

PERMISSION-SENSITIVE REPAIR OF SYNTHETIC WORKOUT SCHEDULES

Armaan Gupta, Arivuchelvan Gnanasekaran
Independent Researchers

Abstract : When a scheduling request is inconsistent, finding a feasible alternative does not establish permission to change the user's requirements. We evaluate this distinction on 384 scripted workout-scheduling conversations derived from 48 base instances in six structural families. Seven variants preserve each original constraint set while changing waiver permissions; an eighth is a feasible control. Two model snapshots are tested with direct generation, direct review, formal compilation, and compilation review, yielding 3,072 observations scored by an exhaustive reference. Under the original alphabetized output schema, compilation achieved 99.7% versus 45.3% behavioral success for GPT-4.1-mini and 82.3% versus 41.1% for GPT-5.4-nano. A post-main control adds 1,536 observations with source schema order restored. The compilation advantage persisted, but narrowed from 54.4 to 21.9 percentage points for mini; it changed from 41.1 to 44.5 points for nano. Review did not consistently improve success, and correct behavior did not guarantee valid conflict certificates or faithful extraction. The contribution is an auditable matched evaluation separating permission-sensitive repair, extraction, execution, and interface effects. Results are limited to these templates, snapshots, and configurations; they establish neither clinical safety nor general superiority over strong reasoning systems.

IndexTerms – Constraint satisfaction, language models, permission-sensitive planning, minimum repair, structured outputs.

1. INTRODUCTION

Consider a request for four sessions, at least 30 minutes per session, with at most 100 minutes in total. The request is inconsistent. Reducing session duration or increasing the time budget can make a schedule possible, but these edits are not interchangeable if the user permits only one of them. If the user later withdraws that permission, the original request remains inconsistent and the planner should return a blocked result rather than silently change a requirement. Ranking a requirement as less important also need not authorize its deletion.

This study asks whether explicit compilation of both requirements and current waiver permissions improves this behavior relative to direct language-model planning. Formal planning, preference-aware infeasibility repair, and permission enforcement are established approaches. Our aim is narrower: to make their interaction measurable under controlled permission changes and to distinguish correct actions from correct explanations of infeasibility.

The study provides three concrete artifacts. First, a generator creates matched conversations with exact reference statuses, all minimum permitted waiver sets, and executable plan checks. Second, a paired two-model experiment compares four procedures under the same task semantics. Third, the analysis separately reports behavioral correctness, certificate correctness, and extraction fidelity, making it possible to locate failures rather than summarize them as a single planning score. A separately labeled exploratory control also audits a request-serialization choice discovered after the main experiment. These are incremental evaluation contributions, not claims of a new general-purpose repair algorithm.

2. RELATED WORK

Hao et al. formalize natural-language planning requests and use verification tools to identify infeasibility and propose modifications [1]. SafeRun separates LLM interpretation from solver enforcement for running plans; its evaluated request pairs are feasible, leaving conflicting requests outside that evaluation [2]. Thus neither solver-assisted planning nor a fitness application is itself novel.

AWARE-US studies infeasible query repair guided by inferred constraint importance [3]. The present task instead supplies explicit permissions and treats importance without permission as insufficient. ConInstruct evaluates conflicting instructions and distinguishes conflict detection from subsequent response behavior [4]. Our executable outcomes add checks of the delivered schedule and the cardinality of authorized clause deletion, rather than relying on verbal acknowledgment alone.

Recent work further narrows the novelty claim. From Errors to Proofs examines formalization fidelity and minimal-core feedback, including comparisons with direct reasoning [5]. Dead Text or Binding Clause? studies revoked constraints and compiled dialogue state [6]. FAVA separates extracted permissions from formal authorization and enforcement [7]. DRIFT-Bench evaluates multi-turn constraint consistency with solver checking in satisfiable trajectories [8]. The ACS benchmark already includes workout-related constraint assessment [9]. We do not claim to discover the extraction bottleneck, revocation failures, or permission compilation. The distinctive artifact here is the matched combination of infeasible scheduling, explicit clause-level waiver permissions, exact minimum repair, and separate behavior/certificate evaluation. This targeted literature assessment does not prove that every aspect of the combination is unprecedented.

