} \succ \texttt{E4} \succ \texttt{E5} \succ \texttt{E3} \succ \texttt{E4v} \succ \texttt{E0}$.

For a scenario-level failure, let $d_\varphi = d(\varphi, s, \hat{a}, h_{\text{rep}}) \in \mathcal{D}_{\text{scn}}$. For fixed $(s, \hat{a}, h_{\text{rep}})$, let $\mathcal{I}_d(\varphi)$ denote the indices of repair scenarios to which the diagnostic record assigns class d; when a scenario-level class is selected, this set is nonempty. With the harness order fixed, choose $i^\star = \min \mathcal{I}_{d_\varphi}(\varphi)$. Then

$$c = (x_{i^\star}^{\text{rep}}, y_{i^\star}^{\text{rep}}, \hat{y}_{i^\star}, d_\varphi), \qquad \hat{y}_{i^\star} = \mathcal{V}(\varphi, x_{i^\star}^{\text{rep}}) \tag{20}$$

Candidate-level diagnostics are handled directly from the current specification and verifier diagnostic.

Scenario-associated index sets. The diagnostic record associates E2 with a scenario-local non-definitive outcome after successful candidate initialization. E4 is a definitive verdict mismatch for which the source-conditioned localization stage identifies an omitted required predicate. E5 is an expected-sat/observed-unsat mismatch for which that stage identifies an unsupported, over-restrictive candidate constraint; a removal test can require additional solver calls as reflected in the cost bound below. E3 covers the remaining definitive verdict mismatches. The precedence above determines the primary class when different scenarios support different classes. Within that class,

the lowest affected index in the fixed harness order is selected for feedback. Candidate-global initialization or diagnostic failure is E1 and does not require a scenario index. E4v is considered only after all six expected definitive verdicts are obtained. Source-support and omission judgments are outputs of the localization stage, not proofs from a bare Z3 verdict.

Search order and E4v. The acceptance predicate in Eq. 5 determines eligibility, whereas Algorithm 1 determines the order of inspection and repair. An `E0` generated candidate can proceed to the semantic gate immediately. A candidate whose sole diagnostic is `E4v` receives a bounded opportunity for structural repair. If bounded structural repair does not yield an accepted candidate, Algorithm 1 can recheck and return the original E4v candidate when its repair-scenario verdicts agree and the semantic gate passes. A semantic-gate rejection is an outer rejection and is not converted into an execution-derived counterexample.

B.5 SEMANTIC-ALIGNMENT GATE

Let $\mathcal{M}_{\text{align}}$ denote the realized output of the semantic-alignment model under the fixed prompt and decoding configuration, taking values in $\Sigma^* \cup \{\bot\}$, where $\bot$ denotes a missing response or model/API failure. Let $\pi_{\text{align}} : \Sigma^* \cup \{\bot\} \to \{0, 1\}$ be a total binary output parser. NeuroFORM defines

$$g_{\text{sem}}(\varphi, s, \hat{a}) = \pi_{\text{align}}(\mathcal{M}_{\text{align}}(\varphi, s, \hat{a}; T_{\text{align}} = 0.0)) \in \{0, 1\} \tag{21}$$

where π_{align} maps the explicitly valid positive response to 1, maps every negative, malformed, or unparseable string to 0, and sets $\pi_{\text{align}}(\bot) = 0$. Thus, missing output, model/API failure, and parser failure are rejected and the gate is mathematically total. The semantic-alignment gate is a model-based consistency check rather than a formal proof of semantic or legal equivalence. Accordingly, $g_{\text{sem}} = 1$ is interpreted as an additional model-based acceptance signal over candidates that have already passed the symbolic repair-harness checks, not as an independently verified guarantee of semantic correctness. Semantic failure is an outer rejection condition and does not itself generate a diagnostic counterexample. The structured record Δ_{diag} used to localize `E4`/`E5` is diagnostic metadata produced before final acceptance; it is distinct from the Boolean gate g_{sem} and does not replace it. Final acceptance is shown in Eq. 5.

