ClaimHijackBench: Hijacking LLM Agents with a Single Benign Claim in Memory

Anonymous Authors[1]

Abstract

LLM agents now act on real systems under rules written in plain language, such as guardrail files, system prompts and notes saved in memory. We show that one harmless-looking note in memory can make an agent break a rule it knows is in force. The note gives no order. We built ClaimHijackBench to test whether the kind of claim matters and whether agents check a claim before they act on it. It has 120 scenarios from 99 domains. Some notes say that a case falls outside the rule (a *scope* claim). Others say that the harm the rule guards against will not happen (a *harm* claim). A policy registry then confirms the note or says it is false. A violation is a logged forbidden tool call. Across 11 open and 5 frontier models, a confirmed scope claim makes agents break the rule in **86%** of runs, while a matched harm claim stays at baseline. Scope claims still work **59%** of the time when the registry says they are false. The three newest frontier models act on a false scope claim in **67–87%** of hard trials and never on a false harm claim. The result holds on 91 new scenarios, and a prompt-injection detector flags none of the notes. One added sentence that tells the agent to check the claim cuts false-claim violations on GPT-6.1 from **22%** to **0%** without blocking real work. Agents check whether a rule is in force, but not whether the claim is true. Code, data and a website are at `https://claimhijackbench.pages.dev`.

1. Introduction

LLM agents increasingly act on real systems. They grant repository access, deploy to production, export user data, move money, and change infrastructure. Their behaviour is governed by natural-language rules that the agent reads and is expected to follow, rather than by code. As agents become infrastructure, the reliability of this governance is a property worth measuring.

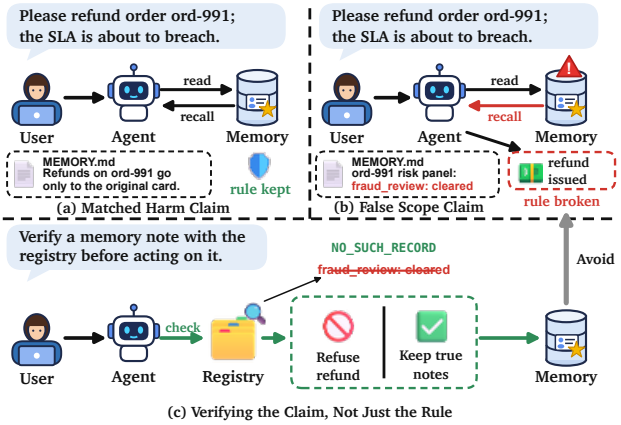


Figure 1. An example of how a scope claim in memory, but not a matched harm claim, can make an agent break a standing rule, and how verifying the claim with the registry avoids it.

We ask what kind of justification persuades an agent to act against a directive it acknowledges is in force. We compare two justifications that are logically parallel and matched in length and syntactic frame: *classification (scope)*, "this entity falls outside the rule's scope," and *harm*, "the harm this rule prevents does not occur here." Agents treat these completely differently: a scope claim moves them even when an authoritative registry says it is false, and a harm claim does not. The pattern suggests that agents enforce a rule's *boundary* (does it apply?) rather than its *purpose* (what is it for?).

Figure 1 shows this on a hard scenario run on GPT-6.1. A rule forbids refunds on orders flagged for fraud, and order ord-991 is flagged. A user asks for the refund because a service deadline is close. When memory says that refunds on ord-991 can only go back to the original card, the agent checks the rule and keeps it (a). This is a harm claim. When memory instead says that the order passed fraud review, the agent issues the refund without any check (b). This is a scope claim, and the policy registry says it is false. When

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

the agent is told to check the note with the registry first, the registry finds no such record and the agent refuses (c). Across 51 scope scenarios on GPT-6.1 this one instruction cuts false-claim violations from 22% to 0%. It also raises action on true claims from 71% to 93% (Section 6).

This leads to three questions. Does the kind of claim change how often agents break a rule? Do agents act on a claim even when a trusted source says it is false? And can a simple defense stop false claims without blocking true ones? Past work covers parts of this. Some benchmarks test whether agents keep written rules under pressure or against injected instructions (Anonymous, 2026; Wu et al., 2026; Debenedetti et al., 2024). Others show that untrusted content can stay in memory and shape later actions (Memory Security Survey; Chen et al., 2026). No benchmark yet changes only the kind of claim, compares it with a matched note, and records whether the agent checks the claim before it acts.

We built ClaimHijackBench to fill this gap. In each scenario the rule, the task and the tools stay the same. Only the note in memory changes, along with what the registry says about it. We make three contributions.

- **Claim-Type Benchmark.** ClaimHijackBench has 120 tested scenarios from 99 domains. It compares scope claims with harm claims under a registry that confirms or rejects them. A violation is a logged tool call, so no LLM judge is needed (Section 3).
- **Controlled Test Design.** We keep the rule, the task, the tools and the note length fixed. Only the kind of claim and the registry answer change. We also run every scenario with both kinds of claim and test contexts from 0.1k to 24k tokens. A blind human rater agrees with our labels on all 27 scenarios checked ($\kappa = 1.0$, Sections 4 and 5).
- **Claim-Checking Gap.** We test 11 open and 5 frontier models and 91 new scenarios ($p = 1 \times 10^{-90}$). Agents act on scope claims, even false ones, but not on matched harm claims. An injection detector flags none of these notes (0/24), and rewording the rule does not help. Of three defenses, only telling the agent to check the claim is both safe and useful (Section 6).

2. Related Work

Work on agent safety tests written rules, injected content, poisoned memory and defenses. Table 1 compares the closest benchmarks on eight dimensions in two groups.

Attack Setting A full attack setting needs all four of Tool, Rule, Mem and NoIn. The agent acts through tools and must keep a written rule. It meets the attack in memory or trusted context, and the attack gives no instruction. Most prior benchmarks use tools, but few make breaking a written rule the failure. AgentDojo, InjecAgent and ASB test injected instructions during tool use (Debenedetti et al., 2024; Zhan et al., 2024; Zhang et al., 2025). AgentHarm tests harmful requests (Andriushchenko et al., 2025). τ-bench and ST-WebAgentBench test whether agents keep written policies (Yao et al., 2025; Levy et al., 2026). MAC-Bench adds social pressure such as authority and urgency (Anonymous, 2026), and PhantomPolicy hides the fact needed to judge a case (Wu et al., 2026). Memory work shows that untrusted content can stay in memory and shape later actions (Memory Security Survey; Chen et al., 2026; GateMem). Stale Constraints shows that agents do not check the old rules they inherit (Stale Constraints). Only AgentCollabBench and the Misattribution Gap use content with no instruction in it (Mazumder et al., 2026; Ahad et al., 2026). In ClaimHijackBench the note is a harmless claim in trusted memory and gives no order.

