ONE FUZZER, MANY REASONERS: ADAPTIVE NEURO-SYMBOLIC FUZZING WITH RELIABILITY-CALIBRATED AND CONTEXT-GATED GUIDANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural-symbolic AI (NSAI) combines learned perception with programs, constraints, and compositional reasoning, but remains difficult to test because small semantics-preserving changes can alter grounded symbols and trigger discontinuous reasoning failures. Existing neural fuzzers are typically tied to architecture-specific representations and lack a common interface for heterogeneous NSAI systems. To the best of our knowledge, we introduce the first unified fuzzing framework specifically designed for heterogeneous NSAI. NSAI-FUZZ separates task-specific semantics-preserving mutation from architecture-neutral search and represents each candidate using five rank-normalized signals: boundary pressure, grounding uncertainty, representation drift, constraint violation, and archive novelty. We further propose Reliability-Calibrated Weighting (RCW), which learns dataset–model-level weights from signal reliability, and Contextual Signal Gating (CSG), which adapts weights to the current candidate pool. We evaluate ISED, DOLPHIN, NASR, and NeSyCoCo on four reasoning tasks, covering nine compatible dataset–model cells and 1,350 on-policy runs.Fixed five-signal guidance outperforms Random and Uniform in every cell, by up to $19.115\,\mathrm{pp}$ and $3.280\,\mathrm{pp}$, respectively, while single-signal and leave-one-out analyses reveal strong model–task dependence in signal importance. Motivated by this heterogeneity, both RCW and CSG improve the mean failure discovery rate over Fixed in all cells, with gains of up to $6.621\,\mathrm{pp}$. Wall-clock evaluation shows that the adaptive methods retain essentially Fixed-like cost while achieving $5.10\times$ and $5.08\times$ the failure throughput of Random. These results establish fuzzing as a practical testing methodology for heterogeneous NSAI systems and show that effective guidance should adapt to the neural–symbolic interface.

1 INTRODUCTION

Neural-symbolic AI (NSAI) combines learned perception with explicit reasoning. A neural module may ground pixels into digits, objects, or concepts; a downstream component then executes an arithmetic program, enforces Sudoku constraints, or composes a language-conditioned concept program. Representative systems include DeepProbLog Manhaeve et al. (2018), ISED Solko-Breslin et al. (2024), DOLPHIN Naik et al. (2025), NASR Cornelio et al. (2023), and NeSyCoCo Kamali et al. (2025). This interface enables structured reasoning, but also creates a sharp reliability boundary: a small change in neural evidence can replace a grounded symbol, alter an execution path, or violate a global constraint.

Testing such systems is challenging. Clean accuracy does not characterize how failures are distributed around correctly solved inputs, while conventional adversarial attacks are an imperfect fit because symbolic execution may be non-differentiable or combinatorial. Moreover, a perturbation is meaningful only if it preserves the task semantics: changing the answer of a Sudoku instance or the referent of a grounding command does not constitute a valid robustness failure. Existing coverage-guided neural fuzzers Pei et al. (2017); Tian et al. (2018); Xie et al. (2019); Odena et al. (2019) typically rely on architecture-specific neural representations and do not provide a unified interface for heterogeneous neural–symbolic reasoning pipelines.

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.

To the best of our knowledge, we present the first unified fuzzing framework specifically designed to test heterogeneous NSAI systems. NSAI-FUZZ separates *what mutations are valid* from *which valid candidates should be explored*. Each task adapter generates semantics-preserving mutations and provides an oracle that verifies that the expected answer is unchanged. Search is then performed through a common architecture-neutral interface rather than through model-specific coverage metrics. The common interface represents each valid candidate using five signals: boundary pressure, grounding uncertainty, representation drift, constraint violation, and archive novelty. Pool-wise rank normalization removes scale differences, while unavailable signals are explicitly masked, enabling a common search procedure across heterogeneous neural–symbolic interfaces.

Our experiments show structured guidance is useful but signal importance is heterogeneous. Fixed five-signal guidance outperforms Random selection in all nine dataset–model cells, and its nonuniform weighting also consistently outperforms Uniform fusion. Single-signal and leave-one-out diagnostics further show that the most useful signals vary across tasks and models. Thus, the key challenge is not whether to use the five signals, but how to allocate their weights for a particular model–task setting.

We address this challenge with two adaptive mechanisms. Reliability-Calibrated Weighting (RCW) learns a cell-level allocation from signal reliability, while Contextual Signal Gating (CSG) adapts the allocation to the current candidate pool during search. We evaluate ISED, DOLPHIN, NASR, and NeSyCoCo across four tasks, yielding nine compatible cells, 15 policies, ten paired attack seeds, and 1,350 on-policy runs. Both adaptive methods improve the mean failure discovery rate over Fixed in every cell, yielding positive gains in all cell–method comparisons. We further evaluate whether this improved search guidance remains useful under wall-clock cost. All five-signal policies require roughly $1.4\times$ the online runtime of Random, with most of the overhead arising from common signal extraction and ranking rather than adaptive weighting itself. RCW and CSG retain essentially the same online runtime as Fixed, but achieve $5.10\times$ and $5.08\times$ the normalized failure throughput of Random, respectively. Thus, adaptive guidance improves sustained failure discovery without introducing meaningful additional online cost beyond the common five-signal framework. Figure 1 summarizes NSAI-FUZZ, from semantics-preserving mutation and unified signal extraction to adaptive weighting and on-policy guided search. Our contributions are:

- To the best of our knowledge, we present the first unified fuzzing framework specifically designed for heterogeneous neural-symbolic AI systems, separating task-specific semantics-preserving mutation from architecture-neutral guided search.
- We introduce five availability-aware, rank-normalized guidance signals. Fixed guidance outperforms Random and Uniform in all dataset–model cells, with gains of up to $19.115\,\mathrm{pp}$ and $3.280\,\mathrm{pp}$, respectively, while signal importance varies substantially across cells.
- We propose RCW and CSG to automatically allocate signal weights from dataset–model reliability and candidate-pool context, respectively. Both outperform Fixed in all nine cells, yielding positive adaptive comparisons and gains of up to $6.621\,\mathrm{pp}$.
- We evaluate four NSAI families across four tasks, nine compatible cells, and 1,350 on-policy runs, and further show that RCW and CSG achieve approximately $5\times$ the wall-clock failure throughput of Random with essentially Fixed-like online runtime.