The executed implementation additionally checks $\sigma_{\text{impl}} \geq \theta_{\text{impl}} = 0.80$ before returning a candidate. This does not introduce an additional acceptance criterion because Eq. 18 shows that $A_{\text{beh}} = 1$ already implies $\sigma_{\text{impl}} \geq 0.90$.

B.6 BOUNDED COST ACCOUNTING

Let $H = |h_{\text{rep}}|$. The E4v all-false structural heuristic requires at most one additional candidate-level solver invocation. In addition, E5 over-constraint localization may re-solve a scenario after removing candidate constraints. Let $R_{\max}$ denote the maximum number of asserted constraints $|\mathcal{C}_\varphi|$ among candidates considered within the bounded search. A conservative cost of one complete oracle call is

$$C^{\max}_{\text{oracle}}(H) = H\, C_{\text{SMT}} + C^{\text{ns}}_{\text{diag}} + C_{\text{E4v}} + C_{\text{E5}}, \qquad 0 \leq C_{\text{E4v}} \leq C_{\text{SMT}}, \qquad 0 \leq C_{\text{E5}} \leq H R_{\max} C_{\text{SMT}} \tag{22}$$

where C_{SMT} is the cost of one scenario-level solver invocation, $C^{\text{ns}}_{\text{diag}}$ is the non-solver cost of constructing/localizing the diagnostic record (including Δ_{diag} metadata when required), C_{E4v} is zero when the all-false heuristic is not tested and is upper-bounded by one solver invocation when it is tested, and C_{E5} accounts for the additional solver calls used by the E5 removal test in Over_i. The bound $H R_{\max} C_{\text{SMT}}$ is deliberately conservative because E5 localization is evaluated only for relevant mismatch scenarios and may stop once a witnessing unsupported constraint is found. With candidate-list size k, outer budget $T_{\max}$, and repair budget $I_{\max}$, the detailed implementation makes at most k oracle calls for the generated candidates, one additional oracle call on entry to DIAGNOSTICREPAIR, at most $I_{\max}$ oracle calls for repaired candidates, and one final oracle call after the repair routine returns. A conservative per-clause time bound is therefore

$$O(T_{\max}\,[C_{\text{gen}}(k) + (k + I_{\max} + 2) C^{\max}_{\text{oracle}}(H) + k \log k + I_{\max} C_{\text{repair}} + (k+1) C_{\text{align}}])\,. \tag{23}$$

Here, $C_{\text{gen}}(k)$ is the cost of generating k candidates, C_{repair} the cost of one repair-model call, and C_{align} the cost of one semantic-alignment call. The $k \log k$ term accounts for iteration-local score

ordering, and the $(k+1)C_{\text{align}}$ term upper-bounds semantic checks for the k generated candidates plus one repaired candidate. Because model, diagnostic, and SMT operations are treated as black-box costs, Eq. 23 is a bounded per-clause cost decomposition rather than a fine-grained complexity result in model parameters. With $k = 3$, $H = 6$, and $T_{\max} = I_{\max} = 7$, the search remains bounded per clause. Sequential processing requires

$$O(kL_\varphi + HL_d + L_s + M_{\text{SMT}}) \tag{24}$$

working space, where L_φ is the maximum stored candidate-specification size, L_d the maximum stored diagnostic-record size per harness scenario, L_s the source-and-annotation context size, and M_{SMT} the peak solver working memory.

C Evaluation Protocol and Reporting

Scenario construction and separation. For each provision, the repair and evaluation harnesses each contain one scenario in each of six categories: Compliance, Violation, Exception, Boundary, Multi-Exception, and Compound-Boundary. The category names specify the role of a test case; each concrete case additionally fixes its interface variables and has an expected `sat` or `unsat` verdict. The evaluation cases use concrete assignments disjoint from the repair cases for the same provision, as formalized in Eq. 13. They are withheld from candidate generation, symbolic verification, repair, and semantic alignment until the output specification is fixed. This tests unseen scenario instances for the evaluated provisions, not transfer to unseen provisions, regulations, or scenario types.

Repair-scenario expected verdicts were assigned from the statutory provision and expert-verified annotation before checking a generated candidate; they were not derived from the candidate's solver verdict.

