Grounded Normative Rule Generation with Structured Search

Anonymous ACL submission

Abstract

Normative rules like institutional charters and workplace policies must be both human-readable and operationally verifiable against actual environment records. However, current language generation and structured-output benchmarks primarily reward surface fluency or schema compliance, leaving operational grounding weakly tested. This creates a critical vulnerability where standard language models generate plausible-sounding policies that fail during enforcement because they rely on unavailable data logs or misaligned scopes. To address this challenge, we formalize the problem as Grounded Normative Rule Synthesis (GNRS) and introduce GNRS-Search, a framework that utilizes Markov Chain Monte Carlo (MCMC) sampling to optimize a discrete, five-slot And-Or Graph (AOG). By explicitly decoupling intermediate operational structure from final prose generation, this method isolates executable feasibility from writing style and allows rule failures to be localized prior to surface realization. We evaluate our approach on GNRS-Bench, a benchmark spanning 116 controlled goals across eight scene families, and RealCharter-Bench, which evaluates transfer to 53 real-world policy tasks with hidden source clauses. GNRS-Search raises average rubric quality from 68.8% to 81.0% and ranks first under a disclosed executable composite metric, while systematic slot interventions confirm that performance gains stem from robust operational logic rather than rhetorical tuning. Ultimately, by transforming automated rule drafting into an inspectable search problem, this work provides a foundational paradigm for deploying verifiable and compliance-ready personal agents within regulated environments.

1 Introduction

Normative rules, such as institutional charters and workplace policies, must fundamentally satisfy a dual requirement: they must be human-readable and operationally verifiable when compliance is audited (Sergot et al., 1986; Solar-Lezama et al., 2006; Governatori and Rotolo, 2010). This dual purpose creates a basic tension for language generation. A generated clause may exhibit high surface fluency while entirely failing as a valid rule by relying on unavailable evidence or untraceable enforcement procedures (Wiseman et al., 2017; Maynez et al., 2020; Ji et al., 2023).

We formalize this problem as Grounded Normative Rule Synthesis, which we abbreviate as GNRS. Given a normative goal, an institutional schema, and a record layer containing environment traces and logs, the task is to write a natural-language rule that is both readable and strictly backed by available data. Figure 1 illustrates this setting with a restaurant attendance example. Direct prompting can produce a fluent policy that invokes unsupported evidence channels, such as tracking systems or staff testimony. The core difficulty goes beyond factual hallucination: rule writing requires coordinating scope, trigger, norm, evidence, and procedure. A small change in one component can invalidate the others; for example, a procedure based on automated logs may fail when the selected evidence is only manual testimony. We therefore treat rule generation as a structured decision problem before surface realization, so that failures can be localized to the component that caused them.

While existing text-generation tools address related problems, they do not enforce operational grounding. Constrained decoding and structured-output methods improve syntactic validity by restricting generations to schemas or grammars (Scholak et al., 2021; Willard and Louf, 2023), but they do not check whether the generated content is supported by available environment records. Iterative self-refinement can improve outputs through feedback loops (Madaan et al., 2023; Shinn et al., 2023), but it lacks an external record layer for verifying evidence channels and procedures. Interac-

tive language agents use environmental feedback to guide action selection (Yao et al., 2022; Shridhar et al., 2020), whereas GNRS asks whether the generated rule text itself is backed by auditable data logs. Our core objective is to identify structural configurations that make a rule operationally grounded before it is rendered as prose.

To solve this problem, we implement GNRS-Search. Instead of generating the final text directly, our framework models the rule as a five-slot And-Or Graph (Zhu and Mumford, 2007; Wu and Zhu, 2011), which we call an AOG, representing intermediate operational commitments across scope, trigger, norm, evidence, and procedure. The system builds a candidate space by combining deterministic pieces from rule templates, context schemas, and record logs with language model proposals. We then use Markov Chain Monte Carlo sampling (Hastings, 1970), or MCMC, to iteratively edit this discrete structure. The search explores candidate changes through four explicit operations including slot replacement, parameter edits, evidence swaps, and joint trigger-evidence adjustments. An internal scoring function guides the search by evaluating record grounding and penalizing hard feasibility violations. This approach successfully turns rule writing into an inspectable search problem where structural errors can be caught and localized before the final text is rendered.

We evaluate our method on the introduced GNRS-Bench, which covers 116 controlled goals across 8 scene families, and test dataset transfer on RealCharter-Bench, which contains 53 real-world policy tasks. Experimental results show that GNRS-Search improves average rubric quality from 68.8% to 81.0% on the main test. It also ranks first under our overall executability metric across different model backbones and out-of-distribution scenes. Human evaluations and sensitivity analyses confirm that these improvements come from correct operational structure rather than better phrasing.

This paper makes four contributions.

- We define GNRS as a structured-NLG task that tests whether a readable rule is also backed by available records.
- We introduce GNRS-Bench with 116 controlled goals and RealCharter-Bench with 53 real-derived policy tasks.
- We implement GNRS-Search, an AOG search method with MCMC proposals for selecting grounded rule structures before rendering.
- We show gains on main, robustness, and real-derived tests. Human calibration supports the rubric, while slot interventions show that the gains come from operational meaning.

2 Related Work

Grounded and constrained generation. Data-to-text and grounded NLG benchmarks study faithful realization from structured inputs (Gardent et al., 2017; Parikh et al., 2020; Nan et al., 2021). Constrained decoding, JSON-mode generation, and grammar-guided decoding improve parseability (OpenAI, 2024; Dong et al., 2025; Tam et al., 2024), and recent work further trains structured-output ability at the schema level (Lu et al., 2025). Best-of-N selection and verifier-style reranking are also common prompting enhancements (Cobbe et al., 2021). GNRS differs because the output is a normative clause whose validity depends not only on format or fluency, but on whether its roles, triggers, evidence, and procedures are grounded in operational records.

Rule grounding, agents, and normative NLP. Rule-oriented benchmarks test rule following or evidence-grounded reasoning (Zhou et al., 2025; Jiayang et al., 2025), while GNRS asks the model to synthesize the rule itself. Interactive agent work uses environmental feedback to guide behavior (Park et al., 2023; Wang et al., 2023; Liu et al., 2024); GNRS instead uses environmental structure as typed primitives and evidence channels that constrain text generation. Legal and normative NLP has studied parsing, retrieval, classification, and compliance reasoning (Hendrycks et al., 2021; Guha et al., 2023; Ren et al., 2024; Holzenberger et al., 2020; Boella et al., 2006). GNRS is complementary, focusing on generating enforceable normative language from goals and operational records.

Symbolic synthesis and constraints. Program synthesis and constraint solving show how discrete structures can be searched under formal specifications (Solar-Lezama et al., 2006; Solar-Lezama, 2008). GNRS-Search follows this spirit by searching over a discrete rule structure before generating text. Unlike program synthesis, however, the target is natural-language policy text, and the constraints are partial grounding diagnostics over available records rather than complete logical specifications.