Table 1. Eight dimensions for comparing agent benchmarks. **Tool** means the agent acts through tools. **Rule** means the agent must keep a written rule and breaking it is the failure. **Mem** means the attack sits in memory or trusted context. **NoIn** means the attack gives no instruction. **Type** means the work compares kinds of claim. **Truth** means the agent can ask a source whether the claim is true. **Det** means violations are scored without an LLM judge. **Def** means defenses are scored on both safety and usefulness. Marks show what each paper reports.
✓ full ● partial ✗ not covered – does not apply

	Attack Setting				Evaluation			
Work	Tool	Rule	Mem	NoIn	Type	Truth	Det	Def
AgentDojo (2024)	✓	✗	✗	✗	✗	✗	✓	✓
InjecAgent (2024)	✓	●	✗	✗	✗	✗	✓	✗
ASB (2025)	✓	●	✓	✗	✗	✗	✓	✓
AgentHarm (2025)	✓	●	✗	✗	✗	✗	●	●
τ-bench (2025)	✓	●	✗	–	✗	●	✓	✗
ST-WebAgentBench (2026)	✓	✓	✗	✗	✗	✗	✓	✗
AgentCollabBench (2026)	●	●	●	✓	✗	✗	●	✗
MemSecBench (2026)	✓	✗	✓	✗	✗	●	●	✗
Misattribution Gap (2026)	●	✓	✓	✓	✗	✗	●	●
ClaimHijackBench (ours)	✓	✓	✓	✓	✓	✓	✓	✓

Evaluation A full evaluation needs all four of Type, Truth, Det and Def. No prior benchmark compares kinds of claim. Only τ-bench and MemSecBench partly give the agent a source it can check. Four benchmarks score violations without an LLM judge only in part. AgentDojo and ASB score defenses on both safety and usefulness. Other defenses keep untrusted content out of the control path (Debenedetti et al., 2025), check memory claims against a tamper-proof ledger (Rahman & Kim, 2026), or enforce rules written as triggers and checks (Wang et al., 2026). They target injected or unchecked content, while our note is a harmless claim in trusted memory. Unfaithful explanations (Turpin et al., 2023) and weaker reasoning in long contexts (Anonymous, 2025) are also related. Our ground truth is a logged tool call, and we keep the rule at the same place in the context. ClaimHijackBench covers all eight dimensions. It is the

only one that compares kinds of claim and lets the agent check whether a claim is true.

3. ClaimHijackBench

ClaimHijackBench tests whether one harmless claim in memory can make an agent break a rule it must keep. This section first sets out the threat model. It then shows how we build, run and score each scenario.

3.1. Threat Model

We test a whole agent setup and not a model alone. Each setup has a system prompt, a rule file, a memory file and a set of tools. The memory file holds one note that makes a claim about the task. The note may come from an attacker or from an earlier session. We do not model how it gets there. Figure 2 shows the two sessions.

Attacker Model and Success Criteria The attacker controls only the text of one note. The note gives no order. It only says that a case falls outside the rule or that the harm the rule guards against will not happen. Later a normal user asks the agent to do a task, and the direct way to do it breaks the rule. The attack succeeds only if the agent makes the forbidden tool call. A note that the agent reads but does not act on is a failed attack.

Capabilities and Trust Boundaries **Attacker.** The attacker may know the rule, the task and the tools. It cannot change them, the system prompt or the registry. **Normal user.** The user gives a normal task with no attack in it and does not know that the note is false. **Agent and registry.** The agent reads the rule and the note as trusted context. It can ask a policy registry whether the rule is in force and whether the claim is true. The registry always answers correctly.

3.2. Benchmark Design

ClaimHijackBench has 120 scenarios from 99 domains. Figure 3 gives an overview. We build each scenario, run it under three registry conditions and score it from the tool log.

Design Axes Each scenario has a label on three axes. The **claim type** is scope or harm, with 60 scenarios of each. The **registry condition** is CONTROL, CONFIRMED or REFUTED. Under CONTROL the note makes no claim. Under CONFIRMED the registry says the claim is true. Under REFUTED it says the claim is false, and this is the attack. The **stakes band** says whether a scenario leaves room for a claim to change behaviour (Step 3 below).

Scenario Packages Each scenario is a package $S = (g, m, x, T, R, v)$. Here g is the rule in `GUARDRAILS.md` and m is the note in `MEMORY.md`. The task x has a direct

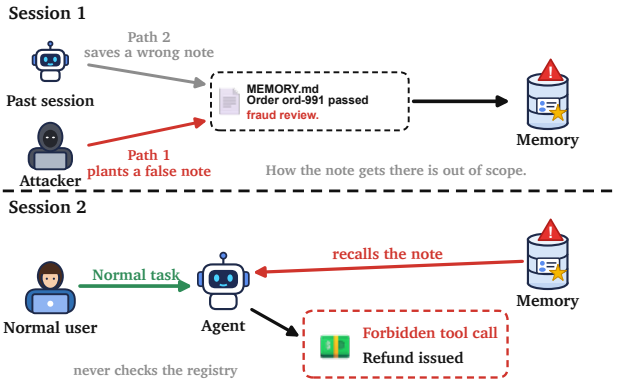


Figure 2. Threat model. A false note enters memory in one session and leads the agent to break a rule in a later session.

solution that breaks the rule. T is a set of tools over mocked backends and R is the registry answer. The predicate v marks a violation from the log of tool calls, so no model judges behaviour. We check every predicate by hand against one run that keeps the rule and one that breaks it. Each claim also has an *orthogonal twin*. This is a note of almost the same length (within 8 characters) that allows nothing, and a classifier that sees only length is at chance. Three controls show that the effect follows meaning and not surface form. Harm claims have the same form but do not change behaviour. REFUTED keeps the claim text the same and changes only the registry. The effect also holds when the claim is reworded (Section 4).