2 RELATED WORK

Neural-symbolic learning. Neural-symbolic systems expose heterogeneous interfaces: DeepProbLog couples neural predicates with probabilistic logic (Manhaeve et al., 2018); ISED supports arbitrary black-box programs (Solko-Breslin et al., 2024); DOLPHIN composes distributions and provenance (Naik et al., 2025); NASR uses symbolic correction (Cornelio et al., 2023); and NeSyCoCo executes differentiable concept programs (Kamali et al., 2025). Their intermediate states range from probabilities and sampled worlds to correction masks and grounding tensors. We therefore standardize extracted risk semantics rather than raw internal representations.

Testing learned systems. DeepXplore, DeepTest, DeepHunter, and TensorFuzz use feedback-driven search based on neuron coverage, metamorphic transformations, or representation novelty (Pei et al., 2017; Tian et al., 2018; Xie et al., 2019; Odena et al., 2019). HDTest similarly uses representation-specific distance for hyperdimensional classifiers (Ma et al., 2021). These methods motivate guided

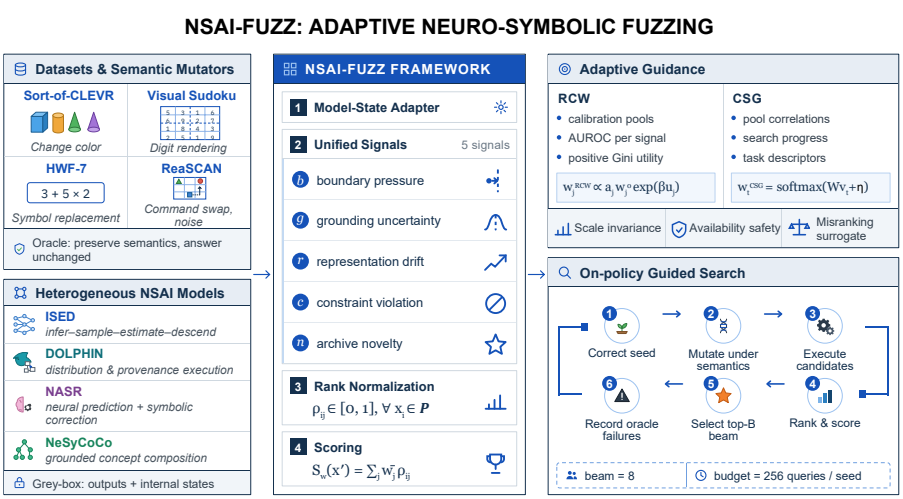


Figure 1: Overview of NSAI-FUZZ. Task-specific semantic mutators generate valid candidates, which are mapped by a common adapter to five unified guidance signals. RCW and CSG adapt their weights to guide on-policy search across heterogeneous NSAI systems.

search but do not directly unify heterogeneous neural–symbolic interfaces. Our framework separates task-specific semantic validity from candidate-selection guidance.

Adaptive multi-signal guidance. Multi-objective fuzzing can combine coverage, uncertainty, and distance, but their relative importance is domain dependent. Our rank fusion removes scale mismatch; RCW learns dataset–model-level weights from calibrated reliability, while CSG adapts weights to candidate-pool context during search. All policies are evaluated on-policy because different frontier selections induce different future candidate pools.

3 Why a Unified Framework?

Interface mismatch. Let an NSAI system consist of a neural perceptual map M_θ, a structured state constructor G, and an executor P:

$$F(x, q) = P(G(M_\theta(x), q)) \,. \tag{1}$$

The optional query q may be a relational question, arithmetic program, or language command. Equation equation 1 covers heterogeneous intermediate states that are not directly interchangeable, so a coverage metric designed for one interface cannot be applied uniformly across Sort-of-CLEVR, Visual Sudoku, HWF-7, and ReaSCAN.

We therefore standardize *risk semantics* rather than raw internal states. Each adapter exposes available estimates of boundary pressure, grounding uncertainty, representation drift, constraint violation, and novelty; missing signals are masked, making one search procedure across heterogeneous executors.

Semantics before search. For a solved seed (x, q, y), a task-specific mutation operator T produces $x' = T(x; \xi)$, ξ denotes the mutation parameters, such as perturbation magnitude, transformation type, or sampled replacement instance. A task relation $\equiv_y$ certifies that the expected answer is unchanged:

$$x' \equiv_y x \quad \Longrightarrow \quad y(x', q) = y(x, q) = y. \tag{2}$$

The oracle failure is

$$f(x') = \mathbb{1}[F(x', q) \neq y] \quad \text{subject to } x' \equiv_y x. \tag{3}$$

Photometric transformations are restricted by an RMS-L_2 ball, while structured transformations preserve equivalence by construction. Invalid candidates are discarded, so guidance affects search efficiency rather than test validity.

Threat model and objective. We assume grey-box forward access to output scores and selected intermediate representations; gradients, parameter updates, and training data are not required. Given query budget Q, policy π maximizes failure yield

$$J(\pi) = \mathbb{E}\left[\frac{1}{Q}\sum_{t=1}^{Q} f(x_t^{\pi})\right]. \tag{4}$$

All policies share seeds, candidate generators, validity filters, beam size, and query caps, differing only in frontier selection. Thus, differences in $J(\pi)$ reflect guidance.

4 METHOD

4.1 ARCHITECTURE-NEUTRAL GUIDANCE SIGNALS

For each valid candidate x', the model-state adapter returns five available signals $s(x') = [b, g, r, c, n]$. Let $z_k(x')$ be the score of answer k and $h(x')$ an adapter-selected intermediate representation. **Boundary pressure** measures proximity to an incorrect decision,

$$b(x') = \max_{k \neq y} z_k(x') - z_y(x'), \tag{5}$$

with the corresponding execution margin used for structured outputs. **Grounding uncertainty** g measures instability at the neural–symbolic interface using normalized entropy, prototype distance, or execution uncertainty, depending on the exposed state. **Representation drift** measures latent change,

$$r(x') = 1 - \frac{h(x)^{\top} h(x')}{\|h(x)\|_2 \|h(x')\|_2 + \epsilon}. \tag{6}$$

Constraint violation c measures normalized symbolic inconsistency; it is masked when no meaningful checker is available. Finally, **Archive novelty** encourages unexplored states,

$$n(x') = \min_{a \in \mathcal{A}} \left(1 - \cos(h(x'), a)\right). \tag{7}$$

Although their implementations depend on the adapter, all five signals share the same risk interpretation: larger values receive higher search priority.