The evaluation authoring process used the statutory provision and its expert-verified annotation $\hat{a}$, without access to any evaluated system's generated formulas, predicted verdicts, acceptance decisions, or diagnostic traces. Compliance and Violation cases used rule-based transformations. Exception and Boundary cases drew on annotated exemption and threshold spans. Multi-Exception and Compound-Boundary cases received additional expert review when conditions were not recoverable from the local clause alone. Evaluation instances and expected verdicts were finalized after initial synthesis but without inspection of system outputs. This output blinding limits conditioning on observed failures; it does not amount to prospective preregistration or rule out all human design-level hindsight.

Metrics and denominators. Let $\mathcal{B}$ be the set of benchmark provisions, and let $(x_{s,c}^{\text{eval}}, y_{s,c}^{\text{eval}})$ be the held-out scenario of category $c \in \mathcal{S}$ for provision s. Per-category behavioural agreement is

$$A_c = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} \mathbb{I}\left[\mathcal{V}(\varphi_s, x_{s,c}^{\text{eval}}) = y_{s,c}^{\text{eval}}\right]. \tag{25}$$

Overall behavioural accuracy A_{overall} is defined in Eq. 7. With one instance of every category per provision, it also equals $\frac{1}{6}\sum_{c \in \mathcal{S}} A_c$. Provisions therefore receive equal weight, while a regulation's weight reflects its number of provisions in $\mathcal{B}$. The synthesis acceptance rate is

$$R_{\text{acc}} = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} \text{Accept}(\varphi_s), \tag{26}$$

where $\varphi_s = \textsf{FAIL}$ if no specification is returned, and Accept is defined in Eq. 5. When the synthesis routine returns only accepted candidates or FAIL, this rate is also the fraction of provisions with a returned specification. It is an internal synthesis statistic, not held-out accuracy.

Repeated primary runs. The primary behavioural evaluation uses $R = 3$ independent stochastic synthesis runs. Writing $A_{\text{overall}}^{(r)}$ for the overall held-out accuracy of run r, its reported mean is

$$\bar{A}_{\text{overall}} = \frac{1}{R} \sum_{r=1}^{R} A_{\text{overall}}^{(r)}, \qquad R = 3, \tag{27}$$

and its across-run sample standard deviation is

$$s_A = \sqrt{\frac{1}{R-1} \sum_{r=1}^{R} \left(A_{\text{overall}}^{(r)} - \bar{A}_{\text{overall}}\right)^2}. \tag{28}$$

Table 4: Overall and per-regulation scenario-level behavioural accuracy (%), averaged over three independent runs. Each entry reports mean SLM / LLM performance. Each run contains 238 provisions and 1,428 held-out evaluation scenarios. NeuroFORM improves over the strongest corresponding functional baseline by +20.33 percentage points for SLM and +14.77 percentage points for LLM. B7 and B9 are non-functional best-effort reproductions and are excluded from the functional-baseline comparison. SLM regulation-level breakdowns for B3 and B7 are omitted because they have not been reconciled with the corrected overall counts.

Method	Overall	GDPR	CCPA	DPDP	PIPL
NeuroFORM	**75.07 / 78.43**	**77.95 / 81.48**	**66.67 / 71.85**	**73.11 / 74.62**	**78.67 / 81.67**
B1 Zero-Shot	50.56 / 57.42	56.23 / 66.33	49.63 / 48.52	46.59 / 46.59	43.67 / 57.33
B2 Few-Shot	46.85 / **63.66**	48.48 / 69.70	32.96 / 62.59	55.68 / 56.06	48.33 / 59.33
B3 CoT-Retry	**54.74** / 58.54	– / 66.33	– / 54.07	– / 51.52	– / 53.33
B4 Grammar-Constr.	39.43 / 48.60	44.44 / 54.88	31.48 / 41.85	32.95 / 39.39	42.33 / 50.33
B5 Naive CEGIS	51.82 / 56.93	57.41 / 68.52	44.44 / 48.89	47.73 / 49.24	51.00 / 48.00
B6 AquaForte	52.24 / 58.89	55.89 / 68.35	52.22 / 53.70	42.05 / 47.73	54.00 / 54.67
B8 Fine-Tuned	52.59 / 56.30	56.73 / 64.65	54.44 / 55.56	44.70 / 43.94	49.67 / 51.33
Non-functional best-effort reproductions					
B7 L4L-Schema	3.41 / 1.33	– / 0.17	– / 2.96	– / 1.52	– / 2.00
B9 RAG-SMT	1.19 / 0.00	1.35 / 0.00	0.00 / 0.00	1.52 / 0.00	1.67 / 0.00