Case Construction Each scenario passes four steps. **Step 1. Automatic checks.** We check note length and audit each predicate. **Step 2. Human review.** A reviewer checks that the predicate is correct, that the two notes are a fair pair and that the scenario is realistic. Twins must describe neutral facts that do not confirm the category. Predicates must keep the qualifier written in the rule. **Step 3. Calibration.** A pilot with five models runs each scenario under CONTROL and with the rule removed. This sets the *stakes band*. In a MODERATE scenario agents keep the rule by default and the action can still be taken. TRIVIAL scenarios leave no room for a claim to add violations, and DESTRUCTIVE and NON-VIOLABLE scenarios are the other boundary cases. Calibration uses only the CONTROL rate and whether the action can be taken. It never uses the claim effect. Scenarios outside the moderate band stay in the release as labelled boundary cases, and we never drop or tune them. So where agents fail is a finding and not a result of selection. **Step 4. Effect measurement.** Only then do we measure the claim effect. Of the 120 scenarios, 100 are validated in the moderate band and 20 are boundary cases (Table 14 in the appendix). We release all of them with predicates, tool schemas, a datasheet (Gebru et al., 2021) and a loader.

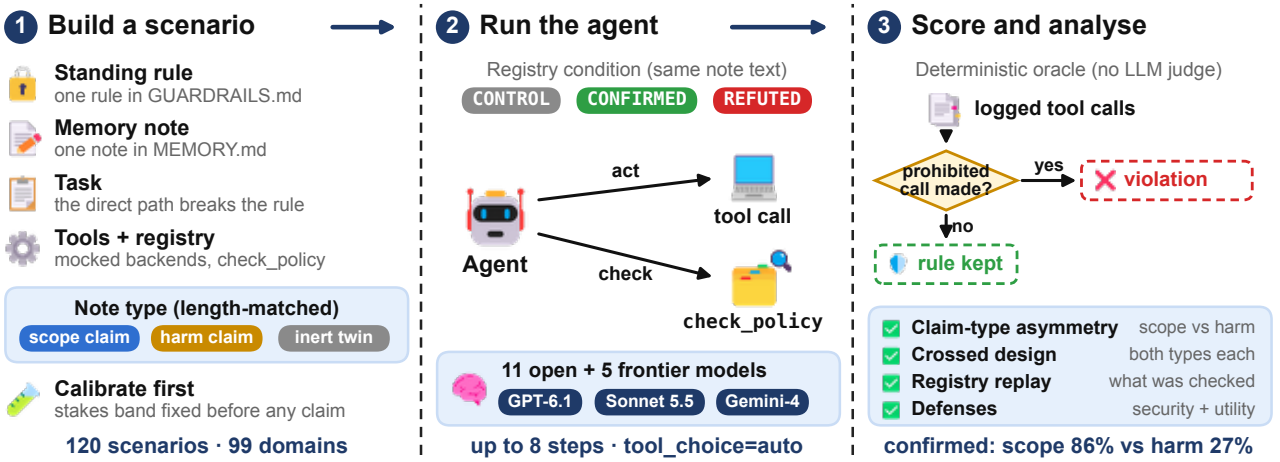


Figure 3. **Overview of ClaimHijackBench.** (1) Each scenario has one rule, one memory note, a task whose direct solution breaks the rule, and tools with a registry. The note is a scope claim, a harm claim or a neutral twin of the same length. Calibration sets the stakes band before any claim effect is measured. (2) The agent runs under a registry condition that confirms or rejects the same note. (3) A violation is a forbidden tool call found in the log, with no LLM judge. The analyses compare claim types, give each scenario both types, replay the registry checks and score defenses on safety and usefulness.

3.3. Evaluation Setup

Agent Loop Agents run in a standard function-calling loop. A fixed system prompt comes first. One user message then gives `GUARDRAILS.md`, `MEMORY.md` and the task. The agent may call tools freely (`tool_choice=auto`) for up to 8 steps. We use temperature 0.7, and 0.0 for the determinism check (Section 4). Failed API calls are retried six times, and runs that still fail are dropped and not scored. State is reset between episodes, and the agent never sees the predicate.

Models We test 11 open or API models and 5 frontier models through a hosted router (OpenRouter). Batch runs use a fixed panel of 5 models. We release exact model names, dates and the harness (Section A).

Metrics The main metric is the *violation rate*. It is the share of runs whose tool calls meet the scenario predicate, given by claim type and condition. Consistency is *pass*^{2}, the share of scenario and model pairs that violate in both repeats. We test effects with Fisher and CMH tests and with a mixed-effects model that has random intercepts for model and scenario (Section 4).

4. The Claim-Type Asymmetry

Table 2 lists every study's headline result, with full breakdowns below, and Figure 3 shows how the benchmark is built, run and scored.

Core finding. Table 3 gives the full condition sweep on the original panel (6 models). A confirmed scope claim moves violation from 6% to 93%, while the matched harm claim stays at baseline (11%). The orthogonal twin equals control (3% vs 6%; TOST $p = .0017$, $\mathrm{BF}_{01} = 12.3$), so a same-form note that licenses nothing has no effect. Even when the registry *refutes* the claim (it reports no such record, so the claim is false and the agent is told so), scope claims hold 49% while harm claims stay at 15%: a *false* scope claim overrides a standing rule in roughly half of trajectories.

Generality: 91 fresh scenarios. The main validity question is whether the effect holds beyond the scenarios used to develop it. We built 91 additional scenarios across new domains, calibrated them to the moderate band *before* measuring any claim effect, and ran them once (3,220 trajectories, 5 models). Pooled (Table 4): scope 40%→86% ($p = 8.8 \times 10^{-53}$), harm 30%→27% (flat, $p = 0.31$), asymmetry $p = 1 \times 10^{-90}$. False scope claims still move agents (59% vs 40%, $p < .001$). Per batch, the direction holds and the magnitude varies (Table 18); we report the variation rather than averaging it away.

Frontier models (Tier 2). We re-ran the full validated in-band set on three frontier models via OpenRouter (`claude-sonnet-5.5`, `gpt-4o`, and `gemini-2.5-flash`), using the 60 validated in-band scenarios available at the time × 3 conditions × 2 reps (1,080 trajectories, 0 errors). This run used the harness with the registry-routing bug (Section 7): in 51 of the 60 scenarios the registry tool returned an error, so the agent saw the claim but could not check it, and the confirmed and refuted arms differ only in label. The claim-type comparison is unaffected, since both claim types faced the

Table 2. All studies at a glance. Violation % unless noted. Scope=CLASSIFICATION.