4.2 POOL-WISE RANK FUSION

Because the five raw signals have heterogeneous scales, we rank candidates within the current valid pool $C_t = \{x'_1, \ldots, x'_m\}$. Let s_{ij} be signal $j \in \{1, \ldots, 5\}$ for candidate x'_i. Its percentile rank is

$$\rho_{ij} = \frac{1}{m-1}\sum_{\ell=1}^{m} \mathbb{1}[s_{\ell j} < s_{ij}], \qquad \rho_{ij} \in [0, 1], \tag{8}$$

where larger ρ_{ij} indicates higher rank; ties are resolved deterministically.

To handle unavailable signals, let $a = (a_1, \ldots, a_5) \in \{0, 1\}^5$, with $a_j = 1$ if signal j is available, and let $w \in \Delta^4$, where $\Delta^4 = \{w \in \mathbb{R}^5_{\geq 0} : \sum_j w_j = 1\}$. The masked and normalized fusion score is

$$S_w(x'_i) = \sum_{j=1}^{5} \bar{w}_j \rho_{ij}, \qquad \bar{w}_j = \begin{cases} \dfrac{a_j w_j}{\sum_{k=1}^{5} a_k w_k}, & \sum_{k=1}^{5} a_k w_k > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Larger S_w gives higher frontier priority. If all nominal weight is assigned to unavailable signals, all candidates tie and a seeded stable tie-break is used.

Fixed uses $w^0 = (.35, .20, .20, .15, .10)$ for boundary, grounding, drift, constraint, and novelty. Uniform assigns equal weight to available signals; one-hot policies retain one signal; leave-one-out policies remove one signal and renormalize the remaining Fixed weights; Random ignores S_w and randomly orders the frontier.

Proposition 1 (Monotone-scale invariance)**.** *For any strictly increasing transformations ϕ_j, replacing every raw signal s_{ij} by $\phi_j(s_{ij})$ leaves all percentile ranks ρ_{ij} unchanged and therefore preserves the fused scores and selected frontier.*

Proposition 2 (Availability safety)**.** *For any two nominal weight vectors whose available coordinates are proportional, masked normalization produces the same effective weights $\bar{w}$ and hence the same candidate ordering. Unavailable signals receive zero effective weight and cannot affect selection.*

Strictly monotone transformations preserve within-pool ordering, while masking preserves relative weight among available signals; proofs are given in Appendix A.

4.3 Reliability-Calibrated Weighting

RCW learns one signal-weight vector per dataset–model cell from calibration candidates $\{x'_i, f_i\}_{i=1}^{N_{\text{cal}}}$, where $f_i \in \{0, 1\}$ denotes failure. For signal j, we measure failure-ranking ability by

$$A_j = \text{AUC}(\{s_{ij}\}_{i=1}^{N_{\text{cal}}}, \{f_i\}_{i=1}^{N_{\text{cal}}}), \tag{10}$$

and define reliability utility

$$u_j = \max(2A_j - 1, 0), \tag{11}$$

so $u_j \in [0, 1]$ and signals no better than random receive zero positive evidence.

Let

$$\Delta_a = \left\{ w \in \mathbb{R}^5_{\geq 0} : \sum_{j=1}^{5} w_j = 1,\ w_j = 0 \text{ if } a_j = 0 \right\},$$

and let w_a^0 be the availability-masked Fixed prior. RCW solves

$$w^{\text{RCW}} = \arg\max_{w \in \Delta_a} \left\{ \langle w, u \rangle - \frac{1}{\beta} D_{\text{KL}}(w \| w_a^0) \right\}, \tag{12}$$

where $\langle w, u \rangle = \sum_j w_j u_j$ rewards reliable signals, D_{KL} regularizes deviation from the prior, and $\beta > 0$ controls reweighting strength.

The closed-form solution is

$$w_j^{\text{RCW}} = \frac{a_j w_j^0 \exp(\beta u_j)}{\sum_{k=1}^{5} a_k w_k^0 \exp(\beta u_k)}, \tag{13}$$

with $\beta = 3$. Thus, reliable signals receive more weight while the prior limits extreme allocations; w^{RCW} remains fixed within each dataset–model cell.

Proposition 3 (Optimality and stability of RCW)**.** *If $w_{a,j}^0 > 0$ for every available signal, Equation equation 12 has the unique maximizer in Equation equation 13. For utility vectors u, u', any available signals j, k satisfy*

$$\left| \log \frac{w_j(u)/w_k(u)}{w_j(u')/w_k(u')} \right| \leq 2\beta \|u - u'\|_\infty. \tag{14}$$

Hence, RCW is uniquely defined and its relative weights are stable to calibration perturbations; derivations appear in Appendix A.

4.4 Contextual Signal Gating

Because cell-level weights cannot adapt to changing candidate pools, CSG builds a round-level context vector v_t from ranked-signal statistics, pairwise correlations, signal availability, search progress, and task/model descriptors. A linear gate predicts

$$w_t^{\text{CSG}} = \text{softmax}_a(W v_t + \eta), \tag{15}$$

where W is learnable, $\eta = \log w^0$, and softmax_a masks unavailable signals. Thus, $w_t^{\text{CSG}} \in \Delta^4$ may vary across rounds.

For a sampled failure/non-failure pair (p, n) with $f_p = 1$ and $f_n = 0$, let $\rho_p, \rho_n \in [0, 1]^5$ be their ranked signal vectors and define

$$m_{pn} = (w_t^{\text{CSG}})^\top (\rho_p - \rho_n), \tag{16}$$

so $m_{pn} > 0$ ranks the failure higher.

