VERIFIED DISPLAY FOR LANGUAGE INTERFACES TO OPTIMISATION-BASED CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Language interfaces can answer questions about recorded runs of optimisation-based controllers. Verifying the structured claims a language model proposes does not ensure that its final prose contains only those claims. Across six models, answers whose claims passed verification still carry a scorer-flagged addition in 21 to 28 percent of paired cases. We introduce a verified display layer. The model proposes typed claims, deterministic checks evaluate them against the recorded solve, and a closed renderer turns only accepted claims into text that decodes back to those claims. In a paired experiment that changes only the displayed text, rendering removes every scorer-flagged addition and introduces no reverse transition; the preregistered split reproduces this result. Human adjudication finds a smaller but persistent rate of false statements in model-written prose. With tool calling, the layer trades coverage for lower displayed risk: useful coverage rises in dispatch and falls in HVAC, and the confidence interval of the pooled change includes zero. The guarantee covers declared predicates over the recorded committed solve, rather than controller correctness, plant behaviour, or causal explanation.

1 INTRODUCTION

Operators may inspect a recorded controller solve by asking natural-language questions after a run. A review of 66 studies of language models for HVAC operation found none in sustained deployment and names advisory interfaces around physics-based controllers as a near-term use (Neubauer et al., 2026). At each control step, the controller solves an optimisation problem, plans future actions, and applies only the first. We call that solve the *committed solve*; its record holds the applied action and the problem that produced it. The record can support narrow statements, such as the value of a control variable at a timestamp or the status of a constraint. A language model may turn such information into a broader statement about why the controller acted or what happened in the plant. These broader statements need not follow from the record. Verification can fail at a second point: the model-written answer may add a claim that was never checked.

Two answers can use the same accepted record evidence while making different claims. "*At 17:30, the CO_2 constraint (lambda_CO2) was slack*" reports a status of the recorded constraint: the limit was not reached and its normalised multiplier was at or below the declared threshold. "*The CO_2 limit forced the controller to raise the damper*" adds a causal account that the same record does not establish. In our building-domain audit, 31 percent of answers had proposed claims that passed all gates but still contained a scorer-flagged statement in the prose shown to the user. The accepted claim and the displayed sentence are separate objects, and free-form generation can change what the user is told. Figure 1 shows one such answer beside its rendered counterpart.

This paper makes three contributions. First, it identifies and measures post-verification additions: claims can pass deterministic checks while the model-written answer adds content that was never checked. Second, it defines a display path in which every factual sentence decodes to an accepted typed claim about the recorded solve. Third, it evaluates the resulting risk–coverage trade-off across six models, two proposal pipelines and three control domains, with a preregistered confirmation, component ablations and human adjudication. We use deterministic template realisation to enforce a display-level trust boundary and evaluate it under matched accepted claims.

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.

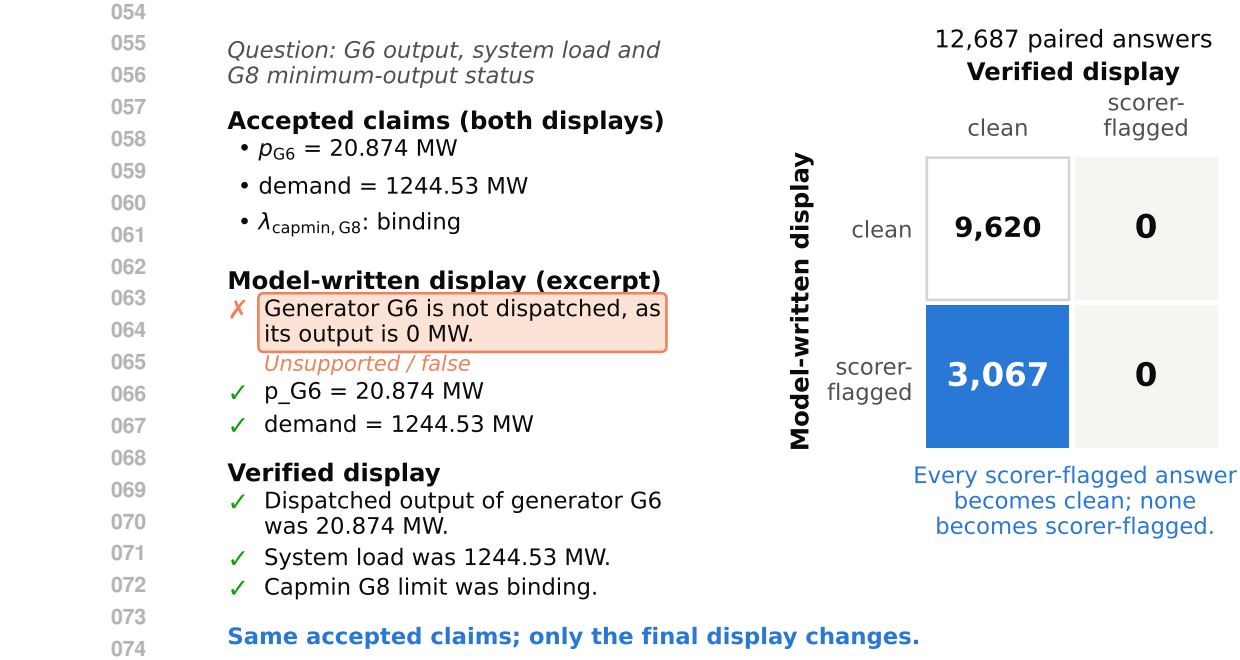

Figure 1: **Same accepted claims, different displayed text.** Left: the model-written answer adds the unsupported statement that generator G6 is not dispatched, although the accepted claim gives $p_{G6} = 20.874$ MW. The verified display renders only the three accepted claims, shown without their timestamps. Right: across 12,687 paired answers with matched accepted claims, rendering changes 3,067 scorer-flagged model-written answers to clean text and produces no transition from clean to scorer-flagged. Table 2 and Appendix C.3 report results by carrier, domain and model.

The interface refuses when the record cannot support the proposed claim. Because refusal can lower error by withholding difficult questions, we also report useful coverage: the share of all questions that receive a responsive unflagged answer or a correct refusal.

The deployed claim language contains numeric readings and constraint-status statements. A constraint is labelled *binding* only when the recorded solution is at its limit, its normalised multiplier is materially nonzero, and a trajectory-specific regularity check permits that reading.

We evaluate zone HVAC, economic dispatch, and longitudinal vehicle control on 60 locked test trajectories. A paired experiment fixes the generator, verifier, retry policy, and accepted claim sets, then changes only the displayed text. Across 12,687 paired answers from two generation pipelines and six language models, rendered displays remove all scorer-flagged additions and introduce no flagged reverse transitions; this is the primary result (Section 5). The comparison of the full layer with plain tool calling is secondary and shows a risk–coverage trade-off that differs by domain.

2 Related Work

Prior systems verify several kinds of intermediate objects. Some translate questions into logic or SMT constraints (Pan et al., 2023; Olausson et al., 2023; Ye et al., 2023), delegate computation to interpreters (Chen et al., 2022; Cosler et al., 2023), or verify plans (Lee et al., 2025). Claims are checked against tables and graphs (Chen et al., 2019; Kim et al., 2023) or cited below the sentence level (Cao & Wang, 2024), and explanations, reasoning traces and certificates are checked in their own right (Rodriguez et al., 2026; Kim et al., 2026; Zhong et al., 2026; Yang, 2026; Rautenberg & Schippkus, 2026; Koomullil, 2026; Rahman et al., 2026; Kwok et al., 2026). Constrained decoding and tool interfaces restrict syntax, schemas and calls (Scholak et al., 2021; Willard & Louf, 2023; Geng et al., 2023; Park et al., 2024; Zabolotnii, 2026; Schick et al., 2023; Patil et al., 2024; Qin et al., 2024). Learned faithfulness scores judge the answer afterwards (Zhang et al., 2023).

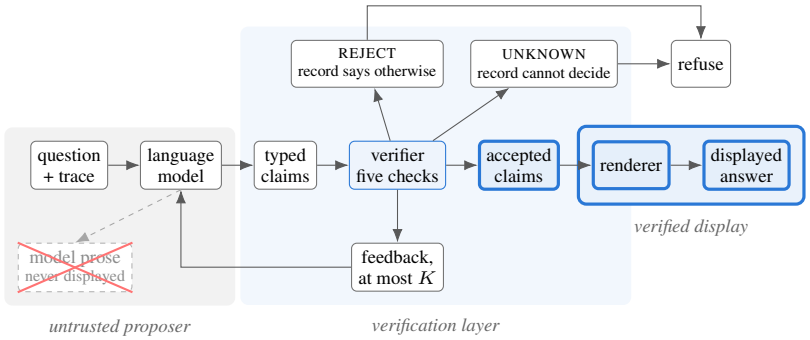

Figure 2: An untrusted model proposes typed claims, deterministic gates decide them, and the renderer converts accepted claims into the only text shown. The residual-only pair takes the same path without the multiplier.

Selective and conformal generation filters claims or abstains when confidence is low (Geifman & El-Yaniv, 2017; Kamath et al., 2020; Mohri & Hashimoto, 2024; Quach et al., 2024; Yadkori et al., 2024; Ren et al., 2023; Lee et al., 2024). Conformal filtering over a deducibility graph keeps retained claims coherent with their premises (Rubin-Toles et al., 2025). Our refusals instead arise when the recorded solve cannot decide a claim.

On the control side, runtime assurance verifies actions rather than statements, through shielding, simplex-style fallback, reachability and safety filters (Alshiekh et al., 2018; Jansen et al., 2020; Carr et al., 2023; Phan et al., 2020; Hobbs et al., 2023; Hsu et al., 2023; Ivanov et al., 2019). Behavioural verification checks the optimisation model itself (Lian et al., 2026). Explainable optimisation derives reasons, counterfactuals, sensitivities, arguments and causal readings of duals (Korikov et al., 2021; Kurtz et al., 2026; Forel et al., 2023; Aigner et al., 2024; Goerigk & Hartisch, 2023; Čyras et al., 2019; Allamaa et al., 2025; Siang & O'Connor, 2025). OptiChat (Chen et al., 2026) and hierarchical causal abduction for nonlinear MPC (Naagarajan et al., 2026) have a language model write the explanation. Our status labels stop at the recorded multiplier and make no causal claim.

Display-stage methods constrain part of the final text. Proof-carrying numbers bind displayed numeric spans to structured claims (Solatorio, 2025). Symbolic references render record values inside model-written text (Hennigen et al., 2023), and a structured health-text system lets code decide which facts may enter prose (Cheng et al., 2026). Closed grammars with parsers are standard in controlled languages and template generation (Kuhn, 2014; Reiter & Dale, 1997; Gatt & Krahmer, 2018).

In most of the systems above, verification does not determine every factual sentence shown to the user. We render the whole answer from accepted claims, so every displayed sentence decodes to one of them. Table 8 and Appendix B.2 compare the approaches in detail.

3 The Verified Display Interface

Figure 2 shows the path from a question and a recorded trajectory to a displayed answer. The trajectory contains one record for each controller solve. A record stores the state, control, and disturbance at that instant. For each inequality constraint j, it stores the residual g_j, the solver-reported multiplier λ_j, and a positive declared scale s_j. We write $x_\tau(e)$ for the recorded value of quantity e at timestamp τ. The domain graph defines the quantities that can be named, their canonical units, readable names, and declared signed relations. The language model proposes typed claims over this vocabulary, deterministic checks decide each claim against the record, and the renderer turns only accepted claims into the displayed answer.

3.1 Typed Claims and Recorded Evidence

The model proposes atomic typed claims. A trace-relative numeric claim is $c = (e, r, v, u, \tau)$, where e is a quantity, $r \in \{=, \leq, \geq, <, >\}$ is a relation, v is a value, u is a unit, and τ is a timestamp.

A status claim names an inequality constraint and assigns it the label *binding* or *slack*. Equality constraints receive no status label (Appendix A). The deployed interface uses numeric and status claims about the committed solve. It can also evaluate declared model-relative relations, such as whether one quantity *increases* or *decreases* with another, against the signed domain graph; results for this optional claim type appear in Appendix C.7.

A *binding* status states that the constraint is at its limit and has a materially nonzero normalised multiplier under the declared threshold. A *slack* status states that the constraint is below its limit and its normalised multiplier is at or below the declared threshold. These are multiplier-qualified labels, not causal explanations: they describe the recorded optimisation result. In a nonconvex domain the recorded solve is one local solution, and another solver or start can return a different solution with a different label. A status label is therefore a property of the recorded local solution; Appendix D.1 measures how often a second solver disagrees.

Residual-only statements say only whether a constraint is *at limit*, $|\hat{g}_j| \leq \varepsilon_g$, or *clear of its limit*, $\hat{g}_j \leq -\varepsilon_g$. They do not depend on the multiplier. Appendix D.4 discusses the wording of all status labels.

3.2 Claim verification

Each proposed claim passes five checks: provenance, coverage, unit, dependency when applicable, and an evidence oracle. Table 19 follows one claim from the raw record to its decoded display. A check returns REJECT when the record contradicts the claim and UNKNOWN when the available record cannot decide it. The provenance check first determines whether the record at the queried timestamp can support a record-relative decision. It requires a solver status from the deployment's declared set and primal feasibility within the declared residual tolerance. If either condition fails, the verifier returns UNKNOWN, because a stored multiplier from an incomplete or infeasible solve is not suitable evidence for a status statement. The declared solver-status set is deployment-specific because solver packages use different labels for a converged solution.

The coverage check requires every named quantity to appear in the domain graph and the requested timestamp to appear in the trajectory. The unit check accepts only the declared canonical unit for that quantity. The deployed interface refuses a mismatched unit instead of converting it, because conversion would require additional assumptions about how the question was interpreted. A deployment that supports conversion can declare it as part of its semantics.

Model-relative claims additionally pass a dependency check: the requested signed relation must appear in the domain graph. The evidence oracle then evaluates the claim against the recorded values. It uses the declared tolerance for equality claims and the literal comparison shown to the user for one-sided claims. A record of 10.096 MW therefore does not support the statement "at most 10.0 MW", although 10.096 lies within the 0.1 MW equality tolerance. Appendix A defines both predicates. A claim reaches the renderer only when every applicable check accepts it.

Raw multipliers depend on the numerical scaling of the objective and constraint formulation, so their magnitudes are not directly comparable across constraints or domains. Each constraint therefore has a declared magnitude s_j for the quantity it bounds, and each domain has an objective magnitude s_f. The verifier uses the normalised multiplier $\hat{\lambda}_j = \lambda_j s_j / s_f$ and normalised residual $\hat{g}_j = g_j / s_j$. Both quantities remain unchanged under positive rescaling of the corresponding constraint or objective. The normalised multiplier is dimensionless. Locally, it represents the change in the objective, relative to s_f, associated with relaxing the constraint by its declared magnitude s_j. Appendix A gives the definitions and the scale-invariance result.

Let θ denote the normalised-multiplier threshold, m the abstention margin, and ε_g the residual tolerance. The verifier accepts *binding* when $\hat{\lambda}_j > \theta + m$ and $|\hat{g}_j| \leq \varepsilon_g$, so the constraint is both at its limit and materially priced. It accepts *slack* when $\hat{\lambda}_j \leq \theta - m$ and $\hat{g}_j \leq -\varepsilon_g$, so the constraint is away from its limit and unpriced. All remaining cases return UNKNOWN. We use $m = 0$, so status abstention arises from the residual condition or the regularity check. The same rules define the verifier outcomes, benchmark labels, and tool responses. A status claim also requires the active-constraint Jacobian to pass the LICQ check. For the fixed recorded formulation and declared scaling, LICQ ensures a unique multiplier vector. Because runtime logs do not contain constraint gradients, we compute this check offline for every recorded instant and bind the resulting mask to its

trajectory. A mask that does not match the record causes refusal. Appendix A.1 reports this audit, and Appendix A.3 explains the conservative treatment of slack claims.