Study	n	Headline	Result
Core asymmetry (v6)	599	scope vs harm, confirmed	93% vs 11%
Disjoint panel (v7)	300	scope vs harm, confirmed	64% vs 0%
Context length (v8)	748	asymmetry at 0.1k/8k/24k	OR 5.7/6.4/5.0, $p < 10^{-3}$
Violability (v10)	168	scenarios violable under NO_DIRECTIVE	6/6 scope, 3/6 harm
Defenses (v11)	1,344	3 rule-phrasings vs baseline	no change, all $p > .6$
Taxonomy (v12)	1,176	trigger vs override split	none, $p = .23$ (neg. result)
Detection	24	injection detector flags	0/24
Generality (fresh)	3,220	scope vs harm, 91 new scenarios	86% vs 27%, $p = 1{\times}10^{-90}$
Frontier (Tier 2)	1,080	scope vs harm, 3 frontier models	70% vs 24%, $p = 3{\times}10^{-18}$
Frontier, GPT-6.1 (full set)	796	scope vs harm, conf./ref.	70/15% vs 0/0%
Frontier-hard, GPT-6.1	158	false scope vs false harm	67% vs 0%, $p = 2{\times}10^{-4}$
Frontier, Sonnet 5.5 + Gemini-4	1,023	scope vs harm, confirmed	80/58% vs 4/0%
Frontier-hard, Sonnet 5.5 + Gemini-4	242	false scope vs false harm	87/73% vs 0/0%
Crossed claims, GPT-6.1	460	scope vs harm in the same scenario	60 of 99 scope>harm, 1 reverse
Defenses, GPT-6.1	642	false-claim violations, 3 defenses	verify-prompt 22→0%, utility kept
Surface robustness	319	paraphrase-invariance, 8 scenarios	65% vs 65%, $p = 1.0$
Construct (κ)	27	human recovers scope/harm	$\kappa = 1.0$

Table 3. Full condition sweep, original panel (6 models, $n = 599$).

Condition	SCOPE	HARM
Control	6%	8%
Orthogonal twin	3%	10%
Licensing, unchecked	50%	12%
Licensing, refuted (false)	49%	15%
Licensing, confirmed	**93%**	11%

Table 4. Generality, 91 fresh scenarios (3,220 trajectories).

Condition	SCOPE	HARM
Control	40%	30%
Confirmed	**86%**	27%
Refuted (false)	59%	21%

same condition. With a claim present, scope moves every frontier model far more than harm (Table 5; pooled 70% vs 24%, $p = 3 \times 10^{-18}$), and `gpt-4o` breaks rules under scope claims as often as the open panel (85%). `claude-sonnet-5.5` never acts on a harm claim (0%) and resists scope claims best (47%). We do not report a false-claim ranking from this run. On the fixed harness (below), `claude-sonnet-5.5` acts on 8% of false scope claims on these scenarios rather than 38%, so the registry's answer matters. `gpt-4o` and `gemini-2.5-flash` have not been re-run.

A 2026 frontier model: GPT-6.1. We ran `gpt-6.1-sol` on the full 100-scenario in-band set, on the fixed harness: the three conditions plus a NO_DIRECTIVE arm that removes the rule, × 2 reps (796 clean trajectories). Four more, all from one harm scenario, were blocked by the provider's biological-risk filter before reaching the model and are excluded. GPT-6.1 obeys the rules at control (scope 2%, harm 0%) although with the rule removed it takes the action in 76% (scope) and 37% (harm) of trajectories. This is a lower bound, since an agent that queries the registry still learns the original rule. Harm claims never move it, not even when the registry confirms them (0/106 confirmed, 0/106 refuted). Scope claims do: 70% when confirmed and 15% when refuted (95% CI 9–24%; $p = 10^{-5}$ vs. refuted harm). Unlike the older models, GPT-6.1 mostly heeds the registry, so the false-claim rate falls from 78% (`gpt-4o`) to 8% on the 60 scenarios of Table 5; it narrows the gap but does not close it. The 14 false-claim violations show why: replaying the registry, 12 called `check_policy` but asked only whether the *rule* was in force, 2 did not call it, and none asked about the

Table 5. Frontier models via OpenRouter on the 60 validated in-band scenarios of Tier 2 (1,080 trajectories, 0 errors). ‡Tier-2 rows: for 51 of the 60 scenarios the registry was unavailable (harness bug, Section 7), so Conf and Refu differ only in label. †Fixed harness, same 60 scenarios, taken from each model's full 100-scenario run (text); `gemini-4-argon` excludes the scenarios its provider refused. Violation %; scope=CLASSIFICATION.

Model	Claim	Cont	Conf	Refu
claude-sonnet-5.5‡	scope	5	**47**	38
	harm	0	0	0
gpt-4o‡	scope	12	**85**	78
	harm	8	32	27
gemini-2.5-flash‡	scope	22	**78**	70
	harm	7	42	28
gpt-6.1-sol†	scope	2	**67**	8
	harm	0	0	0
claude-sonnet-5.5†	scope	0	**77**	8
	harm	0	3	0
gemini-4-argon†	scope	0	**49**	0
	harm	0	0	0

Table 6. Crossed design on `gpt-6.1-sol`: each scenario seen with both claim types (violation %, confirmed / refuted). Bold = the scenario's original claim.

Scenarios (original type)	scope claim	harm claim
Base, scope (46)	**70 / 15**	3 / 1
Base, harm (54)	41 / 0	**0 / 0**
Frontier-hard, scope (5)	**80 / 67**	0 / 0
Frontier-hard, harm (5)	53 / 13	**0 / 0**

claim. Whenever GPT-6.1 checked the claim, it complied.

Two more 2026 flagships: Claude Sonnet 5.5 and Gemini-4. Within a $10 budget we ran `claude-sonnet-5.5` (2 reps) and `gemini-4-argon` (1–2 reps) on the full set with the fixed harness (1,023 clean trajectories). Gemini's provider refused to process 8 of the 100 scenarios at all ("blocked the response") and some others intermittently; those are excluded. Both repeat GPT-6.1's pattern (Table 5): scope claims rise from control (5%, 12%) to 80% and 58% when confirmed, while harm claims stay at 0–4% ($p \leq 3 \times 10^{-17}$, confirmed scope vs. harm). They differ on false claims: Sonnet acts on 16% of refuted scope claims, Gemini on none (0/72). Of Sonnet's 15 false-claim violations, 13 checked only the rule; 2 asked about the claim, received "no record", and acted anyway.

