Exploitability is a property of a verifier and its candidate pool, not the verifier alone

W. Schulz

Jul 29, 2026

Abstract

The same verifier's exploitability ranged 6.2× (0.153–0.949) depending only on its prefilter, so ε belongs to the verifier–pool pair. Per-pool regime detection fails, but averaging over a window of ~50 pools succeeds. Against adaptive generators, decorrelated panels help only partly; an out-of-span channel did about 8× better.

The verifier-bottleneck track exists to answer one question: how far can you push test-time sampling before an imperfect verifier turns more compute into *worse* answers? Its founding move was to split "imperfect verifier" into two orthogonal axes — discrimination, which sets the height of the accuracy ceiling, and exploitability, the rate at which wrong candidates happen to be things the verifier scores highly, which decides whether more sampling helps or hurts. The safe sample budget is governed by the second, not the first.

Four experiments run on 29 July 2026 changed that picture in one specific way, and then showed how to ship the change.

Exploitability is a property of a verifier *and its pool*

The promotion came from a toy result in a different strand, where the intuitive deployment rule — put your most trustworthy checker last — turned out to be false at tight budgets. Brought into this track's harness with its own conventions, one fixed verifier was placed behind five different prefilters, and its measured exploitability was recorded.

At a funnel width of ten, the same verifier's exploitability ranges from 0.153 to 0.949 — a factor of 6.2 — depending only on what the upstream filter hands it. Exploitability is not a property of a verifier. It is a property of a verifier–pool pair. Every safe-budget table and consensus threshold in this track is therefore implicitly conditioned on the pool distribution the number was calibrated on, which is now documented as a caveat on the released command-line tool: the exploitability you feed it must be calibrated on the *post-prefilter* pool the verifier will actually face.

Two pieces of structure confirmed the mechanism. Starvation is real and monotone — a clean prefilter drives exploitability from 0.73 down to 0.06 as the funnel tightens, because exploits are denied before the verifier ever sees them. And the policy ranking contains an inequality worth pinning up: a prefilter correlated with the finalist is worse than no prefilter at all, because it *concentrates* exactly the finalist's blind spots. Stacking correlated verifiers is negative-value, not merely redundant.

Three of six registered checks missed, and they share one cause: at tight funnel widths, pool-*size* effects dominate the registered pool-*density* story. The predictions modelled prefilters as reshaping exploit den-

sity only, and missed two mechanisms — that independently chosen exploit sets still overlap by chance, so an "independent" prefilter certifies exactly the double-exploiters through, and that a prefilter's own blind spots crowd correct candidates out of a tight pool. So the corrected rule has three regimes rather than one. At loose funnels anything non-correlated works. At moderate funnels decorrelation is as good as cleanliness, so buy the cheap one. At tight funnels cleanliness pays — and a correlated cut should never be placed anywhere upstream.

Detecting the regime: why the obvious guard fails and the window works

A parallel result had established the rescue rule and its danger. When a candidate pool is exploit-dominated, discarding the top eight scores before selecting lifts accuracy from 0.075 to 0.533. When the verifier is honest, the same rule *costs* you, dropping 0.920 to 0.388. Neither is a safe default, so something has to detect which regime a deployment is in — and a decorrelated reference channel can do it without labels.

The direct implementation was a per-pool guard: use a weaker, independent reference verifier, see where it ranks the main verifier's champion, and switch rules when the champion looks suspicious. It scored 2 of 4, and a corrected post-hoc detector scored 1 of 3. The reason is not conceptual but statistical, and the experiment measured it: the separation between honest and exploited pools in the guard's own statistic is about 0.19, against a per-pool standard error of about 0.35. There is simply not enough evidence in a single pool of a hundred candidates.

But regime shift is a property of the deployment, not of the pool. Average the same statistic over a window of pools and the standard error shrinks as the square root of the window length — which predicts near-perfect detection at a window of about fifty using the same weak reference channel.

That prediction was pre-registered with four checks and passed 4 of 4. At a window of fifty, the false-alarm rate is 0.000 and the miss rate is 0.010. The separation grows as $\sqrt{T}$ within the registered tolerance across the whole grid. End to end, the windowed guard reaches at least 95% of the *oracle* rule's accuracy at both ends of the regime axis — matching argmax where argmax is right and the trimmed rule where trimming is right — and it stays safe in the mid-regime where the two rules cross. Detection error falls from a coin flip at one pool to essentially nothing at fifty.

The shippable form of the result: do not try to classify each pool; classify the deployment. A weak independent channel that is useless per-request is decisive over a window, and the conversion factor is $\sqrt{T}$.