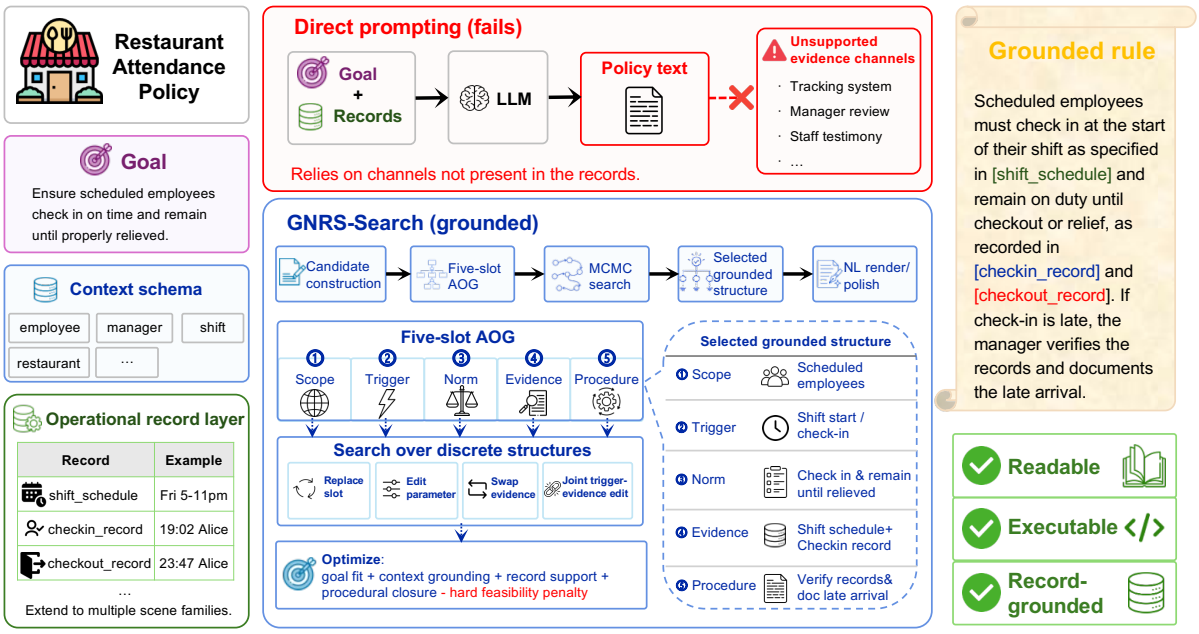

Figure 1: Normative rules must be both readable and operationally grounded. Direct prompting often satisfies the surface-language requirement but introduces unsupported evidence channels or vague enforcement procedures. GNRS-Search addresses this by searching over a five-slot operational structure, covering scope, trigger, norm body, evidence, and procedure, before rendering the selected structure into natural language grounded in concrete operational records.

3 Preliminaries

We study normative text generation as a problem of producing language with an operational interpretation. Let $x = (g, C, D)$ denote a rule-writing context: g is the normative intent, C is the institutional context schema containing roles, locations, objects, and action vocabulary, and D is the operational record layer containing event traces, logs, observations, and evidence channels. The output is a natural-language rule y. Unlike ordinary open-ended text generation, the quality of y depends not only on fluency but on whether it preserves the intent and can be operationalized against C and D.

$$V(y; x) = (v_I, v_G, v_E, v_C, v_R).$$

The components of V measure intent preservation, grounding, evidence use, procedural closure, and readability. A scalar task score is any monotone functional $F(y; x) = u(V(y; x))$. We assume that a normative rule has a latent operational structure a that mediates between intent and surface text. Let K be the set of five slots: scope, trigger, norm, evidence, and procedure. Then $a = (a_k)_{k \in K}$. This structure is not the task output: the task output remains natural language. It is instead a modeling assumption that lets us reason about where grounding fails. A slot assignment is grounded when its referenced actors, actions, evidence channels, and procedures can be interpreted against C and D.

Direct prompting estimates y in one step. Our view separates operational interpretation from surface realization: construct a candidate space $\mathcal{A}(x)$, search for an operational structure a^*, and then realize it as text, where S is an internal score over structures and R is a realization function:

$$a^* = \arg\max_{a \in \mathcal{A}(x)} S(a; x), \qquad y^* = R(a^*, x).$$

We keep S distinct from s_{eval}, the evaluation score over final language, because a system should not win merely by optimizing the same metric used to judge it.

The following proposition states the formal role of structure search. Let $G(a; x)$ be the task-validity score that a perfectly faithful realization of structure a would obtain. Suppose S is calibrated to G within ϵ_C, search returns an ϵ_S-optimal structure under S, and realization loses at most ϵ_R task validity.

Proposition 1. *If $|S(a; x) - G(a; x)| \leq \epsilon_C$ for all $a \in \mathcal{A}(x)$, $S(\hat{a}; x) \geq \max_a S(a; x) - \epsilon_S$, and $F(R(\hat{a}, x); x) \geq G(\hat{a}; x) - \epsilon_R$, then*

$$F(R(\hat{a}, x); x) \geq \max_{a \in \mathcal{A}(x)} G(a; x) - \epsilon_S - 2\epsilon_C - \epsilon_R.$$

Proof. Let $a_G \in \arg\max_a G(a; x)$. By calibration and search optimality,

$$\begin{aligned} G(\hat{a}; x) &\geq S(\hat{a}; x) - \epsilon_C \\ &\geq S(a_G; x) - \epsilon_S - \epsilon_C \\ &\geq G(a_G; x) - \epsilon_S - 2\epsilon_C. \end{aligned}$$

Combining this with the realization bound gives the result. □

4 Evaluation Benchmarks

We evaluate grounded rule synthesis using two complementary benchmarks. The primary testbed is GNRS-Bench, which provides a controlled simulation environment where available operational records are fully typed and systematically varied. To test real-world generalization, we introduce RealCharter-Bench, a collection of evaluation scenarios derived directly from authentic institutional policies. Comprehensive domain distributions, record-layer vocabularies, and document source provenances are detailed in Appendix A.

GNRS-Bench. The benchmark evaluates whether a synthesis framework can write verifiable rules using only the information that an environment can actually log. Following the formalization in Section 3, each instance provides a target management goal alongside an institutional context schema and an operational record layer. For example, in an attendance scenario regulating restaurant shift coordination, the schema defines relevant roles and locations such as employees and service rooms, while the record layer exposes explicit data logs like check-in records. A valid synthesis method must anchor its generated scope, trigger, evidence, and procedure in these available logs instead of relying on unrecorded channels or subjective testimony.

The benchmark contains 116 goals distributed evenly across four functional categories including attendance, property boundaries, task execution, and mixed crisis objectives. We reserve 16 goals for development tuning and 88 goals for core in-distribution testing. To measure out-of-distribution performance, we hold out the entire `open_space_navigation` scene family as a separate 12-goal OOD split. Crucially, the underlying simulator distinguishes transient environmental observables from permanent evidence logs. This design ensures that generated rules can be evaluated after an interaction finishes by checking their compliance against durable data streams rather than fleeting environment states.

Evaluation protocol. The evaluation protocol decouples text generation from reference text imitation. During testing, a method receives the target management goal and the allowed context schema but is denied access to any human-authored golden rule. The framework must output a single natural-language rule clause whose semantic and programmatic commitments remain interpretable against the available environment logs. Final rule drafts are scored by a blinded multi-dimensional rubric to measure linguistic quality, while actual operational feasibility is verified via the binary HARD-OK gate over the underlying structural choices.

RealCharter-Bench. RealCharter-Bench provides an external check to ensure that synthesis frameworks generalize to authentic public policies. It comprises 53 tasks derived from 15 foundational open-source governance guidelines and corporate manuals such as the Python PSF and the GitLab Handbook. To prevent direct token copying, the original human-authored text is completely hidden from the generator. Instead, the dataset represents each task as a normalized policy card that exposes explicit fields for context, triggers, required conduct, and evidence requirements. This setup evaluates whether grounded structured search transfers effectively to real-world policy environments under a unified input-output abstraction.

5 Method

The core design principle of GNRS-Search relies on separating surface prose generation from structural commitment synthesis. Instead of prompting a language model to output the final rule text y in an end-to-end pass, our framework first constructs grounded operational candidates for individual slots, executes a discrete optimization search over complete structural configurations a, and realizes the optimal layout as natural language. This design enforces the scientific objective of anchoring free-text rule generation to an executable and auditable interpretation. We mathematically model the latent structure a as an And-Or Graph, abbreviated as AOG, which spans the five essential operational slots introduced in Section 3 including scope, trigger, norm body, evidence policy, and procedure.

Candidate construction. For each operational slot k, our framework builds a localized can-

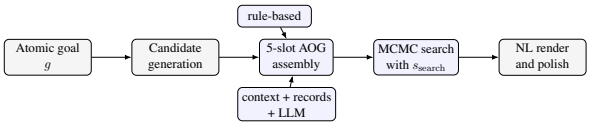

Figure 2: System architecture of the GNRS-Search pipeline. Slotted candidate primitives derived from heterogeneous sources are assembled into a joint five-slot AOG structural space, optimized via an MCMC structural sampler against an internal record-grounded score, and rendered into verifiable normative prose.

didate repository $\mathcal{C}_k$ from four distinct structural sources comprising symbolic rule templates, context-schema primitives, trace-derived event records, and LLM-augmented proposals. The complete structural search space is defined as the Cartesian product of these sets, formulated as

$$\mathcal{A}(x) = \prod_{k \in K} \mathcal{C}_k(x).$$

While the LLM-augmented fragments are highly valuable for injecting context-aware vocabulary and natural phrasing into rule pieces, they undergo a strict validation filter against the environment schema and active record logs to minimize the introduction of ungrounded or imaginary payloads. Across the 88 main evaluation goals, the deterministic symbolic tracks contribute an average of 14.0 template-backed candidates, 9.3 schema-derived primitives, and 10.2 record-backed traces per goal. The generative language model component contributes 16.7 additional fragments per configuration while the grounding filter discards an average of 1.3 unsupported candidates per execution, maintaining a stable 92.5% candidate retention rate.