3.3 VERIFIED RENDERING

The renderer maps each accepted claim in C to a fixed sentence form and appends a fixed scope statement. It receives no model-written prose and can therefore express only the entities, relations, values, units, and timestamps present in C. Before rendering, each accepted equality takes the recorded value the oracle compared it against, so a value inside the tolerance is never displayed in place of the record. A repeated claim is shown once. The decoder reconstructs the committed claim sequence from the rendered text. Numeric values use a representation that the decoder recovers exactly; if decoding fails, the answer is not displayed. The renderer–decoder round trip is defined over a formal normal form, so it establishes that the displayed text decodes to the accepted claims.

A verified status claim can render as "At 10:05, the comfort constraint (lambda_T) was binding." The readable name comes from the domain declaration, and the identifier preserves the link to the recorded constraint used by the verifier and decoder. Names and wording are fixed for a campaign because changing either changes the displayed text. A model-relative relation is rendered as a statement about the declared model graph. The renderer has no sentence form for causal accounts such as "the controller kept the room comfortable because solar radiation was low." Such content cannot appear in a displayed answer.

3.4 RETRIES AND REFUSAL

After a failed claim set, the proposer receives the gate outcome and may try again up to K times. Each attempt is verified independently. The renderer runs only after every claim in a proposed set has been accepted. If no claim set passes within the retry budget, the interface returns a fixed refusal that identifies the relevant failure category; Appendix C.9 reports their counts. Retries can increase coverage by giving the model another opportunity to propose supported claims, but they do not alter the conditions required for a displayed claim. Retrying cannot make an out-of-vocabulary claim acceptable, so a question that requires such a claim cannot receive an adequate verified answer under the deployed vocabulary (technical supplement, Proposition 5.4).

4 WHAT THE INTERFACE GUARANTEES

Assume correct record fields and declarations, a matching regularity artifact for status claims, a sound verifier, and a correct renderer–decoder pair. Then, under the closed-grammar assumption, every displayed proposition corresponds to an accepted claim, and each accepted claim satisfies its declared predicate on the recorded committed solve. The displayed text is the rendering of the committed claim sequence or a fixed refusal, so model-written prose cannot add a proposition after verification. Appendix A.4 states these properties with their assumptions.

The guarantee is relative to the record. It does not establish that the log is authentic, that the controller is correct, or that the operator reads a statement as intended. Normalising by the declared scales makes the status rule invariant to positive rescaling of a constraint or the objective. When the LICQ check fails, several dual solutions may assign different multipliers to the same solve, so the status label is withheld. The loop then refuses rather than substituting a residual-only statement (Appendix A.3).

5 EVALUATION AS SELECTIVE GENERATION

5.1 SETUP

Table 1 summarises the evaluation. A trajectory is one independent controller run and is the unit of resampling, because questions from the same trajectory are correlated. Thresholds, tolerances and policy settings are selected on the development split by minimising a prespecified loss.

Varying the status threshold, the margin or the solver setting moves displayed status labels only to UNKNOWN, except at two HVAC instants where a re-solve reaches a different local optimum (Ap-

Table 1: Evaluation design.

Domains	zone HVAC, economic dispatch, longitudinal vehicle control (simulated)
Trajectories	10 development and 20 locked test trajectories per domain
Test questions	1,456 from a factorial benchmark
Models	six open-weight models; primary model LLAMA 3.1 8B
Proposal pipelines	direct claim generation; tool calling
Primary endpoints	paired change in flagged rate; useful coverage against plain tool calling
Resampling unit	trajectory

pendix A.2). The LICQ gate withholds status labels where the active constraints are rank-deficient, which is common in HVAC and absent in dispatch (Appendix A.1).

Questions come from a factorial benchmark design, and each question specifies the claims required for an adequate answer (Appendix B.3). It holds out controller configurations, trajectories, and disturbance windows. Transfer to a public simulator is tested only on the BOPTEST building; transfer to an unseen question source or claim schema is not tested. Every method receives the same question and trajectory identifier. Appendix B.4 specifies the record fields and interaction budget available to each method. The symbolic renderer, an oracle skyline, receives gold claim requirements.

Let A indicate that a method displays an answer, Φ that the answer contains no statement the scorer marks as false against the record, and Adq that the answer supplies every entity and relation required by the gold claim set. Coverage is $\Pr(A = 1)$, with parsing failures counted as refusals. Selective risk is $\Pr(\Phi = 0 \mid A = 1)$. Adequacy is evaluated only for questions with a gold claim set. We distinguish false, unsupported, and unverifiable content throughout the analysis (Appendix C.10).

Useful coverage is $U(\Pi) = \Pr(A = 1, \Phi = 1, \mathrm{Adq} = 1) + \Pr(\text{unanswerable}, A = 0)$. Appendix B.1 defines the remaining terms. The second term credits a refusal when the record cannot answer the question. Useful coverage is a benchmark proxy: it scores answers against gold claims, not against what an operator can use. Neither metric conditions on displaying an answer, so a method cannot improve its result merely by refusing difficult cases. Appendix C.9 assigns every test question to exactly one outcome.

We prespecified two primary endpoints: the paired change in scorer-flagged rate when only the displayed text changes, and useful coverage relative to plain tool calling. All other comparisons are secondary and use the same family-wide multiplicity correction. The two primary endpoints are confirmatory through a preregistered split on fresh trajectories; all other numbers are exploratory.

Two verifier rules changed after locked-split outputs were inspected (Appendix C.13). Every main-text number comes from code after these changes. The locked campaign predates the cross-solver mask, so its tables report the system with the mask off; the confirmatory split runs with it on.

All methods run on every model except the two tool-calling conditions on the 70B model, so those comparisons use five models (Appendix C.11). We macro-average point estimates across the three domains, so each domain contributes equally.

5.2 PRIMARY RESULT: CONTROLLED DISPLAY COMPARISON

We isolate the effect of rendering by fixing the generator, verifier, retry policy, generations, and accepted claim sets, then changing only the text shown to the user. Coverage and adequacy are therefore identical by construction, and any change in scorer-flagged content comes from the display step (Appendix C.2). For direct claim generation, 1,477 of 6,909 paired answers are flagged as prose and clean when rendered. For tool calling, 1,590 of 5,778 paired answers are flagged in prose and clean when rendered. No answer was clean as prose but flagged after rendering. The paired reduction in flagged rate is 0.214 $[0.198, 0.231]$ for direct claim generation and 0.275 $[0.252, 0.300]$ for tool calling (Table 2).

Every one of the 33 domain–model cells improves under rendering, and the largest one-sided upper bound on a cell-level reverse-transition rate is 0.024 (Appendix C.3). Four hosted frontier models show no reverse transition either (Appendix C.4). On the primary model, direct claim generation

has the same coverage (0.886) and adequacy (0.699) in both display conditions. Rendering reduces scorer-detected selective risk from 0.253 to zero observed and increases useful coverage from 0.497 to 0.620 (Table 3).

Table 2: Paired display transitions, test split, six models pooled per carrier. F→C counts answers flagged as prose and clean when rendered, C→F the reverse, and Diff their difference in flagged rate with a 95% trajectory-clustered interval. Every F→F count is 0.

Carrier	Pairs	F→C	C→F	C→C	Diff	Cells	Worst C→F
direct claim generation	6,909	1,477	0	5,432	0.214 [0.20,0.23]	18/18	0.016
tool calling	5,778	1,590	0	4,188	0.275 [0.25,0.30]	15/15	0.024

5.3 Secondary result: the risk–coverage trade-off with tool calling

Table 3: Primary model, locked test split of 1,456 questions, macro-averaged over three domains. Risk and adequacy condition on answering; useful coverage is over all questions. Small figures are 95% intervals; a zero risk is observed, not bounded. † The symbolic oracle skyline receives the gold claims; it is an upper bound, not a baseline.

Method	Answered	Flagged	Coverage	Risk	Adequacy	Useful
gated, model text	1,281	328	0.886	0.253	0.699	0.497 [0.47,0.52]
gated and rendered (ours)	1,281	0	0.886	0.000	0.699	0.620 [0.59,0.65]
tool calling	1,456	243	1.000	0.173	0.770	0.605 [0.59,0.63]
tool calling + verified display	1,164	0	0.813	0.000	0.802	0.639 [0.61,0.67]
gated + rendered, multipliers in feed	1,280	0	0.878	0.000	0.738	0.659 [0.63,0.68]
symbolic renderer (oracle skyline)†	1,370	0	0.942	0.000	1.000	1.000 [1.00,1.00]

We next apply the complete layer to tool calling, the always-answer baseline with the highest useful coverage on the primary model. This comparison changes verification, rendering, and retries together, and therefore evaluates the full interface on a common proposal carrier (Appendix B.4). Plain tool calling answers all questions, with selective risk 0.173. Tool calling with verified display answers 0.813 of questions and has zero observed scorer-flagged answers (Table 3). The multipliers-in-feed row also shows the proposer the oracle status labels (Appendix C.5).

Two estimands summarise useful coverage. The prespecified one pools all questions, and the macro-average in Table 3 gives each domain equal weight. For the pooled estimand, useful coverage changes by $+0.021$, with a 95% interval of $[-0.023, 0.062]$. The interval includes zero, so the primary endpoint is not met on the primary model.

The macro-average rises from 0.605 to 0.639. It is larger because it weights the three domains equally, whereas HVAC, where the layer loses, holds 41 percent of the questions. Split into its parts, useful answers give 0.619 against 0.605, and correct refusals of unanswerable questions, which plain tool calling always answers, add 0.020.

By domain, the difference is $+0.146$ in dispatch, interval $[0.118, 0.172]$; $+0.051$ in vehicle control, interval $[-0.006, 0.105]$; and -0.093 in HVAC, interval $[-0.164, -0.030]$. Withheld status labels account for part of the HVAC loss. A preregistered confirmatory split on 60 fresh trajectories shows no reverse transition, the same sign in each domain, and a pooled difference, over all questions, of 0.037, interval $[-0.002, 0.076]$ (Appendix C.12).

Across the five models that run tool calling, selective risk is lower with the layer in every domain–model cell. Table 11 gives the comparison pooled over all models. Figure 4 and Appendix C.2 report useful coverage for each domain and model. Zero observed flagged answers over 60 trajectories gives a one-sided 95% upper bound of 4.9 percent on the probability that a trajectory contains a flagged answer, under an equally weighted mixture of the three domains; this is not an answer-level or domain-specific bound (technical supplement, Proposition 7.2). Appendix C.8 breaks the comparison down by perturbation class. On public SEC filings and the public BOPTEST heat-pump

Table 4: Selective risk by component, primary model.

Verification	Closed rendering	Selective risk
no	no	0.395
no	yes	0.524
yes	no	0.253
yes	yes	0 observed

building, rendering again removes every scorer-flagged addition; the useful-coverage endpoint is met on SEC filings and not on BOPTEST (Appendices C.14 and C.15).

5.4 Component ablations

Table 4 separates the two components on the primary model. Rendering unchecked claims raises risk, because the renderer does not check record support. Verification with model-written prose lowers risk but leaves flagged statements in the displayed text.

Constraining decoding without verification has the same limitation. A constrained decoder that emits only well-formed claim blocks over declared entities and units answers every question and produces no unparsable output. Its selective risk is still 0.301, ranging from 0.057 in HVAC to 0.597 in dispatch. Its adequacy is 0.481 and useful coverage is 0.358. In this benchmark, syntactic well-formedness alone does not ensure record-supported output.

A policy in the manner of proof-carrying numbers keeps the prose and hides every unverified number. It lowers selective risk only from 0.237 to 0.095 for direct generation and from 0.265 to 0.095 for tool calling. False status, comparison and action statements contain no number to hide (Appendix C.6). Regenerating prose from accepted claims and checking it afterwards, or by reading it back, leaves useful coverage at 0.233 and 0.236, against 0.619 for the closed renderer in the same campaign (Appendix B.5).

The oracle skyline reaches adequacy 1.000 when given the required claims, so the renderer can produce adequate answers; the deployed gap includes question-to-claim proposal and commitment. The full ablations and span-level policies appear in Appendices C.5 and C.6.

5.5 Relational extension and expressivity boundary

The display guarantee is unchanged when the claim language includes declared model-relative relations, but useful coverage then depends on proposing the relation type the question requests. The deployed interface supports trace-relative numeric claims and multiplier-qualified constraint-status claims. Its base commitment policy accepts the first supported claim set and does not require that set to contain a model-relative relation when the question asks for one. Its adequacy is therefore zero on the 344 of 1,456 test questions whose gold answer requires a declared relation. The remaining limitation is question-to-claim alignment: the intent component must identify the required claim type, and the proposer must generate an accepted claim of that type.

Across six models, 0.834 of the 2,269 inadequate answers produced by the base deployed systems concern relational questions. The base claim language cannot express the relation these questions ask for. A typed-intent component, run as a separate campaign, conditions proposal on the required relation type. On the 344 relational questions it raises useful coverage to a macro-average of 0.441, and every selected claim still passes the verifier and renderer (Appendix C.7).

5.6 What the scorers measure

Two scorers measure faithfulness, and their results mean different things. For rendered answers, the structural scorer checks conformance to the closed grammar; factual interpretation is assessed separately in the blinded audit. In the blinded audit described below, the scorer agreed with the adjudicated label on every rendered answer. Human adjudication found no false statement among the 29 answered items of the two verified-display arms, a one-sided bound of 9.8 percent.

For prose, the span scorer is a measurement, and a noisy one. In a blinded audit of 120 outputs by two non-author control engineers, it agreed with the adjudicated label on 0.701 of prose answers. Against the audit labels (Appendix D.2), it has precision 0.409 and recall 0.562 on prose, and the adequacy scorer misses 25 of the 60 answers that readers judged inadequate. The audit is stratified by the scorer's verdict, so these rates do not transfer to all answers. On planted errors, the scorer attaches fewer numbers to a quantity and detects fewer errors in prose than in rendered text (Table 24). Its recall therefore understates prose errors, while its precision overstates them; Appendix D.3 reports further failures.

Reweighting rater labels within each verdict stratum of the 65 answered prose items gives a false-statement rate of 0.176 for direct-generation prose and 0.184 for tool-calling prose, against flagged rates of 0.214 and 0.275. Counting statements the record cannot settle raises these rates to 0.673 and 0.677. Reported prose-arm risk and useful-coverage values are therefore measurements under a noisy instrument.

An open-weight judge, calibrated on the same labels ($\kappa = 0.524$ for false statements), read 657 paired answers against their records (Appendix D.3). It found an error that neither scorer can see. By hand, 7 rendered answers display a proposed value that lies within the declared tolerance but misstates the record at the precision shown. The renderer described in Section 3 removes this error by displaying the recorded value. Re-rendering every stored answer this way changes no equality or bound verdict and no scorer decision, so the tables stand. Afterwards no rendered answer the judge flags is false on inspection. The calibrated paired effect of rendering on false statements is then 0.212 for direct generation and 0.114 for tool calling.

