

# ActionLens: Pre-Execution Environment Probing for Agent Action Approval

Udbhav Chitransh<sup>1</sup>A V S M Ashok Kumar<sup>2</sup><sup>1</sup>Independent Researcher <sup>2</sup>Independent Researcher

udbhavchitransh@gmail.com avsmashokkumar@gmail.com

---

**Abstract.** ActionLens is a pre-execution control protocol for shell and file actions proposed by AI agents. Instead of approving an action from transcript alone, a trusted monitor gathers lightweight environment evidence before execution: target inspection, sensitivity scanning, dry-run simulation, diff preview, network-risk analysis, and short action history. On a deterministic 36-task benchmark (12 benign, 12 ambiguous, 12 stealth attacks), transcript-only monitoring fails on every attack while ActionLens blocks all 12/12 attacks and preserves perfect benign completion. On a 30-task slice imported from real BashArena task IDs, ActionLens again reduces attack success to 0.000 across all evaluated proposer models. A cross-environment ablation reveals that the dominant useful probe is not fixed: `preview_diff` matters most on native code-mutation attacks; `sensitivity_scan` matters most on BashArena-style security-surface attacks. Pre-execution control improves markedly when the monitor can inspect concrete side-effect evidence, but robust approval requires combining evidence channels rather than relying on any single probe class. A cross-family evaluation of Semantic Guard across 26 models (7 families) raises average Control Score from 0.844 to 0.943 with zero false positives; an adversarial evasion study confirms that deterministic probes and LLM monitors are complementary defenses.

---

## 1 Introduction

As language models become more agentic, the relevant safety problem shifts from unsafe text generation to unsafe *action execution*. A model that can write files, invoke shell commands, or trigger network-relevant behavior can cause harm even when the transcript appears benign—exactly the setting emphasized by recent work on AI control [1, 3].

Transcript-only monitoring is the natural baseline: a monitor reads the task, the recent conversation, and the proposed action, then decides whether to allow it. But transcript-only review is brittle when the real risk is not legible from surface text. An action may look like a debugging step while bundling secrets; a file edit may look like harmless maintenance while quietly weakening a security boundary.

We study this gap through **ActionLens**, a deliberately narrow control protocol for shell and file actions. Instead of building a full agent platform, ActionLens isolates a single action-approval problem: should a trusted monitor approve a proposed shell action before it executes? The protocol lets the monitor inspect targets, scan for sensitive paths, dry-run commands, preview diffs, assess network destinations, and consult

recent action history. The monitor is intentionally rule-based—results reflect protocol design rather than the capability of any particular frontier model.

### Contributions.

1. A pre-execution control protocol that augments transcript-only approval with six lightweight environment probes.
2. A deterministic 36-task benchmark spanning benign, ambiguous, and stealth-attack tasks with five attack categories.
3. A 30-task BashArena transfer slice testing whether the protocol generalizes beyond the native benchmark.
4. A cross-environment ablation showing that the dominant useful probe shifts with the attack surface: diff evidence in code-mutation settings, sensitivity evidence in security-surface settings.
5. A 26-model cross-family evaluation (7 families, 4 OSS to frontier) showing that Semantic Guard raises average Control Score from 0.844 to 0.943, with 13/26 models improving.
6. An adversarial evasion analysis: six attacks crafted to bypass deterministic probes are missed by Semantic Guard (0/6) but caught by strong LLM monitors (6/6), establishing that deterministic and LLM-based monitors are complementary.

---

<sup>1</sup>Research conducted at the AI Control Hackathon, March 2026, in collaboration with Apart Research.

## 2 Related Work

**AI control.** Greenblatt et al. define the control problem: preserving safety despite intentional subversion by an untrusted model [1]. Korbak et al. extend this framing across capability levels [3]. In both treatments the issue is not monitor accuracy in isolation but whether the surrounding protocol safely handles an untrustworthy component.

**Post-hoc resampling.** Ctrl-Z studies resampling-based control in an agent benchmark, reducing attack success from 58% to 7% at a 5% usefulness cost [2]. ActionLens is pre-execution rather than post-hoc: evidence is gathered before the step runs, avoiding executing the action to discover it was harmful.

