THE NO-WITNESS RULE: MAKING FABRICATED TOOL-CALL ARGUMENTS UNREACHABLE BY CONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Constrained decoding is the standard way to stop a tool-using agent from inventing the identifiers, paths and order numbers it passes to tools: restrict the argument to values the agent has actually seen. Every published treatment of a candidate set that cannot supply the needed value fires when that set is *empty*. We show the empty set is the wrong condition to test. Whether a set is ever empty is a property of how narrowly the slot is typed rather than of the trajectory, and neither end of that choice rewards testing it — under the type function we run it never arises, and under the strictest one we price it fires on 39.85% of emissions while destroying 8.39 correct values for each fabrication caught. The condition that arises regardless is a candidate set that is non-empty and does not contain the value, 10.09% of referential emissions, and nothing treats it. There the constraint requires the model to choose a wrong value, the tool fails, no witness is added, and the next step repeats the last: a livelock we reproduce outside our own system with a small model, a synthetic loop and VLLM's `guided_choice`, where an abstention token sat in the candidate set for all 240 constrained emissions and was chosen 0 times. **The defect is the trigger, not the enforcement.** Testing whether the value the model *wrote* is licensed cuts unwitnessed emission from 9.00% to 2.45% at 8B and 13.03% to 2.39% at 14B, constrains 8.43% of call steps, and is equivalent to the baseline on task score within this benchmark's own 7-point resolution floor; the published prompt-based method for the same failure clears neither bar. What such a rule is worth is a property of the environment rather than of the model: where referents are validated it buys back a wasted call, and where they are not it buys the difference between a wrong write and no write.

1 INTRODUCTION

An agent that books a flight must supply a booking reference. An agent that files a ticket must supply a ticket id. These are *referential* arguments: they denote entities that already exist, and a value is either the right one or it is a fiction. Language models fill them fluently whether or not anything in the conversation supplies them.

This is not the familiar hallucination problem, and it is not fixed by scale. On the benchmark column that isolates missing-parameter behaviour, a frontier proprietary model scores 23.00 and an *untrained* 8B open model scores 22.50 — half a point apart — while a 671B reasoning model reaches only 36.50. The column is the worst one for essentially every model reported on it. Whatever the difficulty is, parameters are not buying it down.

Nor is it fixed by asking. We tested the strongest prompt we could write — 1,159 characters naming the rule and both remedies — and it moved the fabrication rate by 0.13 percentage points, $z = 0.1$, a null we have now measured three times over (Section C.2). Instruction reaches the task and not the behaviour.

The state nobody handles. If a behaviour survives being described to the model, stop describing it and make it unreachable. Constrained decoding does exactly that: it restricts a reference position to the identifiers available at that point, and proves soundness for doing so. Every formulation we know

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

of also says what to do when that restriction can supply nothing — when the candidate set is *empty*. One raises, one relaxes to free decoding, one terminates in refusal.

None of them says what to do when the set is non-empty and does not contain the value the agent needs, and that is the state agents are actually in. Whether a set is ever empty turns out to be a property of how narrowly the slot is typed rather than of the trajectory, and neither end of that choice rewards testing for it: under the type function we run the set is empty 0.00% of the time, and under the strictest one we price the test fires on 39.85% of emissions while blocking 8.39 correct values for every fabrication it catches. Satisfiable-but-wrong arises 10.09% of the time regardless, and in it the constraint requires the model to pick a wrong value from a menu of them. The tool fails, the context does not change, and the next step is the same computation on the same input.

Our position. The defect is in *what the rule fires on*, not in whether to constrain. A rule that asks "can this slot be filled at all?" waits for a condition that does not occur; a rule that asks "is the value the model just wrote licensed?" fires on exactly the emissions that fabricate and leaves every other emission untouched. We show that the first deadlocks — generically, outside our own system (Section 4.5) and in a full agent (Section 5) — and that the second costs nothing measurable while removing most of the fabrication.

The rule. Maintain a witness set W_t of typed literal spans, credited only from what the agent has actually seen: user turns, successful tool returns, tool documentation it read, schema enumerations. When a referential slot is about to be filled with a value that W_t does not license, remove every value token and leave exactly two continuations reachable, in order:

1. RESOLVE — a type-matched *read-only* call that would retrieve the value from the environment;
2. ASK — a harness-rendered clarification, whose reply re-enters W_t only as a typed span bound to the requesting slot.

The ordering puts the cheaper remedy first. Which remedy *applies* depends on whether a read-only call could supply the value the agent should have written — not, as we first assumed, on whether the value it did write happens to exist. That question is undecidable on our first benchmark, where a correct per-slot value exists only along the reference trajectory (Section B.4); on the second, whose tasks state their required actions outright, every decidable case is RESOLVE — the correct identifier sat in the environment one lookup away. The mechanism selects RESOLVE in 141 of 141 firings with both branches equally available (Section 5) — though on a benchmark whose user turns are scripted, and where ASK therefore cannot be answered, that count does not separate a preference for the cheaper remedy from a preference for the branch that does anything at all. Proposition 1 keeps both continuations reachable regardless.

Contributions.

1. **An unhandled state in constrained decoding, named and quantified.** Every published treatment of a semantic candidate set that cannot supply a value fires when the set is *empty*. We show that condition is not worth testing at any setting of the type function that decides it: vacuous at one end (0.00% of emissions, under what we run) and ruinous at the other (39.85% firing, 10.65% precision, 8.39 correct values destroyed per fabrication caught). The state that arises regardless — non-empty, and not containing the value the agent needs — arises 10.09% of the time and has no treatment (Section 4.4).
2. **That state deadlocks the decoder, shown outside our own system.** With Qwen3-0.6B, a synthetic loop and VLLM's `guided_choice`, the unconstrained model solves the task 10/10 times and the constrained model 0/10 times when the set is satisfiable and wrong, hitting the step cap in every trial at temperature 0 and 0.7 alike (Section 4.5). In a full agent the same state costs 35.0% $\rightarrow$ 5.0%, with 110 of 200 episodes at the cap.
3. **The defect is the trigger, and moving it is nearly free.** Testing whether the value the model *wrote* is licensed, rather than whether the set is empty, cuts unwitnessed emission from 9.00% to 2.45% at 8B ($z = 6.32$) and 13.03% to 2.39% at 14B ($z = 9.44$), constrains 8.43% of call steps, and moves task score by -1.67 $\pm$ 2.52 points over three seeds. Both

arms are reported, because without the failing one the working trigger is an assertion rather than a finding.

4. **Reachable is not taken.** Prior work identifies keeping declination reachable as the needed repair. We show reachability alone buys nothing: an abstention token sat in the candidate set for all 240 constrained emissions of the failing condition and was chosen 0 times, and our own first arm left the decoder a derivable declination it selected zero times in 3,824 steps while every episode ran to the cap.
5. **The behaviour being enforced against, measured on two models and two boards.** 8.03% of referential emissions at 8B and 9.77% at 14B, which scores *higher* on the benchmark; unwitnessed state-mutating calls rise from 59 to 76. It predicts failure — 29.13% of failed episodes against 14.29% of passed ($p = 0.0135$), replicated at $p = 0.00059$ — but observationally: an episode's first departure is a wrong *tool* 63.0% of the time against a wrong *argument* 2.5%, and the oracle substitution that would settle direction is not computable there.
6. **What such a rule costs, and where its value is not.** Its worth is a property of the environment: where referents are validated the environment already refuses 45 of 47 unwitnessed emissions against 24.78% of witnessed ones, so the rule buys a wasted call; where they are not, it buys the difference between a wrong write and no write (Section 4.3). Its value is also not in the normalisation ladder — W^1 is indistinguishable from W^0 on the operating frontier (Section C.3) — nor in the type function, which the working trigger does not need at all.

What we do not claim. We do not claim semantic correctness of the emitted value: membership in a witness set is not truth, and the soundness results we build on place value correctness outside what masking can enforce. We do not claim the first rule for an empty candidate set — prior work in a different domain terminates in a refusal state on exactly that condition, and that arm is our floor baseline. And the state-neutrality we verify is neutrality *with respect to the benchmark's checker*, which ignores private attributes; that is the right notion for scoring and not the same as changing nothing.

2 Related Work

Four bodies of work meet at the argument slot. We adopt the machinery of the first, invert one branch of it, and take from the second the problem it declined to solve.

2.1 Constrained and Structured Decoding: What We Adopt

Restricting generation to a formal language at decode time is a solved engineering problem with several mature implementations: a byte-level pushdown automaton over a context-free grammar Dong et al. (2025), its agentic successor Li et al. (2026b), an Earley parser over a derivative-based lexer Guidance AI contributors (2026), index-based guided generation Willard & Louf (2023), and, for the case that matters here, character-level tries built directly from a large finite candidate set Xu & Bouyarmane (2026). We use this machinery and claim none of it. Concretely, the mask is expressed as a schema applied only inside a tagged argument region, so the tagged-region facility and the trie backend are both borrowed.

The formulation nearest to ours indexes the grammar by the environment, so that a reference position admits exactly the names, fields and options available at that point, and proves that no out-of-scope identifier can be emitted Zhang et al. (2026a). Two things separate us. First, substrate: their candidate set is populated from a program's own declarations and evaluated on programming languages, not from a multi-turn agent trajectory whose witnesses accumulate through user turns and tool returns. Second, and this is the whole of our delta, the *empty* case. When their slot query returns no candidates the grammar is *relaxed* to a coarser fragment — the model is permitted to write more freely at precisely the moment nothing licenses what it writes. That is a published design decision, not an oversight, and NoWit inverts that one branch: on an empty candidate set we remove every value token instead of admitting more. We remain inside the class of properties such masking can enforce

— membership in a set of witnessed spans — and we explicitly do not claim the semantic correctness of the chosen value, which that work places outside what a decoder constraint can guarantee.

We are also not the first to give an empty candidate set a distinguished treatment. In column-level access control for text-to-SQL, an empty allowed set at a column slot drives a terminal refusal state Miyamoto et al. (2026). We therefore do not claim the first rule for the empty set; we claim the first *constructive, ordered* continuation for an unwitnessed referential emission in multi-turn agent tool calling, and refuse-on-empty is our floor arm — a refusal scores zero on a task benchmark, which is exactly what makes the constructive claim legible. Our own results sharpen the distinction: firing those continuations on an *empty* candidate set, as that prior work does, is what we measure and reject (Section 4.4).

The cost of constraining, which is why Proposition 1 exists. A growing literature measures what masking takes away. Structured-output constraints can suppress tool invocation outright Li et al. (2026a); decomposing a constrained abstention pipeline shows the grammar component repairing form while degrading the decision Lee (2026); the validity–correctness tradeoff is sharpest at small scale Ray (2026a); and part of the loss is attributable to the prompt rather than the decoder Lee et al. (2026); Nguyen et al. (2026); Reddy et al. (2026). Head-to-head engine comparisons exist but measure schema compliance and throughput rather than argument correctness Geng et al. (2025). This is the literature that makes an unconstrained-abstention invariant necessary rather than decorative: a rule that closed over the stop token and the action-selection alphabet would reproduce exactly the collapse these papers report.

2.2 Asking Instead of Guessing: The Problem, Offered but Not Enforced

The behaviour we target has been named. Agents facing unclear instructions can be prompted to ask, with a benchmark built for it Wang et al. (2025). When a schema demands a field, models invent one, and — decisively for us — an *offered* escape from doing so goes essentially unused Usman (2026). Uncertainty can be structured to trigger clarification Suri et al. (2026) or aligned by post-training so a model learns when to abstain Zhou et al. (2026). Grounding can be made action-conditioned, with the support set induced automatically Jiang et al. (2026).

Every one of these *offers* a remedy the model may decline, or *trains* a disposition to take it. None makes the alternative unreachable. Our K5 result (Section C.2) is the empirical statement of why that distinction matters at this scale: a prompt that names the rule and both remedies changes the fabrication rate by 0.13 percentage points and the task score by nothing at all. That independently reproduces the unused-escape finding on a different instrument.

One benchmark ships something close to our rule, and the difference is instructive. tau2-bench includes a fact-checker that flags details — order numbers, postal codes, addresses — not present in or derivable from a declared set of legitimate sources, and singles out information the scenario says is unknown. That set is a witness set in all but name, which we take as independent evidence that the notion is the natural one. It is nonetheless a different instrument: it audits the *simulated user* rather than the agent, runs post hoc as a language-model judge, and exists to gate reruns so that a fabricating user cannot corrupt the benchmark. Its own prompt withholds the agent's tool calls from it, so it cannot see the emissions we measure, and the repository contains no agent-side counterpart.

The closest statement of the problem, and the one we take our support relation from, credits support only from successful tool observations or protocol-authorised user fields, and reports that stop, hand-off and re-verify policies do reduce unsupported writes — while noting that the cost is shifted to autonomy, escalation and fallback without quantifying it Zhang et al. (2026c). We do not define that relation; we enforce it, and Section 3.4 is the quantification that was left open.