6 Trusted components and limitations

The trusted components are the record and domain declarations, the regularity artifact, verifier, renderer–decoder pair, parser, and commitment rule; the language model and evaluation scorer are outside this set. In the enumerated and property-based tests of prompt injection, infeasible records, multiplier rescaling, and renderer–decoder round trips, no unsupported display was observed. An independent verifier re-decides 133,463 claims and differs only in 42 refusal subtypes (Appendix D.1).

Status labels are a property of the recorded local solution. In HVAC, re-solves that reach a different local optimum contradict displayed status sentences at 2 instants of one trajectory, which neither the LICQ gate nor the locked-split mask catches (Appendix A.2). Locked-split results omit the cross-solver mask. An extended mask that also withholds these two instants changes every arm's macro-averaged useful coverage by at most 0.007 (Appendix C.12). The interface verifies individual claims, not the completeness or causal adequacy of an answer, and a displayed answer can be correct while omitting a relevant constraint or event. The prose flag rate does not fall consistently with model size (Appendix C.4). The evaluation therefore establishes correspondence with recorded simulated solves; operator comprehension and decision quality remain deployment questions.

7 Conclusion

A language model can add unsupported statements while verbalising claims that have already passed verification. In a paired intervention over 12,687 answers, rendering accepted claims through a closed grammar removed every scorer-flagged addition and introduced no reverse transition. Reweighting the scorer results with blinded human labels gives estimated false-statement rates of 0.176 for direct-generation prose and 0.184 for tool-calling prose. Under the closed grammar, the verified display makes accepted typed claims the only source of factual content in the displayed text. The guarantee remains relative to the recorded committed solve. These results identify post-verification generation as a separate failure surface and show how to close it for a declared claim language.

Ethics statement

This work studies language interfaces for inspecting completed optimisation-based controller runs. Displayed statements are generated from claims accepted against a supplied record under declared

checks. A central deployment risk is over-trust: a user may read a narrow statement about a recorded solve as a broader assurance, or may act incorrectly after a refusal. Restricting displayed text to a closed claim grammar and reporting refusal reasons limits the content that reaches the user.

The experiments use simulated controller trajectories, public disturbance data, and a synthesised vehicle trace; no personal data are used. Two non-author control engineers completed a blinded internal audit of 120 answers. They were unpaid colleagues in the authors' laboratory. A deployment would require authenticated logs, reviewed declarations, and human-factors evaluation.

REPRODUCIBILITY STATEMENT

The anonymised release at `https://anonymous.4open.science/r/iclr_paper3-1C00/README.md` contains the verifier, renderer, decoder, benchmark builder, strategy suite, and analysis scripts used for every reported table and figure. It also includes trajectories, benchmark questions, run records, prompts, retry-feedback templates, raw generations, the locked split, a benchmark content hash, the regularity artifact, the cross-solver mask, which the runner enables by default, the confirmatory preregistration with its two amendments, and an environment lockfile. The frontier-model records in Appendix C.4 were generated with the released prompts and verifier, through a command-line relay and an API proxy. The release contains both runners, with the operator's endpoint and request references removed. A technical supplement with full statements and proofs is included in the supplementary material. Each record stores a configuration hash covering the numeric-semantics version, regularity-gate stamp, renderer wording version, tolerance map, thresholds, residual tolerance, benchmark hash, retry and token budgets, seed, model revision, and code commit.

The analysis aborts when a locked field differs within a reported cell, preventing a table from combining incompatible configurations. Replay recomputes all reported values from frozen generations and records on a CPU. Regeneration reruns the models on a GPU and may vary across hardware because batching is not bitwise reproducible, even with identical prompts and seeds. The release ships the generator of every reported table and figure, and its replay recomputes each stored paper fact from the records and compares it with the shipped value. Before submission, every decimal and grouped count in the manuscript was checked against these facts and the generator outputs.

AI USE STATEMENT

Large language models were used as evaluation subjects and as tools for boilerplate code, LaTeX formatting, and prose editing. Language models are also the subjects of the evaluation: six open-weight models and four hosted models generate the answers the paper measures. The authors reviewed and verified all AI-assisted text and code. The authors take responsibility for the final paper, including its text, claims, code, and experimental artifacts.

REFERENCES

Kevin-Martin Aigner, Marc Goerigk, Michael Hartisch, Frauke Liers, and Arthur Miehlich. A framework for data-driven explainability in mathematical optimization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 20912–20920, 2024.

Antonio Alcántara, Aysegül Kahraman, Anosh Arshad Sundhu, and Spyros Chatzivasileiadis. Conductor: An llm-orchestrated digital twin for uncertainty-aware distribution grid operations. *arXiv preprint arXiv:2606.24609*, 2026.

Vassili Alexiadis, James Colyar, John Halkias, Rob Hranac, and Gene McHale. The next generation simulation program. *Institute of Transportation Engineers. ITE Journal*, 74(8):22, 2004.

Jean Pierre Allamaa, Panagiotis Patrinos, and Tong Duy Son. Exampc: the data-driven explainable and approximate nmpc with physical insights. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2976–2983. IEEE, 2025.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control: Jae andersson et al. *Mathematical Programming Computation*, 11(1):1–36, 2019.

Biswas Rudra Jyoti Arka, Sadman Sakib, Md Zahidul Islam, and Shamsun Nahar Edib. Explainable post-disaster grid observability recovery using human-oversight agentic llms. *arXiv preprint arXiv:2609.16774*, 2026.

Shuyang Cao and Lu Wang. Verifiable generation with subsentence-level fine-grained citations. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 15584–15596, 2024.

Steven Carr, Nils Jansen, Sebastian Junges, and Ufuk Topcu. Safe reinforcement learning via shielding under partial observability. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 14748–14756, 2023.

Hao Chen, Gonzalo Esteban Constante-Flores, Krishna Sri Ipsit Mantri, Sai Madhukiran Kompalli, Akshdeep Singh Ahluwalia, and Can Li. Optichat: Bridging optimization models and practitioners with large language models. *INFORMS Journal on Data Science*, 5(3):199–220, 2026.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*, 2019.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Kai-Chen Cheng, Haejun Han, and David Q Sun. Think fast, talk smart: Partitioning deterministic and neural computation for structured health text generation. *arXiv preprint arXiv:2605.29652*, 2026.

Matthias Cosler, Christopher Hahn, Daniel Mendoza, Frederik Schmitt, and Caroline Trippel. nl2spec: Interactively translating unstructured natural language to temporal logics with large language models. In *International Conference on Computer Aided Verification*, pp. 383–396. Springer, 2023.

Kristijonas Čyras, Dimitrios Letsios, Ruth Misener, and Francesca Toni. Argumentation for explainable scheduling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 2752–2759, 2019.

Alexandre Forel, Axel Parmentier, and Thibaut Vidal. Explainable data-driven optimization: From context to decision and back again. In *International conference on machine learning*, pp. 10170–10187. PMLR, 2023.

Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170, 2018.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. *Advances in neural information processing systems*, 30, 2017.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured nlp tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10932–10952, 2023.

Marc Goerigk and Michael Hartisch. A framework for inherently interpretable optimization models. *European Journal of Operational Research*, 310(3):1312–1324, 2023.

Lucas Torroba Hennigen, Shannon Shen, Aniruddha Nrusimha, Bernhard Gapp, David Sontag, and Yoon Kim. Towards verifiable text generation with symbolic references. *arXiv preprint arXiv:2311.09188*, 2023.

Kerianne L Hobbs, Mark L Mote, Matthew CL Abate, Samuel D Coogan, and Eric M Feron. Runtime assurance for safety-critical systems: An introduction to safety filtering approaches for complex control systems. *IEEE Control Systems Magazine*, 43(2):28–65, 2023.

Kai-Chieh Hsu, Haimin Hu, and Jaime F Fisac. The safety filter: A unified view of safety-critical control in autonomous systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 7, 2023.

Radoslav Ivanov, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pp. 169–178, 2019.

Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. Safe reinforcement learning using probabilistic shields. In *31st International Conference on Concurrency Theory (CONCUR 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

Amita Kamath, Robin Jia, and Percy Liang. Selective question answering under domain shift. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 5684–5696, 2020.

Spyros A Kazarlis, AG Bakirtzis, and Vassilios Petridis. A genetic algorithm solution to the unit commitment problem. *IEEE transactions on power systems*, 11(1):83–92, 1996.

Jaechang Kim, Sunung Mun, Seungjoon Lee, Jaewoong Cho, and Jungseul Ok. Towards faithful agentic xai: A verification method and an open-world benchmark for better model faithfulness. *arXiv preprint arXiv:2605.27879*, 2026.

Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. Factkg: Fact verification via reasoning on knowledge graphs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16190–16206, 2023.

Seong-Tae Kim. Pjm hourly energy consumption, 2025. URL `https://dx.doi.org/10.21227/wqg6-9917`.

George Koomullil. Proof-carrying certificates for llm pipelines: A trust-boundary architecture. *arXiv preprint arXiv:2605.16407*, 2026.

Anton Korikov, Alexander Shleyfman, and Chris Beck. Counterfactual explanations for optimization-based decisions in the context of the gdpr. In *ICAPS 2021 workshop on explainable AI planning*, 2021.

Tobias Kuhn. A survey and classification of controlled natural languages. *Computational linguistics*, 40(1):121–170, 2014.

Jannis Kurtz, Ş İlker Birbil, and Dick den Hertog. Counterfactual explanations for linear optimization. *European Journal of Operational Research*, 329(1):24–41, 2026.

Jacky Kwok, Shulu Li, Pranav Atreya, Yuejiang Liu, Yixing Jiang, Chelsea Finn, Marco Pavone, Ion Stoica, and Azalia Mirhoseini. Llm-as-a-verifier: A general-purpose verification framework. *arXiv preprint arXiv:2607.05391*, 2026.

Christine P Lee, David Porfirio, Xinyu Jessica Wang, Kevin Chenkai Zhao, and Bilge Mutlu. Veriplan: Integrating formal verification and llms into end-user planning. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–19, 2025.

Minjae Lee, Kyungmin Kim, Taesoo Kim, and Sangdon Park. Selective generation for controllable language models. *Advances in Neural Information Processing Systems*, 37:50494–50527, 2024.

Junbo Jacob Lian, Yujun Sun, Huiling Chen, Chaoyu Zhang, Hanzhang Qin, and Chung-Piaw Teo. Reloop: Structured modeling and behavioral verification for reliable llm-based optimization. *arXiv preprint arXiv:2602.15983*, 2026.

Christopher Mohri and Tatsunori Hashimoto. Language models with conformal factuality guarantees. *arXiv preprint arXiv:2402.10978*, 2024.

Ramesh Arvind Naagarajan, Zühal Wagner, and Stefan Streif. Hierarchical causal abduction: A foundation framework for explainable model predictive control. *arXiv preprint arXiv:2605.10624*, 2026.

Alexander Neubauer, Tianzhen Hong, Han Li, Mengbo Yu, Amin Darbandi, Yannick Fürst, and Martin Kriegel. Large language models for hvac operations in building energy systems: A critical review of methods, applications, and deployment readiness. *arXiv preprint arXiv:2609.05314*, 2026.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN 9780387400655. URL `https://books.google.de/books?id=VbHYoSyelFcC`.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5153–5176, 2023.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, 2023.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding. *Advances in Neural Information Processing Systems*, 37:24547–24568, 2024.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

Dung T Phan, Radu Grosu, Nils Jansen, Nicola Paoletti, Scott A Smolka, and Scott D Stoller. Neural simplex architecture. In *NASA Formal Methods Symposium*, pp. 97–114. Springer, 2020.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi Jaakkola, and Regina Barzilay. Conformal language modeling. In *International Conference on Learning Representations*, volume 2024, pp. 11654–11681, 2024.

Sumaya Abdul Rahman, Seckhen Ariel Andrade Cuellar, Ghani Raissov, and Mohammad Raza. Verisimpl: Robust optimization modeling from natural language using simplification-based verification. *arXiv preprint arXiv:2607.20474*, 2026.

Nils Rautenberg and Sven Schippkus. Probabilistic guarantees for reducing contextual hallucinations in llms. *arXiv preprint arXiv:2601.00641*, 2026.

Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural language engineering*, 3(1):57–87, 1997.

Allen Z Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, et al. Robots that ask for help: Uncertainty alignment for large language model planners. *arXiv preprint arXiv:2307.01928*, 2023.

Andoni Rodriguez, Alberto Pozanco, and Daniel Borrajo. Cycle-consistent neural explanation of formal verification certificates. *arXiv preprint arXiv:2606.24414*, 2026.

JV Roig. Scalable and reliable evaluation of ai knowledge retrieval systems: Riker and the coherent simulated universe. *arXiv preprint arXiv:2601.08847*, 2025.

Maxon Rubin-Toles, Maya Gambhir, Keshav Ramji, Aaron Roth, and Surbhi Goel. Conformal language model reasoning with coherent factuality. In *International Conference on Learning Representations*, volume 2025, pp. 42086–42114, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 9895–9901, 2021.

Lim C Siang and Daniel L O'Connor. Explainable lp-mpc: Shadow price contributions reveal mv-cv pairings. *arXiv preprint arXiv:2512.06194*, 2025.

Aivin V Solatorio. Proof-carrying numbers (pcn): A protocol for trustworthy numeric answers from llms via claim verification. *arXiv preprint arXiv:2509.06902*, 2025.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1): 25–57, 2006.

Brandon T Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.

Yasin Abbasi Yadkori, Ilja Kuzborskij, David Stutz, András György, Adam Fisch, Arnaud Doucet, Iuliya Beloshapka, Wei-Hung Weng, Yao-Yuan Yang, Csaba Szepesvári, et al. Mitigating llm hallucinations via conformal abstention. *arXiv preprint arXiv:2405.01563*, 2024.

Yujiao Yang. True: A trustworthy unified explanation framework for large language model reasoning. *arXiv preprint arXiv:2602.18905*, 2026.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Satlm: Satisfiability-aided language models using declarative prompting. *Advances in Neural Information Processing Systems*, 36:45548–45580, 2023.

Serhii Zabolotnii. Docua at crf filling 2026: Llm structcore—schema-guided reasoning condensation and deterministic compilation. In *Proceedings of the Third Workshop on Patient-Oriented Language Processing (CL4Health)@ LREC 2026*, pp. 375–383, 2026.

Kun Zhang, Oana Balalau, and Ioana Manolescu. Factspotter: Evaluating the factual faithfulness of graph-to-text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10025–10042, 2023.

Ninghan Zhong, Ahmet Ege Tanriverdi, Kaan Kale, and Sriram Vishwanath. Verytrace: Verifying reasoning traces through compilable formalism and structured verification. *arXiv preprint arXiv:2606.24124*, 2026.

A Formal semantics

Complete proofs of soundness, scale invariance, multiplier identifiability, and display non-interference appear in the technical supplement. The renderer has no sentence form for causal explanations, unverified comparisons, hedges or newly introduced entities, and it performs no unit conversion, localisation or name resolution; a conformance suite tests these cases. A recorded value need not be treated as exact. The oracle may return a declared set S of plausible values; it accepts a claim only when the claim predicate holds for every value in S, rejects it only when the predicate fails for every value in S, and otherwise returns UNKNOWN. The reported experiments use exact records, for which S is a singleton. Equality constraints have multipliers and enter the LICQ check through the active Jacobian, but they receive no status label: an equality is imposed by the formulation, so calling it binding would describe the problem rather than the recorded solve. An equality

multiplier with an operational meaning is declared as a quantity and checked as a numeric claim; the dispatch marginal price is the power-balance dual.

