
Monotone Guardrails: A Learned Veto That Cannot Loosen Agent Permissions

Eduardo Perez Denadai
Optimality.ai

Abstract

The final execution boundary for an autonomous agent tool call should be governed by a discrete binary decision accompanied by an auditable stage identifier, rather than a probabilistic score. Scores are the wrong contract: they must be emitted for every input including ones outside their training distribution, they degrade silently, and they have no vocabulary for the most common real answer—a *required fact was never stated*. We present a two-key release pipeline, which answers one question per action—may this run without a human?—with binary noul and the stage that signed it: blocked-by-authority with the rule that fired, abstained-by-authority with the missing fact named, vetoed-by-reader with its verdict, or released with both keys countersigned. The architecture is two-key and monotone by construction: an exact, rules-as-data authority over a closed verb dialect does all the classifying, and a trained temperature-0 reader sits in a veto lane it cannot escape—it never sees a non-ROUTE, its error fails closed, and turning it off degrades to a named authority-only release. We prove the no-loosen property and enforce it with a spy that asserts the reader generates zero calls on denied states; the selftest runs on every commit, and the release is blocked unless it passes. Measured on an independent 100-case action battery with a reference-free hazard method: **zero real danger-misses**, a per-stage audit trace (authority block 52, reader veto 10, released 38), zero determinism violations. On a real 746-action model log: 547 routed, 38 escalated, 161 refused—every refusal naming the fact the arguments never contained. Action refusals provide actionable diagnostic records: each denial is logged with a specific parameter missingness tag, enabling targeted data-level policy remediation; the two-key invariant is excluded from the backlog by construction. The boundary with scalar decision heads is stated as an invariant—a number may rank the escalation queue, but a number never overrides a guard denial, and a guard release never depends on a number.

1 Introduction

Agent frameworks increasingly execute tool calls autonomously, and the industry’s answer to “is this safe?” is converging on learned scores: a classifier probability, a judge’s rating, a threshold. Scores are the wrong contract for an execution boundary. A score must be emitted for every input, including inputs outside its training distribution; a score degrades gradually and silently; and a score has no vocabulary for the most common real answer—a *required fact was never stated*.

The gate answers one question per action with a bit and a named stage:

```
1 / released          ·      1 / released-authority-only          ·      0 /  
blocked-by-authority:<rule> · 0 / abstained-by-authority:<fact> · 0 /  
vetoed-by-reader:<verdict> · 0 / reader-error
```

Three design commitments follow.

Preprint.

- (G1) **Closed vocabulary, fail-closed default.** The authority is an exact policy over a fixed verb dialect; anything unclassifiable abstains to a human. The system enforces fail-closed defaults: if a required decision parameter is omitted from the input context, the authority explicitly identifies the missing attribute.
- (G2) **Monotone two-key release (no loosening by learning).** A released action requires authority ROUTE *and* reader ROUTE; the reader is never consulted on a denied action and cannot upgrade one. Reader error fails closed.
- (G3) **Policy is data.** The dialect map and rules are tables, not code paths; adding a verb is a reviewed policy change. The gate contains no learned parameter on the authority path.

Contributions: (1) the two-key pipeline with the no-loosen property, stated as a proposition and enforced by an offline selftest that injects a spy reader (§3); (2) the named-refusal contract—every denial carries the rule or missing fact, making refusals an actionable queue rather than an error rate (§4); (3) evaluation on a 100-case independent battery and a 746-action real log with per-stage audit traces (§5); (4) the improvement lifecycle in which every refusal becomes data (dialect row, fact derivation, or battery case) and the two-key invariant is explicitly *not* part of the backlog (§6).

2 Related work

Guardrails for LLM agents. Constitutional and rule-based input/output filters, NeMo-style dialogue policies, and safety-classifier model families (Llama-Guard class) gate model text; sandboxed simulation harnesses [Ruan et al., 2024] probe tool-use risk systematically and document single-factor blind spots in action vetoes—the motivation for our ten-axis battery. The contrasts are structural. Against the Llama-Guard family [Inan et al., 2023]: a single learned classifier assigns safety classes to text, so a model error is a policy error; here the learned component sits in a veto lane that can only tighten an exact policy, so a reader error can only add denials. Against NeMo-style dialogue rails [Yang et al., 2023]: rails govern what the model may *say*; this gate governs what the agent may *execute*, keyed two ways. Against OPA-style engines (below): those provide the exact allow/deny posture, but no learned component is permitted near the decision; we add one, quarantined. In every case, every outcome names its signers.