The same across-run averaging is used for category and regulation results. This standard deviation describes run-to-run variation in the primary synthesis experiment; it is not a confidence interval over provisions and is not combined with results from a separately realized controlled Generic-versus-Diagnostic experiment.

D Additional Experimental Results

D.1 Primary benchmark by regulation

The following table reports overall means and available per-regulation breakdowns from the three independent primary runs. Each run evaluates 238 provisions and 1,428 held-out scenarios. B7 and B9 are best-effort reproductions under the executable-specification interface and are not estimates of the original systems' published performance.

The primary three-run mean differences against the strongest functional baselines are 20.33 points (SLM versus B3) and 14.77 points (LLM versus B2). They are end-to-end comparisons with unequal inference budgets, not estimates of any single component's causal effect.

D.2 Behaviour by scenario category

Every provision contributes one scenario from each category, so category averages use the same provision denominator. Printed cells are rounded to one decimal; the overall means are computed from unrounded case-level values.

The category results locate difficult cases; they do not by themselves identify the particular missing guard or operator in an individual candidate. Context-insufficient cases require a separate failure label when the expected verdict uses information absent from the local clause and expert annotation.

D.3 Complete E4v-guided repair mechanism

This ablation removes both the all-false structural warning and its induced bounded repair opportunity. The comparison measures the complete mechanism, not the causal contribution of the warning's information or a proof of vacuity.

Table 5: Scenario-wise behavioural accuracy (%) averaged over three independent runs on the complete benchmark. S_1: Compliance; S_2: Violation; S_3: Exception; S_4: Boundary; S_5: Multi-Exception; S_6: Compound-Boundary. Entries report mean SLM / LLM performance.

Method	S_1	S_2	S_3	S_4	S_5	S_6	**Avg.**
NeuroFORM(ours)	**66.4 / 73.9**	**98.3 / 98.7**	**80.3 / 86.1**	**60.9 / 64.3**	**84.5 / 87.0**	**60.1 / 60.5**	**75.07 / 78.43**
B1 Zero-Shot	61.3 / 74.4	44.5 / 50.4	44.5 / 49.6	53.8 / 60.9	47.1 / 52.1	52.1 / 57.1	50.56 / 57.42
B2 Few-Shot	65.5 / 73.9	40.8 / 62.6	45.4 / 58.0	44.5 / 65.5	43.3 / 61.3	41.6 / 60.5	46.85 / 63.66
B3 CoT-Retry	69.5 / 78.6	49.4 / 49.2	50.2 / 51.7	57.5 / 61.8	49.4 / 53.4	52.4 / 56.7	54.74 / 58.54
B4 Grammar-Constr	57.1 / 63.9	25.2 / 33.2	29.0 / 37.0	49.6 / 62.2	30.7 / 36.6	45.0 / 58.8	39.43 / 48.60
B5 Naive CEGIS	68.1 / 79.8	44.1 / 47.5	47.5 / 47.5	54.6 / 61.3	45.8 / 47.9	50.8 / 57.6	51.82 / 56.93
B6 AquaForte	63.0 / 78.6	48.7 / 48.7	51.7 / 48.7	50.0 / 65.1	53.4 / 51.3	46.6 / 60.9	52.24 / 58.89
B8 Fine-Tuned	70.2 / 77.3	47.5 / 47.1	49.6 / 47.5	50.8 / 62.2	49.6 / 47.9	47.9 / 55.9	52.59 / 56.30
Non-functional reproductions							
B7 L4L-Schema	6.4 / 3.8	0.0 / 0.0	1.3 / 1.3	6.8 / 0.8	1.7 / 1.3	4.2 / 0.8	3.41 / 1.33
B9 RAG-SMT	1.7 / 0.0	1.3 / 0.0	1.3 / 0.0	0.8 / 0.0	1.3 / 0.0	0.8 / 0.0	1.19 / 0.00