A trajectory T is a sequence of controller records, one per solve. For quantity e at timestamp τ, $x_\tau(e)$ is its recorded value. For inequality constraint j, g_j and λ_j are its residual and solver-reported multiplier. Each constraint has a declared positive scale s_j, and each domain has an objective scale $s_f > 0$. We use the normalised residual and multiplier

$$\hat{g}_j = g_j/s_j, \qquad \hat{\lambda}_j = \lambda_j s_j/s_f.$$

The status threshold, abstention margin, and residual tolerance are θ, m, and ε_g. The verifier records a diagnostic subtype with every UNKNOWN: a provenance failure, a regularity flag, or the status case that caused it, namely a multiplier inside the abstention band, an active but non-binding constraint, a primal violation, or a KKT-inconsistent record (technical supplement, Definition 1.6). Each quantity e has an equality tolerance $\mathrm{tol}(e)$. The signed domain graph is G, and K is the retry budget.

The status predicates are

$$\mathrm{binding}(j) \iff \hat{\lambda}_j > \theta + m \ \wedge\ |\hat{g}_j| \le \varepsilon_g, \qquad \mathrm{slack}(j) \iff \hat{\lambda}_j \le \theta - m \ \wedge\ \hat{g}_j \le -\varepsilon_g.$$

All other status cases are UNKNOWN. A multiplier-qualified status claim additionally requires the LICQ artifact to accept the active set at that record. The artifact is computed offline from constraint gradients, which are absent from the runtime log. The verifier accepts a claim only when every applicable check accepts it; REJECT and UNKNOWN both lead to refusal.

For a predicate P_c, the oracle returns ACCEPT when $P_c(x)$ holds for every $x \in S$, REJECT when it fails for every $x \in S$, and UNKNOWN otherwise. This rule reduces to ordinary point evaluation when S is a singleton. The renderer maps an accepted ordered claim set C to fixed sentence forms. Its decoder D satisfies

$$D(R(C)) = \mathrm{nf}(C),$$

where nf is the renderer's claim normal form. Numeric and status claims retain their entity, relation, value, unit, and timestamp under normalisation; relational claims retain their source, relation, and target. The normal form keeps the first occurrence of a repeated claim. The closing and refusal sentences decode to no factual claim.

Verifier soundness. If all applicable checks accept a claim c, then c satisfies its declared predicate on the recorded solve. Coverage and unit checks ensure that the predicate is defined over declared quantities, timestamps, and units. The evidence oracle evaluates numeric and status predicates, the dependency check evaluates declared relations, and the LICQ artifact is required for multiplier-qualified status claims. Thus acceptance implies truth under the claim semantics above.

Three-way oracle. For any claim predicate P_c and plausible-value set S, acceptance implies that P_c holds for every $x \in S$, and rejection implies that it fails for every $x \in S$. The remaining cases are assigned UNKNOWN. This follows directly from the universal acceptance and rejection rules above.

Scale invariance. Under constraint rescaling $g_j \mapsto a g_j$, $s_j \mapsto a s_j$, and $\lambda_j \mapsto \lambda_j/a$ for $a > 0$, both $\hat{g}_j$ and $\hat{\lambda}_j$ are unchanged. Under objective rescaling $f \mapsto cf$, $s_f \mapsto c s_f$, and $\lambda_j \mapsto c\lambda_j$ for $c > 0$, $\hat{\lambda}_j$ is unchanged. The status verdict therefore remains unchanged under either positive rescaling.

A.1 MULTIPLIER-STATUS ASSUMPTIONS AND LICQ AUDIT

Numeric claims require a usable record and declared units. Multiplier-qualified status claims additionally require a unique multiplier vector at the recorded solution. We use the linear independence constraint qualification (LICQ): the gradients of all active inequalities and equalities are linearly independent. Under LICQ, the KKT stationarity equation has a unique multiplier vector for the fixed formulation and declared scaling. The runtime record supports checks for solver status, primal feasibility, and declared units. Because it does not contain constraint gradients, the LICQ condition is evaluated offline and supplied to the verifier as a trajectory-bound regularity artifact. Recomputing the artifact for all 16,140 recorded instants took 69 seconds of wall-clock time: 49 in HVAC, 19 in

vehicle control and 1 in dispatch. A deployment could avoid the offline pass by logging the rank of the active-constraint Jacobian at solve time, since the solver already evaluates that Jacobian at the solution.

All 16,140 recorded solves satisfy the declared convergence requirement, and no gated method accepts a claim in an undeclared unit; the oracle gate alone, which omits the unit check, accepts 39 such claims.

Because the runtime logs omit constraint gradients, we re-solve every recorded instant offline and construct the active-constraint Jacobian using the declared, possibly state-dependent, constraint scales. We normalise each gradient to unit length and mark an active set rank-deficient when a singular value is below 10^{-8}. Repeating the audit with thresholds 10^{-6} and 10^{-10} leaves all domain-level counts unchanged. All 16,140 recorded solves reconstruct successfully. Their re-solved stage-0 controls and multipliers agree with the stored values to within a relative deviation of 4.2×10^{-4}. The audit produces a regularity mask indexed by trajectory, record, and timestamp. Each mask is bound to the trajectories used to create it by digest. At runtime, the verifier refuses a status claim when the artifact does not cover the supplied record. The reproduction check compares artifact decisions rather than raw digests, because equivalent regenerated artifacts can have different stamps. All 4,638 campaign refusals attributed to the regularity check correspond to an entry in the mask.

Table 5: LICQ audit over recorded solves. *Deficient* counts instants with a rank-deficient active-constraint Jacobian. *Binding* counts stage-0 constraint-instants accepted by the binding predicate. The final columns count binding labels in deficient active sets and labels whose multiplier is non-unique.

Domain	Instants	Deficient	Binding	In deficient set	Non-unique
HVAC	8,640	2,019	5,301	2,339	1,570
Vehicle control	6,000	76	144	42	36
Dispatch	1,500	0	12,700	0	0

Table 5 shows that rank deficiency is common in HVAC, rare in vehicle control, and absent in the evaluated dispatch records. The audit finds 2,019 rank-deficient HVAC instants and 76 vehicle instants. Of the 5,301 HVAC stage-0 constraint-instants labelled *binding* by the status predicate, 1,570 have a non-unique multiplier; the corresponding vehicle count is 36 of 144, and the dispatch count is zero. A rank-deficient record can still determine primal activity, but it does not identify a unique dual price. The interface therefore withholds a multiplier-qualified status label at these records. Residual-only statements, *at limit* and *clear of its limit*, do not depend on multiplier uniqueness and remain in the claim language, but the loop does not substitute them for a withheld label (Appendix A.3).

The absence of rank-deficient dispatch records is a property of these evaluated programmes, not of linear programming in general. Two checks support the result: every dispatch active-constraint Jacobian has full row rank, and an interior-point solver and dual-simplex solver reproduce all normalised multipliers within 8.33×10^{-17}. The latter agreement is consistent with a unique dual representation for the recorded formulations and scaling.

Against labels it does not generate, the status rule agrees with the solver-reported active set at 84,190 of 84,320 constraint-instants, returns UNKNOWN at 129, differs at one, and never contradicts activity recomputed from the normalised residual.

A.2 Threshold, margin and solver settings

The deployed normalised-multiplier thresholds are 2×10^{-8} for HVAC, 3.3×10^{-6} for vehicle control and 5×10^{-8} for dispatch, with margin $m = 0$ and residual tolerance $\varepsilon_g = 10^{-6}$ in every domain. We re-decide frozen primary-model claim sets over a logarithmic sweep of the normalised-multiplier threshold in each domain. The records, proposed claims, verification configuration, and renderer remain fixed, so this analysis isolates the effect of θ on verifier outcomes. At the deployed threshold, the re-decision reproduces all campaign verdicts: 1,020 in HVAC, 787 in vehicle control, and 747 in dispatch. Across the sweep, the accepted binding set remains unchanged at 102 constraint-instants in HVAC, 58 in vehicle control, and 115 in dispatch. Threshold changes move

only near-zero inactive cases between *slack* and UNKNOWN. No scorer-flagged rendered answer is observed at any threshold in the sweep.

The thresholds are selected on development trajectories, not derived from solver tolerances. Each lies inside an empirical gap between the largest normalised multiplier on a constraint labelled *slack* and the smallest on one labelled *binding*. The gap runs from 1.99×10^{-8} to 2.28×10^{-6} in HVAC, from 1.96×10^{-6} to 1.30×10^{-4} in vehicle control, and from 0 to 2.48×10^{-5} in dispatch. Any threshold inside the gap yields the same binding set, which is why the sweep leaves it unchanged. In HVAC the deployed threshold sits at the lower edge of the gap, within one percent of the largest slack multiplier. The binding side retains its margin: the smallest binding multiplier exceeds the reconstructed dual interval of Appendix A.3, 8.81×10^{-7}. A *slack* label additionally requires the residual to be clear of the limit, where complementarity forces a zero multiplier. Replacing normalised multipliers with raw multipliers in HVAC changes 56 verdicts, including five accept-to-reject changes, and widening the declared uncertainty set turns accepted claims into UNKNOWN, not REJECT.

Figure 3 re-solves every test instant that a benchmark question queries and relabels each displayed status sentence of the locked campaign with the deployed rule. HVAC and vehicle control use 24 Ipopt settings: tolerance 10^{-6}, 10^{-8} or 10^{-10}, monotone or adaptive barrier, initial barrier 0.1 or 10^{-4}, and a warm or cold start. Dispatch uses 5 HiGHS settings. The deployed setting reproduces every stored label.

In dispatch no displayed sentence changes under any setting, out of 824. In vehicle control at most 16 of 875 sentences change, all to UNKNOWN. In HVAC, loose tolerances move up to 798 of 1,213 sentences to UNKNOWN. Twelve of the 24 HVAC settings also contradict displayed sentences, 12 or 24 at a time. All of them concern 2 instants of one trajectory, where the alternative solve reaches a different local optimum with the temperature band active. The cross-solver mask does not list these two instants; the extended mask of Appendix C.12 does. Across thresholds from 0.03 to 31.6 times the deployed value and margins up to the threshold, no accepted status label switches to the contradictory label in this sweep; labels move only to UNKNOWN.

A.3 Residual-only fallback and status by constraint family

When the LICQ gate withholds a status label, the record still decides whether the constraint is at its limit. For the primary model, the gated rendered arm had 52 status claims withheld by this gate and tool calling with the verified display had 115. Every one of them has an accepted at-limit or clear-of-limit sentence, yet the loop never substitutes it: all 50 and all 108 affected questions were refused. With the fallback, 39 and 57 of these questions would receive a fully accepted answer. In HVAC, 45 of the 109 questions the tool layer loses involve such a withheld claim, and 22 are fully recoverable. Crediting every recoverable answer would move the HVAC difference from -0.093 to -0.057; the gold answers ask for binding or slack, so the current adequacy test credits fewer. A less conservative rule could bound the multiplier over the reconstructed dual-optimal set instead of withholding every rank-deficient case. It is not deployed, because the reconstruction fails its controls: at unique-dual HVAC instants the reconstructed interval reaches 8.81×10^{-7} instead of collapsing to a point, and at one audited vehicle instant it excludes the solver-reported multiplier. The interface therefore retains the conservative LICQ rule. Table 6 gives the outcomes by constraint family. Withheld claims concentrate on the CO_2 limit and comfort band in HVAC and on the safe following gap and actuator limits in vehicle control; dispatch has none.

A.4 Specification and guarantees

The properties below concern the specification of Appendix A. Their hypotheses are that the recorded values are the values the controller realised, that every declared unit conversion is exact, that the regularity artifact is bound to the trajectory it was computed from, and, for the deployed system, that the implementation agrees with the specification. The conformance suite tests the last hypothesis (Appendix D.1). The technical supplement gives the proofs. $V(c) \in \{\text{ACCEPT}, \text{REJECT}, \text{UNKNOWN}\}$ denotes the verdict on a claim c.

Soundness and the certifiable fragment

Property A1 (Soundness relative to the record)**.** *Let* $V(c) = \text{ACCEPT}$.

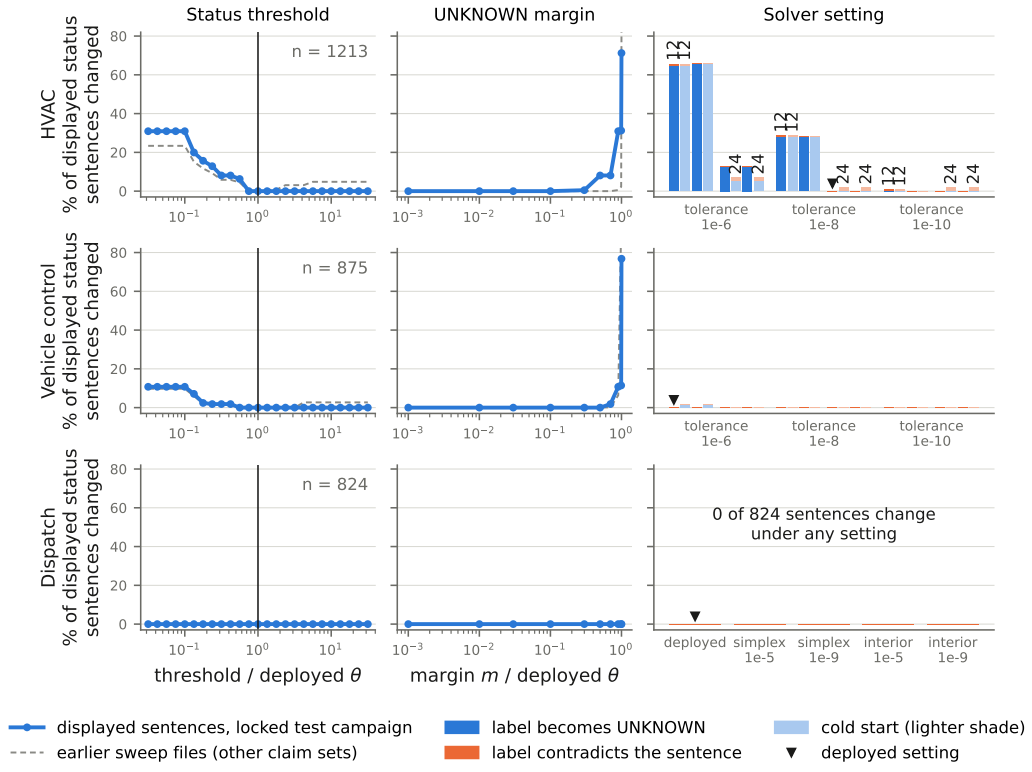

Figure 3: Share of displayed status sentences whose label changes, by domain, as the status threshold, the UNKNOWN margin and the solver setting vary. Orange marks labels that contradict the displayed sentence; the triangle marks the deployed setting. Within each tolerance group the bars are monotone barrier (left four) and adaptive barrier (right four), each with initial barrier 0.1 then 10^{-4}, warm then cold start. Dashed lines repeat the earlier sweeps, which count a different claim set.

Table 6: Status claims by constraint family, primary model, locked test split. Unknown counts withheld claims; the number in brackets is withheld by the LICQ gate. No ramp or rate constraint is declared.