Policy engines and rules-as-data. Production allow/deny engines (OPA-style) express policy as data and keep evaluation exact; we marry that posture to an LLM veto reader, keeping the trusted policy path parameter-free.

Scalar decision heads. A companion calibrated head [Denadai, 2026] answers the same prompt with $\text{noul} \in [0, 1]$ for triage. The two compose under a strict boundary: a decide number may rank an escalation queue; it never overrides a guard denial, and a guard release never depends on a decide number. Selective-prediction theory [Chow, 1957] explains when abstention is optimal; we add the requirement that abstention be *named* so it can be discharged.

Veto-based and monotone architectures. Two-key release resembles monotone security architectures and fail-safe defaults in safety-critical systems: untrusted components may restrict but never extend authority. The reader is treated exactly as an untrusted reviewer with a single capability: veto.

3 Architecture

3.1 The pipeline

Figure 1: (1) a structural parser extracts the proposed tool call from the prose state—a state with no parseable call is not decidable here and abstains (`no_action_parsed`); (2) the dialect table maps SQL-flavoured verbs onto the authority’s closed (tau-bench) vocabulary—an unmapped verb is an action the authority cannot classify, so it abstains (`unknown_verb`: ·); (3) the authority evaluates the mapped action exactly: `ESCALATE` → 0 with the rule that fired, `ABSTAIN` → 0 with the missing fact, `ROUTE` → the veto lane; (4) the veto reader, a quantized 1.5B policy at temperature 0 with byte-exact battery-validated templates, answers one line—ROUTE releases (1), anything else vetoes (0), and an exception is `reader-error` (0, fail-closed).

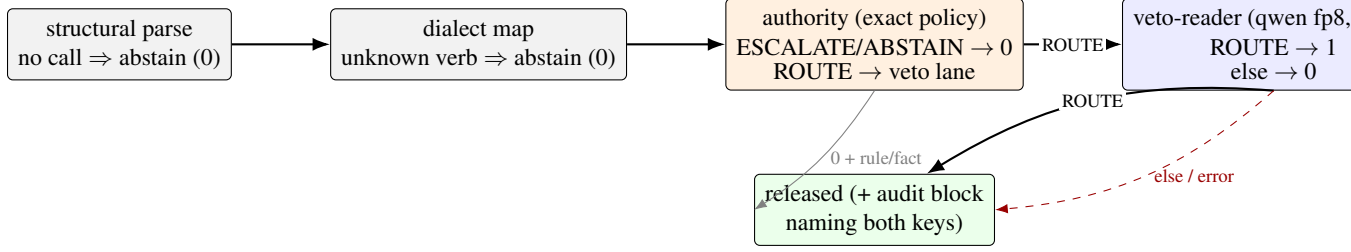


Figure 1: The two-key pipeline. Order is the design: the reader is only ever reached through an authority ROUTE; every exit carries a stage string.

Proposition 1 (No loosening). *For every state $s \in \mathcal{S}$,*

$$\text{guard}(s) = 1 \implies (\text{authority}(s) = \text{ROUTE} \wedge \text{reader}(s) = \text{ROUTE}),$$

where reader is evaluated only when deployed and only on authority-ROUTE states. Hence no learned component can release, upgrade, or reclassify a denied action.

Proof. Let $C(s)$ be the event “the reader is consulted on s .” By the pipeline order in Figure 1, the reader branch is guarded by $\text{verdict} == \text{ROUTE}$, so $C(s) \Leftrightarrow \text{authority}(s) = \text{ROUTE}$ and all other verdicts return before reader invocation. The release rule requires $C(s)$ and $\text{reader}(s) = \text{ROUTE}$, so $\text{guard}(s) = 1$ implies both conjuncts. Contrapositive: $\text{authority}(s) \neq \text{ROUTE} \Rightarrow \neg C(s) \wedge \text{guard}(s) = 0$ —denials depend only on the exact policy, the dialect table, and the structural parser, all parameter-free. $\square$

The property is machine-tested: the offline selftest monkeypatches the reader entry point with a counting stub, replays a fixture set covering all six stages, and asserts the call counter is exactly zero for every non-ROUTE state—unknown-verb and escalation fixtures generate *zero* reader calls, alongside fail-closed reader-error, binary-only contract enforcement (a non-binary noul is rejected), and stage correctness—11 checks, run in CI without GPU or weights; a failing check blocks the release.