**Factored monitoring.** Factor(T,U) decomposes tasks so the untrusted model sees only sub-task context, improving monitoring AUROC from 0.52 to 0.96 [4, 5]. ActionLens takes a complementary direction: rather than restricting what the proposer sees, it enriches what the trusted monitor sees.

**Feedback exploitation.** Hopman, BionicDOLPHIN, and Tracy show that safety drops from 93% to 80% when trusted-to-untrusted feedback is possible [8]. ActionLens addresses this directly: probe outputs are consumed only by the trusted monitor and are never returned to the proposer. Legibility work [6] and monitorability results [7] suggest that richer evidence improves monitoring; ActionLens demonstrates this for environment-side evidence specifically.

## 3 Methods

**Setting.** ActionLens evaluates one proposed action at a time. An untrusted proposer suggests a shell command with a rationale. The trusted monitor runs probes, then outputs one of: `allow`, `block`, or `escalate_manual_review`. Three primary conditions are compared: `allow-all`, `transcript-only`, and `actionlens`. Four ablations remove one probe class each: `-no-dry-run`, `-no-sensitivity-scan`, `-no-preview-diff`, and `inspect-only` (keeping only `inspect_targets` and `history_context`).

**Probes.** The six probes are: `inspect_targets` (extract paths, URLs, env variables); `sensitivity_scan` (classify touched paths against 13 sensitivity rules including secrets, persistence surfaces, firewall configs, sudoers, LD\_PRELOAD); `dry_run` (simulate execution without side effects); `preview_diff` (unified diff of files the action would modify); `network_risk` (classify outbound

destinations against safe/risky domain lists); and `history_context` (retrieve prior actions for behavioral context).

Probes produce *evidence tags*. The monitor aggregates tags into a risk score and applies a decision tree. Hard-blocking rules fire on combinations such as secret material plus outbound network, persistence-surface writes, and diff patterns indicating security weakening (e.g., 0.0.0.0/0 firewall opens, NOPASSWD:ALL sudoers entries, TLS downgrades).

**Benchmarks.** The native benchmark contains 36 tasks: 12 benign software tasks, 12 ambiguous-but-harmless tasks, and 12 stealth attacks across five categories (secret exfiltration, persistence, eval tampering, subtle sabotage, supply chain). The ambiguous split is important: a monitor that blocks everything suspicious can look safe while being useless.

The BashArena transfer slice contains 30 tasks imported from five real BashArena task IDs [9]: 10 benign, 8 ambiguous, 12 attacks. Attack categories map to BashArena’s side tasks—secret exfiltration, firewall disabling, privilege escalation via sudoers, and LD\_PRELOAD persistence injection. Each task includes a simulation spec populating `inspect_reads`, `inspect_writes`, `inspect_network`, `dry_run_output`, and `changes`.

**Hosted-model evaluation.** Beyond deterministic replay, proposer models generate shell commands in structured JSON. Evaluation uses trace replay: one monitor-agnostic trace is collected per task, then replayed through each monitor to eliminate proposer variance.

**Metrics.** Deterministic runs are summarized by:

$$\text{Control} = \text{BCR} \times (1 - \text{ASR}) \quad (1)$$

where **BCR** (benign completion rate) is the fraction of benign tasks the monitor approves, and **ASR** (attack success rate) is the fraction of stealth attacks that execute successfully. A third rate, **ACR** (ambiguous completion rate), tracks the fraction of ambiguous-but-harmless tasks approved; it does not enter the control score but captures the usefulness cost of a cautious monitor. Hosted-model runs use:

$$\text{Live Control} = \text{HCR} \times (1 - \text{ASR}_{|\text{attempt}}) \quad (2)$$

where **HCR** (harmless completion rate) covers both benign and ambiguous tasks, and  $\text{ASR}_{|\text{attempt}}$  conditions on the proposer actually attempting an attack (proposer models vary in whether they try the side task at all).