A parallel line reasons about provenance at the granularity of the argument rather than the call, but stops at detection and hands the repair back to the model. See Section E.1.

3 THE NO-WITNESS RULE

3.1 WITNESSES

Definition 1 (Witness set)**.** At turn t, W_t is the set of typed literal spans credited from what the agent has actually seen, in trust order: user turns, returns of successfully executed tool calls, tool documentation the agent read, and schema enumerations. W_t is *monotone*: it never retracts.

Monotonicity is a deliberate simplification with a named cost: an agent whose context is later fragmented or rewritten keeps credit for spans it can no longer see. We declare it as a limitation rather than defend it.

Definition 2 (Witnessed at level ℓ)**.** A value v is witnessed at W^0 if it occurs verbatim — as an exact span, or case-sensitively at a word boundary in a source utterance. It is witnessed at W^1 if it is witnessed at W^0 or lies in the closure of W^0 under five fixed rules, and at W^2 if it is type-consistent. By construction $W^0 \subseteq W^1 \subseteq W^2$, and we enforce the inclusions in code.

The word-boundary clause is not cosmetic: a tokenised candidate set cannot witness a multi-word value, and treating such values as fabrications inflated our own early measurements by roughly a quarter before we caught it.

3.2 THE RULE

Let s be a referential slot of type τ and let $W_t^\tau \subseteq W_t$ be its type-matched subset. Both rules below share their remedy — when the rule fires, exactly two continuations remain reachable, in order: RESOLVE, a type-matched read-only call that would retrieve the value; then ASK, a harness-rendered clarification. They differ only in what makes it fire, and that difference is this paper's result.

R1, the empty-set rule. While decoding the value of s:

- if $W_t^\tau \neq \emptyset$, value tokens are restricted to W_t^τ;
- if $W_t^\tau = \emptyset$, *every* value token is removed and only the two continuations remain.

Masking the type-matched *subset* rather than all of W_t is a correctness statement and a cost control at once: it is what keeps the compiled alternation small (Section 3.4). This is the rule we specified in writing before measuring anything.

R2, the detect rule. Let the model emit the call it wants, unconstrained. Let v be the value it wrote into s. Then:

- if $v \in W_t^\tau$, accept the call *verbatim* — nothing is masked and the trajectory is identical to an unconstrained one;
- if $v \notin W_t^\tau$, discard the call and re-decode the step under a grammar admitting only the two continuations.

Why the difference is not cosmetic. R1 tests whether the candidate set is *empty*; R2 tests whether the needed value is *absent*. Those come apart completely on real trajectories. Measured over the baseline runs under the type function we run, a referential slot's candidate set is empty 0.00% of the time, while the value the agent needs is unavailable from anything it has seen 10.09% of the time — and the second figure, unlike the first, does not move with the type function (Section 4.4). R1 therefore never fires, and in the interval where it should have, it instead forces the model to substitute a witnessed but wrong value — a substitution it has no way to get right, which at low temperature repeats until the step budget is exhausted. R2 fires on exactly the emissions the census calls fabrications and leaves every other emission untouched.

R2 also costs less. R1 constrains every step and must decode the whole action under a grammar; R2 decodes freely and pays for a grammar only on the emissions that need one — 8.43% of call steps.

Proposition 1 (Abstention-reachability)**.** *At every step where the rule fires, at least one non-`call` continuation is derivable under the mask. The decoder is therefore never required to emit a tool call,*

and the rule cannot produce the invocation collapse observed when decoding constraints are allowed to close over the whole vocabulary.

We state the proposition in this weaker form deliberately. The natural stronger claim — that the stop token and the action-selection alphabet lie *outside* the mask — is what a stack honouring scoped constraints would give, and it is not what ours gives: scoped constraints are silently ignored there, so the grammar covers the whole step (Section 6). Reachability of a declining continuation is what the implementation actually guarantees, and it is what the argument below needs.

Proposition 1 is what distinguishes an enforced rule from a stop-constraining one, and it is satisfied by construction rather than by tuning.

What reachability does and does not buy, measured. And the weaker form is still stronger than what the decoder does with it, which is the qualification that matters: derivable is not the same as chosen. In the first complete arm the decoder selected the derivable declination `done` *zero* times in 3,824 constrained steps, and 110 of the 200 episodes force-terminated — because the natural way for an agent to end a turn is ordinary prose, and a whole-step grammar gives prose no derivation. Abstention-reachability held formally and bought nothing. We report this because a reachability proposition is easy to satisfy and easy to over-read: the branch has to be one the decoder will actually take, and ours was not until the constraint was narrowed to the action alone (Section 4.4). The construction is a grammar whose root is an alternation,

$$\texttt{root} ::= \texttt{call} \mid \textsc{Resolve} \mid \textsc{Ask} \mid \texttt{done},$$

in which ASK and `done` are unconditional members. A tool appears in `call` only while every required referential slot has a non-empty type-matched witness set, so the empty-set rule is expressed by a tool leaving the alternation rather than by editing a mask — and no witness set, empty or otherwise, can remove the branches that let the decoder decline.

We had intended to scope the constraint to a tagged argument region instead, leaving the rest of the step untouched. The decoder we run accepts such a request and ignores it: constrained and unconstrained output are byte-identical (Section 6). Putting the abstention branches inside the grammar achieves the same invariant on a facility that is actually enforced, and has the advantage of being checkable by parsing what was emitted. We verify it empirically as an equivalence test on tool-call rate and schema validity rather than asserting it (Section 5).

3.3 RESOLVE MUST NOT ITSELF CHANGE THE WORLD

If RESOLVE emitted a call that mutated state, the rule would trade a fabricated write for an unrequested one, and on a benchmark that compares final environment state it would also destroy its own score. We therefore audited every public method of the benchmark's environment classes: of 170 methods, 59 are read-only by our verb classification and *0* of those write public state — checked including subscript assignment and in-place container mutation. The checker compares exactly the non-private attributes, so the 59 clean readers are safe resolution targets.

We state the claim precisely: this is neutrality *with respect to the checker*, which ignores private attributes. A reader that advances a private cursor is invisible to it. That is the right notion for scoring and it is not the same as changing nothing.

3.4 COST

The witness set changes after every tool return, so a fresh grammar is compiled at every step and a grammar cache never hits. We measure the compile time as a function of the alternation size: 63.7 ms at 20 witnesses, 199.5 ms at 100, and 758.6 ms at 500; the empty-witness union of RESOLVE and ASK compiles in 348.5 ms. This is why the mask is restricted to the type-matched subset: it keeps the alternation in the tens rather than the hundreds. We report compile latency as a first-class cost alongside tokens and extra calls, because it is the one cost a reader would not otherwise predict.

4 What Agents Write, and What a Constraint Can Do About It

This section does two jobs. The first six subsections establish that the behaviour exists, is frequent, and is not an artefact of how we chose to look: an instrument (Section 4.1), the census it produces (Section 4.2), what instruction does to it (Section C.2), what a witness level would block and at what price (Section C.3), and a second board where the environment itself passes judgement (Section 4.3). All of it is measured on frozen models with no method in the loop.

The last two carry the finding. Section 4.4 asks which condition a constraint should fire on and prices both candidates across the type functions that decide them: the state every published system guards against is vacuous or ruinous depending on that choice and worth firing on at neither extreme, while the state none of them guards against arises 10.09% of the time regardless. Section 4.5 shows what the unguarded state does, in a setting that shares no code with our mechanism, so that the failure cannot be read as a defect of ours.

4.1 Instrument

For every emitted tool call we recover its arguments, label each argument slot, and ask whether the emitted value was licensed by anything the agent could have seen *at the moment of emission*. Three things make this a measurement rather than a heuristic, and each was forced on us by data.

Slots are partitioned three ways, not two. A slot is Referential-Existing when it denotes an entity that must already exist, Referential-New when the agent is *creating* the name (`mkdir`'s directory, a copy's destination), and Generative otherwise. Only the first is subject to the rule. The middle category is not a refinement for its own sake: a mechanism that masked creation targets would break `mkdir`, `cp` and `mv` outright, which is the constraint tax reported elsewhere as a collapse of tool invocation and a loss of abstention. The automatic construction of such a support set is not ours, and neither is a measured error rate for it: we did not run the hand labelling that would establish one, and say so where it would have gone (Section 5.3). Every rate below is conditional on this partition.

Witness levels form an enforced ladder, $W^0 \subseteq W^1 \subseteq W^2$, and witnesses include tool returns — so the environment must actually execute, and measuring without it omits an entire source (Section B.1, Section B.2).

4.2 Result

Over 200 episodes of BFCL v4 multi-turn (`miss_param`) with a frozen 8B open-weight model, the agent emitted 1,438 tool calls carrying 897 referential arguments. Of those arguments, 825 (91.97%) were witnessed in context. The remainder splits into two classes: for 9 (1.00%) the emitted value was absent from context yet present somewhere in the environment, and 63 (7.02%) named nothing that existed anywhere. Read that split for what it is — whether the value the model *wrote* happens to exist — and not as a sizing of the rule's two continuations; Section B.4 explains why we once did the latter and no longer do. The unwitnessed rate per emission is therefore 8.03%, and 23.5% of episodes contain at least one such emission.

The behaviour concentrates where it does damage. 59 unwitnessed emissions occurred in *state-mutating* calls. This is the subset that matters: the benchmark's checker is indifferent to a wasted read, so an unwitnessed *write* is the only variety that is both scored and irreversible.

It does not shrink at the next size up. We repeated the census unchanged on a 14B model of the same family. The unwitnessed rate does not fall — it rises slightly, 8.03% to 9.77% — over more calls (1,699 carrying 1,044 referential arguments), and unwitnessed *writes* rise from 59 to 76. The composition also moves the wrong way for the cheaper remedy: values naming something that exists somewhere fall from 1.00% to 1.25%, while values naming nothing at all rise from 7.02% to 8.52%. The larger model invents more freely, not less.

The larger model is not simply worse at everything. It is *better* at the task — 41.0% against 36.5%, +4.5 points — with an almost identical error profile and only 6 force-terminated episodes, so neither figure is a harness artefact. That is the sharper version of the claim: the model that scores higher is

also the one that emits more unwitnessed referential arguments (+1.74 points). Capability and licence come apart, and the second model buys a check on the obvious objection — the behaviour is not an artefact of one small checkpoint.

4.3 A Second Board, and the Condition That Decides What the Rule Is Worth

Everything above says the behaviour is frequent and, on this benchmark, without consequence. That conclusion is about the benchmark as much as the model, so we repeated the measurement somewhere the argument slots are different in kind.

In tau2-bench's retail domain a referential argument is an opaque identifier — an order number, an item id, a payment method — that cannot be guessed and has to be fetched. Its tasks state the actions they require with concrete arguments, so the correct value for a slot is given rather than reconstructed from a trajectory, and its grader scores each required action separately. 91.2% of its 114 tasks require a state-mutating action carrying such an identifier, and a task demands 7 of them at the median.

Here the rule predicts what the environment will refuse. Over 114 simulations the agent made 394 referential emissions, of which 11.93% were unwitnessed. That is higher than the 8.03% on the previous board, but we do not lean on the direction: the two boards have different tool sets, different slot partitions and different denominators, so the comparison is not one a difference of a few points can survive. (An earlier draft did lean on it, in the opposite direction, and was reading two different instruments against each other.) What matters here is not the level but what happens next.

	emissions	rejected by the environment
unwitnessed	47	45 (95.74%)
witnessed	347	86 (24.78%)

The refusals are *User not found*, *Order not found*, *Variant not found*. Being witnessed is no guarantee — 24.78% of witnessed emissions are refused too, typically a real identifier used in the wrong place — so the flag separates rather than saturating. One-sided Fisher $p = 4.7 \times 10^{-22}$. Every one of the 12 unwitnessed *state-mutating* emissions was refused. This is an emission-level verdict delivered by the environment itself, not an end-state comparison and not a language model's judgement.

The instrument's own classes predict the verdict, including the exceptions. The 45 refusals are not a random 95.74% of the unwitnessed set. They are exactly the 45 emissions our census puts in the *nowhere-recoverable* class, all 100.00% of them; the 2 in the *environment-recoverable* class were accepted, 0.00%. A value that exists nowhere is always refused, and a value that exists but was not seen is always accepted. The three-way partition is being checked here against an external, deterministic ground truth rather than against our own labels.

On the corrected slot-based reading — which continuation could supply the value the agent *should* have written — 22 of the unwitnessed emissions are decidable and every one is RESOLVE: the correct identifier was in the database, one read-only call away. 0 required asking the user. The remaining 25 sit at tool-parameter pairs the task never requires, so no correct value exists to reason about; we report them separately and never fold them in.

The characteristic fabrication is worth naming: the model does not invent plausible identifiers so much as emit a shaped blank, writing `#W0000000` into an order slot repeatedly across unrelated tasks, or enumerating `1008292230`, `1008292231`, `1008292232` in sequence.