		Gated and rendered				Tool calling + verified display			
Domain	Family	Prop.	Acc.	Rej.	Unknown	Prop.	Acc.	Rej.	Unknown
HVAC	CO_2 limit	92	66	4	22 (22)	279	198	20	61 (61)
HVAC	power cap	54	47	7	0 (0)	249	199	50	0 (0)
HVAC	comfort band	145	115	11	19 (7)	346	280	42	24 (10)
Vehicle	actuator limit	85	78	4	3 (3)	284	245	23	16 (16)
Vehicle	safe gap	57	37	0	20 (20)	114	80	6	28 (28)
Vehicle	speed limit	43	43	0	0 (0)	101	94	7	0 (0)
Dispatch	generator capacity	107	107	0	0 (0)	298	298	0	0 (0)
Dispatch	generator minimum	115	114	1	0 (0)	275	275	0	0 (0)
All	all families	698	607	27	64 (52)	1,946	1,669	148	129 (115)

(i) If $c = (e, r, v, u, \tau)$ is numeric, then e is declared, τ is recorded, u is the canonical unit of e, and

$$|x_\tau(e) - v| \le \mathrm{tol}(e) \quad (r \text{ is } =), \qquad x_\tau(e)\ r\ v \quad (r \in \{\le, \ge, <, >\}).$$

(ii) If $c = (\lambda_j, \mathrm{binding}, \tau)$, then $\hat{\lambda}_j > \theta + m$ and $|\hat{g}_j| \le \varepsilon_g$ at τ. If $c = (\lambda_j, \mathrm{slack}, \tau)$, then $\hat{\lambda}_j \le \theta - m$ and $\hat{g}_j \le -\varepsilon_g$ at τ.

(iii) If $c = (a, r, b)$ is relational, then $(a \xrightarrow{\sigma} b)$ is an edge of G with $\sigma = +$ for $r = \text{increases}$ and $\sigma = -$ for $r = \text{decreases}$.

Theorem 1.7 of the technical supplement gives the proof.

The certifiable fragment cov is the set of claims that satisfy three conditions. First, every named quantity is declared; for a trace-relative claim, the timestamp is recorded and its solve converged; for a numeric claim, the unit is canonical. Second, a status claim has an unmasked instant and a status token in $\{\text{BINDING}, \text{SLACK}\}$. Third, a relational claim names an edge of G with sign $+$ or $-$. cov is a function of the record and the declarations only; the language model, the prompt and the retry policy do not enter it. Every accepted claim lies in cov, hence so does every displayed claim. On cov the verifier is complete:

$$V(c) = \text{ACCEPT} \iff c \text{ is true}, \qquad V(c) = \text{REJECT} \iff c \text{ is false}, \qquad c \in \mathrm{cov},$$

so UNKNOWN does not occur on cov (technical supplement, Theorem 3.3).

SCALE INVARIANCE

Property A2 (Invariant status quantities)**.** *Let $a_j > 0$ and $\beta > 0$. Consider the constraint rescaling $g_j \mapsto a_j g_j$, $s_j \mapsto a_j s_j$, $\lambda_j \mapsto \lambda_j / a_j$ and the objective rescaling $f \mapsto \beta f$, $s_f \mapsto \beta s_f$, $\lambda_j \mapsto \beta \lambda_j$. Then*

$$\frac{a_j g_j}{a_j s_j} = \hat{g}_j, \qquad \frac{(\lambda_j / a_j)(a_j s_j)}{s_f} = \hat{\lambda}_j, \qquad \frac{(\beta \lambda_j) s_j}{\beta s_f} = \hat{\lambda}_j.$$

Hence every status verdict, and every verdict on a numeric claim about $\hat{\lambda}_j$ or $\hat{g}_j$, is unchanged.

Theorem 2.4 of the technical supplement gives the proof.

IDENTIFIABILITY OF STATUS LABELS

Property A3 (Multiplier uniqueness under LICQ)**.** *Fix the optimisation formulation and the declared scaling. Let $z^\star$ be the recorded solution, $\mathcal{A} := \{j : g_j(z^\star) = 0\}$ its active set and $J_\mathcal{A}$ the matrix whose rows are the gradients at $z^\star$ of the active inequality and of the equality constraints. If $J_\mathcal{A}$ has full row rank (LICQ), then the KKT multipliers at $z^\star$ are unique:*

$$(\lambda_\mathcal{A}, \nu) = -\left(J_\mathcal{A} J_\mathcal{A}^\top\right)^{-1} J_\mathcal{A} \nabla f(z^\star), \qquad \lambda_j = 0 \quad (j \notin \mathcal{A}).$$

Hence every solver that returns $z^\star$ with a KKT multiplier returns the same $\hat{\lambda}_j$, and the status predicate at $z^\star$ does not depend on the solver.

Theorem 4.3 of the technical supplement gives the proof for the declared scaling.

The property concerns a fixed formulation and scaling. Equivalent physical models with different constraint decompositions have different multipliers, and the verifier does not claim invariance to such reformulations. It checks the multiplier of the declared constraint row in the recorded optimisation problem. Appendix D.1 reports the second-solver and formulation checks of the implementation audit.

The LICQ requirement applies to both *binding* and *slack*. For a strictly inactive inequality, $g_j(z^\star) < 0$ and complementarity $\lambda_j g_j(z^\star) = 0$ already give $\lambda_j = 0$, so LICQ is not needed for that conclusion. The same gate is applied to both labels as a conservative numerical policy: near-zero residuals and solver tolerances can make an apparent slack status depend on the active-set reading. Appendix A.3 reports the cases behind this policy.

A local-stability result under perturbation of the problem data would require strict complementarity and the strong second-order sufficient condition. The deployed margin is $m = 0$, so that result is not used, and no persistence of status labels under perturbation is claimed. The guarantee concerns the recorded committed solve and its declared semantics.

DISPLAY NON-INTERFERENCE

Property A4 (Display non-interference)**.** *Fix a trajectory T, a question q, the declarations and a retry budget K. For a model transcript m, let $\mathrm{gate}_K(m)$ be the first claim sequence of m whose*

claims are all accepted, or $\varnothing$ if there is none, and let κ replace the value of each accepted equality claim by the recorded value it was compared against. Put $C := \kappa(\mathrm{gate}_K(m))$. The displayed output is

$$\mathrm{disp}(m) = \begin{cases} R(C) & C \neq \varnothing, \\ z_\varnothing & C = \varnothing, \end{cases}$$

$$\mathrm{prop}(\mathrm{disp}(m)) = \{P_{\mathrm{nf}(c)} : c \in C\}, \qquad |\,\mathrm{prop}(\mathrm{disp}(m))| \leq |C|,$$

where $z_\varnothing$ is a fixed refusal with no factual proposition. Hence $\mathrm{disp}(m) \in R(\mathrm{Acc}(T, q)) \cup \mathcal{F}$ for every m, where $\mathrm{Acc}(T, q)$ is the set of accepted claim sequences and $\mathcal{F}$ is the fixed set of refusals.

Proposition 6.5 of the technical supplement gives the proof, and the conformance suite checks the implementation against it (Appendix D.1).

By Properties A4 and A1, every displayed factual proposition is the predicate of an accepted claim, and that predicate holds on the recorded solve. An empty claim sequence renders as $z_\varnothing$, which asserts no factual proposition.

B Experimental protocol

B.1 Glossary

Table 7 defines each evaluation term.

Table 7: Terms used in the evaluation.

Term	Meaning
Exposure	$\Pr(A = 1, \Phi = 0)$: flagged answers as a share of all questions.
Conformance	Agreement of the implemented verifier, renderer and decoder with the declared semantics; it does not establish that the semantics describe the plant.
Regularity (LICQ) mask	The recorded instants whose active constraint gradients are rank-deficient; status claims there are withheld.
Cross-solver mask	The recorded instants where a second solver assigns a different status; status claims there are withheld when the mask is enabled.
Trajectory	One independent controller run; the unit of resampling for every interval.

B.2 Position against recent verification work

Table 8 compares methods by the object they verify and by the guarantee that reaches the displayed response. The relevant distinction is whether verification constrains the final sentence shown to the reader or an intermediate object that a model may later describe in free prose. Appendix B.5 evaluates this distinction under matched accepted claim sets, where the final display policy is the only component that changes.

B.3 Domains and benchmark

We evaluate three simulated control domains, each with 10 development trajectories and 20 locked test trajectories fixed before evaluation. Zone HVAC controls temperature, humidity, and carbon dioxide using a nonlinear program in CasADi (Andersson et al., 2019) solved with IPOPT (Wächter & Biegler, 2006); its disturbances combine a published air-handling log with a typical meteorological year. Longitudinal vehicle control follows a lead vehicle on the same optimisation stack, using stop-and-go profiles patterned on the NGSIM I-80 study (Alexiadis et al., 2004). Economic dispatch allocates a ten-unit fleet from Kazarlis et al. (1996) against recorded PJM load (Kim, 2025) through a linear program solved by HiGHS. The three domains provide numeric readings, constraint-status cases, and declared model relations under different solver and regularity conditions. Controller implementations are included in the release.

Table 8: Comparison with recent verification-based language interfaces. *Object* is the representation checked by the method. *Displayed text* describes the guarantee for the response shown to the reader. *Decoder* indicates whether displayed text maps exactly back to the checked representation. *Cost* indicates whether the work reports the effect of the restriction on answer coverage or adequacy. Entries describe what each cited paper reports; *not reported* means the paper does not describe the property, not that its system lacks it.

Work	Object	Displayed text	Decoder	Cost
Agentic XAI verification (Kim et al., 2026)	explanation drafts	model-generated text follows verification	not reported	not reported
Cycle-consistent certificates (Rodriguez et al., 2026)	formal certificates	model text is accepted after checker round-trip	partly reported	not reported
VeryTrace (Zhong et al., 2026)	reasoning traces	verified trace; final prose is unconstrained	not reported	not reported
Proof-carrying pipelines (Koomullil, 2026)	deterministic computations	certificate accompanies model output	not reported	not reported
Contextual-hallucination guarantees (Rautenberg & Schippkus, 2026)	generated statements	probabilistic guarantee on generated output	not reported	partly reported
Proof-carrying numbers (Solatorio, 2025)	displayed numeric spans	numeric spans are source-bound and assigned a verification status; surrounding model prose is displayed	spans only	not reported
Symbolic references (Hennigen et al., 2023)	values from structured data	referenced values are rendered from the source; surrounding prose remains free	values only	not reported
Verified display interface	typed claims about a committed solve	closed renderer produces the displayed response from accepted claims	yes, exact	yes

BENCHMARK CONSTRUCTION

The benchmark evaluates the interface on distinctions represented by the claim language and verifier. It does not estimate the distribution of questions asked by control-room operators. Recent grid-operation agents are evaluated on a real 60 kV distribution network (Alcántara et al., 2026) and on operator questions during grid restoration on IEEE test systems (Arka et al., 2026). Items are generated from author-written templates over numeric readings, constraint statuses, model-relative relations, and conjunctions that require multiple claims (Table 9). The factorial design varies trajectory, constraint regime, question type, claim-set size, wording, and perturbation. Perturbations include false premises, unit mismatches, stale timestamps, unsupported quantities, negation, and hedging. Each perturbation modifies a base item while retaining its source trajectory and timestamp.

Every item specifies the claims required for an adequate answer and whether the question is answerable within the certifiable fragment. An item is unanswerable when it names a quantity outside the declared graph or requests a status predicate that the record cannot decide. Gold claims are derived from the declared record-relative semantics, not from human annotation. RIKER likewise fixes ground truth first and generates its evaluation documents from it (Roig, 2025). The verifier, benchmark labels, and structural scorer therefore use the same claim predicates. This establishes conformance with the declared task specification; Section 4 states what that specification does not cover. Appendix A.1 compares the status rule with solver-reported active sets and residual-based activity, which are produced independently of the multiplier threshold and margin.

Items inherit the split of their source trajectory, so no trajectory contributes questions to both development and test sets. Each trajectory uses a sampled controller configuration. The split holds out trajectories, configurations, and disturbance windows. It does not hold out the simulator, the template generator, the claim language, or the configuration family. Evaluation on an unseen site or independently authored question distribution would require a separate benchmark.

Table 9: Benchmark data card. Questions are generated from author-written templates, with four wording families and five paraphrases per family. Items vary factorially over trajectory, regime, and question type; perturbations are appended when applicable. Gold claims are derived from the solver trace under the declared semantics. No human annotation is used for benchmark labels. The final row gives the content-hash prefixes of the locked item lists.

Benchmark	HVAC	AV speed	Dispatch	Pooled
Trajectories dev / test	10 / 20	10 / 20	10 / 20	30 / 60
Items dev / test	322 / 599	158 / 413	225 / 444	705 / 1,456
numeric	262	163	187	612
status	213	134	164	511
conjunction	216	144	164	524
relational	230	130	154	514
none	664	408	480	1,552
false premise	24	17	18	59
unit trap	35	25	27	87
stale timestamp	50	31	36	117
unsupported entity	50	30	36	116
negation	49	30	36	115
hedged	49	30	36	115
Perturbed share, declared / realised	0.45 / 0.28	0.45 / 0.29	0.45 / 0.28	0.45 / 0.28
Phrasing families (paraphrases each)	4 (5)	4 (5)	4 (5)	4 (5)
Answerable / unanswerable	854 / 67	538 / 33	633 / 36	2,025 / 136
Content SHA-256 (prefix)	`80df1aff`	`e2e5974e`	`a2bfcdd7`	`80df1affe2e5974ea2bfcdd7`

Question-type rows are not mutually exclusive because conjunctions may include numeric, status, or relational claims. Perturbation rows count applied perturbations and may also overlap across item types; the total item counts appear in the "Items dev / test" row.

B.4 Baselines and protocol

All methods receive the same benchmark question and trajectory identifier. They differ in the record fields they access, the representation they produce, the verification they apply, and the text shown to the user. Table 10 summarises these choices. The controlled display comparison in Section 5 is the only experiment that fixes proposer inputs, proposed and accepted claims, verifier outcomes, retry behaviour, and answer availability while changing only the displayed text.

Table 10: Claim source, verification, and display policy. Free-text, retrieval, and tool-calling baselines read the record at the queried instant and show model-written text. Filter-and-regenerate and cycle-check methods use accepted claims from the common pipeline. The oracle skyline receives the gold claim requirements. Structured-output conditions and the oracle gate alone are ablations reported in Table 11.

Method	Claim source	Verification	Display
Free text, retrieval, tool calling	model or tool output	none	model prose
Gated prose	typed claims	five gates	model prose
Verified display	typed claims	five gates	closed renderer
Filter and regenerate	accepted claims	post-hoc check	model prose
Cycle check	accepted claims	read-back check	model prose
Oracle skyline	gold requirements	five gates	closed renderer

The ablations include structured output with and without rendering, which applies the output schema without the verifier; the oracle gate alone, which applies the evidence oracle without coverage, provenance, unit, or regularity checks; and a gated condition in which the model receives constraint

multipliers in its input. The oracle gate alone displays 209 answers that rely on a status claim withheld by the deployed verifier. The symbolic renderer receives gold claim requirements.

Tool calling retrieves record information through a multi-turn interaction. In the tool-calling baseline, the model writes the displayed answer. In the verified-display condition, the model proposes typed claims, which pass through the same gates and renderer used by direct claim generation; retry feedback is available for up to K attempts. This contrast evaluates the complete interface on a shared proposal carrier. The paired experiment in Section 5 separately isolates the display step by holding accepted claims fixed.

B.5 Prose regeneration from accepted claims

We compare the closed renderer with two policies that generate prose from accepted claims: filter-and-regenerate and cycle checking. Both retain model-written text as the final displayed response. The comparison uses the same accepted-claim representation, claim language, record checks, prompts, and retry budget across conditions. It therefore evaluates the final display policy under a common proposal representation, not the full released systems of the cited work.