Structured search. A concrete state $a \in \mathcal{A}(x)$ corresponds to a fully populated five-slot AOG structural layout. To navigate this combinatorial optimization space, we implement a discrete Markov Chain Monte Carlo, or MCMC, structural sampler driven by four specialized proposal distributions including single-slot replacement, parameter-only tuning, evidence-policy swapping, and coordinated trigger-evidence structural edits. The evaluation function linearly combines structural grounding indicators and linguistic quality features, penalizing hard operational feasibility violations according to the scoring function

$$s_{\text{search}}(a; x) = w^\top \phi(a, x) - \lambda \, \mathbf{1}[\neg h(a, x)].$$

The descriptive feature vector $\phi(a, x)$ computationally operationalizes multiple programmatic dimensions covering intent alignment, schema grounding, record track compatibility, procedural closure, and value consistency, while $h(a, x)$ acts as a binary predicate flagging execution failures. Given a proposed structural state change from a to a', the sampler accepts the mutation with a standard Metropolis-Hastings probability formulated as $\min(1, \exp((s_{\text{search}}(a'; x) - s_{\text{search}}(a; x))/\tau))$. This stochastic accept-reject gate provides the search mechanism with sufficient exploration freedom to escape local optima while consistently steering the trajectory toward record-grounded configurations. Our evaluation layout utilizes 200 optimization steps per run with the temperature parameter fixed at $\tau = 0.08$.

The optimization objective s_{search} functions as a structural prior and is decoupled from the downstream validation metric. It reflects domain-agnostic properties that any verifiable organizational clause should satisfy, prioritizing trace-backed evidence channels over unrecorded claims and structurally closed enforcement flows over ambiguous procedures. The parameter weights are calibrated on the 16 development goals and remain entirely frozen before test-set execution. Crucially, no weight is optimized against the test scenarios or the automatic evaluator outputs. Quantitative verification details regarding alternative uniform weightings, the ablation of hard constraints, and a baseline comparison using a mathematical CP-SAT solver (Perron et al., 2023) are detailed in Appendix B, isolating this structural search bias from test-time metric overfitting. A worked AOG schematic and construction-and-search trace are provided in Figure 6.

Realization and evaluator separation. The optimized latent graph layout is processed by a surface realization function R into an initial rule clause formulated as $y = R(a^*)$, which is subsequently refined through a payload-preserving text polish workflow. A programmatic alignment guard monitors this editing phase, automatically rejecting any sentence rewrites that omit essential structural entities or discard required evidence logs. Crucially, the internal search scorer s_{search} is strictly isolated from the final performance evaluation setup. We report our headline achievements using s_{eval}, an independent multi-dimensional rubric score computed over the polished natural language text, alongside HARD-OK, a binary execution feasibility checker tracking true operational grounding within the envi-

Method	Spec	Exec	Flu	Read	Faith	Avg	v2 Avg	HARD-OK	OVERALL
Direct Prompting	60.4	61.7	83.1	**79.8**	59.2	68.8	64.9	0.0	55.1
Structured Outputs	72.7	53.8	54.1	47.7	68.4	59.3	68.8	4.2	48.3
Best-of-N	72.0	52.7	54.5	47.4	68.7	59.1	69.0	6.1	48.5
Self-Refine	72.7	55.4	50.2	46.4	68.3	58.6	68.6	3.8	47.6
In-Context Grounding	70.2	51.9	59.8	49.9	67.8	59.9	69.7	0.0	48.0
Norm Synthesis	73.8	55.3	63.9	54.5	68.4	63.2	71.3	2.7	51.1
No record grounding	75.0	56.9	64.4	55.2	69.1	64.1	71.4	74.6	66.2
No search	74.0	56.7	64.5	56.1	67.0	63.7	71.7	46.2	60.2
No LLM candidates	75.6	57.7	74.4	60.5	68.1	67.2	70.5	**77.3**	69.2
GNRS-Search	**80.2**	**73.3**	**92.5**	79.5	**79.2**	**81.0**	**86.4**	51.1	**75.0**

Table 1: Main test results on 88 goals $\times$ 3 seeds. Rubric columns are GPT-5.1 judgments on a 1–5 scale, reported as percentages. v2 Avg is a stricter diagnostic rubric that penalizes generic evidence vocabulary. HARD-OK is a binary AOG feasibility diagnostic. OVERALL is the disclosed executable composite $0.80\times$ Avg $+0.20\times$ HARD-OK.

ronment records. This systematic boundary ensures that performance enhancements are driven by genuine structural grounding rather than superficial convergence toward the evaluation metric.

6 Results and Analysis

6.1 Evaluation Setup

The in-distribution evaluation suite uses 88 distinct goals across 3 random seeds. Baselines encompass direct prompting, structured outputs (Willard and Louf, 2023), best-of-N (N=8) reranking (Nakano et al., 2021), self-refinement (Madaan et al., 2023), retrieval-style in-context grounding (Lewis et al., 2020), and a single-shot CRSEC norm-synthesis adaptation (Ren et al., 2024), while pipeline ablations examine the removal of record grounding, structural search, and language model candidates. The CRSEC-style baseline adapts only the norm-creation and representation prompt. The primary scoring instrument is a blinded GPT-5.1 rubric evaluating five dimensions including specificity, executability, fluency, readability, and faithfulness. We additionally report HARD-OK as a binary feasibility diagnostic over the selected AOG layout, alongside a disclosed executable composite score defined as OVERALL = 0.80 Avg + 0.20 HARD-OK. Metric-weight sensitivity is reported in Appendix B.1; human calibration and a second-model rubric check are reported in Appendix B.4, Tables 16–18; rubric prompts are listed in Appendix D.

6.2 Main Benchmark Results

Table 1 shows the head-to-head performance on the main test split. GNRS-Search improves average rubric quality to 81.0%, achieving a notable margin over both direct prompting and the closest ablation framework. The primary source of this improvement stems from the executability dimension, where our approach raises the average performance to 73.3%. This gain highlights a clear quality-feasibility frontier illustrated in Figure 3, where the language model candidate ablation represents the most conservative baseline under the binary feasibility gate, while GNRS-Search achieves the optimal balance under the composite evaluation metric. Paired goal-level bootstrap analysis confirms that this ranking remains stable across alternative weights, whereas shifting the optimization objective heavily toward binary gate metrics compromises overall text quality. Furthermore, our method-blinded human calibration study indicates strong directional alignment with the automatic scorer, showing a row-level Spearman correlation of 0.77 and yielding a positive human score margin of 0.69 over non-search alternatives.

Ablation	Avg	Δ	HARD-OK
GNRS-Search full	81.0	0.00	51.1
Uniform weights	80.4	-0.53	60.6
Random candidates	79.9	-1.03	36.0
No hard penalty	80.1	-0.86	53.8
100 search steps	80.2	-0.77	47.0
AND-only evidence	80.6	-0.35	52.7
CP-SAT control	64.8	-16.18	60.6

Table 2: Ablations on GNRS-Search. Δ is the Avg difference. All rows run on the same 88 goals and three seeds; CP-SAT uses the LLM-augmented candidate inventory with an OR-Tools backend.