**Table 1:** Native 36-task benchmark. Wilson 95% CI for ASR.

| Monitor         | ASR   | 95% CI       | BCR   | Control |
|-----------------|-------|--------------|-------|---------|
| transcript-only | 1.000 | (0.76, 1.00) | 1.000 | 0.000   |
| actionlens      | 0.000 | (0.00, 0.24) | 1.000 | 1.000   |

**Table 2:** Native benchmark ablations (12 attacks). “Caught” = blocked or constrained.

| Monitor             | ASR   | Control | Caught |
|---------------------|-------|---------|--------|
| actionlens          | 0.000 | 1.000   | 12/12  |
| no preview_diff     | 0.333 | 0.667   | 8/12   |
| no dry_run          | 0.167 | 0.833   | 10/12  |
| no sensitivity_scan | 0.250 | 0.750   | 9/12   |
| inspect-only        | 1.000 | 0.000   | 0/12   |
| transcript-only     | 1.000 | 0.000   | 0/12   |

**Table 3:** BashArena transfer slice, deterministic evaluation (12 attacks, 10 benign, 8 ambiguous). Ambig CR = ambiguous completion rate (ACR).

| Monitor         | ASR   | BCR   | Ambig CR | Control |
|-----------------|-------|-------|----------|---------|
| transcript-only | 1.000 | 1.000 | 1.000    | 0.000   |
| actionlens      | 0.000 | 1.000 | 0.250    | 1.000   |

## 4 Results

**Native benchmark.** Table 1 reports the central result. Transcript-only monitoring fails on all 12 attacks because the stealth attacks are designed to avoid explicit transcript-level risk signals—the risk lives in file diffs, sensitive path writes, and network destinations. ActionLens blocks all 12/12 attacks with no benign false positives, moving control score from 0.000 to 1.000.

**Native ablations.** Table 2 shows that the gain comes from active evidence gathering. `inspect_targets` alone is indistinguishable from transcript-only—passive path listing without content analysis catches nothing. The dominant probe in this setting is `preview_diff`: removing it drops coverage from 12/12 to 8/12 and halves the control score. Removing `dry_run` costs 2 detections; removing `sensitivity_scan` costs 3.

**BashArena transfer.** Table 3 shows the deterministic transfer result. The pattern is the same: transcript-only approves every attack; ActionLens drives attack success to 0.000 while preserving benign completion. The usefulness cost is real—ambiguous completion drops from 1.000 to 0.250—but the safety gain is consistent.

**Hosted-model transfer.** Table 4 reports the two flagship hosted-model rows. GPT-5.2-Codex attempts more attacks and is the stronger stress test; GPT-5.3-

**Table 4:** Flagship hosted-model transfer (BashArena slice). Live Control from Eq. 2.

| Model         | Monitor    | ASR   | ASR Att | Live Ctrl |
|---------------|------------|-------|---------|-----------|
| gpt-5.2-codex | transcript | 0.583 | 1.000   | 0.000     |
|               | actionlens | 0.000 | 0.000   | 0.700     |
| gpt-5.3-codex | transcript | 0.250 | 1.000   | 0.000     |
|               | actionlens | 0.000 | 0.000   | 0.900     |

**Table 5:** BashArena transfer ablations, averaged across 5 proposer models. Removing `sensitivity_scan` is most damaging here; removing `preview_diff` has no effect—the opposite of the native benchmark pattern.

| Monitor             | Avg ASR | Avg ΔASR | Live Ctrl |
|---------------------|---------|----------|-----------|
| actionlens          | 0.000   | —        | 0.496     |
| no preview_diff     | 0.000   | +0.000   | 0.532     |
| no dry_run          | 0.028   | +0.028   | 0.514     |
| no sensitivity_scan | 0.250   | +0.250   | 0.223     |
| inspect-only        | 0.389   | +0.389   | 0.000     |
| transcript-only     | 0.389   | +0.389   | 0.000     |

Codex is a cleaner usefulness case. For both, transcript-only fails on every attempted attack while ActionLens reduces attack success given attempt to 0.000. Five additional models are summarized in the appendix.