3.2 Templates are part of the checkpoint

The reader’s prompts are byte-identical to the ones the battery validated, and the measured facts they encode are documented in the code: raw prose asks are the measured failure regime; and a call-line-only prompt drops stated-context hazards—the three real dangerous misses of an earlier round—so the shipping template carries both the action *and* its stated context. The reader is married to one checkpoint and one template regime: a new reader model means the battery reruns, not a configuration bump. Two distinct sensitivities must be separated. *Semantic* sensitivity is measured: omitting the stated-context block (the action-only template) caused the three real dangerous misses of the earlier round, which is why the shipping template carries both. *Formatting* sensitivity—minor prompt paraphrase or context reordering—has *not* been measured; the deployed guarantee is byte-identical templates, and robustness beyond that is an open question (§9), not a claim.

4 The refusal contract

Every denial is data. `blocked-by-authority:<rule>` names the policy rule; `abstained-by-authority:<fact>` names the fact that was never stated (`missing:money`, `missing:backup`, ...) or the verb the dialect table does not know; `vetoed-by-reader:<verdict>` names the reader’s verdict line. Downstream, refusals become obligations for the proof engine: a refusal with a named missing fact is precisely the shape of claim a verifier can later discharge, closing the loop from “cannot decide” to “here is what would decide it.” The response envelope is the standard decision-model wire format (binary noul only) with an added decision audit block—stage, authority verdict and reasons, reader surface and verdict, and latency—so a report can always tell which keys signed. Concretely, every stage has one wire shape:

```

REL  {"answers":{"safe_to_run":{"noul":1.0}},
      "decision":{"stage":"released",
                  "authority":"ROUTE","reader":"ROUTE"}}
BLK  {..., "decision":{"stage":"blocked-by-authority",
                  "authority":"ESCALATE",
                  "reason":"no-approval-declared"}}
ABT  {..., "decision":{"stage":"abstained-by-authority",
                  "authority":"ABSTAIN",
                  "missing":["money"]}}
VETO {..., "decision":{"stage":"vetoed-by-reader",
                  "authority":"ROUTE",
                  "reader":"UNSAFE_CONTEXT"}}
ERR  {..., "decision":{"stage":"reader-error",
                  "authority":"ROUTE","noul":0.0}}

```

(released-authority-only carries the same envelope with "reader":"off".) Refusal stages always present noul = 0 and a populated diagnostic; releases always present both keys.

5 Evaluation

Offline selftest. Eleven checks (contract envelope, binary enforcement, dialect mapping, unknown-verb and no-parse abstention, authority-only release, authority escalation, reader veto with exact-context surface, reader release, reader-error fail-closed, and the no-loosen spy) pass with a stub reader—the authority path is pure Python and runs without GPU or weights, which is what makes the invariant testable on every commit.

Independent 100-case battery. The Mercury-100 action battery (ten axes; 41 single-factor probes, an 8-cell confounded array, 51 exact repeats) with a reference-free hazard lexicon as the leak detector. Round-5 record for the two-key pipeline: **zero real danger-misses**, per-stage audit trace with the histogram 52 blocked-by-authority / 10 vetoed-by-reader / 38 released, and zero determinism violations. The cells where the gate is more permissive than the reference model are the battery’s own documented array-noise cases: near-identical read-only previews that the reference scores inconsistently. The composition matters as much as the totals: 38 releases all carry both keys; every one of the 62 denials names its stage; and the authority-only ablation (reader off) isolates each key’s contribution on the same states.