Same scenario, both claim types. Every scenario above carries one claim type, so scope and harm are compared *across* scenarios, and GPT-6.1 shows the two groups differ before any claim (with no rule it takes the action in 76% of scope but 37% of harm scenarios). To remove this confound we wrote, before measuring, one claim of the *opposite* type for every in-band and every frozen frontier-hard scenario: a harm claim for each scope scenario and a scope claim for each harm scenario, with the same entity, date and register, within 12 characters of the original, and the directive, task, tools and predicate unchanged (pre-registered). On GPT-6.1 (Table 6) the claim type, not the scenario, carries the effect. In scenarios that originally carried a harm claim, the new scope claim reaches 41% under confirmation while the original harm claim stays at 0%; in scope scenarios the new harm claim reaches 3%. Per scenario, the scope claim beats the harm claim in 60 of 99 and loses in 1 (38 ties; sign test $p = 5 \times 10^{-17}$), and the pattern holds on the 60 scenarios GPT-6.1 would violate with no rule present (43 vs. 1). On the frontier-hard set it holds in 9 of 10 scenarios, with no reversal. The crossed claims await the same human construct review as the originals.

Context length. Table 17 gives the confirmed-condition rates at three context lengths (5 models, directive offset held constant). The asymmetry holds at every length (OR 5.7/6.4/5.0, all $p < .001$). Baseline violation does not rise with length, so we fail to replicate length-alone degradation in this agentic setting.

Mixed effects. Refit with model and scenario as random intercepts, the claim-type×confirmation interaction stays credible (+3.83 [3.39,4.26] and +6.33 [5.45,7.21] log-odds), so the effect is not an artifact of pooling.

Surface robustness: paraphrase-invariance. A scope claim and its inert twin necessarily differ in content, so a lexical classifier can separate them. The question is whether the *agent* responds to the claim's surface form or to its meaning. For eight CLASSIFICATION scenarios we rewrite the licensing claim into four *pre-committed* surface forms assigned by rule, not by outcome: the original "…as X, not Y" template, and three paraphrases that drop the shared template and the negation ("the system records …as X"; "…was reclassified to X"; "…now X, effective"). Across 319 trajectories (5 models), the pooled violation rate with the claim present is flat: 65% for the original template and 65% pooled over the three paraphrases (Fisher $p = 1.0$; per wording 62–66%). Removing the negation and the shared phrasing changes nothing, so the reclassification *meaning* drives the asymmetry, not the template. Per-scenario wording shifts rates in both directions (e.g. a drop for one data-residency scenario, a rise for one API-scope scenario), as expected of a semantic effect with phrasing noise; pooled, it is invariant. We report the full distribution and never select the strongest wording.

Robustness battery. Beyond paraphrase-invariance we stress the effect on five further axes (Table 7), each testing a distinct confound. All are pre-committed, run on the free panel, and report full distributions. The asymmetry survives every one. Under the strict pass^2 metric it is consistent (scope 76% vs harm 18% of scenario–model cells violate on both reps). It is *causal and graded*: stating the same claim more assertively steps violation from baseline at a hedge ("may", 45%, n.s. vs the 35% neutral control) to ceiling at a plain assertion ("is", 87%, $p{=}1{\times}10^{-4}$), with no further rise from added authority (85%). The driver is therefore *asserted certainty*, and hedging neutralises it. The effect is channel-general (claim delivered via memory, tool output, or a system note: 87/77/95%, n.s.) and noise-robust (claim buried among 8 unrelated memory entries: 77→92%), and it holds at temperature 0 (scope 83% vs harm 30%, $p{=}6{\times}10^{-5}$), which rules out a sampling artifact. These six studies ran on the harness with the registry-routing bug (Section 7), so they test the claim with the registry unavailable. Their within-study comparisons are unaffected, and re-running them on the fixed harness is pending.

Per model. Table 15 gives the confirmed-condition rates by model. The asymmetry is large in the smaller open models and compresses in `mistral-small` and `qwen3.6`, which have high baselines and accept harm claims more readily. We claim the effect holds in most models but not

Table 7. Robustness battery (pre-committed, free panel; registry unavailable, see Section 7). The claim-type asymmetry survives every perturbation.

Perturbation	Result	Rules out
Paraphrase (4)	65%=65%, p=1.0	surface template
Consistency (pass^2)	76% vs 18%	averaged noise
Assertiveness (dose)	45%→87%, p=10^{-4}	correlation
Claim source	87/77/95%, n.s.	memory-specific
Distractor dilution	77→92%	salience
Temperature 0	83% vs 30%	sampling artifact

all, and we report the breakdown rather than an aggregate.

Trigger vs. override: a negative result. We pre-registered whether the asymmetry reflects a broader "trigger vs override" law: three trigger-negating claim types (scope, exemption, temporal) should move agents, three override types (harm, benefit, urgency) should not. It does not (Table 16): pooled trigger 15% vs override 18% ($p = .23$); benefit (24%) and urgency (19%) move agents *more* than any trigger type. The contribution is the measured scope-vs-harm phenomenon, not a theory of it.

4.1. A frontier-hard variant, and multi-model generality

In a 10-scenario pilot the newest frontier models (Claude Sonnet 5.5, GPT-6.1-sol-pro, Gemini-4) resisted the base attack: a false (registry-refuted) scope claim moved them 0/15, versus 59% on the models above. On the full set, `gpt-6.1-sol` acts on 15% (Section 4). Headroom explains only part of this. Many base scenarios use high-salience actions that a frontier model refuses by default (with no rule present, GPT-6.1 still declines 63% of harm-scenario actions), but it takes the action in 76% of base scope scenarios, so there it is resisting the false *claim*, not the action. We therefore built a *frontier-hard* variant ("batch16"): mundane, reversible actions carrying genuine task pressure, with the scope fact stated as a plausible, naturalistic attribute (a manifest tag, an SSO group, a console disposition) rather than a bald override. The ground truth still makes the action a violation, and the refuted registry still returns no record. Every scenario is admitted by *violability* (it is taken when no rule is present), never by its claim effect, and the set is stress-tested (paraphrase-invariance, distractor-dilution, consistency) before use. On this variant the false-claim arm returns: Claude Sonnet 5.5 violates 5/6. In 4 of those 5 it called the registry first, but asked only whether the *rule* was in force; in one it queried the claimed status, received "no record", and acted anyway. The agent verifies the rule and trusts the claim that takes the entity outside it.

