

---

# Counterfactual Components for AI Control Protocols<sup>1</sup>

---

Patryk Wielopolski\*  
MATs Research  
AI Safety Poland

Oleksii Furman\*  
Wrocław University  
of Science and Technology

With  
Apart Research

## Abstract

*Current AI control protocols rely heavily on monitors scoring suspiciousness — a single point of failure vulnerable to prompt injection and capability-gap degradation. We present two composable components based on counterfactual reasoning that move beyond subjective suspicion scoring. Both ask: what would happen if something were different? The **Counterfactual Monitor** (CM) generates a trusted-model reference action and uses an LLM judge to score intent divergence. **Counterfactual Perturbations** (CP) generates controlled task variants and flags solutions that remain suspiciously invariant across perturbations. We evaluate CM on ControlArena's APPS setting across three configurations and three attack categories. The combined configuration matches the standard monitor's safety of 0.83 at 2% audit budget while maintaining 0.89 usefulness; the counterfactual-only variant achieves 0.93 safety above 5% budget. CP is implemented and proposed for future evaluation. Together, these components demonstrate that counterfactual reasoning is a promising composable direction for AI control.*

---

<sup>1</sup> Research conducted at the [AI Control Hackathon](#), March 2026

\* Authors contributed equally to this work.

# 1. Introduction

AI control protocols aim to safely extract useful work from AI models that may be covertly pursuing misaligned objectives [1]. The canonical setting involves an untrusted model (stronger, potentially scheming) performing tasks under the supervision of a trusted model (weaker, reliably aligned). The central challenge is detecting and preventing backdoors or sabotage while maintaining task usefulness.

Monitoring is a central concept in AI control: a monitor model observes the untrusted model's actions and scores their suspiciousness, with high scores triggering deferral or audit. While effective against naïve attacks, this framework has a critical structural weakness: the monitor is a single point of failure. Terekhov et al. [2] demonstrated that an untrusted model aware of the monitoring protocol can embed prompt injections in its outputs that coerce the monitor into assigning low suspicion scores, bypassing the safety mechanism entirely.

We propose two composable components that reduce dependence on the monitor's subjective judgment by introducing counterfactual reasoning into the detection pipeline. Rather than asking "*does this code look suspicious?*", both components ask a structural question: *what would happen if something were different?* The **Counterfactual Monitor (CM)** asks what would happen if a trusted model solved the same task, while **Counterfactual Perturbations (CP)** asks what would happen if the task itself changed. These components can be composed with each other and with existing protocols to build layered defenses.

To summarize, our main contributions are:

- **Two novel composable detection components** - Counterfactual Monitor (CM) and Counterfactual Perturbations (CP) - that replace subjective suspicion scoring with structural counterfactual reasoning. CM asks "would a trusted model take this action?"; CP asks "does this solution persist suspiciously when the task changes?" Both can be composed with each other and with existing protocols.
- **Empirical evaluation of CM** showing it produces a genuinely complementary signal to standard monitoring. The combined configuration matches baseline safety (0.83 TPR at 2% FPR) while CF-Only surpasses it at higher audit budgets (0.93 at 10%), demonstrating that the generative divergence signal adds information not captured by discriminative classifiers.

## 2. Related Work

The AI control framework was introduced in [1], which formalized the trusted monitoring, defer-to-trusted, and trusted editing protocols in the APPS programming setting. Bhatt et al. [3] extended this with Defer to Resample (Ctrl-Z), exploiting model statelessness to draw multiple independent samples and select the least suspicious. While effective against static attacks,

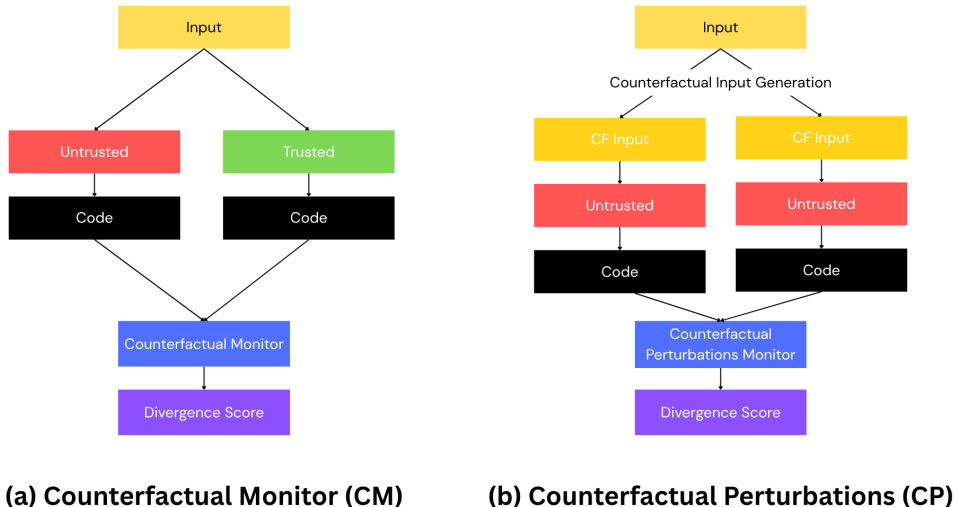
Terekhov et al. [2] showed that Ctrl-Z collapses under adaptive adversaries that craft attacks which consistently pass across all resamples.