6.3 Mechanism and Sensitivity

As shown in Table 2, removing individual algorithmic components degrades synthesis quality. Random candidate selection causes the largest absolute

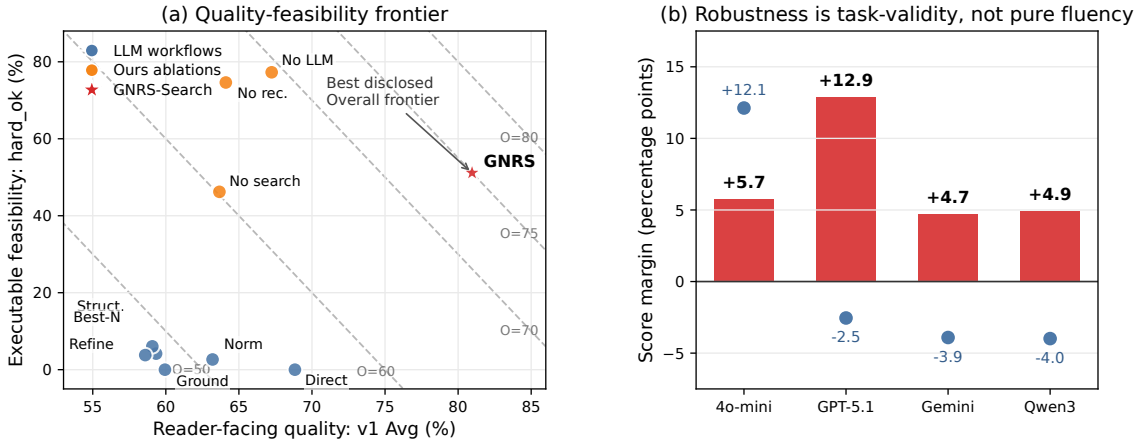

Figure 3: Analytical view of the headline results. Left: v1 Avg and HARD-OK form a quality-feasibility plane; dashed contours show the disclosed OVERALL composite, and points are labeled with abbreviated method names. Right: red bars show GNRS-Search's executable-OVERALL margin over the best non-GNRS-Search method, while blue dots show GNRS-Search minus Direct Prompting on pure Avg.

performance drop, which confirms that candidate ranking characteristics directly impact framework execution. Weakening the search space to 100 steps or removing the hard feasibility penalty also reduces overall performance, suggesting that the task rewards systematic structured decomposition over localized search heuristics. Crucially, the mathematical CP-SAT solver baseline performs substantially below our full framework on both average quality and composite metrics, demonstrating that the performance advantage cannot be replicated by simply replacing our sampling policy with an off-the-shelf discrete optimizer.

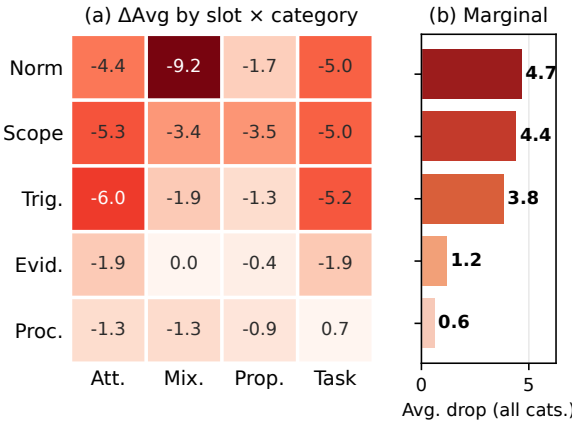

Figure 4: Slot sensitivity. Each cell is the Avg change after replacing one slot's candidate pool with distractors from other goals.

To locate where synthesized rules remain most vulnerable, our slot sensitivity intervention degrades candidate sets one slot at a time using cross-goal distractors. Figure 4 rejects the initial hypothesis that evidence aggregation details would dictate overall fragility. Instead, quality loss concentrates heavily within the semantic core of the rule text, where degrading the norm body slot causes the largest average performance drop, followed closely by scope and trigger modifications. Conversely, procedure slots and evidence policy formulations remain highly resilient. Static pool audits confirm that this stability is not an artifact of small distractor sets, indicating that downstream verification properties are often structurally recoverable from environmental observables once the core compliance duties are correctly specified.

6.4 Robustness and Real-World Transfer

We evaluate generalization capabilities across a held-out scene family and an external cross-model generator grid. In the out-of-distribution experiment holding out open-space navigation goals, Table 3 confirms that GNRS-Search maintains its performance advantage by scoring 79.8% on average quality and 82.7% on the joint composite metric. Cross-model robustness evaluations summarized in Table 4 show that our search framework ranks first under the composite task-validity metric across all tested commercial and open-source backbones. While open-ended direct prompting can achieve competitive surface text fluency on stronger model generators, it consistently yields zero binary feasibility scores due to its tendency to produce ungrounded generic clauses. This confirms that grounding-aware structural search remains neces-

sary to satisfy joint executability requirements.

Method	Avg	HARD-OK	OVERALL	s_{search}
Direct	65.6	0.0	52.4	0.546
Structured	58.0	0.0	46.4	0.805
Best-of-N	58.1	0.0	46.5	0.814
Self-Refine	58.2	0.0	46.6	0.806
Grounding	52.7	0.0	42.1	0.803
Norm Synth.	57.8	0.0	46.2	0.790
No record	57.7	**100.0**	66.1	0.995
No search	56.7	86.1	62.6	0.888
No LLM	60.9	**100.0**	68.7	0.917
GNRS-Search	**79.8**	94.4	**82.7**	0.904

Table 3: OOD results on the held-out open-space navigation family (12 goals $\times$ 3 seeds). OVERALL uses the same disclosed $0.80/0.20$ weighting as Table 1. Method names are abbreviated in the table body for width.

Generator	Direct	Refine	No LLM	GNRS-Search	Δ
GPT-4o-mini	55.1	47.6	69.2	**75.0**	+5.7
GPT-5.1	67.3	66.2	69.1	**82.0**	+12.9
Gemini Flash Lite	61.6	51.7	69.0	**73.7**	+4.7
Qwen3-235B	63.1	54.9	69.2	**74.1**	+4.9

Table 4: Cross-model executable OVERALL with a fixed GPT-5.1 scorer. Δ is GNRS-Search minus the best non-GNRS-Search row for that generator.

External evaluation on RealCharter-Bench indicates that our structured optimization translates effectively to authentic public policies. As reported in Table 5, GNRS-Search achieves an average text quality score of 93.4% alongside a perfect feasibility rate, leading the strongest iterative self-refinement baseline. Figure 5 details this relationship across the quality-feasibility plane, demonstrating that the external transfer performance gains are driven by structured search optimization rather than template formatting effects. Crucially, the perfect feasibility alignment reflects the explicit enumeration of target slots within normalized task cards, which reduces the likelihood of introducing unsupported environment payloads compared to open-ended simulation settings.

Method	n	Faith	HARD-OK	OVERALL
Direct	53	4.64	62.26	82.26
Self-Refine	53	4.77	77.36	86.09
No LLM	53	4.50	47.17	76.07
No search	53	2.68	0.00	50.56
GNRS-Search	53	**5.00**	**100.00**	**94.69**

Table 5: RealCharter-Bench results. Faith is the 1–5 task-faithfulness score; HARD-OK and OVERALL are percentages. All methods receive the same normalized task card, while the source clause text is hidden.

In summary, our empirical analysis yields three clear findings regarding verifiable rule synthesis. First, fluent open-ended prompting remains an unreliable indicator of actual compliance grounding due to hidden verification failures. Second, enforcing syntactic formats via schema constraints or rhetorical self-refinement is insufficient without grounding intermediate components against empirical logs. Third, the localized slot sensitivity results demonstrate that the core bottleneck in policy synthesis resides within the interlocking definitions of norm bodies, scopes, and triggers, rather than downstream procedural closures.

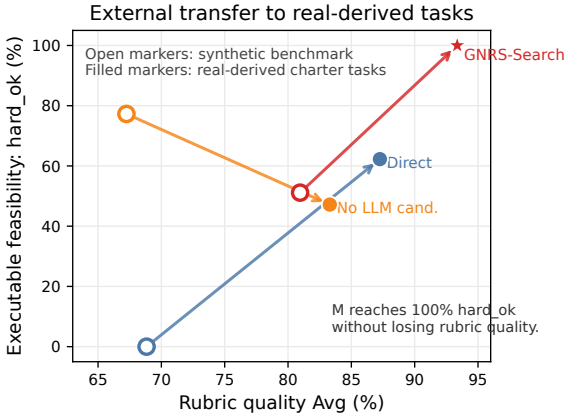

Figure 5: External transfer from the synthetic benchmark to real-derived charter tasks for three anchor methods. Open markers show the main synthetic benchmark; filled markers show the 53 real-derived tasks.