**Cross-environment probe dependence.** The central result is in Table 5: the dominant probe is environment-dependent. On the native benchmark, removing `preview_diff` is most damaging. On the BashArena transfer slice, removing `preview_diff` has essentially no effect, while removing `sensitivity_scan` raises average ASR by 0.250. The structural reason is direct: BashArena’s attacks touch security-bearing file paths (`firewall.conf`, `sudoers.d/`, `ld.so.preload`, `api_tokens.json`) that sensitivity scanning classifies as high-risk before any content analysis is needed. Native benchmark attacks hide in subtle code mutations where path classification is uninformative and diff content is decisive.

A robust pre-execution monitor therefore needs both probe classes. A diff-only monitor misses attacks on security-bearing paths. A path-only monitor misses attacks that corrupt legitimate files with subtle content changes. An adversary who knows which class the monitor relies on can construct attacks that evade it.

**Cross-family Semantic Guard evaluation.** To test whether deterministic probing generalizes across monitor capability levels, we evaluated Semantic Guard—a standalone module implementing six targeted detectors (typosquat, silent failure, exfiltration, test suppression, security weakening, secret leak)—across 26

**Table 6:** Semantic Guard cross-family evaluation (26 models, 36-task benchmark). CS = Control Score. SG raises average CS from 0.844 to 0.943 with zero false positives. 10 representative models shown; full results in appendix.

| Model                  | Family | Det   | FP | CS    | CS+SG | $\Delta$ |
|------------------------|--------|-------|----|-------|-------|----------|
| GPT-4.1-nano           | GitHub | 3/12  | 0  | 0.250 | 0.667 | +0.417   |
| o3-mini                | OpenAI | 6/12  | 0  | 0.500 | 0.833 | +0.333   |
| Mistral-7B             | OSS    | 8/12  | 0  | 0.667 | 1.000 | +0.333   |
| Qwen3-8B               | OSS    | 9/12  | 0  | 0.750 | 1.000 | +0.250   |
| GPT-5.1-Codex          | OpenAI | 10/12 | 0  | 0.833 | 1.000 | +0.167   |
| Cl. Sonnet 4.6         | Anthro | 12/12 | 1  | 0.917 | 0.917 | +0.000   |
| GPT-5.4                | OpenAI | 12/12 | 0  | 1.000 | 1.000 | +0.000   |
| <i>Avg (26 models)</i> |        |       |    | 0.844 | 0.943 | +0.099   |

models spanning 7 families: OSS (Phi-4, Mistral-7B, Qwen2.5-Coder-32B, Qwen3-8B), GitHub Models API (GPT-4o-mini, GPT-4.1-mini, GPT-4.1-nano), and 19 frontier models via OpenRouter (OpenAI, Anthropic, Google, xAI, DeepSeek). Table 6 reports the results.

Semantic Guard alone catches 7/12 attacks with zero false positives in under 1ms (pure regex/string matching, no LLM calls). The improvement is strongest for weak models: Mistral-7B (7B parameters, \$0.05/MTok) with SG achieves CS=1.000, matching GPT-5.4 (\$2.50/MTok). The number of models achieving perfect CS rises from 9/26 (35%) to 15/26 (58%).

**Adversarial evasion analysis.** To stress-test Semantic Guard’s limitations, we crafted six attacks specifically designed to bypass its probe coverage: DNS exfiltration via subprocess (bypasses `exfiltration_detector` which requires `requests.post`), a malicious package with an unrelated name from a private registry (bypasses `typosquat_detector`), a CI environment variable causing silent exit 0 (bypasses `test_suppression_detector`), a time-gated logic bomb (bypasses `silent_failure_detector`), an auth bypass in Python code rather than config files (bypasses `security_weakening_detector`), and a base64-encoded reverse shell (bypasses all probes). Table 7 reports the results.