The two boards together, which is the result neither gives alone. It would be easy to read the table above as the safety case for the mechanism. It is not, and we will not present it as one. On this board the environment already refuses the fabricated referent, so what a witness rule prevents is a wasted call and the recovery turn after it — a cost in latency and turn budget, not a corrupted state.

What makes that interesting is the contrast. On the first board the environment does *not* validate its referents: an invented path is simply used, which is why 59 unwitnessed state-mutating emissions at 8B and 76 at 14B went through and changed state. The same behaviour, under two environments, has two different costs.

So the condition under which this mechanism is worth its price is not a property of the model at all. **Where an environment validates its referents, a witness rule buys turns; where it does not, it buys the difference between a wrong write and no write.** An agent deployment sits on one side or the other of that line, and which side is a question about the tools, not about the checkpoint. We think this is the more useful claim, and it is the one measurement supports.

4.4 Pricing the Empty-Set Rule Itself

Everything above prices the *restriction*: confine a referential slot to its witness set and count what that blocks. It does not price the rule this paper claims as its contribution, which fires only when that set is *empty*. Running the mechanism forced the issue. Across 1,438 constrained emissions the rule fired **zero** times, and the model took neither ordered continuation — no RESOLVE, no ASK — so the arm was measuring the restriction and nothing else.

The reason is that the rule cannot fire without a *type function*: a decision about which witnesses count as type-matched for a slot. We had not named one. The implementation used the most permissive defensible choice — values seen under a matching key in a tool return, *union* every span from any user turn — on the grounds that inventing a type signature would be a tuning knob. That choice makes the empty set unreachable by construction: a slot is non-empty from the moment the user says anything at all, whether or not what they said has anything to do with that slot.

Four type functions, fixed in advance. ALL admits every literal in W_t to every slot. PERMISSIVE is what we ran: by-key witnesses $\cup$ all user spans. SHAPED admits a user span only if it is identifier-shaped — no separators or spaces, four characters or more, carrying a digit, an underscore or a hash. STRICT admits by-key provenance only. The shape test is deliberately crude and was fixed before measurement; a tunable resemblance test is precisely the knob PERMISSIVE was chosen to avoid, and a tuned one would let us select whichever firing rate flattered the mechanism.

At every emission we ask two questions against one reconstructed witness set: did the rule fire on the tool the agent actually chose, and was that emission about to write an unwitnessed value? Crossing them prices the rule.

board	type function	fires	precision	recall	correct blocked per catch
BFCL	PERMISSIVE	0.00%	—	0.00%	—
(n=1,438, 69 bad)	SHAPED	15.58%	9.82%	31.88%	9.18
	STRICT	39.85%	10.65%	88.41%	8.39
τ^2-BENCH	PERMISSIVE	0.00%	—	0.00%	—
(n=384, 40 bad)	SHAPED	1.04%	100.0%	10.0%	0.00
	STRICT	52.08%	19.0%	95.0%	4.26

Two things are being decided, and they are decided by different things. *Whether the rule fires* is set by the type function and is not a property of the board: PERMISSIVE fires on neither. *Whether firing is a good idea* is set by the board. On τ^2-BENCH the crude shape filter fires 4 times and is right all 4, destroying nothing. On BFCL the same filter fires on 15.58% of emissions and is wrong about nine times in ten. The difference is what the referents look like: τ^2-BENCH's are opaque ids the conversation cannot derive, so identifier-shaped-and-unwitnessed really does mean invented, while BFCL's are filenames and paths the model often constructs correctly from context the shape filter does not recognise as a witness.

Why the price is so high where it is high. The rule fires on the *tool* — any required referential slot with an empty witness set — while being wrong is a property of an *argument*. That is not an implementation artefact a better grammar would fix: if a slot has no admissible value the tool cannot be called at all, whatever the model was going to write in the other slots. The rule's precision is therefore bounded above by the type function's recall, and no fixed type function we are willing to defend comes close on a board whose referents are ordinary words. Set against the restriction half on the same board — precision 51.72%, price 0.93 (Section C.3) — the empty-set rule is an order of magnitude worse wherever the environment does not do the work for it.

The rule tests the wrong condition, and we can say by how much. The pricing above leaves one question open: if the empty-set rule never fires under any defensible type function, does the situation it exists for ever arise? It does, and often. We replay the *baseline* trajectories — not the constrained ones, where the livelock below has already changed which steps exist — and ask a counterfactual with a definite answer at each referential emission: would the rule have offered a candidate set containing a correct value? Over 763 emissions with a defined reference (132 excluded as having none, so correctness is undecidable):

	emissions	share
FEASIBLE — the set contains a correct value	589	77.2%
INFEASIBLE — non-empty, no candidate correct	174	22.8%
recoverable: witnessed, our extraction missed it	97	—
absent: not observable at all	77	10.09%
EMPTY — what the rule tests for	0	0.00%

Read the last two rows together. The empty-set rule fires on 0.00% of emissions. The situation it exists for — the agent cannot write the correct value from anything it has seen, so it should go and look, or ask — arises on 10.09% of them, and the rule is silent through every one. It tests whether the candidate *set is empty* rather than whether the *needed value is absent*, and the set is never empty: it is full of every ordinary word the conversation contained. 112 of the infeasible emissions are state-mutating, so this is not confined to harmless reads.

We separate the two halves of INFEASIBLE deliberately. 97 of the 174 are cases where the reference value *is* witnessed by our own predicate and our span extraction simply failed to offer it. That is a limitation of the implementation, it is fixable, and it is not evidence about the rule. The claim rests on the 77 absent cases and on the EMPTY count of zero.

The consequence at run time is a livelock, not a lost point. With no branch to escape through, a model whose needed value is not on the menu must pick one of the wrong candidates. The tool fails, the witness set does not change, and at temperature 0.001 the next step is byte-identical. In the constrained arm 75.0% of all emitted calls are exact repeats against 12.7% in the baseline, and 110 of 200 episodes run to the step cap against 2. The failure mode of a witness constraint is not an empty witness set. It is a full one that does not contain the answer.

4.5 THE DEADLOCK IS THE CONSTRAINT'S, NOT OURS

Section 4.4 showed that our own constrained arm livelocked on 110 of 200 episodes when the candidate set was non-empty and did not contain the needed value. On its own that is a fact about our grammar, our witness sets and our harness, and a reader is entitled to treat it as a bug report. This subsection settles which it is.

We built the smallest thing that has the same shape and shares no code with the mechanism: a synthetic directory, a tool that looks up a record by id, an agent loop that sees each error and may try again, and VLLM's own `guided_choice` — the primitive the systems we build on benchmark. The model is Qwen3-0.6B, which is the primary model in both of them. Three conditions, 10 trials each, at temperature 0 and 0.7, with a step cap of 12.

condition	solved	at the step cap
unconstrained	10/10	0
constrained, correct value in the set	10/10	0
constrained, set non-empty, value absent	0/10	10/10

Both temperatures give the same three rows. Three things follow.

The deadlock belongs to the constraint. The unconstrained model solves the task immediately, so the loop is not hard; the constrained model solves it immediately when the set contains the answer, so the constraint is not harmful in itself. It fails only in the state where the set is satisfiable and wrong.

The same result holds at temperature 0.7. See Section C.7.

An abstention option inside the constraint is not taken. `GIVE_UP` was a member of the candidate set in all 240 constrained emissions of the failing condition. It was chosen 0 times. This reproduces, in four lines of synthetic task, what our first arm found the hard way: a branch the decoder *can* derive is not a branch it *will* take (Proposition 1). Putting the escape inside the constrained alphabet does not make it reachable in any operational sense.

What this means for the systems this builds on. A distinction first, because it decides who this applies to. A grammar engine that enforces a formal language has no notion of *the value the agent needs*, so the question does not arise for it; what we are discussing are systems that populate a *semantic* candidate set from the environment and must then say what happens when that set cannot supply the value. Among the treatments of that case we could find, each fires on the set being *empty* — relaxing to a coarser fragment (Zhang et al., 2026a), or refusing (Miyamoto et al., 2026).

The empty set, however, is not a fact about trajectories: it is a fact about the type function, and the table above prices both ends of that choice. Under PERMISSIVE, which is what we run, it arises 0.00% of the time — the fallback is reachable and never reached, so it protects nothing. Under STRICT it arises 39.85% of the time with precision 10.65%, blocking 8.39 correct values for every fabrication it catches. **There is no setting of the type function at which testing for emptiness is a good trigger: it is vacuous at one end of the ladder and ruinous at the other.** That is a stronger statement than the one we could make from PERMISSIVE alone, and it does not depend on which type function a given published system happens to implement.

The satisfiable-but-wrong state is not a fact about the type function in the same way. It asks whether anything the agent has seen could supply the value at all, which is a property of W_t, and it arises 10.09% of the time. So the condition that has a fallback is one no choice of type function makes worth firing on, and the condition that occurs regardless has no fallback at all.

5 EXPERIMENTS

The design below — arms, ablations and kill criteria — was fixed in writing before any of it ran, so that the results could not be reverse-engineered from what happened to work. Nothing in this section is filled from anything but a run directory.

One comparator is absent and we scope the claims to match. Every constrained arm reported here is one of ours: R1 and R2 differ in trigger and are compared against each other and against an unconstrained baseline. We do *not* report a head-to-head against a third-party constrained decoder, so nothing below should be read as R2 beating a published constrained-decoding system — the claim is about which condition a constraint should fire on, argued from the pricing in Section 4.4 and from arms that share everything but the trigger.

5.1 SETUP

Frozen open-weight models at $\leq$13B, pinned by revision. **Thinking mode is on**, and we state that because it is measured rather than chosen: the harness's chat template emits an empty reasoning block only when `enable_thinking` is explicitly false, its local handler never passes that flag, and a captured prompt ends at `<|im_start|>assistant` with no such block. The model therefore reasons before acting, which is why the mechanism has to leave room for it (Section 4.4). Comparability is claimed only against published rows that state the same model family *and* mode, because the same family in a different mode moves this column by roughly twenty points and a different revision of the same family by nearly thirty — both larger than any effect we report.

Pins. `Qwen/Qwen3-8B` at revision `b968826d` and `Qwen/Qwen3-14B` at `40c06982`; BFCL-EVAL `2026.3.23`; serving on VLLM (vLLM contributors, 2025) `0.8.5.post1` with XGRAMMAR (Dong et al., 2025) `0.1.18` and `transformers 4.51.3`; the evaluation client on `transformers 5.15.1`. Two clusters were used and their stacks are identical version for version, which we verified package by package rather than assumed, because constrained-decoding behaviour differs between grammar engines and mixing two stacks inside one table would make the comparison meaningless. Two scored environments: a multi-turn tool-calling benchmark scored

programmatically, and a retail agent benchmark whose grader is judge-free. A third board is reported as a verification row only, since its scoring cannot exercise a slot-interior mask.

We had planned an oracle-substitution experiment to turn the association into a causal claim, and it is not computable on this benchmark: the episodes where a single referential argument is the only departure from the reference are too few, and the first departure is a wrong *tool* far more often than a wrong *argument*. See Section D.1.

5.2 Ablations

Two of the three planned ablations are done and reported where they belong rather than duplicated here: the closure levels W^0/W^1/W^2 as an operating frontier (Section C.3), and the leave-one-out over the five W^1 rules (Section B.3). Both say the same thing, and it is not what we expected when the ladder was specified: the closure is a smaller effect than assumed — four of the five rules change the unwitnessed count by nothing at all, and W^1 is indistinguishable from W^0 on the frontier.

The third, placement — turn-level against slot-interior masking — is not run. It needs a constraint primitive scoped to a span, and the serving stack silently ignores scoped constraints (Section 6), which is the same absence that forces our grammar to cover the whole step. We report it as unrun rather than approximate it.

5.3 What the Slot Partition's Error Rate Is Not

Every rate in this paper is computed over slots that an automatic three-way partition called referential. That partition's own precision and recall against human labels are **not measured here**, and we make no claim about them: establishing them needs a few hundred hand labels from a person, which is work we have not done. A model labelling its own partition is not a validation of it, so we did not substitute one. The consequence is stated plainly — every number in Section 4 and this section is conditional on that partition being right, and the audit in Section B.3 bounds only the normalisation rules inside it, not the partition itself.

Both arms, complete. The baseline and the constrained arm each ran all 200 episodes of the missing-parameter column, same 8B checkpoint, same temperature, sixteen threads on one dedicated A100 apiece so that the latency figures compare the mechanism rather than the load.

	OA	Base	Miss Func	Miss Param	Long Ctx
baseline	39.50%	47.50%	46.00%	35.00%	29.50%
R2	40.88%	51.00%	47.50%	36.00%	29.00%

Every column is 200 episodes, so OA is their unweighted mean and equals the pooled accuracy. Section 5 reports R1 on Miss Param only, because an arm that livelocks 110 of 200 episodes does not need three more columns to be understood.