Schema design is also an established concern. Lin studies schema instructions and intermediate reasoning fields [10]; OrderBench separates structural validity from executable semantics in ordering agents [11]. Constrained Sensitivity Lab reports accuracy changes under reordered structured-output fields [12]. Our supplementary experiment is therefore a study-specific robustness check, not the first observation that schema presentation can affect model behavior.

3. TASK AND REFERENCE SEMANTICS

A schedule assigns each of five days, numbered 0 through 4, either no session or one session. A session lasts 20, 30, or 40 minutes and uses one of three abstract circuit labels: bodyweight, band, or dumbbell. At least one session is required. All circuit types are

globally available; an allowed-circuit clause is a synthetic restriction that can be waived only when authorized. The domain has 10 choices per day and therefore $10^5 - 1 = 99,999$ nonempty schedules. The task does not model training adaptation, injury risk, equipment ownership, or medical appropriateness.

Let D be this finite domain, C the set of numbered clauses, and P a subset of C whose deletion is currently permitted. A valid repair is a set W contained in P for which at least one schedule x in D satisfies every clause in C outside W . The objective minimizes $|W|$. We accept every waiver set achieving that minimum, and every schedule satisfying the retained clauses. Clause deletion is the entire edit language: a one-minute budget change is not automatically a cheaper repair than deleting a session-count clause. Global domain constraints cannot be waived.

The reference status is feasible when all clauses in C are jointly satisfiable, repaired when they are inconsistent but a permitted deletion makes them satisfiable, and blocked when no permitted deletion suffices. Equivalently, an inconsistent request is repairable exactly when the protected set C outside P is satisfiable: deleting all permitted clauses supplies a witness to existence, after which subset search determines the minimum. This elementary property is used as a consistency check, not offered as a novel theorem.

A response also supplies a conflict certificate. For a repaired answer, it is an inclusion-minimal unsatisfiable subset of the original C . For a blocked answer, it is such a subset of the protected clauses. Removing any single member must restore satisfiability. Inclusion minimality is distinct from minimum cardinality. Feasible answers have no certificate or waivers; blocked answers have no schedule or waivers.

Seven clause types express exact session count, a total-minute upper bound, a per-session duration lower bound, allowed days, allowed circuits, a lower bound on one circuit's session count, and a minimum difference between each pair of session day numbers. The generation and compilation schemas preserve rule identifiers so permissions remain attached to the same clauses after changes in presentation order.

4. EVALUATION DESIGN

4.1 Dataset construction

A 30-case development pilot used six base problems and five variants per problem. It informed implementation fixes and the choice to separate behavioral success from certificate success. Its scores are reported only as development information in the accompanying artifact and are excluded from the main results.

The main generator uses seed 2026090901 and samples eight distinct base instances in each of six families: time-budget conflict, insufficient allowed days, spacing conflict, circuit restriction conflict, incompatible circuit minima, and coupled time/day conflicts. Numeric values, allowed sets, rule identifiers, rule order, and wording differ from development examples. Exact original development clause sets are excluded after removing identifiers and ordering. Nevertheless, many main instances are isomorphic; this is generalization within known templates, not to unseen task families.

Each base instance has eight variants. Allow A and allow B permit different clause deletions. Allow all permits deletion of any numbered clause. Deny permits none. Revoke grants A and then cancels all permissions. Switch grants A and then replaces it with B only. Priority deny describes selected clauses as less important but explicitly keeps them mandatory and denies waiver permission. The feasible control changes the conflicting clauses and allows no deletion. There are 192 repaired, 144 blocked, and 48 feasible references. All conversations contain five messages, including two fixed assistant acknowledgments; the evaluated model did not generate these intermediate turns.

The coupled family illustrates why minimum repair matters. If too many sessions must fit both a restricted day set and a tight time budget, deleting the count clause can resolve both conflicts at once. Permission to delete only the time and day clauses instead requires two deletions. The scoring therefore cannot use a single preferred repair independent of permission.

4.2 Procedures and models

Direct generation returns a status, schedule, waived identifiers, and conflict identifiers. Direct review receives the original dialogue and its first answer, then makes one correction pass. Compilation extracts all original clauses and the current permitted identifier set into a constrained JSON schema. A deterministic search procedure then generates the answer. Compilation review audits the first extraction against the original dialogue before the same search procedure runs. Neither review receives oracle feedback; compilation review does not see a solver result.