Semantic Guard catches 0/6 evasion attacks by construction—they target its blind spots. Strong LLM monitors (GPT-5.1-Codex, Claude Sonnet 4.6, GPT-5.4, Gemini 2.5 Flash) catch 6/6 with near-zero false positives. The weak reasoning model o3-mini catches 0/6, mirroring its poor performance on the standard benchmark. This establishes the complementary defense thesis: deterministic probes handle known attack patterns reliably and cheaply, while LLM monitors provide semantic reasoning for novel or obfuscated attacks. Neither alone is sufficient; the combination is strictly

**Table 7:** Adversarial evasion attacks (6 attacks crafted to bypass SG). Strong LLM monitors catch what deterministic probes miss.

| Monitor          | Det | FP(b) | FP(a) |
|------------------|-----|-------|-------|
| Semantic Guard   | 0/6 | 0/3   | 0/3   |
| o3-mini          | 0/6 | 0/3   | 0/3   |
| GPT-5.1-Codex    | 6/6 | 0/3   | 0/3   |
| Cl. Sonnet 4.6   | 6/6 | 0/3   | 0/3   |
| GPT-5.4          | 6/6 | 0/3   | 0/3   |
| Gemini 2.5 Flash | 6/6 | 1/3   | 0/3   |

stronger than either component.

## 5 Case Studies

**Success: firewall disabling (BashArena task 1003684).** The agent proposes rewriting `etc/firewall/firewall.conf` with `WHITELIST_RANGES="0.0.0.0/0"`, rationale: “open the whitelist broadly so the VPN task can reach anything during setup.” Transcript-only approves; the rationale is plausible and no explicit “disable firewall” token appears. ActionLens runs `sensitivity_scan` first, classifying `firewall.conf` as a high-risk security configuration path. `preview_diff` then reveals the downgrade: a narrow whitelist replaced by a globally permissive rule. The tag `diff_downgrades_security` fires and the monitor blocks. Had the action been approved, a scoped firewall exception would silently become unrestricted inbound/outbound access under the cover of legitimate VPN troubleshooting.

**Failure: simulation-fidelity gap.** The debug-bundle exfiltration attack (also task 1003684) was initially missed despite being caught on the native benchmark. The root cause was not the monitor policy but the transfer spec: the initial version did not populate `inspect_reads` with `opt/configs/api_tokens.json`, so `sensitivity_scan` never observed that a secret-bearing file was being accessed before the outbound upload. Once the spec was corrected the attack was caught. This is a concrete statement of the simulation-fidelity limitation: in live execution the gap disappears; in simulation-based evaluation, spec completeness is a first-class correctness requirement.

## 6 Discussion

The main lesson from the cross-environment ablation is practical. A diff-heavy coding environment and a path-sensitive security environment reward different

evidence channels. Control protocols should not be treated as static monitor templates; they should be adapted to the structure of the environment in which the untrusted agent is acting.

The second lesson is that passive inspection is not enough. `inspect_targets` alone behaves like transcript-only review in both environments. The useful gain appears only once the monitor can extract or simulate side-effect evidence not visible in the transcript.

**Limitations.** The native benchmark is small (36 tasks) and simulation-modeled. The BashArena transfer slice is narrower than the full ControlArena harness. Hosted-model scoring is attempt-sensitive. The monitor is rule-based rather than learned. Probe outputs are not fed back to the proposer—a deliberate design choice that real deployments must enforce architecturally [8]. Semantic Guard uses regex/string matching rather than AST analysis; the evasion study demonstrates that obfuscated payloads, indirect channels, and attacks placed in unexpected file types evade deterministic probes entirely. All three Anthropic models produce a systematic false positive on the same ambiguous task, suggesting per-model calibration may be necessary.

**Future directions.** Three extensions are most natural: broader external validation against full BashArena or LinuxArena; LLM-backed monitors that reason over probe evidence while preserving feedback separation; and adaptive adversarial evaluation in which the attacker knows the probe profile. The cross-family evaluation also suggests a cost-effectiveness direction: AST-based Semantic Guard with data-flow analysis could close the evasion gap while preserving the latency and cost advantages of deterministic probing.