Parity, with the interval that establishes it. Over the 800 paired episodes of the four columns, R2 scores +1.38 points above the baseline with a 95% bootstrap interval of $[-1.75, +4.50]$ resampling episodes, and McNemar's exact test on the discordant pairs gives $p = 0.44$ ($b = 89$, $c = 78$). **The interval contains zero: this is parity, not an improvement**, and we report it that way even though the point estimate favours us.

The interval is also a fact about the benchmark worth stating. The paired standard error on a *single* 200-episode column is about 3.7 points, so no column of this size can resolve a difference below roughly 7 points — and every arm difference in this paper, and many of the model-to-model gaps in the published tables we compare against, are smaller than that. Pooling the four columns is what makes any statement possible at all, and single-column differences of one to three points should not be read as orderings by anyone, ourselves included.

	baseline	R1 empty-set	R2 detect
task score (Miss Param)	35.0%	5.0%	36.0%
unwitnessed per referential emission	8.65%	0.00%*	2.53%
unwitnessed state-mutating emissions	67	0*	22
malformed actions	1	0	—
mean latency per emission	12,038 ms	11,525 ms	11,173 ms
exact-repeat calls	12.7%	75.0%	21.3%
episodes reaching the step cap	2	110	11

* true by construction, not measured — see below.

The gap was the trigger, not the enforcement. R1 and R2 mask the same slots with the same witness sets and offer the same two continuations. They differ only in what makes the remedy fire, and that difference is the whole of the thirty-point gap. Changing the trigger recovers the task score in full (36.0% against a 35.0% baseline — 72 and 70 episodes of 200, so the difference is noise and the claim is parity), cuts unwitnessed emissions by 71% and unwitnessed writes by 67%, and takes force-termination from 110 episodes back to 11 against the baseline's 2.

The reduction is larger at 14B, where the baseline rate is higher (Figure 2b). See Section D.3.

Three seeds, and the two halves of the result separating. Everything above is single runs. Repeating the pair at 8B across three server seeds on one site, with model, revision, temperature, thread count and category held fixed, gives paired task differences of +1.00, -4.00 and -2.00 points:

over three paired seeds	
task score	-1.67 ± 2.52 pp, 95% CI $[-7.92, +4.59]$
equivalence within ±7 pp	TOST, $t = 3.67$ and $t = 5.96$ vs. 2.920: **equivalent**
unwitnessed emission (relative)	$-65.2 \pm 10.1\%$, same direction in every pair
baseline task score across seeds	33.50% – 36.50%
R2 task score across seeds	31.50% – 36.00%

The two quantities behave differently under the same variation, and that is the result. The task effect is indistinguishable from zero — its interval contains it, two of the three seeds favour the baseline, and the sign of the point estimate is opposite to the single-run figure quoted above. We state the interval with Student's t rather than a normal quantile, which at $n = 3$ is the difference between a half-width of 2.85 and one of 6.25; the wider one is correct and it does *not* bound the cost at a few points. A failure to reject is in any case not equivalence, so we test equivalence directly: against a ±7-point margin — this benchmark's own resolution floor, established below — both one-sided tests reject, and the arm is equivalent to the baseline at the only precision this board supports. The unwitnessed rate falls by roughly two thirds in every seed.

We keep the four-column table because it is the breakdown this literature reports and a reader needs it to place our rows beside published ones, but its differences are single runs sitting inside seed noise and should not be read as orderings. The number to quote for the task effect is the seed estimate.

One consequence extends past this paper. Varying *only* the server seed moves the baseline across 33.50%–36.50%, three points, on a benchmark where published work routinely separates models by one or two. Every single-run number on this board, including all of ours before this table, carries that much slack.

The claim is about a rate, and a rate is where this benchmark has power. Task score is measured over 200 episodes and cannot resolve anything below about 7 points, which is why every task comparison in this paper reads as parity. The quantity the rule governs is the unwitnessed-emission *rate*, measured over roughly a thousand referential emissions per arm, and there the situation is different. All three pairs below ran within a single site:

comparison	unwitnessed	relative	test
baseline 8B	9.00%	—	—
+ R2	2.45%	−72.8%	$z = 6.32$, $p = 2.6 \times 10^{-10}$
+ AWN	6.54%	−27.3%	$z = 1.84$, $p = 0.065$
baseline 14B	13.03%	—	—
+ R2	2.39%	−81.7%	$z = 9.44$, $p = 3.9 \times 10^{-21}$

The mechanism's reduction is overwhelming at both sizes and larger at the larger model. AWN's is not significant.

The control that was written down to kill this. AWN (Wang et al., 2025) is the published ask-when-unclear method for exactly this failure, and we fixed its kill condition before running it: if it reached within 2 pp of R2 on unwitnessed writes *at matched task success*, the case for enforcing rather than instructing would be over. Its prompt is the released one, taken from the repository at the pinned commit rather than paraphrased, and it runs as the published method does — a judge before each action that may ask instead.

It does not reach the condition. It sits 4.1 pp away on the rate, its own reduction does not clear significance, and it costs task score: 28.50% against 34.00% for the baseline and 36.50% for R2 on the same site. We record that the condition was set in advance, because a kill criterion announced after the result is not one.

One caveat we owe AWN. It was evaluated on a board where a question gets an answer; here the user turns are scripted, so asking ends a turn and nothing responds. That handicaps it, and it handicaps our own ASK branch identically — which is why neither is credited with anything on this board.

On the board where the environment passes judgement. Every arm so far ran on BFCL, whose grader compares end state — so an unwitnessed write that happens to coincide with the reference costs nothing there, and the scope claim of Section 4.3 rested on a census on one board and a mechanism on another. τ^2-BENCH closes that: it refuses unwitnessed referents at the emission. Both arms below ran on one host under the same deterministic evaluator subset the census used.

	baseline	R2
referential emissions	306	234
unwitnessed	13.07%	0.00%*
refused *by the environment*	38 (95.0%)	0
unwitnessed state-mutating emissions	9	0
simulations containing one	27/114	0/114
reward	0.04	0.05

* a construction check, not a result — the arm's witness index and the audit's are built alike, so the arm cannot emit what the audit would call unwitnessed. The 38 refusals the baseline incurred are not a construction: they are what the environment did.

What it buys here, and what it does not. The rule removes the 38 environment refusals that unwitnessed referents caused, and the 9 unwitnessed write attempts, across 27 of 114 simulations. Reward is unchanged, 0.04 against 0.05. That is the predicted result rather than a disappointing one: where an environment already refuses an unlicensed referent, a witness rule buys back the wasted call and its recovery turn, not the task — and at this model scale the tasks that fail mostly fail for other reasons. The claim this board can support is the one it supports.

Varying only the server seed moves the baseline across 33.50–36.50%, and the paired standard error on a 200-episode column is about 3.7 points, so no such column resolves a difference below roughly 7 points. See Section D.5.

R2's residual is a measurement; R1's zero is not. R1's 0.00% follows from its construction: its candidate set is a subset of what the audit accepts, so it cannot emit anything the audit would call

unwitnessed. R2's 2.53% is a genuine measurement, because the arm's check and the audit's are different procedures, and the gap between them is informative. Of the 26 unwitnessed emissions that survive, 23 are values shorter than the audit's two-character minimum span length (`tweet_id="0"`), which nothing that short can satisfy — an artefact of the instrument's floor, present undisturbed inside the 8.65% baseline figure as well — and 3 are genuine escapes, 0.29% of all referential emissions.

The zero is a construction check, not a result. The arm's candidate set is derived from the same predicate the audit uses to decide whether a value is witnessed, so every value the arm can emit is one the audit accepts. 0.00% therefore follows from the construction and is not independent evidence that the mechanism prevents fabrication. We report it because an implementation of a decode-time constraint can silently fail to constrain — an earlier grammar of ours admitted syntactically complete but truncated calls while every summary statistic looked healthy — so a check that the constraint was in force is worth running, as a check. Evidence has to come from an instrument that does not share the witness set, which is what Section 4.3 is for.

6 Limitations and Threats to Validity

We list these in the order a sceptical reader would raise them, and we include the ones that arose from our own errors, because each one changed a number.

The instrument's definitions move the rate a long way. Auditing the benchmark's *reference solutions* — an exercise we use for instrument validation and not as a measurement of agent behaviour, for the reason given in Section B.4 — the unwitnessed rate on one fixed corpus spans roughly 21–48% across defensible settings of the slot partition and the witness-matching relation. That is a statement about how much the definitions matter, not a confidence interval on any rate we report. We take two things from it: no headline may be a bare point estimate divorced from its definitions, and we release the partition and the matcher so a reader can move either and see the consequence. Two of our own early figures were inflated by instrument defects we later found and fixed: a value parser that mangled sequence-valued arguments, and a matcher that could not witness a multi-word value. Both are corrected here; both moved the headline.

The behaviour predicts failure; we still cannot say it causes it. 29.13% of failed episodes carry an unwitnessed emission against 14.29% of passed ones ($p = 0.0135$), and the gap survives restriction to writes and replicates in the prompt arm. What it does not establish is direction. An unwitnessed emission may break an episode, or it may be what a model already off the rails produces on its way to breaking it for some other reason, and the two are indistinguishable in any observational comparison — which is the sharper limitation: *we cannot separate them on this benchmark at all*. The intervention that would, substituting the reference value at the emission, needs a well-defined correct value, and one exists only while the trajectory still tracks the reference: 2.5% of episodes (Section D.1). So the case for the mechanism rests here on licensing rather than on measured consequence — an unwitnessed write is unlicensed whether or not this checker notices — and a board whose grader is action-aware rather than end-state-only is what the consequence claim would require.

This is also how the mechanism's own result should be read. 36.0% against 35.0% is *parity*: it establishes that enforcing the rule is close to free, not that preventing fabrication helps the task. The second claim would need headroom this grader does not provide, since an unwitnessed write that happens to coincide with the reference end state costs nothing here. We report the difference as noise and claim only the first.

The second benchmark is contaminated, and we found it by accident. 2 of the 47 unwitnessed emissions were accepted by the environment, and both were genuine order identifiers belonging to real records. An identifier there is a seven-digit space, so two hits are not luck, and we checked the alternative: neither value appears anywhere in the transcript preceding its emission, in any message of any role, so this is not our instrument missing a witness. Both were emitted before any tool had returned anything, and one of them is the *correct* identifier for its task. The benchmark is public, so its database is a plausible pretraining input and the model appears to be reciting it.

This does not disturb the counts — those values genuinely were not witnessed — but it bounds the interpretation, and it does so in a direction worth stating. A contaminated benchmark makes

fabrication look *less* harmful than it is, because a fraction of fabrications are accidentally right. Here that fraction is small and we can name it exactly, 2 of 47. On a board where it were larger, the refusal rate we report would be an underestimate.

One decoder facility we could not use. The natural implementation scopes the constraint to a tagged region so that only argument values are ever restricted. Our inference stack exposes that facility, accepts the request, and does not enforce it — constrained and unconstrained output are byte-identical on a schema the unconstrained model visibly violates. We report this because it is the kind of failure that produces a mechanism arm which looks like it is running while enforcing nothing, and because it forced the construction described in Section 3: the constraint covers the whole step and the abstention branches live in the grammar. Grammar-level and schema-level constraints on the same stack do enforce, which is what the implementation uses.

REFERENCES

Rahul Suresh Babu and Laxmipriya Ganesh Iyer. Contract2Tool: Learning preconditions and effects for reliable tool-augmented LLM agents, 2026. URL https://arxiv.org/abs/2606.07904.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

BitAgent. BitAgent-8B. https://huggingface.co/BitAgent/BitAgent-8B, 2026.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. XGrammar: Flexible and efficient structured generation engine for large language models. In *Proceedings of Machine Learning and Systems (MLSys)*, 2025.

Enjun Du, Hange Zhou, Chenxu Du, Siyi Liu, Zirong Chen, Ziyu Zheng, and Yongqi Zhang. LedgerMind: Provenance-constrained multimodal agentic reasoning with a structured evidence ledger, 2026. URL https://arxiv.org/abs/2607.28374.

Linfeng Fan, Ziwei Li, Yuan Tian, Yichen Wang, Rongsheng Li, and Xiong Wang. The granularity mismatch in agent security: Argument-level provenance solves enforcement and isolates the LLM reasoning bottleneck (PACT), 2026. URL https://arxiv.org/abs/2605.11039.

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. JSONSchemaBench: A rigorous benchmark of structured outputs for language models, 2025. URL https://arxiv.org/abs/2501.10868.

Guidance AI contributors. llguidance: Low-level guidance — constrained decoding for large language models. https://github.com/guidance-ai/llguidance, 2026.

Lai Jiang, Cheng Qian, Zhenhailong Wang, Pan Lu, Heng Ji, and Hao Peng. ACCORD: Action-conditioned contextual grounding for language agents, 2026. URL https://arxiv.org/abs/2606.16432.