The two snapshots are gpt-4.1-mini-2025-04-14, with temperature zero, and gpt-5.4-nano-2026-03-17, with reasoning effort set to none and temperature unspecified. Both use a maximum of 2,048 output tokens and strict structured outputs. Model availability was checked on 9 September 2026. These are inexpensive configurations from a single provider, not a survey of frontier reasoning capability. Direct generation is instructed to reason carefully, but does not receive tools or an explicit scratchpad; review alone does not equalize all computational differences with exact search.

Each model processes each of the 384 cases once per method, producing 3,072 observations. Initial direct answers are shared with direct review, and initial compilations with compilation review. Consequently, paired conditions are dependent and review costs include their shared initial call. Execution order is deterministically shuffled, with eight workers and at most two new requests per second per model. Requests and raw responses are retained, with no credential included in the research artifact.

4.3 Verification and metrics

The answer-producing backend uses pruned recursive search. The reference implementation independently enumerates schedule features and uses bit masks to evaluate satisfiability, all minimum waiver sets, delivered plans, and conflict certificates. Both implementations agree on every generated main reference and its witness. An additional model-independent audit checked all 480 possible permission subsets across the 48 original base instances; statuses and minimum repairs agreed throughout. Development checks cover rule subsets, 100 seeded randomized boundary cases, invalid output, and paired-run bookkeeping. This is implementation independence, not independent human validation: AI assistance produced both implementations and the templates.

The primary outcome, behavior success, requires a well-formed response and the correct status. Format validity includes admissible session fields, unique in-domain days, and allowed duration and circuit values, not merely parseable JSON. An out-of-domain or duplicate-day schedule therefore fails this outcome before clause-level scoring. For feasible or repaired answers, the supplied schedule must satisfy every retained clause and the waived set must be an authorized minimum repair. For blocked answers, both schedule and waived set must be empty. The secondary strict outcome also requires the appropriate valid conflict certificate. A response can therefore pass behavior while failing strict scoring, but cannot pass behavior merely by declaring the correct status.

We additionally report status accuracy, protected-clause violations, exact extraction of rules and permissions, and success across all seven permission variants of a base instance. A protected-clause violation is assessed on a nonempty delivered schedule or a nonblocked answer; correctly empty blocked responses are inapplicable. Invalid responses count as primary failures. The compilation parser clears fields unused by a clause type before validating its schema. It never changes operative parameters, identifiers, or permissions, and preserves the original extraction. This normalization was adopted during development because unused-field conventions otherwise rejected semantically equivalent clauses.

A local protocol fixed the main cases, prompts, metrics, and analysis before evaluation calls. It was not publicly preregistered; the protocol checksum record was created after execution began. The four primary contrasts compare compilation with direct generation, and their respective review variants, within each model. Descriptive 95% percentile intervals use 10,000 bootstrap draws (seed 2026090902), resampling eight base instances within each family while keeping all variants and paired procedures together. These intervals describe variation under the fixed synthetic generator. We make no population inference about real users, no significance-test claims, and no claim that 384 transcripts are 384 independent planning problems.

4.4 Exploratory schema-order control

Inspection after the main run identified an unanticipated implementation choice: canonical JSON serialization alphabetized schema property dictionaries. The main answer order was conflict, plan, status, waived; individual compiled clauses began activities, days, id, kind, number. The provider documents that output keys follow schema order [13]. A separate protocol, written before the additional calls but after the main results, specified 1,536 new observations: direct and compilation for both models on all 384 cases, with source insertion order preserved. This restored answer order status, plan, waived, conflict; compilation order rules, permitted; and clause order id, kind, number, days, activities. Prompts, schema values, required-field arrays, model settings, and scores were unchanged. A separate cache namespace prevented reuse of original responses.

The two runs were not counterbalanced, and this intervention changes multiple nested field orders, not solely conflict position. It is exploratory and potentially confounded by run timing and output variability. The original matrix remains the prospectively specified result; the control tests how narrowly that result should be interpreted.

5. RESULTS

5.1 Aggregate behavior and certificates

All 3,072 planned observations were saved. Table 1 reports the complete matrix; every cell has 384 cases, and malformed outputs remain in that denominator.

Table 1. Primary behavior success, secondary strict success, and correct status. Model names abbreviate the pinned snapshots in Section 4.2.