These are adaptations of the published policies, with our claim language, checks and prompts, because the original methods were not released for this setting.

On the 1,456 test questions with the primary model, filter-and-regenerate displays answers for 0.736 of questions at selective risk 0.134. The cycle check displays answers for 0.650 at risk 0.067. The closed renderer displays answers for 0.880 at zero observed risk. Useful coverage is 0.233 for filter-and-regenerate, 0.236 for the cycle check, and 0.619 for the closed renderer. This experiment uses a separate jointly generated campaign because the two prose policies require regenerated answers; its coverage therefore differs slightly from the 0.886 reported for the main paired direct-generation condition.

On the larger model, useful coverage is 0.415 for filter-and-regenerate, 0.398 for the cycle check, and 0.578 for the closed renderer. The cycle check reaches selective risk 0.006 by displaying fewer answers. These results concern the final display policy under a common proposal representation. A model writes the displayed sentence in both prose-based conditions, and their checks depend on a model-based span or read-back comparison. The closed renderer instead produces the displayed text directly from the accepted claims and verifies the render–decode correspondence deterministically.

C Additional results and resource use

C.1 Full results and reporting

Table 11 reports every metric for the evaluated arms, pooled over model–domain runs. Risk and adequacy condition on displaying an answer; useful coverage is evaluated over all questions. The symbolic renderer receives gold claim requirements.

C.2 Matched component comparisons

The complete tool-calling comparison changes gating, rendering, and retries together. Table 12 separates these components using arms already produced in the campaign. The first three contrasts hold generation fixed, so their differences come only from verification or display policy. The final contrast changes the retry budget from one to three attempts and increases completion tokens by a factor of 1.61.

All intervals resample trajectories. The display-only contrasts reduce scorer-flagged risk without changing generation resources. The complete tool-layer comparison changes exposure substantially; its pooled useful-coverage difference includes zero, and on the primary model the per-domain differences range from -0.093 in HVAC to $+0.146$ in dispatch. The retry comparison quantifies the additional generation cost required to increase coverage after a failed proposal.

Table 11: Full primary comparison, pooled over evaluated model–domain runs. Risk and adequacy condition on answering. *Rej.* and UNK give refusal rates by gate verdict. † marks the symbolic oracle skyline, which receives gold claim requirements. Intervals are trajectory-resampled 95% intervals. Tool-calling rows pool 5 models and all other non-oracle rows pool 6.

Method	Cov.	Risk	Adq.	Useful	Rej.	UNK
free text	1.000	0.352	0.449	0.264 [0.25,0.28]	0.000	0.000
retrieval	1.000	0.243	0.710	0.517 [0.50,0.53]	0.000	0.000
tool calling	1.000	0.285	0.680	0.489 [0.48,0.50]	0.000	0.000
structured output	1.000	0.334	0.461	0.303 [0.29,0.31]	0.000	0.000
structured output, rendered	0.917	0.283	0.486	0.362 [0.35,0.38]	0.000	0.000
oracle gate alone, model text	0.815	0.223	0.668	0.448 [0.44,0.46]	0.081	0.000
gated and rendered, no retries	0.512	0.000	0.585	0.359 [0.34,0.38]	0.301	0.022
gated, model text	0.792	0.215	0.660	0.435 [0.42,0.45]	0.082	0.021
gated and rendered (ours)	0.792	0.000	0.660	0.540 [0.52,0.56]	0.082	0.021
symbolic renderer (oracle skyline)†	0.942	0.000	1.000	1.000 [1.00,1.00]	0.000	0.000
gated and rendered, multipliers in the feed	0.845	0.000	0.695	0.609 [0.59,0.63]	0.068	0.024
tool calling, gated, model text	0.800	0.286	0.749	0.466 [0.45,0.48]	0.121	0.029
tool calling + verified display, no retries	0.724	0.000	0.757	0.579 [0.56,0.60]	0.165	0.038
tool calling + verified display	0.800	0.000	0.749	0.608 [0.59,0.63]	0.121	0.029

Table 12: Matched component comparisons on the primary model, macro-averaged across domains. Values are shown as reference / comparison. Tokens is the completion-token ratio per record. The first three rows hold generation fixed.

Contrast	Change	Tokens	Coverage	Risk	Useful
tool → tool + layer, $K=1$	gates and renderer, one attempt	1.00	1.000 / 0.730	0.173 / 0.000	0.605 / 0.627
tool + gates, prose → tool + display	display only, tool carrier	1.00	0.813 / 0.813	0.265 / 0.000	0.492 / 0.639
gated, prose → ours	display only, direct carrier	1.00	0.886 / 0.886	0.253 / 0.000	0.497 / 0.620
tool + layer, $K=1$ → tool + display	retry budget, one to three attempts	1.61	0.730 / 0.813	0.000 / 0.000	0.627 / 0.639

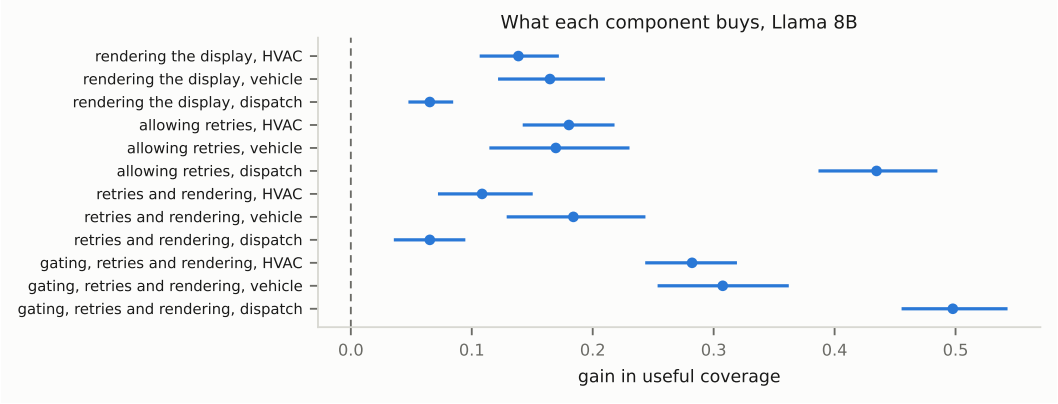

Figure 4: Component effects on useful coverage for the primary model. Markers show paired differences with 95% trajectory-resampled intervals for gating, display rendering, and retries. Hollow markers indicate intervals containing zero.

C.3 PAIRED DISPLAY TRANSITIONS IN EVERY CELL

The paired display comparison covers 33 domain–model cells, with 152 to 537 paired answers per cell. Every cell contains flagged-to-clean transitions and no clean-to-flagged transitions. The largest one-sided upper bound on a cell-level reverse-transition rate is 0.024. Every test trajectory contains

at least one flagged-to-clean transition on both carriers. The pooled one-sided upper bounds on the reverse-transition rates are 5.3×10^{-4} for direct generation and 6.4×10^{-4} for tool calling. These bounds treat paired answers as independent. Answers cluster within trajectories, so the conservative statement is at the trajectory level: with no reverse transition in any of the 60 trajectories, the probability that a trajectory contains one is at most 4.9 percent for the equally weighted mixture of the three domains.

C.4 HOSTED FRONTIER MODELS

Table 13: Hosted models: requested identifiers and settings. All four run the gated display pair with three attempts on the locked test split.

Identifier	Access	Reasoning effort	Output limit (tokens)
`gpt-5.6-luna`	command-line agent	high	not set
`claude-opus-5`	API proxy	medium	16,000
`claude-sonnet-5`	API proxy	medium	16,000
`gpt-5.5`	API proxy	medium	16,000

We repeated the direct-generation display comparison with a hosted reasoning model on the locked test split of all three domains. Table 13 lists the identifiers and settings of every hosted model. This is a separate campaign and enters no headline number. Prompts, gates, masks, retry budget, benchmark and scorer are those of the locked campaign, and both displays share one generation per question. The model ran with high reasoning effort through the provider's command-line agent rather than its API; the specification written before the run named medium effort and the API. That interface exposes no model revision, sampling seed or decoding setting, so this arm cannot be regenerated exactly. Its records are released with the other campaign records.

Of the 1,456 questions, 1,138 are answered by both displays, and the two displays carry identical accepted claims for every question. The scorer flags the gated prose in 0.128 of these answers: 0.077 in HVAC, 0.027 in dispatch and 0.276 in vehicle control. The pooled rate for the six open models is 0.214. By model it is 0.256 for Llama 3.1 8B and 0.222 for the quantised Llama 3.1 70B, 0.227 for Mistral 7B and 0.232 for Mistral 24B, and 0.145 for Qwen 2.5 7B and 0.194 for Qwen 2.5 32B. Within these families the rate does not fall consistently with size, so how leakage changes with model scale remains open. Rendering turns all 146 flagged answers clean and flags none that were clean, a paired difference with interval $[0.097, 0.161]$. Flagged-to-clean transitions occur in 42 of the 60 test trajectories. Reading flagged vehicle-control answers shows scorer errors among them, such as a correct rebuttal of a false premise counted as a false status claim. The prose rate is therefore a scorer-flagged rate, not a rater-judged error rate (Section 5.6). Retry feedback can state recorded values, as in every gated arm, so coverage here measures verifier-assisted answering rather than unaided accuracy.

Three further hosted models ran through the same runner at the prespecified settings of this arm: medium reasoning effort, a 16,000-token output limit and three attempts, through an API proxy. For each model the two displays carry identical accepted claims. Rendering turns 719 of 1,347, 448 of 1,156 and 223 of 1,208 answered pairs from scorer-flagged to clean, and flags none that were clean. We do not report these models' prose flag rates as leakage. The external judge gives calibrated paired effects of 0.140 to 0.146 for the four hosted models (Appendix D.3). In the flagged answers we read, the scorer often attaches a number to a neighbouring quantity, for example an occupant count read as a CO_2 reading, and longer answers give it more occasions to do so.

C.5 PROPOSAL QUALITY

Direct claim generation provides a simple proposal carrier for the display comparison. On the primary model, the gated-and-rendered condition displays 1,281 answers with coverage 0.886, adequacy 0.699, and useful coverage 0.620. An alternative proposer condition that supplies constraint multipliers to the model increases useful coverage to 0.659 at coverage 0.878. This condition also shows the model each constraint's status label as computed by the oracle, so its status claims can be copied from the checker rather than inferred. The headline arm keeps the sensor feed of the free-text and gated-prose baselines, so its comparisons isolate verification and display. The display guarantee

holds unchanged for either feed. This change affects proposal quality because it changes the information available to the model. In the verified pipeline, retries raise coverage from 0.522 to 0.886, and every displayed claim still passes the same checks.

C.6 SPAN-LEVEL POLICIES

Span-level verification checks displayed values while leaving the remaining proposition unconstrained. Table 14 compares a fail-closed policy, which refuses an answer containing a wrong displayed number, with a source-rendered policy, which replaces displayed values with record values. Both policies can still display unsupported statuses, comparisons, actions, or untracked quantities. The closed renderer cannot express these forms outside its accepted claim grammar.

Table 14: Span-level verification versus claim-level closure, pooled over all models. *Fail-closed* refuses an answer with a wrong displayed value. *Source-rendered* corrects displayed values from the record. *Still unsupported* counts answers containing an unsupported status, comparison, action, or untracked quantity.

Arm	Answers	Fail-closed			Source-rendered
		displayed	still unsupported	share	share
unverified prose	8,736	6,355	2,508	0.395	0.414
retrieval	8,736	7,008	2,446	0.349	0.379
tool calling	7,280	5,674	2,354	0.415	0.438
oracle gate, prose	7,119	6,124	2,319	0.379	0.399
structured output, prose	8,736	7,004	3,278	0.468	0.467
gated, prose	6,909	5,980	2,158	0.361	0.381
pooled	47,516	38,145	15,063	0.395	0.414
closed renderer	25,428	–	0	0.000	0.000

Across 47,516 prose answers, fail-closed verification displays 38,145 and leaves 15,063 unsupported propositions, a share of 0.395. Source rendering leaves a share of 0.414. For gated prose, which begins from claims accepted by the deployed verifier, the fail-closed share is 0.361. The closed renderer has zero such cases among 25,428 displayed answers because its grammar cannot express propositions outside accepted claims.

We also evaluate both policies as live arms. On the primary model, the fail-closed policy displays 1,190 of 1,456 answers; 192 of the 914 answers assessed by the span detector contain unsupported content, a share of 0.210. The source-rendered policy displays 547 answers, and 110 of 429 assessed answers contain unsupported content, a share of 0.256. On a second model, the corresponding shares are 0.213 and 0.315. These results show that correct displayed values do not ensure that the complete sentence is supported.

A third family keeps the model's prose and checks it one sentence at a time, in the manner of proof-carrying numbers and symbolic references. Marking the sentences whose spans no accepted claim covers leaves almost no scorer flag, but 74.6% of direct-generation answers and 75.7% of tool-calling answers, or 50.6% without lines of JSON formatting, still show an unchecked sentence whose causal, intent, or narrative content no check reads. A policy closer to proof-carrying numbers keeps every sentence and hides only numbers that no accepted claim covers. It leaves selective risk at 0.095 on both carriers, with useful coverage 0.564 and 0.600, because a false status, comparison or action statement contains no number to hide.

C.7 RELATIONAL-EXTENSION CAMPAIGN

We evaluate an untrusted typed-intent component for questions that require a model-relative relation. The component classifies the question, conditions proposal on the required relation type, and ranks candidate claim sets accordingly. Every selected claim still passes the same verifier and renderer, so an incorrect intent prediction can reduce adequacy or cause refusal but cannot add an unsupported displayed proposition. Relations are evaluated against the declared signed graph, and the gold rela-

Table 15: Relational questions in the separate frozen campaign. The sampled control and intent reranker use the same sampling configuration and differ only in whether candidate selection requires a relation. Risk is scorer-detected selective risk conditional on answering.

Domain	Questions	Control useful	Rerank useful	Rerank risk
HVAC	151	0.000	0.563	0.000
AV speed	91	0.000	0.505	0.000
Dispatch	102	0.000	0.255	0.000
Macro-average	–	0.000	0.441	0.000
Every question	1,456	0.592	0.696	0.000

Table 16: Intent reranker results on relational questions. Adequacy is conditional on answering.

Domain	Coverage	Adequacy	Refused	Tokens
HVAC	0.570	0.988	65	68
AV speed	0.505	1.000	45	69
Dispatch	0.255	1.000	76	31
Macro-average	0.443	0.996	–	–

tion labels use that same graph. This experiment therefore evaluates question routing and proposal, not a causal property of the recorded run.

On the locked relational subset of 344 questions, the intent-conditioned reranker increases useful coverage by 0.456 $[0.389, 0.516]$ relative to a sampled control without a relation requirement. The control has zero useful coverage in every domain. The reranker reaches 0.563 in HVAC, 0.505 in longitudinal vehicle control, and 0.255 in dispatch, for a macro-average of 0.441 (Table 15). It has zero observed scorer-detected risk in all domains. Conditional adequacy is 0.988 in HVAC and 1.000 in the other domains (Table 16).

On all 1,456 questions, the deterministic compiler identifies every relational question as relational and does not label non-relational answerable questions as relational. Its remaining errors concern unanswerable questions, which it sometimes labels as answerable. The component engages only for predicted relational questions. Useful coverage therefore rises from 0.592 to 0.696, a paired increase of 0.108 $[0.091, 0.124]$, while the 1,112 non-relational questions remain unchanged. Both arms of this campaign sample four candidates rather than decoding greedily, so its control differs from the greedy 0.620 of Table 3. Coverage falls from 0.899 to 0.774 because the reranker refuses when it cannot obtain a supported relation claim. Conditional adequacy increases from 0.640 to 0.878.