Real action log. A 746-action log from a model’s own trajectory: 547 routed, 38 escalated (each naming the human-decision reason), 161 refused (each naming the missing fact—e.g., missing:money: the arguments never named an amount). Exit code 1 on any refusal, so guard && deploy behaves as written. Stratification: every one of the 161 refusals carries its stage and missing-fact tag in the run’s refusals.md, so the log stratifies by failure type at the artifact level (fact gap vs. rule block vs. unmapped verb), and per-verb tallies are recomputable from the same artifact; the verb domains present are the SQL-flavored set (scope, §7). Downstream resolution—how many refusals were cleared by parameter clarification versus dialect expansion—was not tracked longitudinally in this run; the ledger records refusal reasons, not their closures. Closing that loop (queue-closure telemetry) is listed among the open challenges (§9). The authority side of this run doubles as the regression corpus for policy changes: rule edits ship as data plus a differential test against all 746 recorded actions, and escalation rate is reported as a bound, never optimized as a target.

Authority census. On the battery’s own dialect, the exact authority classifies 100 states as 48 ROUTE / 48 ABSTAIN / 4 ESCALATE—the closed vocabulary doing most of the refusing, with the reader adjudicating only the 48 routed actions.

6 Improvement lifecycle

Driver: *coverage without loosening*. Every refusal lands in one of three queues, and each queue has exactly one first-line fix:

- (Q1) **Dialect gap** (`unknown_verb:·`) → add a reviewed DIALECT row, mapped to a verb the authority classifies *honestly*—never to a verb that dodges a rule.
- (Q2) **Fact gap** (`missing:·`) → decide whether the frontend can derive the fact, or the abstention *is* the product answer.
- (Q3) **Reader queue** (false vetoes) → battery cases for the next reader round; reader rounds are battery-gated with the bar registered before the run: zero real danger-misses, monotonicity on every release, zero determinism violations.

Improvement is data-first: a dialect row or policy rule before a checkpoint, a checkpoint before (never) a code-level `if` in the authority. In operation we collect the stage histogram (it sizes the queues directly), human outcomes on the only two labels that matter (released-but-shouldn't, blocked-but-shouldn't), and reader latency and trigger rate—if the trigger share grows, that is an authority coverage question, not a reader speed question. Any `reader-error` is an incident, not a metric. The two-key invariant is explicitly outside the backlog.

7 Limitations

The authority's vocabulary is closed by design: coverage grows only through reviewed data changes, so recall on genuinely novel verb families is zero until a human adds them—intended, but a real usability cost. The maintenance burden is quantifiable and bounded: one reviewed data row per surface verb (plus aliases), each change gated by a differential test against all 746 recorded actions, so a dialect edit cannot regress the corpus silently. Schema evolution in *external* APIs behaves differently from verb drift: argument renames hit the fact layer, surfacing as `abstained-by-authority:⟨fact⟩` (e.g. a renamed amount field reads as `missing:money`), not as `unknown_verb`, because the verb lexicon is stable across argument schemas; the abstention is the correct fail-closed response and the fact-gap queue is the remediation path. Semi-automated proposal of dialect rows (schema/AST mining producing *candidate* rows under human review) is future work, not deployed: the row is the policy, and a proposal is not a policy until reviewed. On cost: the authority path is pure Python, sub-millisecond, GPU-free; the reader costs ~ 0.5 s per firing on a GPU and fires only on authority-ROUTE states (the battery census routed 48/100), so amortized overhead is $\text{trigger-rate} \times \text{latency}$ and is an authority-coverage question before it is a compute question. At high call frequency the veto lane dominates end-to-end latency; distilling the reader into a lighter discriminative veto—with battery-gated parity as the acceptance bar—is future work. The battery is $n = 100$, one domain (SQL-flavoured tool calls), with documented noisy cells; danger-miss counts on natural action distributions remain unmeasured at scale. The dialect mapping is hand-authored, and the multi-domain question (shell, cloud, webhooks) is open (§9). Finally, the reader inherits its training distribution from a single post-trained checkpoint—version coupling is enforced, but transfer beyond the battery's grammar is exactly the open question the queues exist to answer.