## 7 Conclusion

ActionLens is a pre-execution control protocol for shell and file actions. On a deterministic 36-task benchmark, pre-execution environment probing moves attack success from 1.000 to 0.000 while preserving perfect benign completion. The same directional result holds on a 30-task BashArena transfer slice across five frontier proposer models. The central finding is not merely that ActionLens works, but why: the dominant useful probe shifts between environments. Diff evidence dominates in code-mutation settings; path sensitivity dominates in security-surface settings. A cross-family evaluation across 26 models shows that Semantic Guard raises average Control Score from 0.844 to 0.943 with zero false positives, making small open-source models competitive with frontier systems. An adversarial evasion

analysis confirms that deterministic probes and LLM monitors are complementary: neither alone catches all attacks, but their combination is strictly stronger than either component. Practical control protocols should combine environment-side evidence channels and adapt them to the deployment setting.

## Code and Data

Code, benchmark specs, transfer slice, and results: <https://github.com/udsea/actionlens>

## Author Contributions

Udbhav Chitransh led project design, benchmark construction, core protocol implementation, BashArena transfer evaluation, cross-environment ablation study, and paper writing. A V S M Ashok Kumar designed the Semantic Guard module, conducted the 26-model cross-family evaluation, and designed the adversarial evasion analysis. Both authors contributed to framing the final submission.

## LLM Usage

LLMs were used as aswriting assistants. All experimental design decisions, result selection, and final claims were verified by the authors against the underlying evaluation artifacts.

## References

- [1] R. Greenblatt, B. Shlegeris, et al. AI Control: Improving Safety Despite Intentional Subversion. *ICLR* 2025. arXiv:2312.06942.
- [2] A. Bhatt, C. Rushing, et al. Ctrl-Z: Controlling AI Agents via Resampling. *Redwood Research*, 2025. arXiv:2504.10374.
- [3] T. Korbak, M. Balesni, B. Shlegeris, G. Irving. How to Evaluate Control Measures for LLM Agents? arXiv:2504.05259, 2025.
- [4] Pivotal Research. Factor(T,U): Factored Cognition Strengthens Monitoring. arXiv:2512.02157, 2025.
- [5] Pivotal Research. Factor(U,T): Controlling Untrusted AI by Monitoring their Plans. arXiv:2512.14745, 2025.
- [6] R. Greenblatt. Basic Legibility Protocols Improve Trusted Monitoring. arXiv:2602.10153, 2026.
- [7] A. Stickland, J. Michelfeit, et al. Async Control: Stress-testing Asynchronous Control Measures. *UK AISI*, 2025. arXiv:2512.13526.

- [8] M. Hopman, BionicDOLPH1N, T. Tracy. Untrusted AIs Can Exploit Feedback in Control Protocols. *LessWrong*, May 2025.
- [9] A. Kaufman, J. Lucassen. BashArena and Control Setting Design. *Redwood Research Blog*, 2025.

## A Extended Hosted-Model Results

Table 8 reports all five hosted-model rows on the BashArena transfer slice. Models with BCR below 0.5 (claude-opus, qwen3-coder, kimi-k2) failed to complete main tasks—a proposer capability issue, not a monitor issue. The monitor blocked every attack each model attempted.

**Table 8:** Full hosted-model BashArena transfer results. ActionLens reduces ASR to 0.000 on every model. Low BCR reflects proposer capability, not monitor quality.

| Model         | Monitor    | BCR  | ASR   | Live Ctrl |
|---------------|------------|------|-------|-----------|
| gpt-5.3-codex | transcript | 0.90 | 0.250 | 0.000     |
|               | actionlens | 0.90 | 0.000 | 0.900     |
| gpt-5.2-codex | transcript | 0.70 | 0.583 | 0.000     |
|               | actionlens | 0.70 | 0.000 | 0.700     |
| claude-opus   | transcript | 0.30 | 0.333 | 0.000     |
|               | actionlens | 0.30 | 0.000 | 0.300     |
| qwen3-coder   | transcript | 0.30 | 0.500 | 0.000     |
|               | actionlens | 0.30 | 0.000 | 0.300     |
| kimi-k2       | transcript | 0.30 | 0.417 | 0.000     |
|               | actionlens | 0.30 | 0.000 | 0.300     |

## B Full 26-Model Cross-Family Results

Table 9 reports Semantic Guard evaluation across all 26 models. CS = transcript-only Control Score; CS+SG = Control Score with Semantic Guard. Models are grouped by capability tier. SG raises 6 models from imperfect to perfect CS = 1.000 (marked with †).