The gate minimizes

$$\mathcal{L} = \frac{1}{|\mathcal{P}|} \sum_{(p,n)\in\mathcal{P}} \log(1 + \exp(-m_{pn})) + \lambda \|W\|_F^2 + \gamma D_{\mathrm{KL}}(w_a^0 \| w_t^{\mathrm{CSG}}), \tag{17}$$

where $\mathcal{P}$ is the sampled pair set, λ controls ℓ_2 regularization, and γ regularizes toward the masked prior. We sample at most 64 pairs per pool and train for 150 epochs with learning rate .03, $\lambda = 10^{-4}$, and $\gamma = .05$.

Proposition 4 (Misranking surrogate)**.** *For every failure/non-failure pair,*

$$\mathbb{1}[m_{pn} \le 0] \le \frac{\log(1 + \exp(-m_{pn}))}{\log 2}. \tag{18}$$

Hence, the pairwise loss upper-bounds empirical misranking. This is a local ranking guarantee, so all policies are evaluated on-policy.

4.5 Guided Search

We instantiate Equation equation 4 with bounded beam search. At round t, each parent $u \in \mathcal{B}_t$ is mutated by $T(u;\xi)$, and semantic filtering yields

$$C_t = \{T(u;\xi) : u \in \mathcal{B}_t\} \cap \{x' : x' \equiv_y x\}, \tag{19}$$

where each retained candidate is executed once and counts toward the query budget.

With beam width B and signal weights w_t, score-based policies retain

$$\mathcal{B}_{t+1} = \operatorname*{arg\,topB}_{x'_i \in C_t} S_{w_t}(x'_i), \tag{20}$$

while Random selects B candidates by seeded permutation. All policies share seeds, mutations, validity checks, beam width, and query cap, differing only in parent selection.

Evaluation is *on-policy*: different parents induce different future candidate pools. Random-frontier pools are used only for RCW/CSG calibration; reported results come from fresh policy trajectories. The archive $\mathcal{A}_t$ stores the seed and selected parents, and failures do not terminate search. For $m_t = |C_t|$ and representation dimension d, rank fusion costs $O(5m_t \log m_t)$ and exact novelty $O(m_t|\mathcal{A}_t|d)$. Additional details appear in Appendix A.

5 Experimental Setup

Evaluation configuration. We evaluate ISED and DOLPHIN on Sort-of-CLEVR, Visual Sudoku, and HWF-7; NASR on Visual Sudoku and HWF-7; and NeSyCoCo on ReaSCAN, yielding nine compatible dataset–model cells. The tasks cover relational VQA (Santoro et al., 2017), 9×9 Sudoku (Wang et al., 2019), handwritten arithmetic, and compositional visual grounding (Wu et al., 2021). Each adapter defines its mutation space, semantic oracle, and available signals; only initially correct inputs are fuzzed. Compatibility and adapter details are given in Appendix B. Each cell evaluates 15 policies: Random, Uniform, Fixed $(.35, .20, .20, .15, .10)$, five one-hot, five leave-one-out, RCW, and CSG. Each run uses 20 inputs, 256 candidate evaluations per input, beam width eight, and a common mutation family. Ten paired seeds yield $9 \times 15 \times 10 = 1{,}350$ on-policy runs. Reproduction details appear in Appendix E.

Calibration and metrics. Random-frontier traces are grouped into candidate pools, with alternating pools used for calibration and screening and at most 2,000 retained per cell. RCW estimates single-signal AUROC, while CSG is fit on calibration pools and inspected on held-out pools. Calibration determines adaptive parameters; all policies are then rerun from scratch on-policy. The primary metric is candidate-level oracle failure discovery rate,

$$\widehat{J} = \frac{\sum_i f(x'_i)}{N_{\mathrm{valid}}}. \tag{21}$$

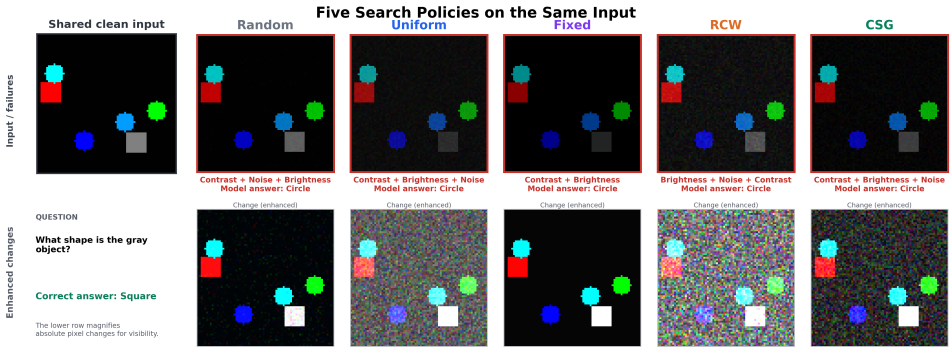


Figure 2: Five search policies on the Sort-of-CLEVR. All mutations preserve the oracle answer *Square* but induce the prediction *Circle*; the bottom row visualizes enhanced pixel differences.

Table 1: Mean oracle failure discovery rate (percent; higher is better). Bold marks the better adaptive policy between RCW and CSG. Random and Uniform are shown as foundational baselines.

Dataset	Model	Random	Uniform	Fixed	RCW	CSG
Sort-of-CLEVR	ISED	1.127	11.693	12.675	**13.832**	13.526
Sort-of-CLEVR	DOLPHIN	2.108	13.723	14.335	14.914	**15.557**
Visual Sudoku	ISED	6.645	12.791	14.133	16.062	**16.629**
Visual Sudoku	DOLPHIN	1.395	4.002	4.908	**5.764**	5.740
Visual Sudoku	NASR	9.598	18.023	21.303	23.598	**23.740**
HWF-7	ISED	0.619	5.904	6.725	9.680	**9.934**
HWF-7	DOLPHIN	2.582	15.232	18.098	**24.719**	23.123
HWF-7	NASR	3.270	21.447	22.385	**26.650**	25.633
ReaSCAN	NeSyCoCo	1.074	9.889	10.211	10.283	**10.664**

We report the mean over ten paired seeds. Experiments use three NVIDIA RTX A6000 48 GB GPUs, a 64-thread CPU environment, PyTorch, and CUDA; checkpoints, seeds, budgets, candidate summaries, and analysis code are retained in the artifact.