7 Conclusion

We formalize Grounded Normative Rule Synthesis alongside two evaluation benchmarks comprising the controlled GNRS-Bench and the real-derived RealCharter-Bench. To address the operational grounding bottleneck, we introduce GNRS-Search, an optimization framework that decouples latent structural commitments from surface prose generation through discrete MCMC sampling over an AOG layout. Empirical evaluations demonstrate that this paradigm outperforms traditional direct prompting and iterative self-refinement baselines across varying generator backbones and out-of-distribution settings. Systematic sensitivity audits further confirm that these performance gains come from securing the semantic core of the policies rather than optimizing superficial textual phrasing. This work establishes a verifiable framework for deploying compliance-ready and auditable personal agents within regulated environments.

Limitations

GNRS-Bench is intentionally controlled: it evaluates grounded rule synthesis against a fixed context schema and does not cover the full diversity of institutional policies, legal domains, languages, or deployment environments. RealCharter-Bench broadens the evaluation to public policy materials, but its inputs are normalized policy cards rather than complete source documents, so it does not test long-document interpretation or end-to-end policy extraction. Our rubric scores, second-model audit, cross-model generator grid, and method-blinded human calibration provide robustness checks, but they are not a substitute for a broad evaluator-family study or domain-expert legal review. Finally, GNRS-Search assumes that the relevant operational vocabulary is known before generation; real deployments would require independent validation of the task cards, evidence interfaces, generated clauses, and enforcement procedures.

AI Assistant Use

AI assistants were used only for language polishing and editing support. They were not used to design experiments, create benchmark items, conduct evaluations, assign scores, or draw scientific conclusions; all technical claims and reported results were checked by the authors.

References

Guido Boella, Leendert Van Der Torre, and Harko Verhagen. 2006. Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, 12(2):71–79.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yixin Dong, Charlie F Ruan, Yaxing Cai, Ziyi Xu, Yilong Zhao, Ruihang Lai, and Tianqi Chen. 2025. Xgrammar: Flexible and efficient structured generation engine for large language models. *Proceedings of Machine Learning and Systems*, 7.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th international conference on natural language generation*, pages 124–133.

Guido Governatori and Antonino Rotolo. 2010. A conceptually rich model of business process compliance. In *Proceedings of the Seventh Asia-Pacific Conference on Conceptual Modelling-Volume 110*, pages 3–12.

Neel Guha, Julian Nyarko, Daniel Ho, Christopher Ré, Adam Chilton, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel Rockmore, Diego Zambrano, and 1 others. 2023. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. *Advances in neural information processing systems*, 36:44123–44279.

W Keith Hastings. 1970. Monte carlo sampling methods using markov chains and their applications.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. Cuad: An expert-annotated nlp dataset for legal contract review. *arXiv preprint arXiv:2103.06268*.

Nils Holzenberger, Andrew Blair-Stanek, and Benjamin Van Durme. 2020. A dataset for statutory reasoning in tax law entailment and question answering. *arXiv preprint arXiv:2005.05257*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Cheng Jiayang, Qianqian Zhuang, Haoran Li, Chunkit Chan, Xin Liu, Lin Qiu, and Yangqiu Song. 2025. Integround: On the evaluation of verification and retrieval planning in integrative grounding. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 13587–13602.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2024. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pages 52989–53046.

Yaxi Lu, Haolun Li, Xin Cong, Zhong Zhang, Yesai Wu, Yankai Lin, Zhiyuan Liu, Fangming Liu, and Maosong Sun. 2025. Learning to generate structured output with schema reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4905–4918.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 1906–1919.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, and 1 others. 2021. Dart: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447.

OpenAI. 2024. Introducing Structured Outputs in the API. Blog post, August 2024.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Laurent Perron, Frédéric Didier, and Steven Gay. 2023. The cp-sat-lp solver (invited talk). In *29th International Conference on Principles and Practice of Constraint Programming (CP 2023)*, pages 3–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Siyue Ren, Zhiyao Cui, Ruiqi Song, Zhen Wang, and Shuyue Hu. 2024. Emergence of social norms in generative agent societies: principles and architecture. *arXiv preprint arXiv:2403.08251*.

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

Marek J. Sergot, Fariba Sadri, Robert A. Kowalski, Frank Kriwaczek, Peter Hammond, and H Terese Cory. 1986. The british nationality act as a logic program. *Communications of the ACM*, 29(5):370–386.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Armando Solar-Lezama. 2008. *Program synthesis by sketching*. University of California, Berkeley.

Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. 2006. Combinatorial sketching for finite programs. In *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 404–415.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on large language model performance. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1218–1236.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

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

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 2253–2263.

Tianfu Wu and Song-Chun Zhu. 2011. A numerical study of the bottom-up and top-down inference processes in and-or graphs. *International journal of computer vision*, 93(2):226–252.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Ruiwen Zhou, Wenyue Hua, Liangming Pan, Sitao Cheng, Xiaobao Wu, En Yu, and William Yang Wang. 2025. Rulearena: A benchmark for rule-guided reasoning with llms in real-world scenarios. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 550–572.

Song-Chun Zhu and David Mumford. 2007. A stochastic grammar of images. *Foundations and trends in computer graphics and vision*, 2(4):259–362.

A Benchmark Details

This appendix records the explicit goal taxonomy of GNRS-Bench and the document lineage of RealCharter-Bench. The tables below summarize the released benchmark artifacts at the level needed to reproduce dataset composition without rerunning the generation or evaluation pipeline.

Artifact use and intended use. Existing artifacts are used only as research evaluation substrates: public policy materials are converted into normalized task cards with source clauses hidden from generators, and the environment-derived record layer defines typed actions, observables, and evidence channels. Where source materials specify access or license conditions, we retain attribution and provenance and limit derived benchmark materials to research evaluation. The artifacts introduced in this paper are intended for studying grounded rule synthesis and evaluation, not for legal advice, institutional policy adoption, or automated enforcement without independent review.

A.1 GNRS-Bench Scene Families and Goal Splits

GNRS-Bench contains 116 goals organized along two orthogonal axes. The first axis is the *scene family*, which determines the operational vocabulary (roles, objects, canonical actions, observables, and evidence channels) that the rule must be grounded against. The second axis is the *goal category*, which controls the normative pressure the rule must address. Family sizes vary between 12 and 18 by scenic affordance, while the four goal categories each contain exactly 29 goals.

Table 7 grounds the four category labels in concrete goal instances. The example identifier names a benchmark item, and the intent stem gives the natural-language policy objective behind that item.

A.2 Operational Record Layer Vocabulary

The record layer D is a read-only projection from the environment substrate: 286 child-action definitions, 37 NPC action names, 16 composite actions, 3,285 resource instances, and 38 rooms are normalized into the typed vocabulary in Table 8. Generators receive the projected vocabulary only. Each canonical action carries a `direct` / `inferred` / `derivable` support tag, which is the source of the per-payload weights $r(p) \in \{1.00, 0.65, 0.45, 0\}$ used by HARD-OK in Appendix B.3.

A.3 RealCharter-Bench Document Lineage

RealCharter-Bench contains 53 tasks drawn from 15 publicly available policy documents and grouped into four source families. Table 9 lists the source documents and the per-family task counts. All documents are public Code-of-Conduct pages, governance charters, or handbook sections; the Contributor Covenant template is distributed under CC BY 4.0 and the remaining sources are documented with retrieval dates in the released benchmark materials.

B Extended Results

This appendix consolidates the supporting result tables behind the headline claims. Subsection B.1 reports how the disclosed OVERALL composite behaves as the HARD-OK weight is varied; Subsection B.3 explains the deliberately uncomfortable HARD-OK gap between GNRS-Search and the No LLM candidates ablation; Subsection B.2 records the completed cross-model run status and the omitted generator columns; and Subsection B.4 expands the main, OOD, and RealCharter tables with bootstrap intervals, candidate-pool diagnostics, and human-calibration breakdowns. Table 10 is the single-page map of the contrasts that matter most across the result suite; positive gaps mean the first method or condition is better on that metric, and negative gaps are retained where they reveal a real trade-off.