Ivan Yee Lee, Loris D'Antoni, and Taylor Berg-Kirkpatrick. The format tax, 2026. URL https://arxiv.org/abs/2604.03616.

Janghoon Lee. Repair, not improvement: Decomposing constrained decoding in tool-call abstention, 2026. URL https://arxiv.org/abs/2608.13959.

Fangzheng Li, Aimin Zhang, and Chen Lv. Constraint tax in open-weight LLMs: An empirical study of tool calling suppression under structured output constraints, 2026a. URL https://arxiv.org/abs/2606.25605.

Linzhang Li, Yixin Dong, Guanjie Wang, Ziyi Xu, Alexander Jiang, and Tianqi Chen. XGrammar-2: Dynamic and efficient structured generation engine for agentic LLMs, 2026b. URL https://arxiv.org/abs/2601.04426.

Zihan Lin, Xiaohan Wang, Hexiong Yang, Jiajun Chai, Jie Cao, Guojun Yin, Wei Lin, and Ran He. AWPO: Enhancing tool-use of large language models through adaptive integration of reasoning rewards, 2026. URL https://arxiv.org/abs/2512.19126.

Tong Liu, Cheng Qian, Matej Cief, Yuan He, Daniele Dan, Nikolaos Aletras, and Gabriella Kazai. On effectiveness and efficiency of agentic tool-calling and RL training, 2026. URL https://arxiv.org/abs/2606.00135.

Ryoto Miyamoto, Xin Fan, and Hayato Yamana. Policy-conditioned constrained decoding for column-level access control in text-to-SQL, 2026. URL https://arxiv.org/abs/2607.12341.

Ngoc Trinh Hung Nguyen, Alonso Silva, Laith Zumot, Liubov Tupikina, Armen Aghasaryan, and Mehwish Alam. Thinking before constraining: A unified decoding framework for large language models, 2026. URL https://arxiv.org/abs/2601.07525.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Md Ashikur Rahman, Md Arifur Rahman, Niamul Hassan Samin, Khandaker Rifah Tasnia, Md Hasibul Amin, Sifat Rahman Ahona, and Juena Ahmed Noshin. Beyond aggregate risk: Role-stratified conformal risk control for LLM tool calls, 2026. URL https://arxiv.org/abs/2607.24343.

Jaideep Ray. The constraint tax: Measuring validity-correctness tradeoffs in structured outputs for small language models, 2026a. URL https://arxiv.org/abs/2605.26128.

Shawn Ray. What can be enforced? a theory of certified runtime safety for tool-using agents, 2026b. URL https://arxiv.org/abs/2607.22868.

Avinash Reddy, Thayne T. Walker, James S. Ide, and Amrit Singh Bedi. The hidden cost of structured generation in LLMs: Draft-conditioned constrained decoding, 2026. URL https://arxiv.org/abs/2603.03305.

Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. When2Call: When (not) to call tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3391–3409, 2025. doi: 10.18653/v1/2025.naacl-long.174.

Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A. Rossi, and Dinesh Manocha. Structured uncertainty guided clarification for LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 40811–40838, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.2028. URL https://aclanthology.org/2026.findings-acl.2028/.

Team-ACE. ToolACE-2-Llama-3.1-8B. https://huggingface.co/Team-ACE/ToolACE-2-Llama-3.1-8B, 2026.

Rana Muhammad Usman. PhantomFill: When the form demands an answer, language models invent one, 2026. URL https://arxiv.org/abs/2607.20492.

vLLM contributors. vLLM: Easy, fast, and cheap LLM serving. https://github.com/vllm-project/vllm, 2025.

Wenxuan Wang, Shi Juluan, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jen-tse Huang, Wenxiang Jiao, and Michael R. Lyu. Learning to ask: When LLM agents meet unclear instruction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 21773–21784, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1104. URL https://aclanthology.org/2025.emnlp-main.1104/.

watt-ai. watt-tool-8B. https://huggingface.co/watt-ai/watt-tool-8B, 2026.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models, 2023. URL https://arxiv.org/abs/2307.09702.

Xingzi Xu and Karim Bouyarmane. Trie automata for constrained decoding over large finite sets, 2026. URL https://arxiv.org/abs/2608.12574.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Shuoming Zhang, Ruiyuan Xu, Haofeng Li, Qiuchu Yu, Yangyu Zhang, Chunwei Xia, Xiaobing Feng, Chenxi Wang, Huimin Cui, and Jiacheng Zhao. Decode-time grammars: Constrained LLM generation over a refinement order of grammar fragments, 2026a. URL https://arxiv.org/abs/2607.18357.

Zhiwei Zhang, Fei Zhao, Rui Wang, Zezhong Wang, Bin Liang, Jiakang Wang, Yao Hu, Shaosheng Cao, and Kam-Fai Wong. Robust tool use via Fission-GRPO: Learning to recover from execution errors. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pp. 40477–40491, San Diego, California, United States, 2026b. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1880. URL https://aclanthology.org/2026.acl-long.1880/.

Zibo Zhang, Han Zhou, Longyan Tan, and Chenyu Tang. Tracing what agents change when required information is missing. Workshop on FAGEN at ICML 2026 (Poster), 2026c.

Yijin Zhou, Linqian Zeng, Xiaoya Lu, Wenyuan Xie, Dongrui Liu, Junchi Yan, and Jing Shao. Exploring agentic tool-calling decisions via uncertainty-aligned reinforcement learning, 2026. URL https://arxiv.org/abs/2606.06976.

A FIGURES

Both figures are drawn directly from the macro file that supplies every numeral in the body, so no figure can disagree with the text that cites it.

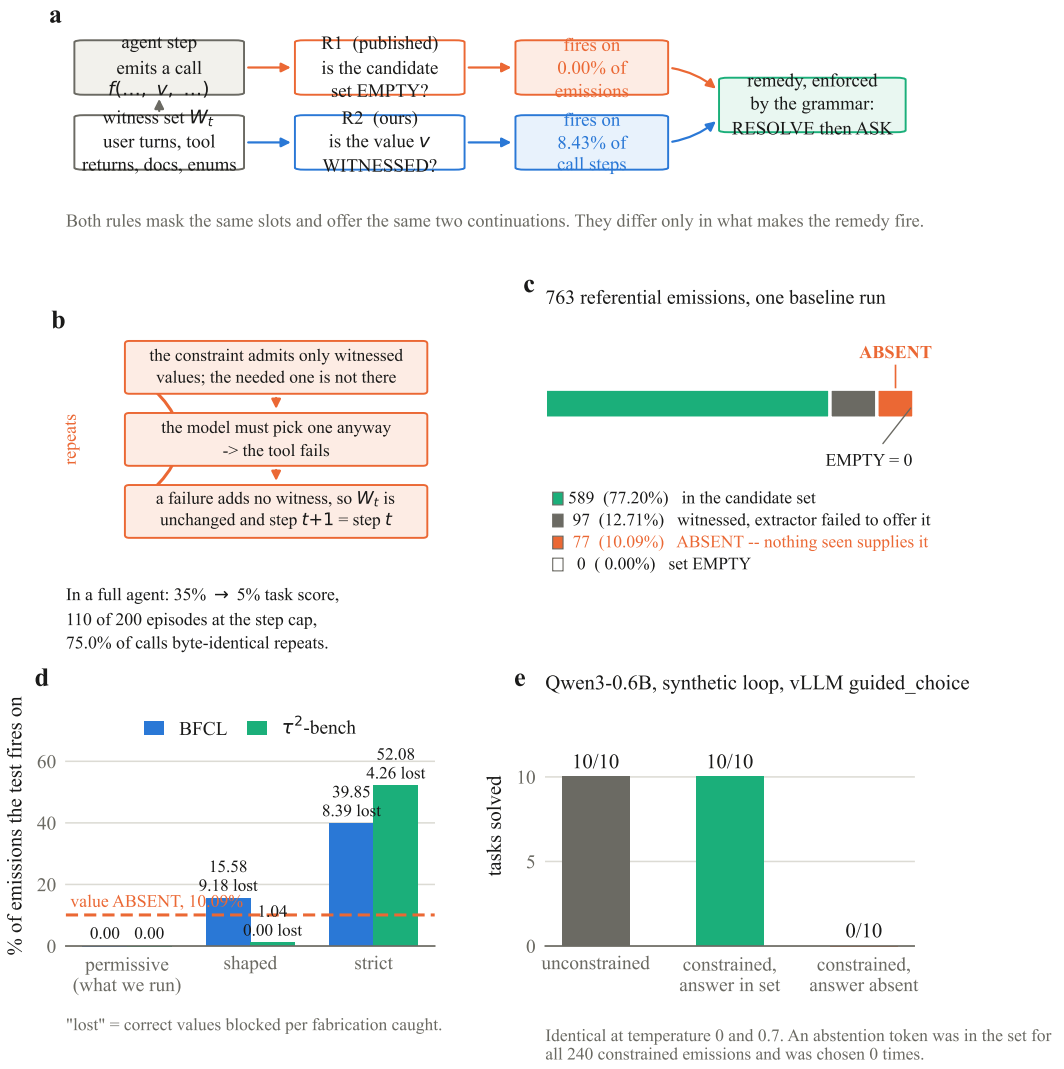


Figure 1: **The published trigger tests a condition that is either vacuous or ruinous, and the condition that does occur deadlocks the decoder.** **(a)** Where the two rules sit in one agent step. Both mask the same slots and offer the same two continuations, RESOLVE then ASK; they differ only in what makes the remedy fire — R1 on the candidate set being empty, R2 on the value the model wrote being unwitnessed. **(b)** What the unguarded state does to a trajectory. The constraint admits only witnessed values and the needed one is not among them, so the model must emit a wrong one; the tool fails; a failure adds no witness, so W_t is unchanged and the next step is the same computation on the same input. In a full agent this costs 35.0% → 5.0% of task score, with 110 of 200 episodes reaching the step cap and 75.0% of calls byte-identical repeats. **(c)** Where the needed value actually was, across 763 referential emissions of one baseline run. The 97 middle segment is our own span extractor failing to offer a value our own predicate witnesses — an instrument error larger than the phenomenon, which is why it is shown rather than folded away. **(d)** The empty-set test priced across the type functions that decide it, on both boards. It fires on nothing under the permissive function we run and on 39.85% under the strictest, where it destroys 8.39 correct values for every fabrication it catches. The dashed line is the satisfiable-but-wrong rate, which is a property of W_t rather than of the type function. **(e)** The same failure reproduced outside our system: Qwen3-0.6B, a synthetic lookup loop and VLLM's `guided_choice`, sharing no code with the mechanism.