8 Conclusion

The system argued here: the final execution boundary for agent actions should be a bit with a named stage, not a probability. The bit comes from an exact, data-defined policy; the learning is quarantined in a veto lane that can only tighten, enforced by construction and by a spy test that runs on every commit; and the refusals are named facts, which is what makes them fixable. Measured: zero real danger-misses on an independent battery with a full per-stage trace (52/10/38), and 746 real actions sorted into routed / escalated / refused-with-a-reason— each outcome an addressable queue entry, not an error bar. Probabilistic decision heads may be utilized to rank human escalation queues, whereas the monotone guardrail functions as the non-bypassable execution gate.

9 Future Work and Open Challenges

This work was conducted within the hybrid-neurosymbolic research program of **Optimality.ai** (neural proposers, symbolic authorities, fail-closed contracts). The open challenges are as follows.

First, dialect scalability: semi-automated proposal of policy rows via schema/AST mining under human-in-the-loop review, so coverage grows without hand-authoring becoming the bottleneck—while the reviewed row remains the unit of policy. Second, multi-domain evaluation: batteries

and logs from heterogeneous tool environments (OS shell, cloud infrastructure CLIs, web APIs), demonstrating that the two-key pipeline scales beyond database operations. Third, reader economics: distill the veto reader into a lighter discriminative or encoder-based classifier with battery-gated parity, and measure template robustness beyond byte-identical prompts (paraphrase and reordering sensitivity are currently unmeasured). Fourth, measure danger-miss rates on real fleet action logs with the only labels that matter (released-but-shouldn't, blocked-but-shouldn't), and instrument queue closure so refusal resolution (clarification vs. dialect expansion) becomes measurable. Fifth, productize the stage-histogram telemetry so the three queues size themselves in operation; and sixth, tighten the loop with the scalar head: the calibrated head ranks the escalation queue, the guard decides, and a number never overrides a denial.

References

- C. K. Chow. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, EC-6(4):247–254, 1957.
- Eduardo Perez Denadai. A calibrated safety-decision head for tool-using agents. In *Optimality.ai Technical Report*, 2026. Companion paper.
- Hamish Inan, Omkar Upadhyay, Zifan He, et al. Llama Guard: Input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*, 2023.
- Joyce Ruan, Wenxuan Di, Yinheng Li, et al. Identifying the risks of LM agents with an LM-emulated sandbox. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.
- Harrison Yang, Vincent Traag, et al. NeMo Guardrails: A toolkit for controllable and safe language model applications with dialograils. *arXiv preprint arXiv:2312.17231*, 2023.

Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Claims are scoped to the two-key invariant (proved and selftested), the 100-case battery record, and the 746-action log; composition is stated explicitly.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Limitations name the closed vocabulary, checkpoint-template coupling, n=100 battery, hand-authored dialect, and unmeasured natural-distribution danger rates.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: The no-loosen proposition is stated with its assumption (pipeline order) and proven in one paragraph; the selftest is the machine check.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Battery construction, stage histograms, selftest contents, dialect table provenance, and the registered reader-round bars are disclosed.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- Reproducibility requires a reasonable avenue for the reported results, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Code and recorded batteries will accompany the final version; the paper discloses stage vocabulary, histograms, and all measured counts.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: Sections 3-5 describe the pipeline, templates, contract envelope, and queues in implementation-level detail.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: Zero danger-misses with per-stage trace, stage histogram, determinism violations reported; battery n and repeat structure stated (no sampling claims beyond n=100).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: Authority path is pure Python milliseconds-scale; the reader is one quantized 1.5B model at 0.5 s per firing on a single GPU; no cluster used.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the code of ethics appropriate to the authors' field and institutions?

Answer: [\[Yes\]](#)

Justification: The authors have reviewed the applicable code of ethics.

Guidelines:

- The answer NA means that the authors have not reviewed a code of ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[Yes\]](#)

Justification: A fail-closed last line of defense with named refusals reduces silent autonomous risk; misuse considerations (over-reliance on the bit) are bounded by the named-stage audit.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms

for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: A safety gate with fail-closed defaults; the reader artifact is a small quantized research checkpoint with a battery gate.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Base model and battery provenance cited with upstream licenses; tau-bench action vocabulary cited to its source.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The battery states, dialect table, and selftest suite are new assets documented for release.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No crowdsourcing.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- Under prevailing codes of ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the applicable code of ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The learned veto reader is a core component; its checkpoint, temperature, and templates are fully specified.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.