Model	Procedure	Behavior	Strict	Status
4.1 mini	Direct	174 (45.3%)	120 (31.2%)	216 (56.2%)
4.1 mini	Direct + review	163 (42.4%)	118 (30.7%)	173 (45.1%)
4.1 mini	Compile	383 (99.7%)	383 (99.7%)	383 (99.7%)
4.1 mini	Compile + review	383 (99.7%)	383 (99.7%)	383 (99.7%)
5.4 nano	Direct	158 (41.1%)	67 (17.4%)	174 (45.3%)
5.4 nano	Direct + review	144 (37.5%)	57 (14.8%)	157 (40.9%)
5.4 nano	Compile	316 (82.3%)	286 (74.5%)	325 (84.6%)
5.4 nano	Compile + review	317 (82.6%)	286 (74.5%)	327 (85.2%)

Table 2. Paired gains in behavior success. Intervals resample base instances within the six fixed families; percentage points (pp), descriptive only.

Contrast	Gain (pp)	95% interval (pp)
4.1 mini: Compile minus Direct	54.4	[51.3, 57.0]
4.1 mini: Compile + review minus Direct + review	57.3	[54.7, 59.9]
5.4 nano: Compile minus Direct	41.1	[36.5, 45.8]
5.4 nano: Compile + review minus Direct + review	45.1	[40.1, 50.3]

For 4.1 mini, 54, 45, 0, 0 responses respectively passed behavior but failed strict scoring in direct, direct-review, compile, and compile-review conditions. These differences count certificate-only failures under the operational metrics, not an estimate of the causal burden of requesting certificates.

For 5.4 nano, 91, 87, 30, 31 responses respectively passed behavior but failed strict scoring in direct, direct-review, compile, and compile-review conditions. These differences count certificate-only failures under the operational metrics, not an estimate of the causal burden of requesting certificates.

5.2 Permission variants and paired histories

Table 3. Behavior-correct counts out of 48 in each variant. D = direct; DR = direct review; C = compile; CR = compile review. Mini and nano identify the two snapshots. The feasible control changes original clauses; the other seven variants preserve them.

Variant	Mini D	Mini DR	Mini C	Mini CR	Nano D	Nano DR	Nano C	Nano CR
allow a	10	3	48	48	2	0	37	36
allow b	12	5	47	47	10	4	36	39
allow all	5	4	48	48	1	2	38	38
deny	48	40	48	48	46	43	44	46
revoke	48	45	48	48	44	39	43	43
switch	1	5	48	48	1	3	37	31
priority deny	48	36	48	48	46	46	45	47
feasible	2	25	48	48	8	7	36	37

For 4.1 mini, all seven permission variants were behavior-correct on 1, 0, 47, 47 of 48 base instances for direct, direct review, compile, and compile review, respectively.

For 5.4 nano, all seven permission variants were behavior-correct on 0, 0, 29, 23 of 48 base instances for direct, direct review, compile, and compile review, respectively.

5.3 Extraction and delivered-plan errors

4.1 mini, compile: exact rules on 380/384 cases, exact current permissions on 383/384, and both exact on 380/384. Extraction diagnostics were available on 383/384; all other cases remain failures in the primary analysis. Unused-field normalization changed 220/384 extractions. Among jointly exact extractions, 0 failed behavior.

4.1 mini, compile + review: exact rules on 380/384 cases, exact current permissions on 383/384, and both exact on 380/384. Extraction diagnostics were available on 383/384; all other cases remain failures in the primary analysis. Unused-field normalization changed 218/384 extractions. Among jointly exact extractions, 0 failed behavior.

5.4 nano, compile: exact rules on 285/384 cases, exact current permissions on 371/384, and both exact on 284/384. Extraction diagnostics were available on 374/384; all other cases remain failures in the primary analysis. Unused-field normalization changed 219/384 extractions. Among jointly exact extractions, 0 failed behavior.

5.4 nano, compile + review: exact rules on 285/384 cases, exact current permissions on 372/384, and both exact on 282/384. Extraction diagnostics were available on 376/384; all other cases remain failures in the primary analysis. Unused-field normalization changed 235/384 extractions. Among jointly exact extractions, 0 failed behavior.

Table 4. Protected-clause violations among assessed answers, with nonassessment shown explicitly. A nonblocked answer or any nonempty delivered schedule is assessed when its format is valid. Correctly empty blocked outputs are inapplicable. Malformed outputs are not silently counted as safe.