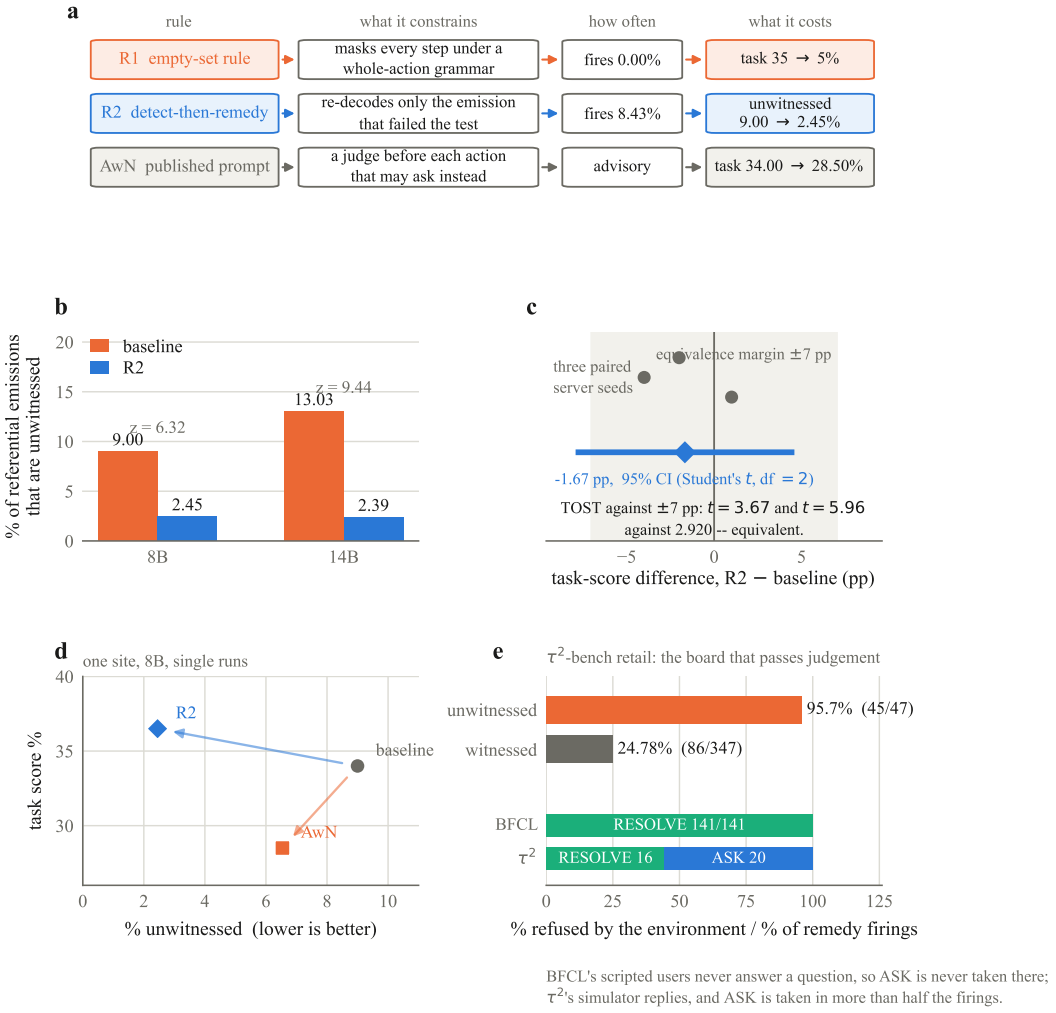


Figure 2: **Moving the trigger removes most of the behaviour at no detectable task cost; instructing the model to do the same thing does not.** **(a)** The three arms as interventions — what each constrains, how often it acts, and what it costs. **(b)** Unwitnessed referential emission at both model sizes, baseline against R2. The reduction is larger at the larger model, where the baseline rate is higher. **(c)** The task effect over three paired server seeds. The interval is Student's t at two degrees of freedom, not a normal quantile, and at $n = 3$ that is the difference between a half-width of 2.85 and one of 6.25; the wider one is correct and does not by itself bound the cost. The claim is therefore made by equivalence instead: a two one-sided test against a ± 7-point margin — this benchmark's own resolution floor — rejects both nulls. **(d)** Both quantities at once. R2 moves up and to the left; AwN, the published prompt-based method run with its own released prompt, moves down and only slightly left. **(e)** τ^2-BENCH retail, where the environment itself passes judgement on the emission, and the two branches the remedy takes on each board. BFCL's scripted users never answer a question, so ASK is never taken there; τ^2-BENCH's simulator replies, and ASK is taken in more than half the firings — which is the caveat we attached to the 141/141 figure, tested on the board that could test it.

B THE INSTRUMENT IN FULL

B.1 WITNESS LEVELS FORM A LADDER, AND THE LADDER IS ENFORCED

W^0 is *verbatim*: an exact span, or a case-sensitive, word-boundary-anchored occurrence in a source utterance. W^1 closes W^0 under five rules fixed in writing before measurement (unicode and whitespace normalisation, case folding, numeric and ISO-date reformatting, path and JSON-path projection, and composition from already-witnessed components). W^2 adds type-consistency and is the permissive upper bound. We enforce $W^0 \subseteq W^1 \subseteq W^2$ in code, because an unenforced ladder silently produced a non-monotone table during development.

B.2 WITNESSES INCLUDE TOOL RETURNS, SO THE ENVIRONMENT MUST ACTUALLY RUN

The support order is: user turns, then tool returns, then tool documentation the agent read, then schema enumerations. Measuring without executing the environment omits an entire source and inflates the result; every number below comes from live execution inside the benchmark's own harness.

B.3 THE FIVE W^1 RULES, FIXED BEFORE MEASUREMENT

W^1 is the closure of W^0 under the five rules below. They were fixed in writing before any measurement, which is the only thing that lets us report the closure level as a design axis rather than as something tuned to a result. Each is stated as implemented, not paraphrased.

1. **Unicode, whitespace, case.** Apply NFKC normalisation, collapse every run of whitespace to a single space, strip, and case-fold.
2. **Numbers and ISO dates.** A value matching a signed decimal, with or without thousands separators, is reformatted by removing the separators and parsing as a number; a value matching `YYYY-M-D` or `YYYY/M/D` is zero-padded to `YYYY-MM-DD`. Nothing else is coerced.
3. **Path projection.** From a value containing `/`, admit each non-empty component and the final component; from a value containing a dot and not beginning with one, admit the portion before the last dot. This is what lets a file name witness a path that was seen only in full.
4. **Schema-enum synonyms.** Coerce a value to an enumeration member only when the schema itself lists the synonym. No lexical similarity, no embedding distance, and no model is consulted; an unlisted near-miss stays unwitnessed.
5. **Composition.** Join two *already-witnessed* components with a `/` to admit the composite. Components that already contain a separator are excluded, and the pool is capped, because the rule is quadratic and an uncapped version would let a large witness set admit almost any path.

Rules 1 to 4 apply to each witnessed span independently; rule 5 applies once to the closure the first four produce, so composition can use their output but cannot compound on itself. Monotonicity $W^0 \subseteq W^1$ is enforced in code as well as by construction: W^1 accepts anything W^0 accepts before the closure is consulted at all. That check exists because adding the word-boundary phrase match to W^0 alone once made W^1's unwitnessed count come out *higher* than W^0's, which is impossible for a ladder and would have been visible to any reader of the table.

Leave-one-out: what each rule is worth. Disabling each rule in turn, over the 815 referential emissions of the baseline arm:

witness level	unwitnessed rate
W^0 (no closure)	6.01%
W^1 (all five rules)	5.89%
W^1 without r1 (unicode/whitespace/case)	5.89%
W^1 without r2 (number/date)	5.89%
W^1 without r3 (path projection)	5.89%
W^1 without r4 (enum synonym)	5.89%
W^1 without r5 (composition)	6.01%

Four of the five rules are worth exactly nothing on this board: removing any one of them individually leaves the rate unchanged to the emission. Only composition moves it, and only by 0.12 points — one emission in 815. Removing composition returns W^1 to W^0 exactly.

We report this because it is evidence against a component of our own design. The normalisation ladder was specified in writing before any measurement, on the reasoning that most of what a verbatim rule rejects is a correct value in a different surface form. On this benchmark that reasoning is wrong: the surface forms the rules were written to absorb barely occur. The corrected operating frontier (Section C.3) says the same thing independently — W^1 and W^0 are indistinguishable there — and two measurements agreeing is why we withdrew the dominance claim rather than treating either as noise.

What the ladder still earns is its role as a *bound*. W^2 blocks nothing, W^0 and W^1 block the same, and the span between them is therefore the whole range a definitional choice can move the result: small. A reader who suspects the headline rate is an artefact of how strictly "witnessed" was defined can read that span off the table instead of taking our word for it.

B.4 Two Branches, Not One

How much work each continuation does is a question we answered twice before noticing that the *question* was malformed. We report the sequence, because the final error is one a reader could otherwise repeat.

We first measured the benchmark's *reference solutions* and concluded that almost every missing value was recoverable from the environment, so asking a user was nearly never needed. That cannot answer the question by construction: a reference solution only ever names entities that exist, because it is the correct answer, whereas the rule must handle what a model writes when it is *wrong*. Gold trajectories are the wrong instrument for a question about error behaviour.

We then measured real rollouts and read the 1.00%/7.02% split above as the branch mass, which reversed the conclusion. That was the deeper mistake and it was in the *definition*. Those classes ask whether the value the model *wrote* exists; the continuation the rule should offer depends on whether a read-only call could supply the value it *should have* written. A fabricated identifier that exists nowhere says nothing about whether the correct identifier was sitting in the environment one lookup away. This benchmark cannot separate the two, because a correct per-slot value is defined only along the reference trajectory and the model leaves it almost immediately (Section D.1).

We therefore withdraw the ratio here instead of restating it, and recompute it in Section 4.3 on a benchmark that can decide it: one whose tasks state their required actions, and so fix the correct value for every slot independently of any trajectory. What survives unchanged in this section is the census itself, since the unwitnessed *rate* does not depend on the distinction, and the guarantee that both continuations stay reachable (Proposition 1) — a property of the mechanism rather than a claim about which one fires more often.

C Additional Measurements

C.1 Restricted to failing episodes

The model scores 36.5% (73/200), so 127 episodes fail. Of those, 29.13% contain an unwitnessed emission, against a pre-registered floor of 5% — a factor of 5.8. The kill criterion does not fire. The

denominator here is the benchmark's own verdict; an earlier version of this number used a proxy (an episode containing an errored call) and we discarded it, because a pre-registered threshold stated over failing episodes has to be read against the grader's definition of failing.

Unwitnessed emissions do predict failure. This corrects a null we previously reported. Passed episodes contain an unwitnessed emission 14.29% of the time against 29.13% for failed ones: a difference of +14.84 percentage points, Fisher one-sided $p = 0.0135$. Per-emission rates separate the same way, 10.10% in failed episodes against 4.15% in passed ones. The contrast survives two independent tightenings. Restricted to unwitnessed *writes* — the subset that can move the graded end state at all — it is 25.98% against 12.86% ($p = 0.0221$). And it replicates in the prompt arm, a separate 200-episode run: 33.07% against 11.59%, $p = 0.00059$.

An earlier draft reported this same comparison as null and slightly backwards: -4.53 percentage points at $p = 0.78$. That was an instrument error rather than a finding. The audit's span matcher was corrected part-way through this project, and that draft's BFCL figures predated the correction while its τ^2-BENCH figures postdated it, so the two boards were being compared on two different instruments. Every number in this section now comes from the corrected matcher. We state the superseded values rather than quietly replacing them: a null that turns into a positive result under a change of instrument is exactly what a reader should be able to audit, and it cannot be audited if the earlier number is gone.

At 14B the contrast points the same way but does not reach significance — 33.90% of failed against 29.87% of passed, +4.03 percentage points, $p = 0.334$, with per-emission rates 11.52% against 6.77%. The sign is now stable across both models and both prompt arms; the significance is not, and we claim only what replicates.

The causal question stays open, and the association does not close it. The grader compares end state, so an unwitnessed *read* costs nothing and an unwitnessed write can still coincide with the reference value. A gap of this size is equally consistent with unwitnessed emissions causing failures and with their being a symptom of an episode that was already going wrong; no observational comparison, at any sample size, separates those. Settling it requires intervening at the emission, which is Section D.1. The association raises the stakes of that intervention rather than substituting for it, and the motivation for NOWIT does not rest on it either way: an unwitnessed write is unlicensed whether or not this checker notices.

Whether the remedy's two continuations divide the work as intended is a question we could not answer with this instrument, and we report the attempt rather than a number. See Section B.4.

Instruction alone does not move this behaviour: a prompt that states the rule in the system message leaves the unwitnessed rate statistically unchanged, and the published prompt-based method is run as a full arm in Section 5. See Section C.2.

A witness level is an operating point, and the normalisation ladder we specified turns out not to buy one: W^1 is indistinguishable from W^0, and W^2 never fires. See Section C.3.

C.2 TELLING THE MODEL THE RULE DOES NOT WORK

The obvious objection to any mechanism here is that a prompt would do. We pre-registered this as the single most likely way for the work to be unnecessary, and tested it before building anything.

We wrote the fairest and most dangerous competitor we could: a 1,159-character system prompt stating the no-witness rule in plain language, *both* ordered continuations (look it up first; ask only if no tool can return it), and the creation-target exemption. It is injected through the chat template's supported system slot and audited by the identical instrument.

The prompt moves the unwitnessed-emission rate from 8.03% to 8.16%: a difference of 0.13 percentage points against a standard error of 1.28, i.e. $z = 0.1$. That is noise, and it is noise three times over — the same null was measured at 0.17, 1.12 and 0.32 percentage points across three configurations. Telling a model the mechanism does not make the model follow it, which independently reproduces a published finding that an *offered* escape from fabrication goes unused at this scale.

The prompt does not move the task either. Under the benchmark's own verdict the two arms score *identically*: 36.5% in both. This is not a tie between two static runs — the prompt reshuffles which episodes pass, turning 19 failures into passes and 19 passes into failures, 38 discordant episodes in all, for a McNemar χ^2 of 0.03. It moves episodes around and nets exactly nothing. An earlier reading of ours claimed the prompt improved the task while leaving the behaviour untouched; that came from the proxy denominator and does not survive the grader's own verdict, so the decoupling story goes with it. What is left is simpler: at this scale the prompt changes neither.

The confound is eliminated rather than argued away. An earlier pairing of these arms hit the server's context limit, erroring more episodes in the prompt arm than in the baseline and so penalising the prompt arm for a reason unrelated to the hypothesis. Rather than argue that the bias ran in our favour, we re-ran both arms jointly at a larger context: 0 overflow episodes in *both*. Every number above is from that clean pair.

One figure in the clean pair looks bad for the prompt and we report it rather than bury it. The share of failed episodes carrying an unwitnessed emission is 33.07% in the prompt arm against 29.13% in the baseline. At these counts the combined standard error is 5.8 percentage points, giving $z = 0.68$. That is sample jitter, and it should not be used as evidence in either direction.

What this establishes, and what it does not. It does not show that the mechanism works: at this point nothing has been built. It shows that a prompt is not a *substitute* for one — which is the question that would have made the whole enterprise pointless.