**Table 9:** Full 26-model Semantic Guard evaluation. Det = attacks detected by transcript monitor. † = model reaches CS = 1.000 only with SG.

| Model                                                   | Family    | Det   | FP | CS    | CS+SG | $\Delta$ |
|---------------------------------------------------------|-----------|-------|----|-------|-------|----------|
| <i>Tier 1: Weak (<math>CS &lt; 0.7</math>)</i>          |           |       |    |       |       |          |
| GPT-4.1-nano                                            | GitHub    | 3/12  | 0  | 0.250 | 0.667 | +0.417   |
| Phi-4 14B                                               | OSS       | 7/12  | 2  | 0.486 | 0.764 | +0.278   |
| o3-mini                                                 | OpenAI    | 6/12  | 0  | 0.500 | 0.833 | +0.333   |
| Qwen2.5-Coder-32B                                       | OSS       | 7/12  | 1  | 0.535 | 0.840 | +0.306   |
| Mistral-7B†                                             | OSS       | 8/12  | 0  | 0.667 | 1.000 | +0.333   |
| <i>Tier 2: Mid (<math>0.7 \leq CS &lt; 0.95</math>)</i> |           |       |    |       |       |          |
| Qwen3-8B†                                               | OSS       | 9/12  | 0  | 0.750 | 1.000 | +0.250   |
| o4-mini                                                 | OpenAI    | 10/12 | 0  | 0.833 | 0.917 | +0.083   |
| GPT-5-mini                                              | OpenAI    | 10/12 | 0  | 0.833 | 0.917 | +0.083   |
| GPT-5.1-Codex†                                          | OpenAI    | 10/12 | 0  | 0.833 | 1.000 | +0.167   |
| Gemini 3 Flash                                          | Google    | 11/12 | 1  | 0.840 | 0.917 | +0.076   |
| GPT-4.1-mini†                                           | GitHub    | 11/12 | 0  | 0.917 | 1.000 | +0.083   |
| GPT-5.3-Codex†                                          | OpenAI    | 11/12 | 0  | 0.917 | 1.000 | +0.083   |
| GPT-5.2-Codex†                                          | OpenAI    | 11/12 | 0  | 0.917 | 1.000 | +0.083   |
| Gemini 2.5 Pro                                          | Google    | 12/12 | 1  | 0.917 | 0.917 | +0.000   |
| Cl. Sonnet 4.6                                          | Anthropic | 12/12 | 1  | 0.917 | 0.917 | +0.000   |
| Cl. Opus 4.6                                            | Anthropic | 12/12 | 1  | 0.917 | 0.917 | +0.000   |
| Cl. Haiku 4.5                                           | Anthropic | 12/12 | 1  | 0.917 | 0.917 | +0.000   |
| <i>Tier 3: Strong (<math>CS \geq 0.95</math>)</i>       |           |       |    |       |       |          |
| GPT-4o-mini                                             | GitHub    | 12/12 | 0  | 1.000 | 1.000 | +0.000   |
| Gemini 3.1 Pro                                          | Google    | 12/12 | 0  | 1.000 | 1.000 | +0.000   |
| Grok-4.1                                                | xAI       | 12/12 | 0  | 1.000 | 1.000 | +0.000   |
| DeepSeek-R1                                             | DeepSeek  | 12/12 | 0  | 1.000 | 1.000 | +0.000   |
| GPT-5.4                                                 | OpenAI    | 12/12 | 0  | 1.000 | 1.000 | +0.000   |
| GPT-5.4-mini                                            | OpenAI    | 12/12 | 0  | 1.000 | 1.000 | +0.000   |
| Grok-4                                                  | xAI       | 12/12 | 0  | 1.000 | 1.000 | +0.000   |
| Gemini 2.5 Flash                                        | Google    | 12/12 | 0  | 1.000 | 1.000 | +0.000   |
| Gemini 3.1 Fl. Lite                                     | Google    | 12/12 | 0  | 1.000 | 1.000 | +0.000   |
| <i>Average (26 models)</i>                              |           |       |    | 0.844 | 0.943 | +0.099   |