The relational extension is a separate frozen campaign, not pooled with the main results, that uses the deployed tolerances, thresholds, margins, regularity artifacts, parser guard and relation-unit policy.

The relation-intent classifier is evaluated on author-written templates and does not establish general natural-language understanding; the remaining gap lies in candidate generation and selection, not in the closed renderer.

The reranker was developed on the development split. An earlier diagnostic on the locked split, which used oracle question types and selected no threshold, prompt, model or decision rule, showed that acceptance-only commitment could stop at a supported numeric claim before proposing the required relation; this motivated ranking candidates by the required type.

C.8 Results by question type and perturbation

Table 17 reports useful coverage by question type, perturbation class and answerability for the primary model. An unsupported-entity question names a quantity the record does not declare, so the correct response is a refusal. Free text, retrieval and tool calling answer every such question and score 0; gated and rendered scores 0.373 and tool calling with verified display 0.267. On false premises, gated and rendered matches tool calling at 0.848, and tool calling with verified display

reaches 0.935. Unit traps are the weakest case for the layer: gated and rendered scores 0.476 and tool calling with verified display 0.444, against 0.587 for plain tool calling. Each question type is phrased by one of five author-written paraphrases. Across the five, useful coverage spans 0.082 for gated and rendered and 0.062 for tool calling with verified display, against 0.114 for plain tool calling. This measures sensitivity to wording inside the benchmark's templates, not transfer to an unseen question source.

Table 17: Useful coverage by question type, perturbation and answerability for the primary model on the test split, pooled over three domains. Columns are arms: free = free text, retr. = retrieval, tool = tool calling, gated, text = gated, model text, ours = gated and rendered (ours), tool + disp. = tool calling + verified display, skyline = symbolic renderer (oracle skyline). Useful coverage credits a faithful and adequate answer on an answerable item and a refusal on an unanswerable one.

Level	n	free	retr.	tool	gated, text	ours	tool + disp.	skyline
numeric	408	0.444	0.630	0.716	0.735	0.826	0.745	1.000
status	351	0.368	0.519	0.772	0.650	0.917	0.906	1.000
conjunction	353	0.210	0.578	0.810	0.547	0.686	0.836	1.000
relational	344	0.000	0.000	0.110	0.000	0.000	0.000	1.000
none	1,040	0.283	0.485	0.644	0.510	0.631	0.650	1.000
false premise	46	0.326	0.500	0.848	0.565	0.848	0.935	1.000
unit trap	63	0.175	0.333	0.587	0.397	0.476	0.444	1.000
stale timestamp	79	0.241	0.380	0.582	0.329	0.570	0.595	1.000
unsupported entity	75	0.000	0.000	0.000	0.373	0.373	0.267	1.000
negation	80	0.238	0.425	0.625	0.537	0.650	0.738	1.000
hedged	73	0.356	0.425	0.616	0.589	0.699	0.603	1.000
answerable	1,370	0.280	0.469	0.647	0.498	0.629	0.647	1.000
unanswerable	86	0.000	0.000	0.000	0.453	0.453	0.360	1.000

C.9 Every question in one outcome

Every test question is assigned exactly one terminal outcome, and the per-arm accounting ships with the release. For the deployed direct pipeline, 4,968 of 8,736 questions receive an answer inside the supported claim fragment, and *Nothing*, no parsable claim, is the main refusal category that a stronger proposer could reduce.

C.10 Statistical reporting

We distinguish three outcomes. A statement is *false* when the record supports the opposite claim, *unsupported* when the record lacks evidence for what the statement asserts, and *unverifiable* when the record cannot decide the statement in either direction. The faithfulness indicator Φ treats false and unsupported content as failures. The annotation audit reports unverifiable content separately because it describes a different reader-facing failure. The prose scorer detects candidate claims in text, whereas the structural scorer decodes the closed rendering grammar; reported scorer results retain this difference.

The two prespecified endpoints are reported with paired tests and trajectory-resampled intervals. Every p-value printed in this paper is an unadjusted two-sided exact McNemar value on paired answers; the multiplicity correction below applies to the secondary family and is reported only as the count of tests that pass it. For the display endpoint, no answer moves from clean rendered text to scorer-flagged prose in any domain; McNemar tests give $p < 10^{-21}$ in each domain. The useful-coverage endpoint compares tool calling with the verified display against plain tool calling on the primary model. The difference is $+0.146$ in dispatch, with interval $[0.118, 0.172]$ and $p < 0.001$. In vehicle control it is $+0.051$, with interval $[-0.006, 0.105]$ and McNemar $p = 0.044$. In HVAC it is -0.093, with interval $[-0.164, -0.030]$ and $p < 0.001$. The pooled difference spans zero.

Trajectory is the resampling unit because questions from the same controller run are correlated. Each domain–model cell therefore contains 20 clusters rather than several hundred independent questions. This design supports the large paired display effect within a cell but does not support a domain–model interaction analysis for useful coverage. We report cell-level intervals and do not claim an

interaction effect. At 80 percent power and a two-sided 5 percent level, the smallest detectable difference in useful coverage for the tool-layer contrast is 0.060 pooled over 60 trajectories. Per domain it is 0.038 in dispatch, 0.079 in vehicle control and 0.096 in HVAC, so smaller differences cannot be separated from zero with 20 trajectories per domain.

We correct comparisons across the full family of 666 domain, model, arm, and endpoint tests using the Benjamini–Hochberg procedure at false discovery rate 0.05. Of these tests, 504 satisfy the correction.

The artifact also contains question-level and record-level accounting, adequacy decompositions by entity, timestamp, and relation, assigned causes of inadequate answers, run-to-run variation, and risk–coverage curves across abstention margins.

C.11 Models, cost, and latency

Table 18 lists the evaluated models. All conditions use greedy decoding with a 768-token completion budget. Tool-calling conditions are omitted for Llama 3.1 70B because their multi-turn cost exceeded the available compute budget. Llama 3.1 70B is the only quantised model, so differences across model sizes also reflect numerical precision and should not be interpreted as scale-only effects. The model set was fixed before evaluation. Each run record stores the loaded weight revision, so an unintended model update is detectable. Under greedy decoding, repeating a cell under four inference seeds is a determinism check: exposure moves by at most 0.015, and every paired effect keeps its sign.

On the primary model, direct verified display has median end-to-end latency 1,123 milliseconds, compared with 863 milliseconds for retrieval. Tool calling has median latency 4,566 milliseconds without the display layer and 5,655 milliseconds with it. Verification and rendering add approximately 0.9 milliseconds per attempt and generate no tokens. These are medians of per-request latency on the HVAC test split, one question at a time; the additional end-to-end latency mainly reflects generation and retries, and Table 12 reports the matched generation-resource comparisons.

Table 18: Model configurations. All conditions use greedy decoding with a 768-token completion budget. Tool-calling conditions are omitted for Llama 3.1 70B because of multi-turn compute cost. Each run record stores the loaded weight revision.

Model	Params (B)	Precision	Decoding	Tool arms	Revision
Llama 3.1 8B	8.0	fp16	greedy	yes	`0e9e39f`
Llama 3.1 70B	70.6	4-bit	greedy	no	`1605565`
Qwen 2.5 7B	7.6	fp16	greedy	yes	`a09a354`
Qwen 2.5 32B	32.8	fp16	greedy	yes	`5ede1c9`
Mistral 7B	7.2	fp16	greedy	yes	`c170c70`
Mistral 24B	24.0	fp16	greedy	yes	`9527884`

C.12 Confirmatory split

The locked split stopped being confirmatory once two verifier rules changed after its outputs had been inspected (Appendix C.13). We therefore froze the system and wrote a preregistration before generating any confirmatory trajectory, question or answer. It has two amendments, both made before any model generation. The first replaced a master seed that overflowed the random-number generator. The second extended the cross-solver mask from four to all twenty trajectories per domain.

The split draws 20 new test trajectories per domain from a new master seed and excludes every real-data window the locked split used. Its benchmark has 1,577 questions. The primary model runs the gated, tool-calling and oracle arms with greedy decoding and unchanged prompts, budgets and thresholds. The regularity and cross-solver artifacts are recomputed on the new trajectories and bound to them by digest. Unlike the locked split, the cross-solver mask is live. The unchanged analysis scores the records, and all three endpoints are reported as they came out.

On the display endpoint, rendering moves 418 of 1,440 gated answers and 338 of 1,291 tool-calling answers from scorer-flagged prose to clean text, and no answer moves the other way in any domain or trajectory. The prose arms are flagged at 0.290 and 0.262.

The pooled useful-coverage difference of tool calling with the verified display over plain tool calling is 0.037, with trajectory interval $[-0.002, 0.076]$, which includes zero. It is $+0.148$ in dispatch, with interval $[0.106, 0.193]$, and $+0.064$ in vehicle control, with interval $[0.000, 0.129]$ and $p = 0.004$. It is -0.077 in HVAC, with interval $[-0.123, -0.033]$. In each domain the confirmatory difference has the same sign as the locked-split difference. Neither split separates the pooled difference from zero.

The HVAC loss has a measured partial source. Consider the HVAC questions that plain tool calling answers usefully and the layer does not. The regularity gate withheld a status label in 45 of 109 on the locked split and 83 of 115 here. The rest were refused for a status or value mismatch, or answered inadequately. Rank deficiency is also more common on the new HVAC trajectories. It affects 2,462 of 5,760 instants, or 42.7 percent, against 2,019 of 8,640 on the locked split.

The secondary endpoint measures the mask end to end. HVAC and vehicle control were rerun with the mask off and everything else fixed. Unlike the offline re-decision in Appendix D.1, this includes retries. In HVAC the mask lowers coverage by 0.006 for gated generation and 0.009 for tool calling, changing 4 and 13 of 640 answers. The useful-coverage cost is 0.005 and 0.006, with $p = 0.25$ and $p = 0.29$. In vehicle control no answer changes, which reflects sample size rather than a free mask. The mask lists 5 vehicle instants, the benchmark queries 1 of them with 2 questions, and neither asks a status. In HVAC, 25 questions fall on 16 of the 194 masked instants, and 8 ask a status.

The locked-split tables report the system as it was run, with the mask off; the confirmatory split reports it on. To measure its cost, we extended the mask by the 3 HVAC instants where a solver setting flips a definite status label (Appendix A.2). These include the two instants of a different local optimum. The extended mask lists 116 test instants and acts on 62 locked test questions, 32 in HVAC and 30 in vehicle control. We ran these questions with the mask on and off, for all six models and every status-checked arm each model runs, under one system version and otherwise unchanged settings. The mask turns 391 of 2,418 answers into refusals and 17 refusals into answers. Useful coverage on these questions falls from 0.510 to 0.354. Three faithfulness verdicts change, all in the gated arm that displays model text, and no rendered answer is flagged. Spread over each domain's test questions, the mask changes every arm's macro-averaged useful coverage by at most 0.007. For the primary model the largest change is 0.017, in HVAC. With the mask on, no cell shows a reverse display transition, and the useful-coverage endpoint is $+0.013$, interval $[-0.033, 0.057]$. The locked-split tables and the comparisons between arms therefore stand. The mask-cost endpoint links the two configurations.

C.13 Changes after inspecting locked outputs

Two verifier rules were adopted after the locked split's outputs had been inspected. A one-sided bound is accepted only when the recorded value satisfies the displayed comparator literally, while equality claims keep their declared tolerance. A multiplier-qualified status claim also requires the regularity artifact to accept the active set (Appendix A.1). The benchmark requires no bound claim for adequacy, so the literal rule has no measured completeness cost. Because both rules followed inspection, the preregistered confirmatory split tests them on fresh trajectories (Appendix C.12). Thresholds, tolerances and policy settings were selected on development trajectories only. The locked-split tables run with the cross-solver mask off, and Appendix C.12 measures what the mask costs. The recorded-value renderer was introduced after an external judge read the stored answers; it changes no verdict and no scorer decision (Section 5.6).

C.14 A domain outside control: SEC financial filings

The interface needs typed values, declared units and a question, not a controller. We therefore built one domain outside control from public data: annual 10-K facts for 30 companies, fiscal years 2019 to 2024, from the SEC EDGAR XBRL interface, with the accession number of every value kept. Companies were drawn by a seeded shuffle and split into 10 development and 20 test companies. The domain declares no constraints, so every claim is a numeric reading; the equality tolerance is 0.5

USD million and 0.005 USD per share. The benchmark has 400 test questions, and the specification was frozen before the full run.

With accepted claims held fixed, rendering lowers the flagged rate from 0.022 to 0 for direct generation, interval $[0.008, 0.035]$, and from 0.298 to 0 for tool calling, $[0.253, 0.342]$, with no answer moving from clean to flagged. Tool calling with the verified display reaches useful coverage 0.792, against 0.603 for plain tool calling, a paired difference of 0.190 $[0.138, 0.245]$. After the run we fixed deterministic scorer artefacts, such as digits in a company name read as a figure, and rescored the same 2,400 frozen records. Under the scorer version that ran, both results hold and are larger: the flagged rates fall from 0.081 to 0.003 and from 0.328 to 0.009, and the useful-coverage difference is 0.212 $[0.142, 0.297]$. The domain has one model, one question source and no status claims, so it tests the display layer and the numeric verifier only.

C.15 A PUBLIC BUILDING TRACE: BOPTEST

BOPTEST provides building emulators with published weather, occupancy, comfort bands and price profiles. We used its `bestest_hydronic_heat_pump` test case, whose single input modulates a heat pump. The plant, weather, occupancy, comfort band and prices come from BOPTEST; the MPC controller, its grey-box prediction model and the choice of 48-hour windows are ours. The prediction model was fitted on a separate excitation run that overlaps no trajectory. Thirty trajectories at a 15-minute step give 5,760 committed solves, and the trajectories covering the two official BOPTEST test periods are all in the test split. The unchanged builder produced 558 test questions. The LICQ audit finds rank-deficient active sets at 73 of the 5,760 instants. A preregistration was frozen before any model generation.

The primary model ran five arms with the settings of the main campaign. Rendering lowers the flagged rate from 0.102 to 0 for direct generation, paired interval $[0.074, 0.132]$, and from 0.148 to 0 for tool calling, $[0.116, 0.182]$, with no answer moving from clean to flagged. Useful coverage is 0.640 for plain tool calling and 0.654 with the verified display, a difference of $+0.014$ $[-0.011, 0.041]$, while the display refuses 0.131 of questions. Unlike the in-project HVAC trajectories, the controller cannot always reach the comfort band here, during recovery from night setback and in warm spells. The scorer was frozen by the preregistration. It reads some statements as the wrong quantity, such as a shadow price read as an electricity price, and this domain has no human audit.

D ASSURANCE AND REPRODUCIBILITY

D.1 IMPLEMENTATION ASSURANCE DETAILS

Deliberate corruptions of declared magnitudes, artifact binding and renderer version each cause refusal; log integrity is not tested, because the records are generated within this project. These tests concern record handling, verification, rendering, and decoding. They do not establish that the supplied record is authentic, that the optimisation model is correct, or that the declared claim semantics capture plant behaviour or user interpretation.