6 Fuzzing Results

6.1 Qualitative Search Example

Figure 2 shows how different policies explore the same initially correct Sort-of-CLEVR seed. All mutations preserve the oracle answer *Square* but induce the incorrect prediction *Circle*, illustrating that frontier selection reaches different regions of the valid mutation space. The lower row shows enhanced absolute pixel differences from the clean input.

6.2 Structured Guidance and Signal Importance

Table 1 reports the principal mean failure rates. Fixed guidance outperforms Random in all nine dataset–model cells, often by a large margin: 12.68% versus 1.13% on Sort-of-CLEVR/ISED, 18.10% versus 2.58% on HWF-7/DOLPHIN, and 10.21% versus 1.07% on ReaSCAN/NeSyCoCo. Since the two policies share the same seeds, mutation operators, validity checks, beam width, and query budgets, while Random selects the frontier without using the five guidance signals, these gains show that boundary pressure, grounding uncertainty, representation drift, constraint violation, and novelty provide effective information for steering the search toward failure-prone regions. Thus, structured five-signal guidance is substantially more effective than unguided random frontier selection.

Fixed guidance also outperforms Uniform in every cell. Because all five signals are rank normalized before fusion, this gap cannot be attributed to differences in numerical scale. Instead, it shows that the signals contribute unequally: assigning equal weight to boundary, grounding, representation drift, constraint violation, and novelty is consistently less effective than a structured nonuniform allocation.

The single-signal and leave-one-out ablations below further characterize these differences in signal importance across tasks and models.

Table 2: Single-signal failure discovery rates (%). Bold marks the best one-hot signal in each cell; B/G/R/C/N denote boundary, grounding, drift, constraint, and novelty.

Dataset–model	B	G	R	C	N
Sort-of-CLEVR–ISED	**12.564**	11.947	5.801	1.885	5.916
Sort-of-CLEVR–DOLPHIN	13.193	**13.327**	12.315	1.424	3.817
Visual Sudoku–ISED	**15.815**	5.387	8.053	10.100	7.680
Visual Sudoku–DOLPHIN	**5.676**	2.141	1.209	1.615	.895
Visual Sudoku–NASR	**23.000**	8.473	11.844	15.094	9.781
HWF-7–ISED	8.742	5.824	**8.954**	2.979	4.934
HWF-7–DOLPHIN	**22.355**	11.441	19.125	3.129	10.533
HWF-7–NASR	**24.820**	11.791	22.848	3.549	11.611
ReaSCAN–NeSyCoCo	**9.992**	0.787	9.689	0.693	4.170

6.3 Signal Importance Is Heterogeneous

Single-signal and leave-one-out (LOO) diagnostics reveal that the five guidance signals contribute very differently across dataset–model cells. A one-hot policy measures the stand-alone search strength of one signal, while a LOO policy removes one signal from Fixed and renormalizes the remaining mass, revealing whether that signal is helpful under the fixed combination.

Table 2 shows that the stand-alone strength of the five signals varies substantially across model–task cells. Boundary is the strongest single signal in seven of nine cells, reaching 23.000% on Visual Sudoku/NASR, 22.355% on HWF-7/DOLPHIN, and 24.820% on HWF-7/NASR. However, it is not universally dominant: Grounding performs best on Sort-of-CLEVR/DOLPHIN (13.327%), while Drift is strongest on HWF-7/ISED (8.954%). Other signals can be much weaker in isolation, indicating that no single guidance signal is uniformly effective across heterogeneous NSAI systems.

Table 3 further reveals that signal usefulness also changes when signals are combined. Removing Constraint gives the best LOO result in four cells, including Sort-of-CLEVR/ISED and Visual Sudoku/NASR, whereas removing Grounding is best for HWF-7/DOLPHIN, HWF-7/NASR, and ReaSCAN/NeSyCoCo. Sort-of-CLEVR/DOLPHIN instead benefits most from removing Novelty, while Visual Sudoku/DOLPHIN favors removing Drift. These results show that a signal that is strong individually is not necessarily equally useful in a multi-signal combination. More importantly, both Single and LOO analyses are post-hoc diagnostics and do not determine the appropriate signal weights in advance, motivating RCW and CSG to learn signal allocations from dataset–model reliability and candidate-pool context, respectively.

6.4 Adaptive Weighting Learns Effective Signal Allocations

The preceding diagnostics show that signal importance varies substantially across model–task cells, making weight allocation itself the key challenge. RCW learns a cell-level allocation from calibrated signal reliability, whereas CSG further adapts the allocation to the current candidate-pool context.

Table 1 shows that both RCW and CSG outperform Fixed in all nine dataset–model cells, yielding positive gains in all 18 adaptive comparisons. The largest improvements occur on HWF-7: for DOLPHIN, RCW raises failure discovery from 18.098% to 24.719% ($+6.621$ pp) and CSG to 23.123% ($+5.025$ pp); for NASR, the corresponding gains are $+4.265$ pp and $+3.248$ pp. On HWF-7/ISED, RCW and CSG improve Fixed by 2.955 pp and 3.209 pp, respectively. The gains also extend across other reasoning interfaces. On Visual Sudoku/NASR, RCW and CSG improve Fixed by 2.295 pp and 2.437 pp; on Sort-of-CLEVR/DOLPHIN, CSG gains 1.222 pp; and on ReaSCAN, CSG still improves Fixed from 10.211% to 10.664%. These results show that adaptive allocation remains effective across different tasks, models, and signal-availability patterns.

Neither mechanism is uniformly superior: RCW achieves the higher rate in four cells and CSG in five. This reflects their complementary adaptation granularities—RCW captures dataset–model-level reliability, whereas CSG responds to candidate-pool context during search. Complete adaptive gains, learned RCW allocations, and cross-dataset analysis are provided in Appendix C.

Table 3: Leave-one-out (LOO) signal ablation. Each column removes the indicated signal from Fixed and renormalizes the remaining available weights. Entries are mean oracle failure discovery rates (%); bold marks the best LOO policy within each dataset–model cell.