The regime flip: what a panel is for, and what it cannot do

The last experiment came out of the formal side of the same problem, and it is the one that changes the design advice.

This corpus held two apparently incompatible stories about why a checker cannot check itself. The measured story says the operative variable is correlation — a critic sharing the generator's context shares its standard of correctness, and no hierarchy is needed to explain the failure. Tarski's theorem says the hierarchy is forced. The reconciliation is that the liar is not *sampled*, it is *constructed*: diagonalization is optimization against the checker's own definition. So the two claims describe different regimes, and the boundary between them is whether the generator is adaptive.

The test was a coverage game: the generator places flaws inside its own control span, panels vary in how correlated their checked sets are, the adaptive generator white-box minimizes its rejection probability

over every possible flaw placement — the diagonal step made literal — and a surplus channel reads flaw leakage *outside* the generator's span. Eight checks, exact probabilities, run once. 7 of 8.

The flip is real and large. At full decorrelation the same panel misses 0.032 of static flaws and 0.559 of adaptive ones. Decorrelation approximately solves the static regime and cannot solve the adaptive one. In the static regime, miss rate rises monotonically with checked-set correlation, which is the correlation law arriving in coverage form.

The coverage cliff is exact. A panel that leaves three channels unchecked misses an adaptive generator with probability 1.0000; full coverage brings it to 0.00002. That is the toy image of *the metalanguage must be at least as expressive as the object language*, arriving as a pigeonhole result.

And surplus is the only near-solution under adaptivity. A placement-independent out-of-span channel floors the adaptive miss at 0.045–0.080 regardless of panel correlation, and wins the architecture decision outright: 0.917 against 0.720 and 0.658 for majority and union panels on a mixed pool.

The single miss tempers the claim in a way worth stating, because the strong version was the interesting one. A tenth in-span decorrelated critic still bought a 36.4% relative reduction under adaptivity, above the registered ceiling of 30.9% — partly a threshold-laxity confound, since five-of-ten is no longer a strict majority. So "only surplus survives adaptivity" is falsified as stated. The graded form stands: in-span additions help sub-proportionally, and the surplus channel is about eight times better than an equal-budget in-span critic.

The deployment rule, in one line

The advice this track shipped a month ago was *keep one decorrelated channel*. It sharpens to keep one out-of-span channel, and the difference is not cosmetic. Decorrelated-but-in-span critics are a static-regime asset that an adversary optimizes around; surplus — features the generator cannot model or route around — is the anti-diagonal asset. Match the panel architecture to the threat regime: redundancy and majority against noise, breadth and surplus against optimization.

Three attributions, stated plainly. The static half is owned — the correlation literature on ensembles and judges is well developed and cited in the track's gate record. The adaptive-failure half is owned too: reward-model ensembles are documented failing under optimization pressure *because* their members share error patterns [1], and unhackability is already known to be pool-relative [2]. The measured two-regime decomposition with a surplus floor and a coverage cliff appeared unclaimed when this was written — *that novelty claim was later withdrawn: see When the Exploit Is Constructed, which traces the two-regime split to Dwork et al. (2015) and the coverage half to Zhuang & Hadfield-Menell (2020)* — and in any case this is a toy. The real test is the large-model version, where an out-of-family reward channel is put up against an extra same-family reward model under actual optimization pressure, and that is the strand's next pre-registered step.

Sources

1. Eisenstein, J., Nagpal, C., Agarwal, A., et al. (2024). "Helping or Herding? Reward Model Ensembles Mitigate but do not Eliminate Reward Hacking." *COLM 2024*. arXiv:2312.09244 — ensembles do not eliminate reward hacking "because all reward models in the ensemble exhibit similar error patterns."

2. Skalse, J., Howe, N. H. R., Krasheninnikov, D., & Krueger, D. (2022). "Defining and Characterizing Reward Hacking." *NeurIPS 35*. arXiv:2209.13085 — unhackability is defined relative to a set of policies.
3. Hodges, W. (2022 revision). "Tarski's Truth Definitions." *Stanford Encyclopedia of Philosophy*. plato.stanford.edu — on Tarski's 1933 paper and the undefinability of truth within the object language.
4. Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., & Roth, A. (2015). "Preserving Statistical Validity in Adaptive Data Analysis." *STOC 2015*. arXiv:1411.2664 — the prior art that later withdrew this chapter's novelty claim.

All measurements in this chapter (funnel exploitability 0.153–0.949, the windowed guard, the coverage game) are this project's own run-once results, in `verifier_bottleneck/` *in the Quantum-Intelligence repository.*

Published by NeuroGenesys (AW3 Technology, Inc.). Canonical version: https://neurogenesys.si/research/exploitability-verifier-pool