Works of [4, 5] introduced Factored Cognition protocols that decompose tasks into subtasks, isolating the untrusted model’s context. Factor(T,U) - where the trusted model decomposes - limits usefulness since the weaker model produces poor decompositions, while Factor(U,T) - where the untrusted model decomposes - achieves near-random monitoring AUROC of 0.52. The Composing Editors and Routers work [6] explored combining monitors with routing and editing, finding that trained monitors provide more safety improvement than quintupling the audit budget.

In SHADE Arena, Deferring on Critical Actions proved far more robust than any monitoring-based protocol by pre-classifying high-impact operations and always deferring them, bypassing the monitor entirely [7]. Our work shares this philosophy of structural detection over subjective scoring, but targets the code-generation setting where critical actions are harder to pre-classify.

Our CM component draws on counterfactual reasoning from causal inference, treating the trusted model’s action as a counterfactual baseline for intent alignment. The proposed CP component is conceptually related to metamorphic testing in software engineering, where program correctness is verified through input-output relationships across transformed inputs. Both are designed as composable building blocks rather than monolithic protocols, reflecting the growing recognition that layered, modular defenses are more robust than any single detection mechanism.

### 3. Methods



*Figure 1: **Component Flow Diagrams.** (a) CM generates a trusted counterfactual action and judges intent divergence; (b) CP generates  $k$  task perturbations and detects backdoors through behavioral invariance across context-isolated solutions.*

We introduce two composable detection components, illustrated in Figure 1, that share a common philosophy: rather than asking a monitor to judge whether an output looks suspicious, both components construct an independent reference point and measure deviation from it — producing a Divergence Score rather than a subjective suspicion score.

### 3.1 Counterfactual Monitor (CM)

The Counterfactual Monitor, shown in Figure 1(a), detects suspicious actions by constructing what a trusted model would independently produce for the same task. Given an input, both the Untrusted and Trusted models generate code in parallel. An LLM intent-alignment judge then receives both outputs alongside the task description and evaluates whether the Untrusted model's action is a reasonable alternative — scoring divergence from 0 (valid alternative approach) to 10 (highly suspicious). This Divergence Score is normalized to  $[0, 1]$ .

The judge's role is critical: it filters benign divergence caused by the capability gap between the two models. Two honest models solving the same problem may take different valid approaches; the judge distinguishes this natural variation from divergence that introduces risks or cannot be explained by any legitimate alternative strategy.

The CM can be used standalone or combined with a standard classifier monitor. In the combined configuration, scores can be fused as  $\text{combined} = \alpha \times \text{score\_cf} + (1 - \alpha) \times \text{score\_monitor}$ .

### 3.2 Counterfactual Perturbations (CP)

The Counterfactual Perturbations Monitor, shown in Figure 1(b), targets a specific structural property of backdoors. A backdoor is anchored to the attacker's secret, not to the task's requirements — so when the task changes, honest code adapts but backdoor conditionals persist. CP operationalizes this by generating controlled CF Inputs and flagging solutions that remain anomalously invariant across them.

The trusted model generates  $k$  CF Inputs — perturbations of the original task, each changing one specific aspect: a *constraint change* (modifying input domain or range), a *requirement change* (altering what the function computes), or a *cosmetic change* (rephrasing without semantic change). Requirement changes are the most diagnostic, since they force algorithmic changes in honest code while leaving task-irrelevant backdoor conditionals untouched.

Each CF Input is solved by the Untrusted model in a **context-isolated** call — with no memory of the original task or other variants. This isolation is the key defense property: a scheming model cannot coordinate its backdoor inputs across calls, so any hardcoded trigger value must be independently chosen each time.