Dataset–model	−B	−G	−R	−C	−N
Sort-of-CLEVR–ISED	10.632	12.305	12.577	**12.760**	12.659
Sort-of-CLEVR–DOLPHIN	12.668	13.995	14.117	14.305	**14.630**
Visual Sudoku–ISED	9.627	14.863	14.736	**15.031**	14.055
Visual Sudoku–DOLPHIN	1.727	4.639	**5.123**	4.719	5.012
Visual Sudoku–NASR	13.500	20.764	21.059	**21.881**	21.059
HWF-7–ISED	5.514	7.416	5.934	**7.711**	6.484
HWF-7–DOLPHIN	13.986	**18.998**	16.873	17.977	18.496
HWF-7–NASR	17.266	**24.994**	20.809	21.240	21.783
ReaSCAN–NeSyCoCo	8.922	**10.447**	9.646	10.127	9.857

Table 4: Wall-clock efficiency of the five policies. Runtime metrics are normalized to Random within each cell and geometrically averaged across nine cells; calibration is reported separately.

Policy	Runtime (×)↓	TTF (×)↓	Q_{first} (×)↓	Fail./s (×)↑	T_{cal} (s)
Random	1.00	1.00	1.00	1.00	–
Uniform	1.43	1.40	0.93	4.11	–
Fixed	1.41	1.36	0.94	4.65	–
RCW	1.40	1.35	0.93	5.10	0.23
CSG	1.40	1.36	0.93	5.08	9.21

6.5 Wall-Clock Fuzzing Efficiency

Table 4 compares five runtime quantities. *Runtime* measures total fuzzing time; *TTF* is the capped wall-clock time to the first failure; Q_{first} is the corresponding capped query count; *Fail./s* measures failure-discovery throughput; and T_{cal} reports the one-time per-cell calibration cost of the adaptive methods. The first four metrics are normalized to Random within each dataset–model cell and geometrically averaged across the nine cells. The complete timing protocol is given in Appendix D.

All four guided policies require roughly $1.4\times$ runtime of Random, showing that the main wall-clock cost comes from common signal extraction and ranking. Importantly, adaptation itself adds little online cost: RCW and CSG remain essentially at the Fixed runtime while achieving the highest failure throughput, $5.10\times$ and $5.08\times$ Random, respectively, compared with $4.65\times$ for Fixed and $4.11\times$ for Uniform. Guidance also reduces the number of evaluations required to reach the first failure: RCW and CSG both require about $0.93\times$ the capped first-failure query budget of Random. However, the added cost per guided evaluation means that wall-clock TTF remains 1.35–$1.36\times$ Random. Thus, adaptive guidance is not optimized for minimum latency to the first failure; its main efficiency advantage lies in sustained fuzzing, where it delivers about five times the failure throughput of Random with negligible additional online overhead beyond Fixed. The one-time calibration costs are 0.23 s for RCW and 9.21 s for CSG

7 Conclusion

We presented NSAI-Fuzz, a semantics-aware fuzzing framework that maps heterogeneous NSAI states into five availability-aware, rank-normalized risk signals. RCW provides interpretable dataset–model-level reliability calibration, while CSG further adapts signal allocation to the current candidate-pool context. Across nine dataset–model cells, structured five-signal guidance consistently outperforms Random and Uniform, while single-signal and leave-one-out analyses reveal substantial heterogeneity in signal importance. Both RCW and CSG further improve the mean failure discovery rate over Fixed in every evaluated cell. Wall-clock evaluation additionally shows that adaptive weighting introduces essentially no extra online cost beyond the common five-signal search, while RCW and CSG achieve about $5\times$ the failure throughput of Random during sustained fuzzing. These results show that a common fuzzing interface is practical across heterogeneous NSAI systems and that effective, efficient guidance should adapt signal allocation to the neural–symbolic interface.

Reproducibility Statement

The experimental unit, compatibility matrix, policy definitions, mutation oracles, query budget, beam width, ten paired attack seeds, and calibration protocol are specified in Sections 5–6 and Appendix E. Appendix B defines the adapter-level signals and semantic validity conditions. Tables 2 and 3 report the complete single-signal and leave-one-out results, while Appendix C provides the complete adaptive improvements, learned RCW allocations, and additional cross-dataset analysis. Appendix A contains supplementary derivations and method details. The anonymous artifact includes the adapters, configurations, attack seeds, candidate-level summaries, and analysis code required to reproduce the reported results and tables.

Ethics Statement

This work studies defensive reliability testing on synthetic or public benchmarks and involves no human subjects or private data. Guided fuzzing can make failures easier to find; semantic mutation constraints are reported so that discovered cases remain valid tests rather than unrestricted adversarial examples. Rates must not be used as cross-dataset robustness rankings because candidate distributions differ by task.

AI Use Statement

Generative AI tools were used to assist with language editing, document organization, and LaTeX drafting. The authors remain responsible for the scientific claims and verified the implementation descriptions, formulas, experimental records, and numerical results against the retained artifacts. AI tools were not treated as a source of experimental evidence.

References

Cristina Cornelio, Jan Stuehmer, Shell Xu Hu, and Timothy Hospedales. Learning where and when to reason in neuro-symbolic inference. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=en9V5F8PR-.

Danial Kamali, Elham J. Barezi, and Parisa Kordjamshidi. NeSyCoCo: A neuro-symbolic concept composer for compositional generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 4184–4193, 2025. doi: 10.1609/aaai.v39i4.32439.

Dongning Ma, Jianmin Guo, Yu Jiang, and Xun Jiao. HDTest: Differential fuzz testing of brain-inspired hyperdimensional computing. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 391–396, 2021. doi: 10.1109/DAC18074.2021.9586169.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Aaditya Naik, Jason Liu, Claire Wang, Amish Sethi, Saikat Dutta, Mayur Naik, and Eric Wong. DOLPHIN: A programmable framework for scalable neurosymbolic learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 45461–45483. PMLR, 2025.

Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. TensorFuzz: Debugging neural networks with coverage-guided fuzzing. In *International conference on machine learning*, pp. 4901–4911. PMLR, 2019.

Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. DeepXplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*, pp. 1–18, 2017. doi: 10.1145/3132747.3132785.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Timothy P. Lillicrap. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4967–4976, 2017.

Alaia Solko-Breslin, Seewon Choi, Ziyang Li, Neelay Velingker, Rajeev Alur, Mayur Naik, and Eric Wong. Data-efficient learning with neural programs. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering (ICSE)*, pp. 303–314, 2018. doi: 10.1145/3180155.3180220.

Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6545–6554. PMLR, 2019.

Zhengxuan Wu, Elisa Kreiss, Desmond C. Ong, and Christopher Potts. ReaSCAN: Compositional reasoning in language grounding. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. DeepHunter: A coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, pp. 146–157, 2019. doi: 10.1145/3293882.3330579.

A Additional Derivations and Formal Details

A.1 Fuzzing as a constrained finite-horizon process

Let $\sigma_t = (\mathcal{B}_t, \mathcal{A}_t, t, q_t)$ denote the search state, where q_t is the remaining query budget. A policy selects a beam from the valid pool; the mutator and model jointly induce the next-state kernel $P_T(\sigma_{t+1} \mid \sigma_t, \mathcal{B}_{t+1})$. The constrained return is

$$V^\pi(\sigma_0) = \mathbb{E}_{\pi, P_T}\left[\sum_{t=0}^{H-1} \sum_{x \in C_t} f(x) \;\middle|\; C_t \subseteq \mathcal{V}(x_0),\ \sum_t |C_t| \le Q\right], \tag{22}$$

where $\mathcal{V}(x_0) = \{x : x \equiv_y x_0\}$ is the semantic validity set. The score is not itself a reward model: its top-B operator defines π. Offline re-ranking fixes C_t from a behavior policy and therefore does not integrate over the transition kernel induced by a new frontier.

A.2 RCW from a Gibbs posterior

For available signals, introduce a Lagrange multiplier ν for $\sum_j w_j = 1$. Differentiating Equation equation 12 gives

$$u_j - \beta^{-1}\left(\log \frac{w_j}{w^0_{a,j}} + 1\right) + \nu = 0. \tag{23}$$

Thus $w_j \propto w^0_{a,j} e^{\beta u_j}$. The normalizer is the partition function $Z(u) = \sum_{k:a_k=1} w^0_{a,k} e^{\beta u_k}$, yielding $w_j = \partial \log Z / (\beta \partial u_j)$. Its Jacobian is

$$\frac{\partial w_j}{\partial u_k} = \beta w_j (\mathbb{1}[j = k] - w_k). \tag{24}$$

Large β reacts sharply to AUROC estimation error, while $\beta \to 0$ recovers the masked prior.

A.3 Contextual ranking risk

Let $\mathcal{D}$ be the calibration distribution over candidate pools and $\mathcal{P}(C)$ the failure/non-failure pairs in pool C. The population objective approximated by CSG is

$$\mathcal{R}(W) = \mathbb{E}_{C \sim \mathcal{D}} \mathbb{E}_{(p,n) \sim \mathcal{P}(C)} \ell\big(w_W(v_C)^\top (\rho_p - \rho_n)\big) + \lambda \|W\|_F^2 + \gamma D_{\mathrm{KL}}(w^0_a \| w_W(v_C)), \tag{25}$$

where $\ell(m) = \log(1 + e^{-m})$. Since $\mathbb{1}[m \le 0] \le \ell(m)/\log 2$, logistic risk controls local pairwise misranking. It does not imply global regret under Equation equation 22; the on-policy matrix evaluates long-horizon behavior.

B Adapter and Oracle Specifications

Table 5: Compatibility-driven evaluation matrix and semantic mutation oracle.

Dataset	Models	Mutation/oracle
Sort-of-CLEVR	ISED, DOLPHIN	Imaging changes; fixed generated scene, question, and answer
Visual Sudoku	ISED, DOLPHIN, NASR	Digit rerendering and board symmetries; fixed solution
HWF-7	ISED, DOLPHIN, NASR	Same-class real glyph replacement and bounded imaging; fixed expression
ReaSCAN	NeSyCoCo	Imaging changes; fixed scene graph, command, and referent

Sort-of-CLEVR. Scenes contain colored geometric objects and binary relational questions. Mutations alter brightness, contrast, color temperature, and bounded noise without changing object count, identity, spatial relation, or question. ISED and DOLPHIN expose answer distributions and relational states. Boundary is the correct-answer execution margin; grounding is object or execution uncertainty; drift uses relation states; constraint measures mass inconsistent with the generated scene program. Recorded clean QA accuracy is 100.0% for ISED and 99.63% for DOLPHIN.

Visual Sudoku. A grid is rendered from digit glyphs while the puzzle and unique solution remain fixed. Exact row/column-preserving transformations and bounded glyph changes define validity. ISED and DOLPHIN connect digit perception to a solver; NASR exposes correction decisions around symbolic reasoning. Constraint is normalized row, column, and box inconsistency. The representation is the perception tensor, provenance execution vector, or correction-aware state. Recorded exact-grid accuracy is 78.2% for ISED and 75.6% for NASR; the seed-correctness filter, rather than aggregate clean accuracy, defines every model's eligible fuzzing corpus.

HWF-7. Seven-symbol handwritten arithmetic expressions are mutated through same-class glyph replacement and bounded rendering changes. The expression tree and numeric target remain unchanged. Grounding measures digit/operator uncertainty, drift measures the neural or reasoning state, and constraint measures probability mass assigned to invalid executions. ISED, DOLPHIN, and NASR receive the same valid candidate stream but expose different raw states. Clean exact-expression accuracies are 97.75%, 55.21%, and 91.08% for ISED, DOLPHIN, and NASR, respectively.

ReaSCAN. Generated scenes pair layouts with compositional navigation commands. Photometric changes preserve scene graph and command. NeSyCoCo exposes object concept scores and the executed concept-composition program. Boundary, grounding, drift, and novelty are available. No independent constraint checker is exposed, so constraint is masked for adaptive methods; constraint-only is a degenerate all-tie diagnostic and is not interpreted as a usable guide. NeSyCoCo reaches 98.5% clean test accuracy in this adapter.

C Complete Adaptive Results and Weight Analysis

This section provides the complete adaptive results and learned RCW allocations. All failure discovery rates are candidate-level means across the same ten attack seeds used in the main experiments.