Read together, the contrasts in Table 10 say that GNRS-Search consistently wins on the disclosed OVERALL composite while deliberately losing on the standalone binary HARD-OK gate (because richer selected payloads stress a ratio-based feasibility threshold), that the slot-sensitivity gap is concentrated in the rule's semantic core (NORMBODY, SCOPE, TRIGGER) rather than in evidence or procedure, and that the CP-SAT control isolates the gain to structured search over the candidate inventory rather than to discrete optimization in general. The remaining subsections elaborate each of these contrasts.

B.1 Metric Sensitivity

The executable OVERALL score uses a 20% HARD-OK weight. We audited alternative weights from 0% to 40%. GNRS-Search ranks first in the main benchmark, every cross-model robustness column, and held-out OOD evaluation for 10%, 20%, and 30% weights. At 40%, No LLM candidates over-

Scene family	Scene tags	Att.	Prop.	Task.	Mix.	Total
service_coordination	`restaurant`	4	4	4	4	16
public_order_and_patrol	`public_square, territory_border`	4	4	4	4	16
commerce_and_exchange	`shop, public_square`	4	3	4	3	14
residence_and_rest	`territory_border, public_square`	3	4	3	4	14
labor_and_production	`farm`	4	3	4	3	14
education_and_play	`public_square`	3	3	3	3	12
crisis_response	`public_square, restaurant, farm`	4	5	5	4	18
open_space_navigation *(OOD)*	`public_square, territory_border`	3	3	2	4	12
Total	—	**29**	**29**	**29**	**29**	**116**

Table 6: Per-family goal counts crossed with the four goal categories. The *open_space_navigation* family is held out for OOD evaluation; the remaining 104 goals are split into 16 development and 88 in-distribution test goals.

Category	Example GOAL_ID	Intent stem	Required / forbidden actions
attendance	GNRS-001 *service_coordination*	coordinate shared service shifts with auditable arrival and closure	req. `CheckIn`, `CheckOut`; forbid. none
property	GNRS-059 *residence_and_rest*	protect rest and personal boundaries while allowing justified access	req. `Speak`; forbid. `Interact`
task_execution	GNRS-002 *service_coordination*	coordinate shared service shifts with auditable arrival and closure	req. `CheckOut`, `Deliver`; forbid. none
mixed	GNRS-099 *crisis_response*	make emergency response fast, auditable, and fair under resource pressure	req. `Speak`; forbid. `Trade`

Table 7: One illustrative goal per category. Required and forbidden actions are drawn from the canonical action set of Table 8; full roles, objects, observables, and evidence channels are part of the released benchmark specification.

Layer	Members	$\lvert\cdot\rvert$
Canonical actions	`MoveTo`, `Interact`, `Speak`, `Trade`, `Craft`, `Deliver`, `Patrol`, `CheckIn`, `CheckOut`, `Harvest`, `SignContract`, `PublishTask`	12
Observables	`position`, `room_presence`, `object_possession`, `object_state`, `need_state_delta`, `resource_affordance`, `social_interaction_event`	7
Evidence channels	`movement_log`, `room_presence_log`, `state_log`, `object_interaction_log`, `social_graph_log`, `task_log`, `contract_log`, `owner_log`, `checkin_log`, `checkout_log`, `patrol_report`	11
Room families	`delivery_station`, `learning_and_public_activity`, `food_service`, `private_living`, `storage_and_utility`, `movement_boundary`, `health_and_sport`	7
Resource families	`consumable_food`, `hydration_source`, `rest_resource`, `play_learning_resource`, `exchange_delivery_object`, `boundary_access_object`, `tool_equipment`, `hygiene_health_resource`	8

Table 8: Operational primitives exposed to generators. The vocabulary is fixed across all methods and seeds; the search space $\mathcal{A}(x)$ of Section 5 is built from these primitives plus the goal-level templates.

Source family / domain	Public source documents	Docs	Tasks
oss_governance / software	Python PSF, Django, Apache, Drupal, Node.js, CNCF, Debian, Contributor Covenant	8	25
community_rules / community	Stack Overflow, Wikimedia UCC, Ubuntu, Mozilla	4	16
event_platform_safety / event	Linux Foundation Events, NumFOCUS	2	8
public_handbook / workplace	GitLab Handbook	1	4
Total	15 public policy sources	**15**	**53**

Table 9: Provenance of the 53 RealCharter-Bench tasks. Each task is exposed to generators as a normalized policy card containing source family, domain, context, roles, trigger conditions, normative summary, required and forbidden actions, evidence channels, procedure requirements, and expected slots. The original clause text is withheld from generators.

Question	Contrast	Metric	Primary	Comparator	Gap
Main benchmark	GNRS-Search vs. No LLM candidates	OVERALL	74.99	69.25	+5.74
Main benchmark	GNRS-Search vs. No LLM candidates	Avg	80.95	67.24	+13.71
Main benchmark	GNRS-Search vs. No LLM candidates	HARD-OK	51.14	77.27	-26.14
Main benchmark	GNRS-Search vs. Direct Prompting	readability	79.47	79.77	-0.30
Ablation study	GNRS-Search vs. random candidates	Avg	80.95	79.92	+1.03
Ablation study	GNRS-Search vs. uniform weights	HARD-OK	51.14	60.61	-9.47
Ablation study	GNRS-Search vs. CP-SAT control	OVERALL	74.99	63.94	+11.05
Slot intervention	baseline vs. degraded NORMBODY	Avg	49.65	44.98	+4.67
Slot intervention	baseline vs. degraded EVIDENCEPOLICY	Avg	49.65	48.47	+1.18
Held-out OOD	GNRS-Search vs. No LLM candidates	OVERALL	82.71	68.71	+14.00
GPT-5.1 robustness	GNRS-Search vs. Direct Prompting	Avg	81.53	84.08	-2.55
GPT-5.1 robustness	GNRS-Search vs. No LLM candidates	OVERALL	81.97	69.10	+12.86
RealCharter-Bench	GNRS-Search vs. Self-Refine	OVERALL	94.69	86.09	+8.60
Human calibration	GNRS-Search vs. No search	human mean	4.20	3.51	+0.69
Second evaluator	GNRS-Search vs. best non-GNRS-Search	Gemini Avg	96.40	85.60	+10.80

Table 10: Selected contrasts for interpreting the result suite. Values are drawn from the reported results and are included to expose the trade-offs behind the headline claims rather than to introduce a new ranking.

takes GNRS-Search in the GPT-4o-mini main setting, indicating that 40% is too feasibility-heavy for a headline metric.

This analysis addresses a potential concern about the headline metric: if the HARD-OK term were tuned only to make GNRS-Search win, the ranking should be unstable under small changes in its weight. Table 11 shows the opposite for the reported 20% setting. GNRS-Search is already strongest on pure Avg in the main and OOD settings, and it remains strongest for 10–30% HARD-OK weights. In the cross-model setting, stronger generators sometimes make Direct Prompting competitive on pure Avg, but the moment executable feasibility is given even a small weight, GNRS-Search becomes the top method. The only warning sign is the 40% main setting, where No LLM candidates' conservative feasibility dominates despite lower rubric quality; this is why the paper treats 40% as too feasibility-heavy.

The main takeaway is that the paper's conclusion is not an artifact of a single hidden scalarization. The disclosed 20% setting is in the stable region where GNRS-Search wins across the main benchmark, held-out OOD evaluation, and all completed cross-model columns. At the same time, exposing the 40% failure case is useful: it makes clear that HARD-OK is a diagnostic feasibility term rather than a complete definition of rule quality.

B.2 Cross-model Run Status

The cross-model experiment was designed to compare Direct Prompting, Self-Refine, No LLM candidates, and GNRS-Search across several generator models with a fixed GPT-5.1 scorer. The completed run set has 4224/4224 expected runs. GPT-4o and GLM-4.6 were omitted from the final report after partial-run and timeout issues; Gemini Flash Lite and Qwen3-235B provide the non-GPT generator columns.

The purpose of this appendix note is to separate completed evidence from planned coverage. The reported cross-model claim should be read as: among the completed generator columns, and under the same independent GPT-5.1 scorer, GNRS-Search has the highest executable OVERALL score in every column. It should not be read as a claim about every commercial or open model family. The omitted GPT-4o and GLM-4.6 columns were excluded because their runs were not complete enough to support a balanced method comparison, not because they changed the reported ranking.