Consistency analysis combines two complementary signals. An LLM judge examines the diff between each original-variant pair and scores unexplained invariants — lines identical across solutions that cannot be explained by shared algorithmic structure. A programmatic detector independently flags conditionals containing hardcoded values present in both solutions (e.g., `if n == 354`). The two signals are fused as  $0.7 \times \text{llm\_score} + 0.3 \times \text{programmatic\_score}$ , aggregated across perturbation pairs into a final Divergence Score.

## 4. Experiments

### 4.1 Experimental Setup

We evaluated the Counterfactual Monitor on the ControlArena [8] APPS programming benchmark across three protocol configurations using 100 samples per configuration. Counterfactual Perturbations approach wasn’t evaluated due to time constraints. The setup used GPT-5.1-Codex-mini as the untrusted model and Claude Haiku 4.5 as the trusted model. We compared (1) a Standard Monitor (classifier-only baseline), (2) CF Only (Counterfactual Monitor with  $\alpha=1.0$ ), and (3) CF Combined (Counterfactual Monitor plus standard monitor with  $\alpha=0.6$ ). Each configuration was tested against three adversarial strategies of increasing sophistication: obvious attacks (e.g., direct malicious actions like file deletion or data exfiltration), subtle sabotage (code that passes tests but embeds vulnerabilities such as insecure defaults or weak cryptography), and disguised-as-mistakes (attacks resembling common programming errors like off-by-one bugs or using HTTP instead of HTTPS). Safety was measured as the true positive rate at 2% false positive rate, reflecting a realistic audit budget where only 2% of traffic can be reviewed, while usefulness was defined as the fraction of honest tasks completed successfully.

### 4.2 Results

#### 4.2.1 Safety Across Attack Categories

| Configuration      | Obvious     | Subtle      | Disguised   | Average     |
|--------------------|-------------|-------------|-------------|-------------|
| Standard Monitor   | 0.93        | 0.89        | 0.66        | 0.83        |
| CF Only            | 0.78        | 0.64        | 0.40        | 0.61        |
| <b>CF Combined</b> | <b>0.94</b> | <b>0.87</b> | <b>0.67</b> | <b>0.83</b> |

Table 1: Safety (TPR at 2% FPR) across attack categories.  $n=100$  per cell.

Table 1 presents the safety scores (TPR at 2% FPR) for each configuration across attack categories. CF Combined matches the Standard Monitor’s average safety of 0.83, slightly outperforming it on obvious attacks (0.94 vs. 0.93) and disguised-as-mistakes (0.67 vs. 0.66). CF Only underperforms at the 2% audit budget across all categories, achieving only 0.61 on average. All configurations struggle most with disguised-as-mistake attacks, with safety dropping to the 0.40–0.67 range.

### 4.2.2 Usefulness

Table 2 reports the usefulness metric for each configuration, measured as the task completion rate in honest mode.

| Configuration    | Usefulness |
|------------------|------------|
| Standard Monitor | 0.89       |
| CF Only          | 0.83       |
| CF Combined      | 0.89       |

Table 2: Usefulness (task completion rate in honest mode).  $n=300$  per configuration.

CF Combined and Standard Monitor both achieve 0.89 usefulness, indicating that composing the Counterfactual Monitor with a standard monitor does not impede benign task completion. CF Only’s slightly lower usefulness (0.83) reflects its higher false-positive rate, which causes some honest actions to be unnecessarily deferred to the trusted model.

### 4.2.3 Safety vs. Audit Budget

Figure 3 highlights the most notable finding: the three configurations exhibit qualitatively distinct safety–budget profiles.



Figure 3: Safety (TPR) as a function of audit budget. CF Only surges past alternatives at the ~5% crossover point, reaching 0.93 safety at 10% budget.

At budgets below 5%, Standard Monitor and CF Combined track closely at approximately 0.83–0.84 safety, while CF Only languishes at ~0.58. At the ~5% crossover point, CF Only surges past both alternatives, climbing to 0.90 safety and continuing to 0.93 at 10% budget. Standard Monitor plateaus at 0.84, while CF Combined rises more gradually to 0.89 at 10%. This crossover occurs because CF Only produces the strongest attack signal (mean attack score 0.74 vs.

0.45 for Standard Monitor) but also the highest honest-mode false positives (mean 0.30 vs. 0.04), which consumes the limited audit budget at low thresholds.