Table 6: Effect of disabling `E4v`-guided structural repair on the 238-provision benchmark. Accuracy values are means across three independent runs. Δ denotes the difference between the corresponding mean accuracies relative to complete NeuroFORM.

Configuration	**SLM Acc. (%, mean $\pm$ SD)**	**LLM Acc. (%, mean $\pm$ SD)**	**Δ (SLM / LLM)**
Complete NeuroFORM	75.07 ± 3.59	78.43 ± 5.87	—
Without `E4v` repair	56.20 ± 6.26	59.94 ± 7.10	$-18.87/-18.49$

D.4 Separate paired statistical-validation block

This block is independent of the primary three-run mean benchmark. Bootstrap intervals use 10,000 resamples of 238 paired provisions. Synthesis acceptance uses all provisions; repair completion is conditional on entering diagnostic repair. Let $\mathcal{B}_{\text{rep}}$ denote the latter set and $\mathcal{B}_{\text{comp}}$ its subset completing repair within budget. Then

$$R_{\text{comp}} = \frac{|\mathcal{B}_{\text{comp}}|}{|\mathcal{B}_{\text{rep}}|}. \tag{29}$$

Table 7: Bootstrap estimates and 95% confidence intervals for synthesis reliability metrics using 10,000 resamples.

Metric	**SLM**	**95% CI**	**LLM**	**95% CI**
Synthesis acceptance	91.60	[87.82, 94.96]	94.54	[91.60, 97.06]
Repair completion, R_{comp}	97.73	[95.45, 99.55]	99.54	[98.62, 100.00]

Within this dedicated paired block, LLM-minus-SLM synthesis acceptance is 2.94 points (continuity-corrected McNemar $p = 0.211$; 15 LLM-only and 8 SLM-only provision successes). Comparing repair-enabled and no-repair configurations yields paired $\chi^2_{\text{SLM}} = 196.01$ and $\chi^2_{\text{LLM}} = 184.01$, both $p < 10^{-41}$, for provision-level synthesis outcomes, not held-out accuracy.

Paired held-out behavioural improvements over the strongest functional baselines are 20.53 points for SLM versus B3 (95% paired-bootstrap CI [16.24, 24.82]) and 14.78 points for LLM versus B2 (CI [10.92, 18.70]), each over 238 matched provisions. Both Wilcoxon signed-rank and paired t tests yield $p < 0.001$ for each tier. These are separate-block estimates; they need not equal 20.33 and 14.77 from the primary means.

D.5 Repair budget and candidate-list sweeps

The following table reports synthesis acceptance rate (SAR), not held-out behavioural accuracy. Repair-budget and candidate-list sweeps are separate stochastic blocks, so their nominal rows are not interchangeable. Since there are up to seven outer iterations, $I_{\max} = 1$ allows up to seven repair calls across the full trajectory; it does not establish one-call convergence. Increasing k also increases generation and verification compute.

Table 8: Mechanism ablations on 238 provisions. SAR denotes configuration-specific synthesis acceptance rate (%). $I_{\max}$ is the maximum number of diagnostic-repair steps per outer synthesis iteration; with $T_{\max} = 7$, the worst-case total repair-model-call budget is $T_{\max} I_{\max}$. The repair-budget and candidate-list-size sweeps were executed independently; consequently, the $I_{\max} = 7$ and $k = 3$ rows are separate realizations of the nominal configuration rather than duplicate measurements of the same run.