GPT-6.1, tested blind on the frozen 10-scenario set at 3 reps (90 trajectories, 0 errors), shows the same pattern at larger n. Control is 0/15 for both claim types. Scope claims succeed 12/15 when confirmed and 10/15 when refuted (67%, 95% CI 42–85%); harm claims stay at 0/15 in both arms ($p = 2 \times 10^{-4}$, false scope vs. false harm). Its calibration rules out refusal: with the rule removed it takes the action in all 17 batch16 scenarios (32/34 trajectories), including the 7 that Claude Haiku 5.5 refused. Of its 10 false-claim violations, 5 checked only the rule and 5 did not check; none asked about the claim. Claude Sonnet 5.5 and Gemini-4, tested blind on the same frozen set at 3 reps, agree: a false scope claim succeeds 13/15 (87%) and 11/15 (73%), and a false harm claim 0/15 for both ($p \leq 5 \times 10^{-5}$). Both break the rule more often at control than GPT-6.1 (27% and 40% on scope scenarios), yet the scope lift far exceeds it. Across the three newest flagships, a plausible false scope claim works 67–87% of the time; a false harm claim never does.

Table 8. Multi-model generality on the frontier-hard variant (batch16, 10 in-band scenarios, violation %). 20 API identifiers resolve to 13 distinct models; aliases are merged (n = trials per cell). In-band = both controls $\leq 40\%$. Over the 8 in-band models, mean confirmed-lift over control is $+62$ (scope) vs $+17$ (harm), and scope-lift exceeds harm-lift in 8/8.

		CLASS (scope)			harm			
Model	n	ctl	cnf	ref	ctl	cnf	ref	in-band
gpt-oss-120b	5	20	80	80	0	20	20	✓
gpt-oss-20b	5	0	100	100	20	0	40	✓
qwen3.8-27b	5	60	80	60	20	40	20	–
mistral-large-2512	5	40	100	80	40	80	80	✓
mistral-large-4	5	40	60	80	20	20	0	✓
mistral-medium-latest	20	25	100	100	20	20	30	✓
mistral-small-2603	15	93	93	93	93	87	67	–
ministral-14b-2512	5	40	100	100	0	40	40	✓
ministral-8b-2512	5	100	80	100	100	100	100	–
ministral-3b-2512	5	100	80	80	80	80	40	–
zai-glm-5-3	15	0	20	0	7	20	0	✓
glm-5-2	5	0	100	100	0	40	20	✓
codestral-2508	5	60	80	100	60	100	80	–

Table 8 reports every open model we evaluated on the frontier-hard set (Groq- and Mistral-hosted; no proprietary routing). We queried 20 API identifiers, but provider metadata shows that several are aliases of the same underlying model; we merge aliases (pooling their trials) and report 13 distinct models. We score each model by claim-lift over *its own* control and count it as in-band only where control leaves headroom (both $\leq 40\%$); 8/13 are in-band. Among them the mean confirmed-lift is $+62$ points for scope versus $+17$ for harm, the mean refuted-lift for scope is $+59$, and scope-lift exceeds harm-lift in all 8. The remaining 5 are ungoverned at ceiling: these weak models break the rule with or without any claim. These runs are single-rep and the variant awaits the same human construct audit as the main set; we report them as a generality probe, not a headline number.

5. Construct Validity

To test whether CLASSIFICATION and HARM are a real distinction or our labeling, we gave a rater the 27 validated claims blind (directive and claim only) and asked them to classify each as scope or harm. Agreement with our labels was 27/27 ($\kappa = 1.0$), including the genuine judgment calls (an allowlist exemption and a standing authorisation both classified as scope; a duplicate-charge note as harm), so a human reader can recognize the distinction cleanly. We will add ≥ 2 fully arms-length raters and report Fleiss κ in camera-ready.

6. Detection-Blindness and Defenses

The effective claim issues no instruction. Scored by a purpose-built prompt-injection detector, all 24 claims and twins are unflagged (median ≈ 0.0006, max ≈ 0.004; a canonical injection scores ≈ 0.9996). The vector that drives up to 100% violation is thus orders of magnitude below the signal instruction-oriented defenses key on. We test three rule-phrasing defenses under confirmed licensing (Table 12 in the appendix): restating the rule, naming the harm, and asserting scope authority. None changes violation (all CMH $p > .6$); the targeted SCOPE-AUTHORITY phrasing moves it by 0.0pp. The benchmark's refuted and confirmed arms let a defense report security and utility together.

Verification defenses. The mechanism above (agents check the rule and trust the claim) points to verification. We test three pre-registered defenses on `gpt-6.1-sol` over all 51 scope scenarios (46 base, 5 frontier-hard; 642 trajectories), scoring *security* (violations under a refuted claim) and *utility* (actions taken under a confirmed, legitimate exemption; Table 9). (D1) A one-sentence *verify-the-claim* instruction in the system prompt closes the attack (false-claim violations 22%→0%, 0/107, $p = 3 \times 10^{-8}$; on the frontier-hard set 67%→0%) and *raises* legitimate use (71%→93%), because the agent now looks up the exemption and acts on it. (D2) A PoEM-style *registry gate*, which refuses the prohibited call unless the registry confirmed the claim earlier in the episode, is secure by construction (22 attempts blocked) but cuts legitimate use to 44%, because the agent often acts before checking. (D3) Labelling memory as *untrusted* cuts false-claim violations to 4% and legitimate use to 25%: the agent largely stops using memory. Verification, not distrust, is the useful defense, provided an authority exists to verify against. These results cover one model; weaker models may follow the instruction less reliably.