Model / procedure	Violations / assessed	Not assessed
4.1 mini / Direct	39 / 73	311
4.1 mini / Direct + review	44 / 203	181
4.1 mini / Compile	0 / 239	145
4.1 mini / Compile + review	0 / 239	145
5.4 nano / Direct	35 / 91	293
5.4 nano / Direct + review	31 / 77	307
5.4 nano / Compile	12 / 206	178
5.4 nano / Compile + review	13 / 205	179

Compilation produced no final answer in 20 observations, all retained as behavioral failures. Another 83 direct or direct-review observations failed the output-format/domain check. Post hoc diagnostics identify repeated day numbers and out-of-domain durations among these cases; categories can overlap. These failures are separate from an unauthorized waiver explicitly named in an otherwise well-formed answer.

False blocking substantially shaped the original aggregate results. Direct 4.1 mini passed all 144 genuinely blocked requests, but only 2/48 feasible controls and 28/192 repairable requests. Direct nano passed 136/144 blocked requests, 8/48 feasible controls, and 14/192 repairable requests. For example, in eval-time-01-feasible, 4.1 mini returned blocked for five sessions of at least 30 minutes under a 155-minute cap, although five 30-minute sessions satisfy the request. These are not all failures to remember permissions.

A separate extraction example is eval-coupled-06-feasible. Nano compilation replaced the explicit allowed-day set [0, 1, 2, 3, 4] with an empty set, while retaining five required sessions and the other numeric clauses. The solver then correctly rejected that incorrectly extracted specification. The same error survived compilation review. This illustrates why exact search cannot establish end-to-end faithfulness without checking the language-to-specification boundary. Conversely, some nonexact extractions still yielded correct behavior; outcome success does not certify complete extraction fidelity.

5.4 Execution and cost

The main run used 3,072 unique paid calls, with \$0.9767 in usage-based token accounting. Including development calls and the retained reserve for one failed initial network attempt, cumulative project accounting was \$1.0559, below the \$20 budget. This accounting uses list input/output prices and ignores cached-input discounts, taxes, and provider price changes; it is not an invoice. Review-condition costs include shared initial calls and cannot be added to initial-condition costs as separate billing. There were 20 observations marked invalid or incomplete by the runner. All 3,072 main API calls returned completed responses. The 20 pipeline-error observations contained parsed compilations that failed subsequent validation; no review call was skipped. No main-run network retry or prompt revision was used.

5.5 Exploratory schema-order results

All 1,536 supplementary observations were completed. For every matched pair, the original and supplementary wire payloads had equal parsed JSON values but different schema property order. Table 5 compares the same cases under both serializations.

Table 5. Exploratory whole-schema ordering control. Each count is out of 384; original = alphabetized properties, source = originally authored property order.

Model	Method	Original behavior	Source behavior	Original strict	Source strict
4.1 mini	Direct	174	292	120	196
4.1 mini	Compile	383	376	383	375
5.4 nano	Direct	158	164	67	67
5.4 nano	Compile	316	335	286	301

For 4.1 mini, direct behavior success rose from 174/384 (45.3%) to 292/384 (76.0%), with 132 cases correct only under source order and 14 correct only under alphabetized order. Compilation fell from 383/384 (99.7%) to 376/384 (97.9%). Its advantage over direct generation consequently narrowed from 54.4 to 21.9 percentage points. The model's source-order compilation still had eight explicitly unauthorized waivers, showing that high plan validity alone does not establish permission fidelity.

For nano, direct success changed from 158/384 (41.1%) to 164/384 (42.7%), while compilation changed from 316/384 (82.3%) to 335/384 (87.2%). The compilation advantage was 44.5 points under source order, compared with 41.1 points originally. Exact rule extraction in source-order compilation was 372/384 for mini and 298/384 for nano; exact permission extraction was 376/384 and 382/384, respectively. All jointly exact extractions again passed behavior.

These paired changes support sensitivity to the complete output interface in this workload, particularly for direct mini generation. They do not isolate one field's causal effect, establish the best schema order, or quantify run-to-run variance. The original comparison survives in direction in both models, but its magnitude is not invariant to serialization. No review conditions were repeated in this control.