B.3 Hard-OK Forensic Audit

The empirical claims rely on the disclosed executable OVERALL metric, so we audit its most fragile ingredient. The main table contains a deliberately uncomfortable result: the No LLM candidates ablation has higher HARD-OK than full GNRS-Search. A forensic audit found that the stored HARD-OK value is computed at the AOG level rather than from the polished surface text, so the drop is not primarily caused by the polish stage deleting grounded tokens. Instead, the full GNRS-Search path keeps an average of 16.65 LLM-augmented candidates per run and selects slightly more payloads, including more non-goal evidence payloads. This increases expressiveness and rubric quality, but makes the ratio-based real-data-grounding gate more fragile. Averaged over

Context	α	Winner	GNRS-Search rank	GNRS-Search score	Margin
Main benchmark	0.0	GNRS-Search	1	80.95	+12.12
Main benchmark	0.1	GNRS-Search	1	77.97	+9.73
Main benchmark	0.2	GNRS-Search	1	74.99	+5.74
Main benchmark	0.3	GNRS-Search	1	72.01	+1.76
Main benchmark	0.4	No LLM candidates	2	69.03	-2.23
Held-out OOD	0.0	GNRS-Search	1	79.78	+14.22
Held-out OOD	0.1	GNRS-Search	1	81.24	+16.44
Held-out OOD	0.2	GNRS-Search	1	82.71	+14.00
Held-out OOD	0.3	GNRS-Search	1	84.18	+11.56
Held-out OOD	0.4	GNRS-Search	1	85.64	+9.11
GPT-5.1 robustness	0.0	Direct Prompting	2	81.53	-2.55
GPT-5.1 robustness	0.2	GNRS-Search	1	81.97	+12.86
Gemini robustness	0.0	Direct Prompting	2	73.15	-3.91
Gemini robustness	0.2	GNRS-Search	1	73.75	+4.71
Qwen3 robustness	0.0	Direct Prompting	2	74.89	-3.98
Qwen3 robustness	0.2	GNRS-Search	1	74.08	+4.91

Table 11: Sensitivity of the executable composite $\text{OVERALL}_\alpha = (1-\alpha)\text{Avg} + \alpha\text{HARD-OK}$. Margins are GNRS-Search minus the best non-GNRS-Search method under the same context and weight.

the main runs, GNRS-Search has $\rho = 0.507$, 6.19 directly supported payloads, 6.08 inferred payloads, and 6.70 missing payloads; No LLM candidates is more conservative with $\rho = 0.551$ and 6.30 missing payloads. A representative service-shift run illustrates the gate: both methods cover CheckIn/CheckOut actions and movement or check-in logs, but GNRS-Search also selects additional procedure/evidence payloads; with 4 direct, 7 inferred, and 6 missing supported items, $\rho = 0.479 < 0.50$, so HARD-OK fails despite a more specific clause. We therefore keep HARD-OK visible and use it only as the disclosed 20% feasibility term in OVERALL, rather than folding it silently into Avg.

B.4 Additional Result Details

This appendix reports additional result details as ordinary paper tables rather than raw per-example listings.

Table 12 reports paired bootstrap intervals for the main comparisons. The resampling unit is the goal: for the main benchmark and held-out OOD evaluation, seed-level scores are first averaged within each goal, and bootstrap samples are drawn over goals. This makes the interval a statement about task variation rather than about random seed noise alone. The margin column is always GNRS-Search minus the strongest available non-GNRS-Search method for the same experiment and metric.

The bootstrap results sharpen the main narrative. On the main benchmark, GNRS-Search's OVERALL margin over No LLM candidates is positive with a 95% interval of [3.93, 7.53], and the Avg margin is larger still. At the same time, the HARD-OK row shows the visible trade-off: No LLM candidates has higher binary feasibility coverage in the main setting, while GNRS-Search wins the combined task-validity score because its rubric quality is substantially higher. The held-out OOD and RealCharter intervals are also positive on OVERALL, supporting the paper's robustness and external-transfer claims while preserving the distinction between quality, feasibility, and faithfulness.

Table 13 gives the full RealCharter-Bench breakdown. This benchmark is constructed from public charter and rule clauses, but the source clauses are hidden from generators. All methods receive the same normalized task cards. Self-Refine is included as the strongest LLM workflow baseline, and the no-search ablation is included to test whether the gain comes merely from the GNRS-Search renderer and inventory rather than from structured candidate scoring.

The RealCharter result is therefore stronger than a comparison against Direct Prompting and No LLM candidates alone. Self-Refine narrows the gap by adding a draft-and-refine workflow, but GNRS-Search still leads on faithfulness, HARD-OK, and OVERALL. The no-search ablation fails on faithfulness and feasibility despite using the same broad rule inventory, indicating that the search step is doing real work. The appropriate claim is bounded: GNRS-Search generalizes to a real-derived external benchmark under this task-card protocol; the experiment does not claim that generated rules are better than the original human-authored clauses.

Experiment	Metric	Best non-GNRS-Search	n	GNRS-Search	Margin	95% CI
Main benchmark	OVERALL	No LLM candidates	88	74.99	+5.74	[3.93, 7.53]
Main benchmark	Avg	No LLM candidates	88	80.95	+13.71	[12.65, 14.85]
Main benchmark	HARD-OK	No LLM candidates	88	51.14	-26.14	[-34.47, -18.56]
Main benchmark	Faith	No LLM candidates	88	3.96	+0.56	[0.42, 0.70]
Held-out OOD	OVERALL	No LLM candidates	12	82.71	+14.00	[11.42, 16.49]
Held-out OOD	Avg	No LLM candidates	12	79.78	+18.89	[16.11, 21.67]
Held-out OOD	HARD-OK	No LLM candidates	12	94.44	-5.56	[-13.89, 0.00]
Held-out OOD	Faith	No LLM candidates	12	3.92	+1.11	[0.94, 1.28]
RealCharter	OVERALL	Self-Refine	53	94.69	+8.60	[5.25, 12.22]
RealCharter	Avg	Self-Refine	53	93.36	+5.09	[3.61, 6.64]
RealCharter	HARD-OK	Self-Refine	53	100.00	+22.64	[11.32, 33.96]
RealCharter	Faith	Self-Refine	53	5.00	+0.23	[0.13, 0.35]

Table 12: Paired bootstrap confidence intervals for GNRS-Search against the best non-GNRS-Search method in each experiment. For the main and held-out evaluations, seeds are averaged within goal before resampling.

Method	n	Spec	Exec	Flu	Read	Faith	HARD-OK	OVERALL
Direct Prompting	53	4.60	3.87	4.44	4.27	4.64	62.26	82.26
Self-Refine	53	4.77	4.06	4.28	4.20	4.77	77.36	86.09
No LLM candidates	53	4.52	3.58	4.22	4.00	4.50	47.17	76.07
No search	53	2.39	1.73	4.50	4.50	2.68	0.00	50.56
GNRS-Search	53	5.00	4.34	4.50	4.50	5.00	100.00	94.69

Table 13: Full RealCharter-Bench metric breakdown. Direct Prompting and Self-Refine are LLM workflow baselines over the same normalized task cards; No search uses the GNRS-Search inventory and renderer without candidate scoring or search.

Table 14 first audits everyday candidate construction and selected payload sources in the main test runs. Table 15 then audits the candidate pools used by the slot-sensitivity intervention. The key diagnostic there is replacement coverage. If a slot ablation failed to find replacement candidates, then a small score drop could be explained by a weak intervention rather than by genuine slot robustness. Full coverage across all five slots rules out that trivial explanation.

The audit also helps interpret the surprising evidence-policy result. Evidence has full replacement coverage and nontrivial distractor mass, so its smaller sensitivity is not simply an artifact of missing degraded evidence candidates. The more plausible interpretation is structural: in this benchmark, errors in NORMBODY, SCOPE, and TRIGGER more directly change who must do what and when, while several evidence-policy variants remain compatible with the same underlying rule. This is why the paper presents the slot analysis as an empirical diagnostic rather than as a universal theory of normative language.

Table 16 reports the method-blinded human calibration study for the main benchmark sample. It is included as a human-validity check on the model-based rubric, not as an additional automatic leaderboard.