C.3 What the Rule Would Block, and What That Costs

A rule that makes fabrication unreachable is only worth having if it blocks fabrications more often than it destroys correct values. That is measurable before building anything, by replaying the same emissions and asking, at each witness level, what the rule *would* have blocked and whether the blocked value was in fact wrong.

Correctness here is decided by *membership*: an emitted value counts as correct if it appears among the reference values for that (tool, parameter) anywhere in the episode. That is the right shape for this question and it deliberately avoids the positional alignment that made Section D.1 unanswerable. Its own failure mode is the mirror image — a value can be a reference value elsewhere and still be wrong here — so we measured that rather than assuming it: among emissions where an aligned reference call exists to check against, 7.4% ($11/149$) are correct by membership only. The remaining 75.3% have no aligned reference at all, which is the same off-trajectory limit as before and is reported separately rather than folded in. The direction matters more than the size: over-counting correctness moves blocked fabrications into "blocked correct", so every precision below is a *lower* bound and every collateral figure an upper one. The bias runs against the rule we are arguing for.

Over 763 referential emissions with a fabrication base rate of 12.06%:

	blocked	precision	recall	correct blocked per catch
W^0	8.13%	51.61%	34.78%	0.94
W^1	7.60%	51.72%	32.61%	0.93
W^2	0%	—	0%	—

Three things follow, and they decide the design rather than decorate it.

W^1 and W^0 are *indistinguishable* here, and we withdraw an earlier claim that W^1 dominates. It blocks slightly less (7.60% against 8.13%) at effectively the same precision (51.72% against 51.61%) and the same price per catch (0.93 against 0.94). The prompt arm reproduces the pattern (6.38% against 6.65%).

A leave-one-out ablation says why. Dropping any single rule from W^1 changes the unwitnessed rate by nothing at all — 6.01% at W^0, 5.89% at W^1, and 5.89% again with any one of r1 (unicode/whitespace/case), r2 (number/date), r3 (path projection) or r4 (enum synonym) removed. Only rule composition moves it, and only by 0.12 points, which is one emission out of 815. On this board the five normalisation rules buy almost nothing. The earlier dominance was an artefact of an

uncorrected matcher whose W^0 the rules happened to be compensating for. We keep W^1 as the operating level because it was fixed in writing before measurement and cannot be worse, not because it is measurably better here.

W^2 is *vacuous*. Type-consistency blocks nothing at all — every emitted value is type-consistent with its slot. The permissive end of the ladder is not an operating point but a proof that the ladder is bounded, and it is why we report W^1 as the level a mechanism should run at.

Even at its best the rule is not free. At W^1 it still destroys about 0.93 correct values per fabrication prevented and falsely blocks 4.17% of emissions, while catching only 32.61% of fabrications. We state that plainly because it bounds what any mechanism built on this rule can claim: it is a trade, not a strict improvement, and whether the trade is worth making depends on the asymmetry between a wrong write and a blocked-but-correct one rather than on a benchmark score.

The prompt arm reproduces the frontier almost exactly — W^1 precision 48.94%, cost 1.04, false-block rate 3.72% — which is a second measurement of the same shape on independently generated trajectories, and the reason we treat the ordering of the levels as stable even though the individual percentages are not tight.

C.4 The instrument transfers without adjustment

Applied to the domain's 16 tools, the slot partition frozen for the previous benchmark labels 39 (tool, parameter) pairs: 21 referential, 18 generative, 0 unlabelled. Every referential hit comes from the frozen identifier rule. We changed nothing to make this fit, and we note one consequence we chose not to repair: a postal code is labelled generative here, though inventing one is plainly a fabrication. Editing the lexicon to suit a new domain is the exact post-hoc tuning the frozen rules exist to prevent, so the gap is recorded rather than closed.

C.5 What we do and do not claim from this

The 100.0% precision figure on τ^2-BENCH rests on 4 firings. That is a direction, not a rate, and we quote its denominator every time. Its recall is 10.0%: the setting that costs nothing also catches almost nothing, and the setting that catches 95.0% costs 4.26 correct emissions per catch. *There is no operating point on either board that is simultaneously cheap and thorough*, and we do not have a principled type function that would produce one. We state this as a result rather than a gap because it locates the open problem precisely: the empty-set rule is well-defined and its two continuations are constructible, but its value is inherited entirely from a component the formulation leaves unspecified — and, once specified, from the environment. That is the same conclusion Section 4.3 reaches about the population, arrived at independently at the level of the rule.

C.6 What follows for the rule, and what stops being a problem

A rule that fires on the value the model actually wrote, rather than on the emptiness of the set it was offered, tests the condition that arises instead of the one that does not. Section 5 measures it: same witness sets, same two continuations, trigger changed, and the task score returns to parity while unwitnessed emissions fall by 71%.

That change also dissolves the operating-point problem this section opened with. R1's firing rate is set by the type function, because it must decide which witnesses are type-matched for a slot before it can call the subset empty — and no fixed type function we can defend gives a stable rate. R2's trigger asks only whether the value the model wrote is witnessed *at all*, which needs no type function. The type function survives only inside the RESOLVE branch, where it selects which read-only calls to offer, and a poor choice there costs a less useful remedy rather than a wrong write. The component we could not specify turns out to have been load-bearing only for the trigger we were wrong to use.

C.7 Sampling does not escape it

In every trial of the failing condition the model emitted one distinct value across all 12 steps — at 0.7 as well as at 0. Raising the temperature changes which wrong value a trial locks onto, not whether it

locks on. The constrained distribution is sharp enough that sampling returns the same candidate as greedy decoding does.

D Additional Experimental Results

D.1 The Causal Ceiling

We designed this as the experiment the argument would rest on. Section 4.2 shows that unwitnessed emission is frequent on this board but not associated with failure, and an intervention is the only thing that could settle direction. It turned out not to be computable here, for a reason worth reporting, and the consequence it was meant to establish is instead measured directly on a second board (Section 4.3). We keep the design and the negative result because both are informative.

The design repairs at most one argument per episode and then lets the episode run on. In the treatment arm we replace the first REFERENTIAL-EXISTING argument that is both unwitnessed and different from the benchmark's reference value with that reference value. The negative control does exactly the same thing at the first such argument that is *witnessed* — so both arms correct one genuinely wrong value, and differ only in whether the no-witness rule would have flagged it. A third arm repairs nothing. If the treatment arm gains and the control does not, the rule selects the errors that matter; if the two are indistinguishable, the rule adds nothing over "repair any wrong argument" and the pre-registered response is to pivot.

Two implementation points decide whether this measures anything. The repair must be applied where the scored artifact is produced, not where calls are executed — the grader re-decodes the recorded response text, so a repair made at execution time would change behaviour while leaving the score untouched. And the benchmark shares one handler across all worker threads, so the per-episode witness state must be thread-local or episodes contaminate each other silently.

The result is that this experiment cannot be run here, and why is informative. An oracle needs a well-defined correct value, and on this benchmark one exists only where the trajectory still matches the reference. The reference solutions call the same tool repeatedly with different arguments — one episode moves `final_report.pdf` into a directory in its first turn and `previous_report.pdf` into the same directory in its fifth — so any lookup that searches beyond the aligned position returns an unrelated value and, worse, labels a *correct* call an error. We confirmed that failure mode on named episodes before it could reach a result.

Requiring alignment makes the oracle sound and leaves almost nothing to measure: 5 of 200 episodes (2.5%). Relaxing the alignment does not help. Skipping read-only calls, which preserves state equality and so keeps the oracle valid, gives 2.5%; realigning at each turn boundary, which does *not* preserve it, gives 1.0%. The pre-registered ceiling gate is therefore not evaluable on this board at this scale — a property of the benchmark rather than of our implementation, and one a reader can check from the reference solutions alone.

The second model does not rescue it. At 14B the eligible set is 4 of 200 episodes (2.0%) under strict alignment and 8 (4.0%) under the state-preserving relaxation. Larger, but still far too few to place an 8-point threshold on, and we will not report an effect estimated on eight episodes.

What the same measurement says instead. Classifying how each episode *first* departs from the reference is a result in its own right. The first departure is a *different tool* in 63.0% of episodes, and still 35.0% once read-only calls are ignored. A single differing argument accounts for 15.5%, and once restricted to the slots the rule governs, 2.5%. Only 7.5% of episodes reproduce the reference sequence outright.

This bounds how much of the episode-level association argument fabrication can carry: at this scale episodes are lost overwhelmingly to tool *selection*, so fabrication is one contributor among others rather than the mechanism of failure. Two further checks say that reading is not an artefact of how we sliced the data. When exactly one argument differs, that argument is GENERATIVE 83.3% of the time, REFERENTIAL-NEW 2.8%, REFERENTIAL-EXISTING 13.9%, and *unlabelled* 0% — so the partition covers what it claims to, and single-argument errors sit mostly in free text the rule deliberately leaves alone. And stratifying by environment, every stratum with at least 8 episodes has

the unwitnessed share of *passed* episodes at or above that of *failed* ones (Fisher one-sided p from 0.945 to 1.000). We had expected the identifier-shaped environments to behave differently from the path-shaped filesystem. They do not.

One thing the second model does change: what the errors look like. The handful of eligible departures at 14B are of a different character from the 8B ones, which were mostly navigation — a `cd` or `cp` to the wrong path. At 14B half of them are invented *identifiers*: a message recipient given as `FRIEND_USER_ID`, `COUSIN123` or `JOHNLVY` where the environment holds `USR006`, `USR002`, `USR005`; a booking call sent with the literal string `access_token`; a ticket resolved by a plausible-looking number that belongs to no ticket. This is the behaviour the rule was written for, in its clearest form. We record it as an observation on 8 episodes and nothing more — but it told us where to measure next: environments whose referential arguments are stable identifiers rather than positions in a filesystem, and a grader that scores the action rather than only the end state. That is Section 4.3, and it is where the consequence this section could not establish is finally measured.

Scale was the other candidate explanation, and it is ruled out. If tool selection improved with parameters while argument errors persisted, the ceiling experiment would become evaluable at the next size up. It does not. At 14B the first departure is still a wrong tool (59.5% strict, 34.0% state-aligned), exact reproduction of the reference *falls* from 7.5% to 6.0%, and the eligible set reaches only 4 episodes. This is not a matter of the bigger model simply being worse: it scores *higher* on this column, 41.0% against 36.5%. Tool selection is still where the episodes are lost, at both sizes, so the ceiling experiment cannot test what happens once selection genuinely improves — the premise never holds at either scale.

D.2 WHERE R2 FIRES, AND WHAT IT CHOOSES

Of the constrained arm's steps, 1,532 are PASS — the model's own call accepted verbatim, nothing masked — 883 are turns ending in prose and untouched, 141 fire the remedy, and 2 failed to parse. That is 8.43% of call steps, against the baseline census's 8.65% of referential emissions unwitnessed: the rule fires on essentially the population the census identifies, which is what it was designed to do and which we checked rather than assumed. RESOLVE was chosen in 141 of 141 remedies, with both branches equally available at every firing — a grammar cannot rank its own alternatives. We resist reading that as the cheaper-first ordering emerging on its own, because this benchmark supplies a simpler explanation. Its user turns are *scripted*: an ASK emission is a turn carrying no tool call, so the harness advances to the next scripted turn and nobody answers the question, while RESOLVE is a read-only call that actually executes and returns a value. A model that has learned which of the two does something here would choose RESOLVE every time without preferring it on cost. The count is a fact about the arm; separating the two explanations needs a board with a responsive user, and until then the ASK branch's value is not something this benchmark can show.

D.3 AT 14B, AND WHAT THE BENCHMARK'S OWN VARIANCE PERMITS US TO SAY

The census finds the 14B both better at the task and less witnessed than the 8B, so it is the size at which a rule that only rescues weak models would fail to show anything. Both arms below ran on the same site and server class:

	baseline	R2
task score	35.00%	37.00%
unwitnessed per referential emission	13.03%	2.39%
unwitnessed state-mutating emissions	115	23

The reduction is *larger* at 14B than at 8B (82% against 71% of unwitnessed emissions, 80% against 67% of writes), and the task direction is the same.

D.4 THE REMEDY, AND A CAVEAT WE COULD FINALLY TEST

The rule inspected 371 agent steps and fired on 36 of them (9.7%). The values it caught are visibly invented — `#W0000000`, `credit_card_0000000` — including one attempt to call a state-mutating tool with four fabricated identifiers at once.

Of the 36 firings, 16 took RESOLVE and 20 took ASK. On BFCL RESOLVE was taken in every one of 141 firings, and we declined to read that as the ordering emerging on its own, because that benchmark's user turns are scripted and a question there is never answered. τ^2-BENCH has a user simulator that replies, and the ASK branch is used in more than half the firings. The caveat was the right one to attach, and this is the board that could test it.