The supplementary run used \$0.4476 in usage-based token accounting. One temporary HTTP 503 interrupted execution after 1,380 saved observations; one manually reviewed retry resumed the remaining cases, with the original \$0.006298 reservation retained. The interruption and original manifest are preserved. Cumulative project accounting after both experiments and development was \$1.5098, including both unresolved-attempt reserves. All 4,608 successful experiment API calls returned completed responses; invalid final answers were still scored as failures.

6. DISCUSSION

Under the original alphabetized-schema condition, compilation outperformed direct generation by 54.4 percentage points for 4.1 mini and 41.1 points for nano. However, the direct conditions' high false-blocking rates and the supplementary ordering control prevent interpreting these gaps as an intrinsic model-capability difference. The comparison is between the complete implemented procedures, including their output interfaces.

Review was not a uniformly beneficial operation in the original matrix. Direct review reduced total behavioral successes by 11 cases for 4.1 mini and 14 for nano. For mini, feasible-control success improved from 2/48 to 25/48 while success on genuinely blocked cases fell from 144/144 to 121/144. For nano compilation, review improved aggregate behavior from 316/384 to 317/384, yet reduced fully correct seven-variant base instances from 29/48 to 23/48. This is a concrete example of a small average gain masking weaker consistency across matched permission histories. These descriptive results do not establish that review is generally harmful.

The schema-order control materially qualifies the main finding. The relatively large improvement in mini direct generation, together with a slight decline in its compilation condition, shows that the original 54.4-point gap should not be presented as an architecture-only effect. Nano behaved differently, so the control also argues against treating restored source order as a universal optimization. The defensible conclusion is a bounded procedural advantage for compilation under both tested interfaces, with model-dependent sensitivity and residual permission/extraction errors.

The measurements distinguish three boundaries: interpreting the current request, solving the extracted mathematical problem, and explaining a conflict. Exact search can enforce a correctly extracted specification, but cannot ensure that extraction matches the user's language. Conversely, an invalid conflict certificate need not imply an invalid schedule. Keeping these boundaries separate prevents both an inflated claim of solver safety and an overly pessimistic assessment of otherwise valid actions.

The permission variants provide controlled comparisons, but do not make the whole task a pure permission-understanding test. Arithmetic, finite search, optimality, identifier tracking, and certificate construction all contribute to performance. In particular, success on deny and revoke can coexist with failures on repairable requests because blocking avoids construction of a schedule. Per-status and per-family results must therefore accompany aggregate success.

7. LIMITATIONS

The task is small, synthetic, and explicitly numbered. The six families were selected during development, and main instances often share the same logical structure. No independent person validated the natural-language templates. Broader evidence would require new families, independently authored paraphrases, ambiguous dialogue, longer histories, and user studies; none is claimed here.

Only two snapshots from one provider were tested, with one response per case and no explicit extended reasoning condition. The study estimates neither run-to-run variance nor performance of stronger configurations. Paired methods share initial responses. Compilation has access to an exact task-specific search procedure whereas direct generation does not, so the contrast measures the complete procedures rather than a causal effect of permission representation alone.

All methods are asked for a formal certificate, even though the primary outcome excludes certificate correctness. Separating scoring does not remove the possible cognitive burden of generating a certificate. A no-certificate prompt condition would be needed to isolate that burden. The six-family bootstrap is conditional and descriptive; it cannot repair the absence of representative sampling.

The oracle exhaustively covers this finite domain but shares AI authorship and task assumptions with the generator and solver. Cross-implementation agreement reduces some implementation risks without establishing independent semantic validity. Model or API changes may also prevent byte-identical response reproduction despite pinned model names and retained inputs.

This is a computer-science scheduling experiment. Abstract workout labels provide an application context, not health evidence. There are no participants, private internship records, or patient data. Permission-sensitive formal repair should not be interpreted as clinical approval of generated exercise plans.

8. CONCLUSION

This study contributes a matched synthetic benchmark for explicit permission-sensitive minimum repair, exact reference and outcome checks, and an auditable two-model comparison. Compilation achieved higher behavioral success than direct generation under both tested schema orders, but the size of the advantage depended on the model and interface. A second model review did not consistently improve outcomes, and behavior, conflict certificates, and complete permission-history consistency gave different assessments. These findings support evaluating authorization, extraction, execution, and schema presentation separately. Broader capability, clinical utility, and publication-level generality remain unestablished.