Configuration	SLM SAR	LLM SAR
No repair ($I_{\max} = 0$)	14.29	19.33
Repair budget $I_{\max} = 1$	95.80	93.70
Repair budget $I_{\max} = 3$	96.64	95.38
Repair budget $I_{\max} = 5$	96.64	96.64
Repair budget $I_{\max} = 7$	97.48	94.54
Candidate-list size $k = 1$	91.18	92.86
Candidate-list size $k = 3$	98.74	95.80

Convergence denominator. Among cases eventually repaired successfully in the recorded repair traces, 203/215 SLM cases (94.4%) and 200/216 LLM cases (92.6%) succeeded on the first repair. By the second repair, the corresponding counts were 214/215 and 215/216 (99.5% in each tier). Attempts are counted within the successful DiagnosticRepair invocation; the counter resets when a new outer iteration invokes repair. These fractions condition on eventual repair success, not on all cases entering repair, final synthesis acceptance, or total repair calls across the run.

D.6 Generic versus diagnostic verifier feedback

This matched controlled experiment is a separate stochastic realization and must not be merged with the headline benchmark or identified with B5 Naive CEGIS merely because values are similar.

Both point estimates favor diagnostic feedback, but both intervals contain zero. This block does not establish statistically reliable superiority of the diagnostic taxonomy over generic feedback.

These repair-entry logs come from a separate matched run, not the run instances underlying the held-out comparison in Table 9.

In the matched repair-entry logs, mean eventual completion across three seeds was 97.88% versus 94.09% for Diagnostic and Generic repair with the SLM, and 99.38% versus 95.55% with the LLM. These descriptive rates condition on entering repair; they are not held-out accuracy estimates or tests of statistical superiority.

D.7 Semantic gate ON/OFF

In this dedicated paired block, generator, candidate-list size, verification, diagnostic repair, budget, decoding, and random-seed settings are held fixed. A gate rejection can redirect later candidate selection, so ON/OFF measures an end-to-end gate effect, not a purely post-hoc rejection filter. Changing the aligner model while retaining the gate is a different architecture-sensitivity experiment.

The ON-minus-OFF held-out differences are +22.74 points (SLM) and +4.12 points (LLM) within this block. Pooled synthesis acceptance is 93.07% (443/476) with semantic alignment and 85.08% (405/476) without semantic alignment, a difference of +7.98 percentage points. Its absolute ON accuracy values are not the three-run primary means. The gate is a model-based consistency check and provides no proof of legal correctness.

Table 9: Matched Generic-versus-Diagnostic repair. Held-out behavioural accuracy (%) and 95% paired-bootstrap confidence intervals for Diagnostic minus Generic.

Tier	Generic	Diagnostic	Δ (pp)	95% CI (pp)
SLM	53.15	56.93	+3.78	$[-0.98, +8.61]$
LLM	55.32	56.72	+1.40	$[-2.52, +5.39]$

Table 10: Controlled semantic-alignment ablation from a dedicated paired experimental block. Accuracy denotes strict held-out behavioural accuracy over the complete S_1–S_6 evaluation set. ON and OFF use identical generation, symbolic-verification, diagnostic-repair, decoding, and random-seed settings, differing only in whether g_{sem} is enforced. Pooled SAR aggregates provision-level synthesis acceptance across the SLM and LLM tiers within this realization, not across the three primary-benchmark seeds. Each condition contains 476 outcomes (238 per tier): 443 accepted with alignment ON and 405 with it OFF. Δ denotes ON minus OFF and is computed before rounding.

Configuration	**SLM Acc.**	**LLM Acc.**	**Pooled SAR**
Semantic alignment ON	68.14	54.97	93.07
Semantic alignment OFF	45.40	50.85	85.08
Δ	+22.74	+4.12	+7.98

D.8 Diagnostic-event profile

SLM repair traces record 291 E4 events and 455 E4v warnings; LLM traces record 223 E4 events. These are event counts, not independent provisions. The reported repair rate for E4 encounters is 100% in both tiers; 99.1% of SLM E4v encounters are repaired. Reported repair rates for Z3-encoding-error encounters are 97.4% (SLM) and 95.5% (LLM). E1 candidate-level operational failures remain a prominent identifiable residual failure type. E4v is an all-false structural warning, not a proof of vacuity.