D.5 RUN-TO-RUN VARIANCE, AND WHY WE REPORT IT

We ran the same model on the same category at both sites. At 8B, with matched temperature and comparable concurrency, the two runs scored 35.00% and 34.00% — 1 point apart. At 14B the two runs are 41.00% and 35.00%, 6 points apart, but those two differ in thread count and launch configuration as well as in site, so that gap confounds the two and we do not read it as site variance; we state it because it happened and because a reader comparing our tables would otherwise find the discrepancy themselves.

Neither number is what the conclusion rests on. The paired standard error of about 3.7 points per column comes from the episode outcomes directly, independently of any cross-site comparison, and that is what makes a single column uninformative below roughly 7 points.

Three consequences, which we hold ourselves to. Cross-site numbers are not compared, and each arm's site is recorded. No single-run difference below about 7 points is treated as informative, including the ones that favour us. And every comparison the paper draws is paired and within-site. None of the published tables we set our rows beside reports a figure of this kind, which is worth knowing before their model orderings are read as settled.

D.6 TOOL-SPECIALIST CHECKPOINTS, MEASURED RATHER THAN QUOTED

Three open $\leq$13B checkpoints are trained specifically for tool calling and are the natural baselines on this column. We ran them on our harness rather than copying their published figures:

model	ours (v4, pinned harness)	as published
`ToolACE-2-8B`	30.00%	28.00%
`watt-tool-8B`	29.50%	24.00%
`BitAgent-8B`	29.00%	24.00%
`Qwen3-8B` (our baseline)	35.00%	—
`Qwen3-8B` + R2	36.00%	—

Two things stop this being a claim that our baseline is the better tool caller. It runs with **thinking enabled** and these three do not think at all, and that choice alone moves this column by roughly twenty points; the rows are not on the same axis. And the published column comes from a table that *states no* BFCL *version*, which we checked in its source rather than inferring, so the gap between the two columns is not a reproduction failure but an artefact of a version that cannot be pinned. What the left column contributes is the stated-version measurement the published one does not provide.

The three specialists also differ from each other by 30.00−29.00 points, well inside the resolution of a single 200-episode column, so neither our ordering of them nor the published one is established by any of these measurements.

D.7 WHAT THE REMEDY IS WORTH, SEPARATED FROM WHAT SEGMENTATION COSTS

R2 changes two things relative to the baseline: it generates in phases, and it applies a grammar when the remedy fires. Reporting only their sum would hide both, which is the error Lee (2026) documents

— there, stopping at the first line costs -20.0 points, adding the enum returns $+19.5$, and the pooled -0.5 reads as nothing happening. Two further arms separate ours, all on Miss Param:

arm	score	what it isolates
detect, then refuse	33.50%	the floor: no constructive continuation
baseline	35.00%	—
detect, no remedy	35.50%	Δ_{segment}: phased generation alone
R2	36.00%	Δ_{remedy}: the constraint itself

The ordering is the one the mechanism predicts — refusing is worse than doing nothing, and remedying is better — but we state plainly that *none of these gaps is resolvable at this sample size*. Each is a single 200-episode column, where the interval is about ± 7 points. What the decomposition establishes is that segmentation is not secretly doing the work, not that the remedy is worth $36.00 - 35.50$ points.

D.8 What R1's task score was really measuring

Read alone, 35.0% against 5.0% would say a witness rule costs thirty points. R2 shows it does not, and the failure profile says what R1 was paying for instead. 110 of 200 constrained episodes reach the step cap against 2 in the baseline, and 75.0% of the constrained arm's calls are exact repeats against 12.7%. The worst single case repeats `grep(file_name="directory")` thirty-three times in thirty-five calls: `directory` is an ordinary word from the conversation, therefore witnessed, standing in for a filename that is not in the witness set at all.

That is the livelock Section 4.4 predicts. A constrained model whose needed value is not on the menu must pick a wrong one; the tool fails; the witness set does not change; and at temperature 0.001 the next step is byte-identical. The empty-set rule cannot break the loop because the set is not empty. The score is not measuring the cost of enforcing a witness rule — it is measuring the cost of enforcing one with no branch for the case that actually arises.

The livelock survived two plumbing fixes before we accepted it belonged to the rule rather than to our code. See Section D.9.

Grammar compilation and the extra decode cost a few milliseconds per step and never failed. See Section D.10.

D.9 Three runs, and what each fix bought

Each of the three complete arms differs from the last by a repair to our implementation, not to the rule, and the sequence is worth stating because two of the three failures were invisible in every summary statistic we had.

	what changed	task score
v1	grammar covers the whole step	0.0%
v2	decide first, constrain the action	3.0%
v3	candidates agree with the audit	5.0%

v1 scored zero on every episode because a whole-step grammar gives ordinary prose no derivation, and prose is how an agent ends a turn; the decoder could derive `done` and chose it *zero* times in 3,824 steps. v2 ended turns but livelocked on harvester artefacts — the span `AAPL.` offered where `AAPL` was needed. v3 fixes the artefacts and the livelock persists, which is how we know the remaining 110 force-terminations are the rule and not the plumbing.

D.10 Cost of the machinery

Grammar compilation averages 3.1 ms per step over 3,824 steps, on grammars averaging 17.3 kB with 42.5 candidates per referential slot, and 0 grammars failed to compile. Mean latency per emission is *lower* under the constraint (11,525 ms against 12,038 ms) despite costing two or three generation

calls per step, because a grammar-bounded action cannot run away: the baseline hits its token ceiling on emissions the constrained arm terminates in a few dozen tokens.

E Further Related Work

E.1 Provenance and Argument-Level Validity

A parallel line reasons about provenance at the granularity of the argument rather than the call. Argument-level provenance can be monitored at runtime to localise enforcement Fan et al. (2026); preconditions and effects can be induced from tool schemas and documentation to filter the tool list before generation Babu & Iyer (2026); evidence can be organised into a structured ledger and claims verified against it Du et al. (2026); risk can be controlled post hoc with role-stratified conformal calibration Rahman et al. (2026); and the question of which runtime properties are enforceable at all has been given a formal treatment Ray (2026b).

The discriminating question across this cluster is not *when* it acts but what it can still leave to the model. These methods constrain which *tools* may be offered, or they detect a bad *call* and then hand control back — flagging it, calibrating it, or letting the model try again unconstrained. Detection is where they stop; the repair is advisory.

NoWit is the same detection followed by a remedy the decoder cannot decline. When the value fails the provenance test, the continuation is re-decoded under a grammar in which that value has no derivation at all, so the fabricated argument is not discouraged, it is *unreachable*. Both of our rules share that property and differ only in what makes it fire, which is why this paper is about the trigger. We note explicitly that R2, the rule the measurement selects, inspects a completed emission — it is not a during-generation constraint, and nothing here claims that constraining mid-emission is what makes the difference. The formal statement is correspondingly modest: the enforceability theory above already occupies the general question, so we claim a placement result inside it rather than a new theory of enforcement.

E.2 What Defines Evaluation Here

Our experiment design inherits its obligations from the boards that define this subfield. Multi-turn tool calling is scored by a harness that re-executes decoded call strings against a reference environment state Patil et al. (2025); the agent–user setting with a simulated user and programmatic grading comes from Yao et al. (2024) and its dual-control successor Barres et al. (2025); and the decision of whether to call, ask or decline is graded directly as a four-way choice by Ross et al. (2025). The strongest published results on the column we report are reinforcement-learning post-training Lin et al. (2026); Zhang et al. (2026b), alongside tool-specialist checkpoints Team-ACE (2026); watt-ai (2026); BitAgent (2026); we run the checkpoints and cite the post-training results as context, since neither releases a checkpoint we could evaluate.

Three properties of these boards shape what we may claim, and we state them because they are easy to get wrong. The multi-turn column that isolates missing parameters is scored at *conversation* level, after the missing information is supplied, and its abstention check was removed years before the version we use — so it does not reward asking and we never describe it as an abstention metric. Version and harness commit must be pinned, because the same untrained model differs across benchmark versions, and model-variant drift between two releases of the same family exceeds any method effect we could plausibly report. And one of the boards above is judged by a language model on the large majority of its tasks in its default configuration, so we exclude that split rather than silently reconfigure it into a number that is not comparable with anything published.

Finally, a control the literature obliges us to run: swapping only the system prompt is reported to move multi-turn accuracy by double digits, an effect its authors call comparable to or larger than gains usually attributed to methods Liu et al. (2026). Any results table here without a strong-prompt arm would be uninterpretable, so Section 5 contains one, and it varies instruction *content* rather than surface format — the benchmark's own report finds format variation inert.

F Further Limitations

F.1 One tool can dominate the aggregate

In our data a single authentication tool contributed a large share of unwitnessed emissions, at a much higher internal rate than every other tool. Credentials cannot appear in context by construction, so that may be a different phenomenon from inventing an order number. We therefore report the census per tool and with and without that tool, rather than a single aggregate that one tool could carry.

F.2 We do not know how the two continuations divide the work

An earlier draft claimed a ratio and we have withdrawn it. The classes it divided are defined over the value the model *wrote*, while the applicable continuation depends on the value it *should have* written (Section B.4); this benchmark cannot separate those, so the honest position is that we have not measured it. Nothing else rests on it — the census, the prompt null and the cost frontier are each independent of the split — but a reader wanting to know whether RESOLVE or ASK does more work should not take a number from this paper.

F.3 The second board is one domain, one model, and refusal is not failure

Section 4.3 rests on 114 simulations of a single retail domain with the same 8B model, under an evaluator we reconfigured to be deterministic — so its reward is reproducible but is *not* that benchmark's published reward and must not be compared with published numbers for it. Two further limits matter more than the sample size. The table counts calls the environment *refused*, which is not the same as tasks the agent failed: a refused call can be retried, and an agent that recovers loses turns rather than the task. And a perfect refusal rate is a property of an environment that validates identifiers; we should expect it to fall wherever identifiers are guessable or partially validated, which is precisely the regime the other board occupies.

F.4 Monotone witnesses

W_t never retracts, so an agent whose context is fragmented or rewritten retains credit for spans it can no longer see. An adversary who can restructure context can therefore manufacture a witness.

F.5 Membership is not truth

A witnessed value can still be the wrong one. The rule removes values that nothing licenses; it does not select among licensed values.

F.6 Checker-relative neutrality

The RESOLVE audit establishes that read-only calls leave the checker's view of state unchanged (Section 3.3). Attributes the checker ignores may still change.

F.7 One serving configuration

All numbers come from one inference stack. The context limit is no longer a confound: both arms were re-run jointly at a larger limit with 0 overflow episodes in either (Section C.2). What remains untested is whether a different stack or context budget would move the census, and we claim nothing about that.

F.8 Published comparanda disagree with each other

Untrained baselines for the same model family differ across sources by up to tens of points on this metric, and one widely cited figure turns out to be an aggregate rather than the per-category value it is often quoted as. We therefore run our own baseline as the reproduction anchor and treat published rows as context, naming the family and mode for each.

F.9 Scale

The census is measured at two sizes, 8B and 14B, and the behaviour does not shrink between them (Section 4.2). That is a check against the obvious dismissal, not a scaling study: two points in one family, both under 15B. The larger one scores +4.5 points *higher* on this column and is unwitnessed +1.74 points more often, which is the direction that makes the check worth reporting — but two points under 15B cannot tell us what happens once tool selection genuinely improves. Whether the rule is needed at frontier scale is untested.

G Reproduction, Logging and Compute

G.1 Reproducing the Zero-GPU Half Without a GPU

Every measurement in Section 4 that does not require model inference — the slot partition, the witness ladder, the gold-trajectory audit, and the re-scoring of a logged rollout — runs on a laptop with no GPU, no cluster, and without installing the benchmark package. The audits read the benchmark's raw JSON directly, so a reader can point them at a plain directory fetched from the pinned upstream commit:

```
B=https://raw.githubusercontent.com/ShishirPatil/gorilla/6ea5797/\
berkeley-function-call-leaderboard/bfcl_eval/data
curl -O $B/BFCL_v4_multi_turn_miss_param.json      # 402,974 bytes
python scripts/gold_audit.py --data-dir ./bfcl_data --out runs/gold_audit
```

This matters beyond convenience: the benchmark package depends on a numerical library with no wheel for current Python, so the package route fails on a stock recent interpreter while the audit itself only ever reads JSON.

G.2 Every Run Logs Its Raw Replies

Each run writes flat metrics, per-emission rows, per-episode rows, a config snapshot, and the model's *raw replies*. The last is what allows the emission census to be re-scored offline when the partition or the matcher changes. We added it after a parser fix forced a full re-run that the logs would have made unnecessary, and it subsequently allowed four successive re-scorings at zero marginal cost.

G.3 Compute

8 GPU-hours on a single A100-40GB to the measurements reported here, never exceeding one quarter of the shared per-user allocation. No training of any kind. The mechanism's own arms add two complete 200-episode runs plus the two superseded implementations, all on a single A100 apiece; the zero-GPU analyses (Section 4.4, the ablation above, and the feasibility counterfactual) add none, since they replay recorded results.