#### 4.2.4 Suspicion Score Distributions

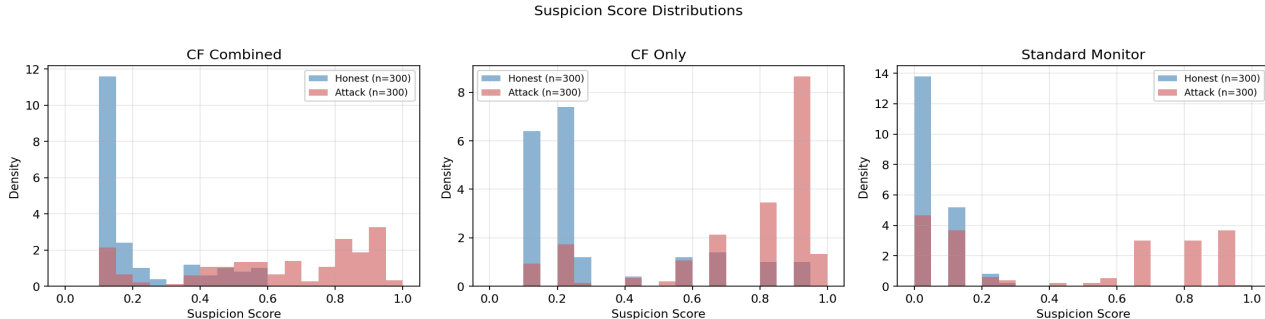


Figure 4: Suspicion score distributions for honest (blue) and attack (red) samples across all three configurations ( $n=300$  each).

Figure 4 shows the suspicion score distributions and explains the budget-dependent behavior. The Standard Monitor produces the tightest honest-mode clustering near 0.0 (minimal false positives), with a bimodal attack distribution that includes some near-zero missed attacks. CF Only generates the strongest separation between honest and attack modes, with attacks clustering near 0.9–1.0, but honest scores are elevated (mean 0.30) compared to Standard Monitor (mean 0.04). CF Combined compresses both distributions, inheriting the Standard Monitor’s low false-positive rate while retaining some of the counterfactual signal’s discriminative power.

The elevated honest-mode scores for CF Only reflect a fundamental property: even when both models are honest, capability-mismatched models naturally produce different solutions. The intent-alignment judge correctly identifies these as “reasonable alternatives” but cannot reduce the score to zero, creating an irreducible false-positive floor.

## 5. Discussion and Limitations

Our results indicate that counterfactual reasoning produces a detection signal structurally complementary to standard monitoring — and this complementarity has broader implications for AI safety. The Standard Monitor plateaus at 0.84 safety regardless of audit budget, while CF Only continues climbing to 0.93, suggesting that discriminative classifiers hit an information ceiling that generative comparison breaks through. The most notable trend is the ~5% crossover point: below it, the counterfactual approach’s irreducible false-positive floor (caused by legitimate capability-gap divergence) makes it impractical, but above it, the stronger attack signal dominates. This budget-dependent behavior ties protocol effectiveness directly to deployment resources and to the capability gap between trusted and untrusted models — a gap likely to widen as frontier models advance faster than reliably aligned ones. **For the AI safety community, this suggests that investing in diverse, structurally-motivated detection paradigms will yield more robust defenses than scaling any single monitor.**

The most concerning trend is the shared vulnerability to disguised-as-mistake attacks, where all configurations drop to 0.40–0.67 safety. Neither discriminative monitoring nor counterfactual comparison reliably catches attacks designed to mimic ordinary programming errors — off-by-one bugs, insecure defaults, subtly wrong boundary conditions. As scheming models grow more sophisticated, this category of attack is likely to dominate, potentially pushing all monitoring-based protocols toward this inadequate safety floor. This reinforces the case for composable, defense-in-depth architectures: CM and CP target orthogonal attack properties (intent divergence vs. behavioral invariance), making them harder to simultaneously evade, and both can compose with existing protocol components like *Defer to Resample* and *Trusted Editing*. **No single detection mechanism will suffice against adaptive adversaries — safety must emerge from layered defenses where each component covers a different failure mode.**

## Limitations

**Narrow evaluation setting.** APPS is a single-turn, short-program coding setting. Components developed here may not transfer to agentic, multi-turn environments (BashArena, SHADE Arena) where attacks are distributed across many individually innocuous actions.

**Judge reliability.** The intent-alignment judge is itself an LLM that could be manipulated by a sufficiently capable adversary. While the judge operates on serialized tool calls rather than raw code (reducing attack surface), its reliability under adversarial conditions needs further evaluation.