CONFORMANCE AND EDGE CASES

Table 20 reports tests of the main implementation boundaries. The renderer receives only accepted claims, so prompt content and model prose cannot directly reach the display. The verifier fails closed for malformed timestamps and units, non-finite values, comparator-boundary cases, and infeasible records under the declared semantics. Under consistent rescaling of a constraint or objective and its declared magnitude, no status verdict changes. We also test corrupted records to distinguish verifier behaviour from the separate requirement that logs be authentic.

The point oracle accepts 25 of 74,144 false claims under corrupted-record tests. A declared uncertainty set removes these accepts by converting the affected cases to abstentions. Replay and frozen-log tests expose a separate requirement: an internally consistent record can describe a different run. Preventing this failure requires authenticated logging and provenance controls outside the claim verifier.

Table 19: Examples from the interface: a rejected one-sided bound, an accepted reading, a status withheld when the multiplier is not identifiable, unsupported free prose, a verified but inadequate answer, and a refusal whose evidence is insufficient for the requested status. The final three examples are drawn from campaign records.

Operator question	*At 05:00, was `p_G10` at most 10.0 MW?* Here `p_G10` is the dispatched output of generator 10, in megawatts.
Recorded evidence	`p_G10` = 10.096 MW at that instant; declared equality tolerance 0.1 MW
Proposed claim	`p_G10` $\leq$ 10.0 MW
Gate decision	REJECT: the record decides a one-sided claim, and $10.096 \not\leq 10.0$
Displayed answer	"I cannot verify the requested statement from this recorded run under the declared checks."
Same instant, asked as a reading	*What did `p_G10` read at 05:00?*
Displayed answer	"At 1998-11-04 05:00:00, the Dispatched output of generator G10 (`p_G10`) was 10.096 MW."
A status claim at a flagged instant	*Was the temperature constraint binding?* at an HVAC record the LICQ audit lists as non-unique. Here `lambda_T` is the multiplier of the zone temperature constraint.
Without the regularity gate	ACCEPT, displayed as "... the temp constraint (`lambda_T`) was binding at the recorded solution (normalised multiplier above threshold)."
With the gate, as deployed	UNKNOWN, displayed as the refusal above
Unverified prose, same record	*Tell me whether the CO_2 constraint was active at 17:30.* Displayed: "The CO_2 constraint is likely not active at 2019-12-14T17:30:00 because the room CO_2 level (793.6 ppm) is below the typical threshold for CO_2 control." The recorded solve does not establish this statement.
Verified but inadequate	*Explain the effect of solar radiation on indoor air temperature.* Displayed: "At 2019-12-15T12:10:00, the Incident solar radiation intensity was 485.67 W/m^2. At 2019-12-15T12:10:00, the Indoor air temperature was 21.0 °C. No additional factual statements are verified against this recorded run." Every sentence holds of the record and none answers the question
A refusal on an undecidable status	*Was the CO_2 constraint binding at 2019-11-01T22:20:00?* Displayed: "No answer is displayed: the recorded run does not settle this question under the declared status checks." The timestamp and constraint are available, but the multiplier-qualified status is not identifiable.

CROSS-SOLVER AUDIT

We rebuild every sampled instant as posed by the controller and re-solve it with an independent solver configuration. Dispatch uses an interior-point solver and dual simplex; the nonlinear domains use a monotone barrier method without warm starts. The deployed status rule is then applied to both solutions. The audit covers all recorded steps from the 20 locked test trajectories in each domain. The solvers support the same status sentence at all 18,800 dispatch constraint-instants. They differ at 13 of 20,000 vehicle constraint-instants, with no pair supporting contradictory status sentences. In HVAC, they differ at 178 of 69,120 constraint-instants, including 137 pairs with contradictory supported status sentences.

The HVAC contradictions arise from materially different primal solutions of the same nonconvex problem, not from near-threshold ties. The LICQ mask does not flag them because it tests uniqueness of the dual representation at a recorded solution, whereas these cases reflect differences between solver-selected primal solutions. We report an optional offline withholding analysis for such cross-solver disagreements. It is not enabled in the deployed configuration and is not part of the display guarantee.

OPTIONAL CROSS-SOLVER WITHHOLDING ANALYSIS

Among 24,729 status claims displayed by the five gated-and-rendered arms, 145 rest on an instant the cross-solver mask lists: 75 of 9,613 in HVAC, 70 of 6,535 in vehicle control, and none of 8,581 in dispatch. Re-deciding frozen records with the mask lowers coverage and useful coverage by 0.005 in

Table 20: Implementation conformance and numerical edge cases, pooled over three seeds and three domains. *Wrong way* counts outcomes that violate the declared verifier or renderer semantics. Log tampering is outside the verifier contract: internally consistent replayed or frozen records can remove true claims without creating a verifier-detectable contradiction.

Case	What is varied	Trials	Wrong way	True claims lost
prompt injection	payload in the question, in a claim string field and in model prose	864	0	–
comparator boundaries	every comparator at the tolerance boundary and one ulp either side	30,240	0	–
timestamp aliasing	a value true at another timestamp, stamped with this one	2,857	0	–
unit restatement	the same quantity restated in another unit, converted and relabelled	4,968	0	–
solver residual	a solve labelled optimal whose residual is not	3,960	0	–
objective rescaling	the objective scaled with its declared magnitude, and the control leaving that magnitude stale	23,760	0	–
log tampering	replay: every record carries the payload of the record one step later	864	–	230
frozen	every record carries the payload of the last record	864	–	598
rendering and decoding	enumerated over the grammar, sampled over the numeric axis	168,973	0	–
independent verifier	every stored decision re-derived from the declarations	133,463	0	–

HVAC and 0.007 in vehicle control. The deployed regularity mask withholds none of these claims, and the verifier reproduces all 1,020 HVAC, 787 vehicle, and 747 dispatch campaign verdicts.

Solver stability and reformulation checks

Cross-solver differences are not explained by sensitivity to nonlinear solver starts. Across 123 sampled nonlinear-domain instants and 492 perturbed starts, no re-solved trajectory changes by more than 10^{-3} in relative norm and none of 4,476 status tokens changes. Condensed, slack-variable, shifted, and rescaled formulations agree with the deployed formulation except for one threshold case. Shifting the initial state by one quarter changes 6,125 HVAC status sentences, confirming that the status rule responds when the underlying control problem changes.

Independent verifier and renderer audit

An independent verifier, implemented without imports from the deployed verifier, re-decides 133,463 stored claims. It agrees on 133,421 claims. The remaining 42 differences are REJECT versus UNKNOWN refusal subtypes and do not change whether text is displayed. The independent implementation makes three rules explicit: multiplier-derived statements require the status conditions, two-sided constraints resolve through their declared halves, and equality comparisons use an explicit tolerance. It also emits a certificate for each displayed answer. A third checker, sharing no code with either verifier, rejects every corrupted certificate in the test suite.

The renderer–decoder suite evaluates 168,973 cases and finds no incorrect round trip. For the recorded-value renderer the suite adds a commitment check on the recorded traces and evaluates 234,553 cases, again with no failure. Enumeration is exhaustive over the finite grammar for each fixed domain; numeric and string slots are tested by property-based cases. The suite found two defects before the final campaign: an empty unit string could render like an omitted unit, and one window template could absorb its closing phrase. Both were corrected before the reported experiments. Test inputs and the complete catalogue are released with the artifact.

Table 21: Blinded audit by two control engineers on 120 items, refusals included. False-statement rates are over all items of an arm and are shown for either rater, both raters, and author adjudication of disagreements. *Unverifiable* counts answers that assert content the supplied record cannot settle; *adequate* counts answers containing all claims required by the question.

Arm	Items	False statement			Unverifiable	Adequate
		Either	Both	Adjudicated		
unverified prose	14	0.357	0.286	0.286	0.429	0.357
gated, prose	22	0.227	0.136	0.182	0.818	0.409
ours	20	0.050	0.000	0.000	0.050	0.550
tool + display	15	0.067	0.000	0.000	0.000	0.467

D.2 Internal blinded scorer audit

The automatic faithfulness and adequacy scorers measure different properties and are evaluated separately. Table 21 reports a blinded audit by two control engineers of 120 displayed answers. A statement is *unverifiable* when the supplied record does not settle it. An answer is *adequate* when it provides every claim required by the question. The audit is an internal validation study of scorer behaviour and displayed-answer interpretation.

The sample is stratified across three domains, seven arms, five question types, and four perturbation classes, with separate strata for refusals and perturbed questions. The engineers scored items independently in different orders using the released written guide. Each item contained the question, displayed answer, relevant readings, and declared constraint information, but omitted the method, model, gold claims, and automatic-scorer decision. Rendered answers and prose remain visually distinguishable, so blinding does not remove all cues to output form. The per-arm sample sizes range from 10 to 22 items and support qualitative comparison rather than precise arm-level estimates.

Raters agree on 0.942 of false-statement labels, 0.917 of adequacy labels, and 0.867 of unverifiability labels, corresponding to Cohen's κ values of 0.777, 0.834, and 0.651. An author adjudicated field-level disagreements against the evidence supplied with each item. The released artifact contains both rater labels, adjudications, and the annotation guide. The author adjudication is a limitation of the audit; the table therefore reports the either-rater and both-rater rates alongside the adjudicated rate.

Against adjudicated adequacy labels, the automatic adequacy scorer has accuracy 0.750, precision 0.875, and recall 0.583 for inadequate answers. It misses 25 of 60 inadequate answers and over-flags five. Automatic adequacy is therefore an optimistic measurement of responsiveness under this scorer. The faithfulness scorer has precision 0.409 and recall 0.562 against the corresponding labels, with errors concentrated in prose. Among 53 rendered items, refusals included, the scorer and adjudicated labels agree on all items; among 67 prose items, agreement is 0.701 (Table 22).

The audit does not estimate a corpus-wide correction to the 3,067 scorer-detected paired transitions because it is stratified by the scorer's output and contains only 18 scorer-flagged items. It estimates agreement within the sampled strata. The 44 answered rendered items fall into two groups. In the 29 from the two verified-display arms, adjudication found no false statement. One rater flagged 1 of 17 answers from our arm and 1 of 12 from tool calling with display. Table 21 reports these as either-rater rates over all items. The 15 answers from unchecked rendering carry 4 adjudicated false statements, and the scorer flags all 4, so rendering unchecked claims does not remove false content. The audit therefore supports the implementation-level display property for rendered answers and shows that automated prose measurements remain noisy.

Table 22: Automatic faithfulness scorer against adjudicated labels, by output form. *Agree* counts matching scorer and adjudicated decisions.

Displayed answer	Items	Agree	Rate	Missed	Over-flagged
rendered from the grammar	53	53	1.000	0	0
written as prose	67	47	0.701	7	13

D.3 DETERMINISTIC SCORER CONTROLS

Faithfulness measurements use a span scorer for prose and a structural scorer for rendered text. Table 23 evaluates both on deterministic controls. A rendered gold claim set is the positive control, and the remaining rows inject errors into otherwise valid text.

Table 23: Correct scorer decisions on deterministic controls, pooled over three domains. For the positive control, a correct decision accepts the unmodified rendered gold claim set. For all other rows, a correct decision detects the injected corruption.

Control	Cases	Span	Structural	Either
positive control	900	900	900	900
extra clause	900	900	900	900
flipped status	444	117	444	444
hedged wrong value	456	394	0	394
wrong timestamp	445	221	0	221
wrong unit	445	36	0	36
wrong value	456	394	456	456

The structural scorer is defined for the closed grammar, whereas the span scorer identifies candidate claims in free prose. Both are evaluation instruments outside the trusted computing base. Unit enforcement is checked directly by the verifier because a wrong unit can evade text-level scoring.

Table 24 measures the span scorer without labels on the paired arms. On both measures the scorer reads rendered text at least as well as prose in five of six domain–carrier cells; in HVAC direct generation the prose binds slightly more numbers.

Table 24: Label-free scorer controls on the paired arms of the reported campaign. *Reach* is the share of displayed numbers the span scorer binds to a declared quantity. *Catch* is the number of answers flagged after one correctly bound numeric value is moved outside all declared tolerances.

		Reach		Catch	
Carrier	Domain	rendered	prose	rendered	prose
Direct generation	HVAC	0.645	0.658	209/210	111/135
Direct generation	Vehicle control	0.889	0.843	250/250	243/250
Direct generation	Dispatch	1.000	0.920	211/211	217/220
Tool calling	HVAC	0.587	0.475	209/210	163/183
Tool calling	Vehicle control	0.843	0.590	250/250	225/245
Tool calling	Dispatch	1.000	0.775	218/218	180/213

EXTERNAL JUDGE

The judge is the instruction-tuned Gemma 2 27B, run greedily with one fixed prompt that shows the question, the record and the answer. Its specification was frozen before any judge output was read. On the 120 audit items it agrees with the adjudicated false-statement label on 0.867 of items, with precision 0.500 and recall 0.750. We judged a stratified sample of 657 paired answers over 33 domain, model and carrier cells, weighted to the paired population, and 588 pairs from the four hosted models. Rates are corrected with the judge's predictive values on the audit items, and intervals resample trajectories. On the text as run, the corrected paired effect of rendering is 0.145, interval $[0.039, 0.252]$, for direct generation and 0.099, interval $[0.028, 0.182]$, for tool calling.

We read every rendered answer the judge called false. Most are judge errors: a value that matches the record at the precision shown, or a status sentence the record supports. Seven are real, one in direct generation, five in tool calling and one from a hosted model, weighted rates of 0.003 and 0.014. Each displays a proposed value that the equality tolerance admitted, either a few hundredths off the record or the value the question attributed to another time. Under the recorded-value renderer we judged the 623 rendered answers whose text changed again. None is judged false, and the stored text of the same answers, judged again in the same run, keeps 615 of its 623 verdicts. The corrected paired

effects become 0.212, interval $[0.120, 0.311]$, and 0.114, interval $[0.045, 0.195]$, and lie between 0.140 and 0.146 for the four hosted models. The audit items were rendered by the earlier renderer, so a rendered answer the judge does not flag is corrected with the miss rate those items give, which is zero.

D.4 Display completeness and reader-facing scope

The interface verifies individual claims but does not guarantee that their set is complete for the question or for the underlying control event. The adequacy taxonomy identifies 319 answers missing a claim required by the benchmark, but it cannot identify an omitted fact outside the gold claim set. A displayed answer can therefore be faithful to every included claim while omitting information that would matter to an operator.

The renderer shows a repeated claim once. As run, 55 of 1,281 displayed answers repeated a sentence verbatim, with 1.72 sentences on average and at most 20. The normal form now keeps the first occurrence of each claim, so the round trip still holds. Re-rendering drops 78 sentences, and answers contain 1.65 sentences on average and at most 9. Recorded values are printed at full stored precision so that the decoder recovers them exactly, which lengthens a changed sentence by about a dozen characters. Rendered answers are also short: the median is 20 words, against 68 for the gated prose answer to the same question, and the rendered answer is shorter on 99.5 percent of questions. These are the only reading-load properties measured; readability for operators is not measured.

The words used in a display can be interpreted more broadly than their declared semantics. A reader may take a verified statement about a recorded solve as a statement about plant behaviour. Likewise, *binding* is a multiplier-qualified label and is narrower than the residual-only engineering sense of an active constraint. The display avoids the word "certified," identifies the recorded run in its scope statement, and includes the declared status qualification in the rendered sentence. In HVAC and vehicle control the status sentence also names its scope, as in "was binding at the recorded solution." Re-rendering the stored answers with this template changed no accepted claim sequence, so every rendered-arm result is unchanged. The rendering and decoding row of Table 20 is the conformance suite run on this grammar.