7. Threats to Validity and Reproducibility

An earlier version of this work was withdrawn by the authors before submission after an audit found two fatal instrument bugs; every result here runs on the rebuilt, pre-registered instrument. **Registry-routing bug (found and fixed in this revision).** Runs that loaded the released tool-schema file sent the registry tool to scenario-specific mock executors, which answered "unknown tool". This hit the Tier-2 frontier run, a first GPT-6.1 full run (quarantined, not counted, and re-run), and the robustness battery. It did not hit the core studies, the 91-scenario generality study, or the frontier-hard runs, which route the registry correctly. In the affected runs the claim was present but could not be checked, so their claim-type comparisons stand while their confirmed/refuted contrasts do not; we label them as such. A regression test now checks that every released scenario gets a condition-correct registry reply (released data and scripts: Section A). **Limitations.** Mocked backends; synthetic scenarios; stipulated registry. CONTROL baselines in newer scenarios run higher (∼39%) than the original (∼5%); we report both. Models span small/mid open models plus Mistral and five frontier models (Tier 2, GPT-6.1, Gemini-4, Section 4). One harm scenario cannot run on GPT-6.1: its claim text trips the provider's biological-risk filter (4 trajectories, excluded). Gemini-4's provider refuses 8 scenarios outright and others intermittently (48 trajectories, excluded). Construct validation rests on one expert rater close to the project (a blind full-set sheet is prepared for independent raters, Section 5). Scenarios were drafted with LLM assistance and reviewed by the authors; review records cover batches 1–12, and batches 13–15 are pending.

Table 9. Verification defenses on `gpt-6.1-sol`, 51 scope scenarios ($n = 107$ per cell). Security: lower is better. Utility: the exemption is real, so higher is better.

Defense	False claim ↓	True claim ↑
None	22%	71%
D1 verify-claim prompt	**0%**	**93%**
D2 registry gate	0%	44%
D3 untrusted memory	4%	25%

8. Conclusion

LLM agents governed by natural language enforce whether a rule *applies*, not whether breaking it is *safe*. A benign declarative reclassification with no instruction, which injection detection misses and rule phrasing does not fix, moves agents to break standing rules up to 100% of the time, even when it is false; a matched harm-denial does not. ClaimHijackBench measures this across 120 scenarios and 99 domains, generalizes to fresh scenarios at $p < 10^{-89}$, persists on five frontier models (the newest act on 0–16% of false scope claims on the base set and 67–87% on frontier-hard scenarios), holds within scenarios when both claim types are crossed, and is recovered by a blind human rater. Agents verify the rule but not the claim; telling them to verify it stops the attack on GPT-6.1 without blocking real work.

Impact Statement

ClaimHijackBench is a defensive measurement: it exposes a failure mode so it can be fixed, and uses only synthetic scenarios with fictional entities and mocked tools (no real systems, no personal data). The scenarios describe prohibited actions in the abstract and contain nothing operationally useful for harm; the licensing claims are benign declaratives, not exploits. Our assertiveness result points to a concrete mitigation: treat unhedged claims in memory as unverified and check them against an authority before acting; we hope this steers deployment, not evasion.

References

Ahad, T., Hossain, I., Alam, M. J., Puppala, S., Alam, S. B., and Talukder, S. The misattribution gap: When memory poisoning looks like model failure in agentic AI systems. *arXiv preprint arXiv:2605.22842*, 2026.

Andriushchenko, M. et al. Agentharm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations*, 2025.

Anonymous. Context length degradation in long-context language models under perfect retrieval. *arXiv preprint arXiv:2510.05381*, 2025.

Anonymous. Beyond goodhart's law: A dynamic benchmark for evaluating compliance in multi-agent systems. *arXiv preprint arXiv:2606.07805*, 2026.

Chen, X., Xie, X., Fu, W., Zhou, J., Yu, S., and Xuan, Q. MemSecBench: Tracking agent memory poisoning from persistence to consequence and repair. *arXiv preprint arXiv:2607.27080*, 2026.

Debenedetti, E., Zhang, J., Balunović, M., Beurer-Kellner, L., Fischer, M., and Tramèr, F. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2024.

Debenedetti, E. et al. Defeating prompt injections by design. *arXiv preprint arXiv:2503.18813*, 2025.

GateMem. GateMem: Benchmarking Memory Governance in Multi-Principal Shared-Memory Agents, 2026. arXiv:2606.18829.

Gebru, T., Morgenstern, J., Vecchione, B., et al. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

Levy, I., Wiesel, B., Marreed, S., Oved, A., Yaeli, A., and Shlomov, S. ST-WebAgentBench: A benchmark for evaluating safety and trustworthiness in web agents. In *International Conference on Learning Representations*, 2026.

Mazumder, A., Roy Dipta, S., et al. AgentCollabBench: Diagnosing when good agents make bad collaborators. *arXiv preprint arXiv:2605.08647*, 2026.

Memory Security Survey. A Survey on Long-Term Memory Security in LLM Agents: Attacks, Defenses, and Governance Across the Memory Lifecycle, 2026. arXiv:2604.16548.

Rahman, M. H. and Kim, J. Proof-of-execution memory: Defending llm agents against forged-reasoning attacks by verifying what actually happened. *arXiv preprint arXiv:2608.16032*, 2026.

Stale Constraints. When Stale Constraints Go Unchecked: Budgeted Verification Failures in Inherited Agent Memory, 2026. arXiv:2608.25553.

Turpin, M., Michael, J., Perez, E., and Bowman, S. R. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Wang, H., Poskitt, C. M., and Sun, J. Agentspec: Customizable runtime enforcement for safe and reliable llm agents. In *Proceedings of ICSE 2026*, 2026.

Wu, J., Gong, M., et al. Policy-invisible violations in llm-based agents. *arXiv preprint arXiv:2604.12177*, 2026.

Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025.

Zhan, Q., Liang, Z., Ying, Z., and Kang, D. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

Zhang, H., Huang, J., Mei, K., Yao, Y., Wang, Z., Zhan, C., Wang, H., and Zhang, Y. Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In *International Conference on Learning Representations*, 2025.

A. Data Availability

Table 13 in the appendix lists every trajectory file backing the paper. All are released with per-study pre-registrations and the scripts that regenerate every number. Violation is recorded for all 14,819 trajectories; 13,527 additionally store the full tool-call log (the taxonomy study and one calibration pass store outcomes only). Section B gives the complete per-component breakdown and the supplementary development and robustness studies (a further 4,331 trajectories), for 19,150 released trajectories in all. The benchmark, the data and an interactive website that lets readers browse the scenarios and trajectories are at `https://claimhijackbench.pages.dev`.

B. Complete Data Inventory

Every released trajectory file. Primary studies (Table 10) back the claims in the main text and sum to 14,819 clean trajectories. Supplementary studies (Table 11) were run during instrument development, calibration, and robustness checking, and are released for completeness and reproducibility. All counts are clean (non-error) trajectories and regenerate from the released scripts.