REPRODUCIBILITY

The accompanying local research package includes the generator, development and evaluation data, exact solver and oracle, frozen prompts, both protocols, source snapshots, raw secret-free model calls, consolidated scored observations, analysis, and manuscript source. Consolidated observations are replayed from original raw responses; the replay reproduces every main summary and paired interval. Per-call timing is not reconstructed or used in the claims. It has not been publicly deposited or peer reviewed. The main data SHA256 is 878b64d714a5d0387bdf5efc8ab5ffd1b5aab7230a6fc520a6fec18f981ac6d. The main run identifier is cd04fab453a6747f. The detailed result table is reports/evaluation-cd04fab453a6747f/analysis.json; per-observation values and errors are retained alongside it. The local frozen protocol is main-protocol.md.

REFERENCES

- [1] Yilun Hao, Yongchao Chen, Yang Zhang, and Chuchu Fan. 2025. Large Language Models Can Solve Real-World Planning Rigorously with Formal Verification Tools. NAACL-HLT, Volume 1, pp. 3434–3483. <https://aclanthology.org/2025.naacl-long.176/>
- [2] Meilin Chen, Zepeng Zhai, Jiaxuan Zhao, and Yuan Lu. 2026. SafeRun: Enabling Determinism in LLM Planning for Running. arXiv:2606.09027v1. <https://arxiv.org/abs/2606.09027v1>
- [3] Mehmet Kurmaz. 2026. AWARE-US: Preference-Aware Infeasibility Resolution in Tool-Calling Agents. arXiv:2601.02643v1. <https://arxiv.org/abs/2601.02643v1>
- [4] Xingwei He, Qianru Zhang, Pengfei Chen, Guanhua Chen, Linlin Yu, Yuan Yuan, and Siu-Ming Yiu. 2026. ConInstruct: Evaluating Large Language Models on Conflict Detection and Resolution in Instructions. Proceedings of the AAAI Conference on Artificial Intelligence 40(37), pp. 30969–30977. <https://doi.org/10.1609/aaai.v40i37.40356>
- [5] Dipankar Sarkar. 2026. From Errors to Proofs: Minimal-Core-Guided Repair for Neuro-Symbolic Constraint Solving. arXiv:2608.14771v1. <https://arxiv.org/abs/2608.14771v1>
- [6] Haoyuan Zhu. 2026. Dead Text or Binding Clause? Measuring and Restoring Constraint Influence in Black-Box LLM Dialogues. arXiv:2608.12599v1. <https://arxiv.org/abs/2608.12599v1>
- [7] Yifan Zhang, Xinkui Zhao, Sai Liu, Hengxuan Lou, Guanjie Cheng, and Chang Liu. 2026. FAVA: Formal Authorization for Verified Agents with Evidence-Backed Permission Graphs. arXiv:2607.27267v1. <https://arxiv.org/abs/2607.27267v1>
- [8] Sebastien Kawada / Kaons Research. 2026. DRIFT-Bench, accompanying Residual Drift Dominates Contradiction in Multi-Turn Constraint Reasoning. Author repository, accessed 9 September 2026. <https://github.com/kaons-research/drift-bench>
- [9] Google DeepMind. ACS benchmark. Author repository accompanying The Ability of Large Language Models to Evaluate Constraint-Satisfaction in Agent Responses to Open-Ended Requests. Accessed 9 September 2026. <https://github.com/google-deepmind/acs-benchmark>
- [10] Sin-Ying Lin. 2026. Your Prompt Is Not the Only Prompt: How Much Do LLMs Weight Structured-Output Schema Descriptions? arXiv:2608.08254v1. <https://arxiv.org/abs/2608.08254v1>

- [11] Yin Li. 2026. When JSON Is Not Enough: Semantic Reliability of Schema-Constrained LLM Ordering Agents. arXiv:2607.18261v1. <https://arxiv.org/abs/2607.18261v1>
- [12] Vaibhav701161. Constrained Sensitivity Lab. Author repository, accessed 10 September 2026. <https://github.com/Vaibhav701161/constrained-sensitivity-lab>
- [13] OpenAI. Structured Model Outputs: Key Ordering. API documentation, accessed 10 September 2026. <https://developers.openai.com/api/docs/guides/structured-outputs>

Copyright & License:

© Authors retain the copyright of this article. This work is published under the Creative Commons Attribution 4.0 International License (CC BY 4.0), permitting unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.