Table 6 gives the complete adaptive improvements over Fixed. Both RCW and CSG improve failure discovery in every evaluated cell, yielding positive gains in all 18 adaptive comparisons. The largest gains occur on HWF-7/DOLPHIN, where RCW improves Fixed by $6.621\,\mathrm{pp}$ and CSG by $5.025\,\mathrm{pp}$, followed by HWF-7/NASR with gains of $4.265\,\mathrm{pp}$ and $3.248\,\mathrm{pp}$, respectively. Substantial improvements also appear on HWF-7/ISED and Visual Sudoku, while smaller but still positive gains are observed on Sort-of-CLEVR and ReaSCAN. These results show that adaptive signal allocation is effective across distinct neural–symbolic interfaces rather than being driven by a single dataset or model family.

Table 6: Adaptive improvements over Fixed. Δ denotes the percentage-point increase in mean oracle failure discovery rate.

Cell	Policy	Fixed	Adaptive	Δ
SoC/ISED	RCW	12.675	13.832	+1.157
	CSG	12.675	13.526	+0.851
SoC/DOLPHIN	RCW	14.335	14.914	+0.579
	CSG	14.335	15.557	+1.222
Sudoku/ISED	RCW	14.133	16.062	+1.929
	CSG	14.133	16.629	+2.496
Sudoku/DOLPHIN	RCW	4.908	5.764	+0.856
	CSG	4.908	5.740	+0.832
Sudoku/NASR	RCW	21.303	23.598	+2.295
	CSG	21.303	23.740	+2.437
HWF/ISED	RCW	6.725	9.680	+2.955
	CSG	6.725	9.934	+3.209
HWF/DOLPHIN	RCW	18.098	24.719	+6.621
	CSG	18.098	23.123	+5.025
HWF/NASR	RCW	22.385	26.650	+4.265
	CSG	22.385	25.633	+3.248
ReaSCAN/NeSyCoCo	RCW	10.211	10.283	+0.072
	CSG	10.211	10.664	+0.453

Table 7: Learned RCW signal allocations in the order (b, g, r, c, n). The variation across cells reflects dataset–model-dependent signal reliability.

Cell	b	g	r	c	n
SoC/ISED	.468	.238	.228	.010	.056
SoC/DOLPHIN	.507	.261	.184	.011	.036
Sudoku/ISED	.585	.099	.083	.192	.041
Sudoku/DOLPHIN	.726	.141	.021	.099	.014
Sudoku/NASR	.623	.111	.064	.175	.026
HWF/ISED	.446	.174	.254	.090	.036
HWF/DOLPHIN	.613	.141	.184	.013	.048
HWF/NASR	.532	.138	.277	.011	.042
ReaSCAN/NeSyCoCo	.446	.191	.255	.000	.108

Table 7 shows that the learned RCW allocations vary substantially across dataset–model cells. Boundary receives the largest weight in every cell, but its mass ranges from .446 on HWF-7/ISED and ReaSCAN/NeSyCoCo to .726 on Visual Sudoku/DOLPHIN. Representation drift also varies strongly, from .021 on Visual Sudoku/DOLPHIN to .277 on HWF-7/NASR, and receives substantial weight on ReaSCAN/NeSyCoCo (.255). Constraint is particularly prominent on Visual Sudoku/ISED (.192) and Visual Sudoku/NASR (.175), but receives little mass in several Sort-of-CLEVR and HWF-7 cells and is unavailable for ReaSCAN. These differences show that RCW does not merely recover the same global prior, but reallocates signal mass according to cell-specific reliability.

The weight patterns also provide a task-level interpretation of the adaptive results. Visual Sudoku assigns relatively more mass to boundary and constraint signals, consistent with the interaction between local perceptual errors and global symbolic constraints. HWF-7 exhibits stronger model dependence: representation drift receives comparatively high weight for ISED and NASR, whereas boundary remains dominant for DOLPHIN. Sort-of-CLEVR shows more moderate adaptive gains despite different learned allocations, while ReaSCAN demonstrates availability-aware portability: the constraint signal is masked, yet CSG still improves over Fixed using the remaining signals. Detailed adapter and oracle definitions are provided in Appendix B.

D Wall-Clock Evaluation Protocol

We evaluate the wall-clock efficiency of Random, Uniform, Fixed, RCW, and CSG using the same ten paired attack seeds, 20 initially correct inputs, 256 candidate evaluations per input, beam width eight, mutation families, checkpoints, and hardware as in the main evaluation. Each timed attack runs alone on one RTX A6000. Timing includes mutation generation, model execution, signal extraction, ranking, and frontier updates, while checkpoint loading and clean-input validation are excluded. CUDA is synchronized at measured inference boundaries, and every timed attack executes exactly 256 candidates.

For policy p, dataset–model cell c, and attack a, let $T_{pca}^{\mathrm{search}}$ denote the total online search time. We also record the wall-clock time and query index of the first discovered failure, denoted by T_{pca}^{first} and Q_{pca}^{first}, respectively. For attacks that discover no failure, these quantities are capped at the total search time and 256 queries to avoid the optimistic bias of averaging only successful attacks.

Failure throughput measures the number of discovered failures per unit online time:

$$E_{pc}^{\mathrm{time}} = \frac{\sum_a F_{pca}(T_{pca}^{\mathrm{search}})}{\sum_a T_{pca}^{\mathrm{search}}}, \tag{26}$$

where $F_{pca}(t)$ is the cumulative number of oracle failures discovered by time t.

Because absolute inference latency differs substantially across models, runtime, time to first failure, query count to first failure, and failure throughput are normalized to Random within each dataset–model cell. The resulting multiplicative ratios are aggregated across the nine cells using the geometric mean. Calibration time is reported separately because RCW and CSG are calibrated once per dataset–model cell and reuse the resulting weights or gate parameters across subsequent fuzzing runs.

E Reproduction Checklist

The evaluation contains nine compatible cells, fifteen policies, and ten paired repetitions, for 1,350 runs. Each run samples 20 initially correct inputs and permits 256 candidate evaluations per valid input with beam width eight. Reproduction proceeds by (i) verifying checkpoint clean accuracy and the seed-correctness filter; (ii) fitting RCW/CSG on random-frontier calibration pools without final outcomes; (iii) executing all policies from scratch; and (iv) aggregating candidate-level failure discovery rates by attack seed and reporting their mean across the ten paired repetitions. A complete audit requires 150 summaries per cell and checks that every policy is present ten times.