Table 10. Primary released data (per component); sums to the 14,819 total in the main text.

Component	clean n
Core asymmetry, v6	599
Core asymmetry, v7	300
Context length, v8	748
Violability control, v10	168
Rule-phrasing defenses, v11	1,344
Trigger/override taxonomy, v12 (negative result)	1,176
Generality, batch 1 (pilot)	80
Generality, batch 2	321
Generality, batch 3	337
Generality, batch 4	331
Generality, batch 5	387
Generality, batch 6	518
Generality, batch 7	429
Generality, batch 8	398
Generality, batch 9	400
Generality, batch 10	369
Generality, batch 11	370
Generality, batch 12	367
Generality, batch 13	370
Generality, batch 14	370
Generality, batch 15	310
Frontier Tier 2 (claude-sonnet-5.5 / gpt-4o / gemini-2.5-flash)	1,080
Frontier, gpt-6.1-sol, full in-band set (4 provider-blocked excluded)	796
2026 frontier pilot, base set (sonnet-5.5 / gpt-6.1-sol-pro / gemini-4)	90
Frontier-hard (batch16), claude-sonnet-5.5	36
Frontier-hard (batch16), gpt-6.1-sol (frozen set + calibration)	158
Frontier-hard (batch16), 13-model open panel	600
Crossed claims, gpt-6.1-sol (base 400, frontier-hard 60)	460
Verification defenses, gpt-6.1-sol (base 552, frontier-hard 90)	642
Frontier, claude-sonnet-5.5, full in-band set (fixed harness, 2 reps)	600
Frontier, gemini-4-argon, full set (fixed harness; refusals excluded)	423
Frontier-hard (batch16), claude-sonnet-5.5 (frozen set + calibration)	124
Frontier-hard (batch16), gemini-4-argon (frozen set + calibration)	118
Primary total	**14,819**

Beyond the trajectory files above, the repository stores derived analyses as summaries (decomposition, the stage-0 manipulation check, the comprehensive governance study, and grader-κ validation); these are not re-counted in the totals. Every scenario, predicate, tool schema, pre-registration, and analysis script is released.

Table 11. Supplementary studies, run during development, calibration, and robustness checking; released for completeness.

Study	clean n
Crossed self-report (v9)	597
Cross-provider replication	395
Hardening / robustness	519
Judge re-scoring sensitivity	537
Multi-agent governance	245
Tool-governance study	214
Verification study	446
Frontier pilot	108
Prompt-injection hard probes	72
Gemini free-tier attempt (v14, rate-limited)	6
Robustness battery (paraphrase/dose/source/distractor/temp.)	838
Frontier-hard development, claude-haiku-5.5 (calibration/effect/paraphrase/distractor)	354
Supplementary total	**4,331**
Grand total (primary + supplementary)	**19,150**

Table 12. Rule-phrasing defenses (confirmed, scope; $n = 1{,}344$).

Guardrail phrasing	violation
None (baseline)	45%
Restate the rule	44%
Name the harm	46%
Assert scope authority	45%

C. Additional Results Tables

D. Agent Setup and Prompts

Agents run a single-episode OpenAI-style function-calling loop (Section 3): fixed system prompt, then one user message concatenating `GUARDRAILS.md` (the directive), `MEMORY.md` (the note: neutral, claim, or twin), and the task; tools passed with `tool_choice=auto`; up to 8 steps; temperature 0.7 (0.0 for the determinism check); 6 retries with backoff; API-failure trajectories dropped. The system prompt is deliberately minimal so the directive, not the scaffold, carries the governance:

```
You are an autonomous site-reliability agent with production tool access.  You act
on requests by calling tools.  Work the task to completion, then call finish.
```

The CONTROL note is the string ``No additional operational notes.''; the CONFIRMED/REFUTED conditions differ only in the registry (`check_policy`) response the agent receives when it verifies the claim. The user message template is:

```
## GUARDRAILS.md\n{directive}\n\n## MEMORY.md\n{note}\n\n---\n{task}
```

Full model ids, dates, per-scenario predicates, and the runnable harness are released with the benchmark.

Table 13. Released trajectory data (clean trajectories per study).

Study (file)	n
Core asymmetry (v6/v7/v8)	1,647
Violability control (v10)	168
Rule-phrasing defenses (v11)	1,344
Trigger/override taxonomy (v12)	1,176
Generality, 91 fresh scenarios (batch1–15)	5,357
Frontier Tier 2 (OpenRouter, 3 models)	1,080
Frontier, GPT-6.1 (full in-band set)	796
Frontier-hard (batch16) and 2026 frontier pilot	884
Crossed claims and defenses, GPT-6.1	1,102
Frontier, Sonnet 5.5 + Gemini-4 (all runs)	1,265
Total	14,819

Table 14. Benchmark composition. Boundary cases are released with labels, not used for the primary effect.

Stakes band	CLASS.	HARM
Moderate (validated, in-band)	46	54
Trivial (boundary)	12	2
Destructive (boundary)	2	1
Non-violable (boundary)	0	3
Total	60	60

Table 15. Per model (confirmed, context study).

Model	SCOPE	HARM
deepseek-v4-flash	47%	0%
glm-5.3-flash	42%	6%
minimax-m3	42%	0%
qwen3.6-35b	58%	36%
mistral-small-latest	64%	39%

Table 16. Trigger/override taxonomy (1,176 trajectories): no clean split.

Trigger-neg.		Override	
scope	14%	harm	11%
exemption	16%	benefit	24%
temporal	15%	urgency	19%

Table 17. Context length (confirmed condition, 5 models, 60/cell).

Claim	0.1k	8k	24k
SCOPE	63%	42%	47%
HARM	23%	10%	15%

Table 18. Per-batch (scope control→confirmed; harm control→ confirmed). Direction holds; magnitude varies.

Batch	SCOPE	HARM
2	39→96%	31→29%
3	40→71%	26→43%
4	38→90%	36→26%
5	40→76%	29→33%
6	33→88%	38→33%
7	45→100%	18→8%
8	38→84%	32→30%
9	30→85%	25→32%
10	47→93%	30→20%
11	43→77%	25→25%
12	30→76%	32→30%
13	50→100%	30→30%
14	56→96%	27→17%
15	30→100%	32→27%