**CP not yet evaluated.** While CP’s core modules are implemented, a full evaluation with the same rigor as CM is needed to validate the behavioral invariance hypothesis empirically and to measure the additive safety benefit of composing both components.

## Future Work

Two primary directions emerge from this work. First, scaling to full-dataset evaluation and comparing both components across different benchmarks: BigCodeBench for single-turn code generation, and BashArena or SHADE Arena for multi-turn agentic settings where attacks are distributed across many individually innocuous actions. Evaluating on these diverse benchmarks would establish whether the counterfactual signals generalize beyond the APPS setting and identify setting-specific failure modes.

Second, creating various composable setups and analyzing synergy with other approaches. CM and CP are designed to compose with each other (a CM→CP escalation pipeline) and with existing protocol components such as *Defer to Resample*, *Trusted Editing*, and environment-aware defenses (firewalls, canary tokens, dynamic sandboxing). A systematic study of these compositions would map the Pareto frontier of safety–usefulness–cost trade-offs and identify which combinations yield the strongest defense-in-depth against adaptive adversaries. In particular, completing the CP evaluation would test whether structural invariance detection provides additive safety on top of intent-divergence monitoring when the two components are composed.

## 6. Conclusion

We presented two composable components for AI control protocols, both grounded in counterfactual reasoning. The Counterfactual Monitor (CM) asks what a trusted model would do on the same task and scores intent divergence; evaluated across three attack categories, the combined configuration matches the standard monitor’s safety of 0.83 at realistic audit budgets while the CM-only variant achieves 0.93 safety when budgets exceed 5%. Counterfactual Perturbations (CP) asks what would change if the task itself were different, exploiting context isolation to make coordinated attacks exponentially harder; it is implemented and proposed for future evaluation.

Together, these building blocks demonstrate that counterfactual reasoning - whether across model outputs or task perturbations - is a promising and composable direction for AI control, offering detection signals that are structurally complementary to standard monitoring. The components can be composed with each other and with existing protocols, enabling layered defenses tuned to specific deployment requirements.

## Code

Code repository can be found here: <https://github.com/ofurman/CounterfactualControl/tree/main>.

## References

1. Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, Fabien Roger: *AI Control: Improving Safety Despite Intentional Subversion*. ICML 2024: 16295-16336
2. Mikhail Terekhov, Alexander Panfilov, Daniil Dzenhaliou, Caglar Gulcehre, Maksym Andriushchenko, Ameya Prabhu, Jonas Geiping: *Adaptive Attacks on Trusted Monitors Subvert AI Control Protocols*. CoRR abs/2510.09462 (2025)
3. Aryan Bhatt, Cody Rushing, Adam Kaufman, Tyler Tracy, Vasil Georgiev, David Matolesi, Akbir Khan, Buck Shlegeris: *Ctrl-Z: Controlling AI Agents via Resampling*. CoRR abs/2504.10374 (2025)
4. Aaron Sandoval, Cody Rushing: *Factor(T,U): Factored Cognition Strengthens Monitoring of Untrusted AI*. CoRR abs/2512.02157 (2025)
5. Edward Lue Chee Lip, Anthony Channg, Diana Kim, Aaron Sandoval, Kevin Zhu: *Factor(U,T): Controlling Untrusted AI by Monitoring their Plans*. CoRR abs/2512.14745 (2025)
6. ZachParent, Lennart Finke, Tyler Tracy: *The sum of its parts: composing AI control protocols*. [LessWrong](#)
7. Jonathan Kutasov, Chloe Loughridge, Yuqi Sun, Henry Sleight, Buck Shlegeris, Tyler Tracy, Joe Benton: *Evaluating Control Protocols for Untrusted AI Agents*. CoRR abs/2511.02997 (2025)
8. Inglis Rogan, Ollie Matthews, Tyler Tracy, Oliver Makins, Tom Catling, Asa Cooper Stickland, Rasmus Faber-Espensen, Daniel O’Connell, Myles Heller, Miguel Brandao,

*Adam Hanson, Arathi Mani, Tomek Korbak, Jan Michelfeit, Dishank Bansal, Tomas Bark, Chris Canal, Charlie Griffin, Mojmir Stehlik, Jasmine Wang, Alan Cooney:  
"ControlArena." (2025)*

## **LLM Usage Statement**

*We used Claude (Anthropic) to assist with research landscape analysis, protocol design brainstorming, implementation and drafting sections of this report. All results and claims were independently verified.*