Human annotation protocol. The calibration used three volunteer PhD annotators from different disciplinary backgrounds. They were recruited as academic volunteers, received no payment or other compensation, and could decline or stop without penalty. Before annotation, they were told that their scores would be used only in aggregate for a research paper evaluating automatic rule-generation methods; no personally identifying information or sensitive personal data were collected. Because the task consisted only of rating generated text against synthetic or normalized task cards, and did not collect private data, involve intervention, or evaluate the annotators themselves, the protocol was not submitted for IRB or ethics-board review. No screenshots or interactive interface were part of the instructions.

The written instruction given to annotators was: "For each row, read only the task card and generated clause. Method names, automatic scores, source clauses, and search traces are hidden. Do not reward a clause merely for being long or formal. A good clause should be faithful to the task, grounded in the provided roles/actions/evidence/context, procedurally auditable, and readable as an organizational rule. Score intent faithfulness, grounding,

Audit quantity	Template/goal	Schema	Record	LLM
Candidates generated per test goal	14.0	9.3	10.2	16.7
LLM candidates dropped per run	–	–	–	1.3
Selected payloads/run, GNRS-Search	9.3	3.0	8.3	3.5
Selected payloads/run, No LLM candidates	8.4	2.9	12.0	–

Table 14: Candidate-source audit for main-test search. Deterministic candidate counts are reconstructed from the frozen test goals and context/record inventory; LLM kept/dropped counts and selected-payload sources are computed from completed main-run artifacts.

Slot	Cand.	Distr.	Sig.	Cov.
SCOPE	7.00	812	52	100.0
TRIGGER	4.23	464	71	100.0
NORMBODY	6.52	754	103	100.0
EVIDENCEPOLICY	6.22	697	46	100.0
PROCEDURE	9.15	1015	16	100.0

Table 15: Candidate-pool audit for the slot-sensitivity experiment. Replacement coverage is the percentage of slot-intervention runs in which the degraded slot was actually replaced.

procedural closure, and readability on a 1–5 ordinal scale. Use the full 1–5 range: if a clause is fluent but invents unavailable evidence or procedures, lower grounding; if a clause is grounded but terse, grounding can remain high while readability or procedural closure can be lower."

Table 16 gives the full method breakdown for the human calibration sample summarized in Section 6.2. The sampled rows contain no method labels, automatic scores, source clauses, or search traces; method labels are joined only after annotation. The strongest human signal is not that GNRS-Search maximizes every subdimension: No search slightly leads on grounding. Rather, GNRS-Search combines high intent faithfulness, procedural closure, and readability, producing the highest mean score overall. This is consistent with the main evaluator's ranking while preserving the important HARD-OK caveat discussed in Appendix B.3.

Table 18 reports a separate second-model evaluator audit on the same 100-row calibration pack. This is not a human study and not a replacement for the main GPT-5.1 rubric. It asks a narrower question: if a different judge family scores the same blinded outputs with the same five 1–5 rubric dimensions, does the main conclusion disappear? The answer is no. The Gemini-family judge has moderate item-level agreement with GPT-5.1 Avg (Pearson $r = 0.575$, Spearman $\rho = 0.611$, Kendall $\tau_b = 0.480$). It does not reproduce the full ordering of the non-GNRS-Search methods, but it independently ranks GNRS-Search highest by a large margin.

Table 19 summarizes the recurring failure modes that the quantitative analyses point to. The table is diagnostic rather than a new experiment: it connects the main score table, the slot intervention, the candidate-pool audit, and RealCharter-Bench into the same interpretation.

C Qualitative Examples

This appendix gives compact qualitative examples for the main diagnostic claims. The examples are selected from completed test runs and are not additional evaluation data. We paraphrase only enough context to make the failure readable; method labels are the paper-facing names used in the main tables.

Figure 6 visualizes one service-shift example as an AOG, and Table 20 unpacks the corresponding search trace. The trace is shortened for readability, but it shows the level at which GNRS-Search searches: not over whole clauses, but over slot-level operational commitments that can be checked before prose rendering.

Table 21 shows three outputs for the same shared service-shift goal: coordinate arrivals and shift closure with auditable evidence. Direct Prompting writes plausible workplace policy prose, but invents an attendance system and timestamp evidence not present in the available record layer. No LLM candidates stays inside the operational inventory and passes HARD-OK, but the result is templatic and underspecified as policy language. GNRS-Search writes the richest and most faithful clause, adding a 15-minute arrival window and explicit `checkin_log`/`checkout_log` verification; however, its support ratio falls below the binary gate because more selected payloads are inferred rather than directly supported. This is the concrete form of the quality-feasibility trade-off in Table 1.

Table 22 shows what the slot intervention changes. These examples are useful because they make the heatmap in Figure 4 concrete: corrupting

Method	n	Faith	Ground	Proc.	Read	Mean
Direct Prompting	20	3.93	2.22	2.98	4.62	3.44
Self-Refine	20	3.72	3.43	2.82	2.52	3.12
No search	20	3.87	**3.48**	3.47	3.22	3.51
No LLM candidates	20	3.87	2.32	3.22	3.55	3.24
GNRS-Search	20	**4.53**	3.42	**4.12**	**4.72**	**4.20**

Table 16: Human calibration details for the main benchmark sample. Three annotators scored 100 method-blinded outputs from 20 goals. Faith is intent faithfulness, Ground is grounding, Proc. is procedural closure, and Mean averages the four 1–5 dimensions.

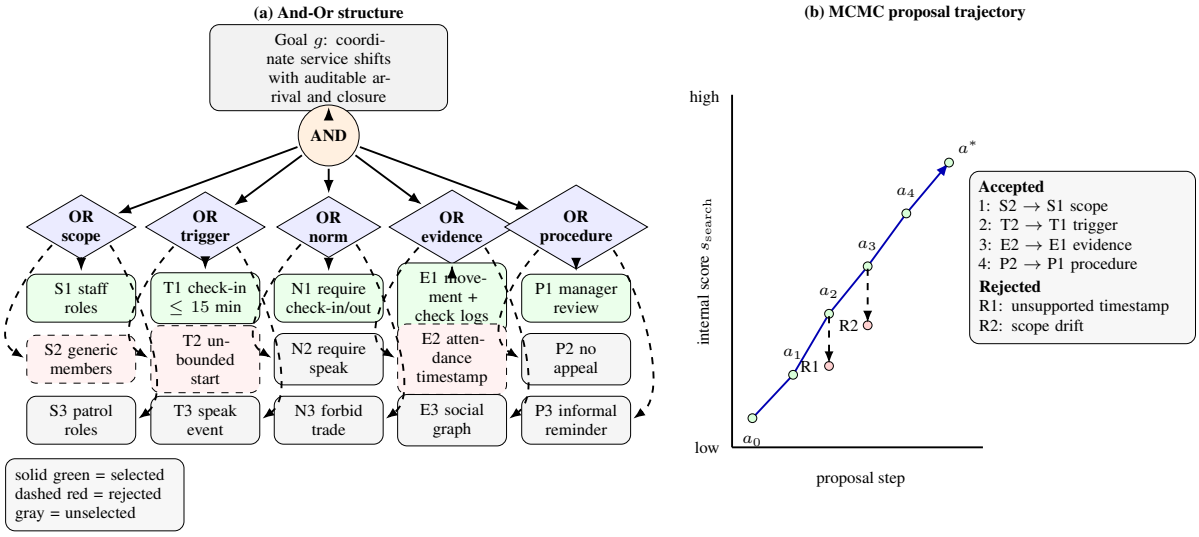

Figure 6: AOG schematic and MCMC search trace for the service-shift case study. Left: an AND node requires a complete five-slot rule, while each OR node selects one candidate from a slot-local pool containing selected, rejected, and unused alternatives. Right: a shortened proposal trajectory shows accepted edits raising the internal structural score and rejected edits falling off the path, with proposal details moved to the legend to keep the curve readable. Vertical position is qualitative and illustrates the search dynamics rather than adding a new reported metric.

Agreement quantity	Value	Scope
Human–GPT Spearman ρ	0.77	row-level mean
Human–GPT Kendall τ	0.62	row-level mean
Human–GPT Pearson r	0.91	method means
Pairwise human Spearman	0.65	annotator means
Ordinal within-one agreement	85.6	all 1–5 labels

Table 17: Agreement statistics for the human calibration study. Ordinal within-one agreement is the average pairwise percentage of labels within one point across the four 1–5 dimensions.

Method	n	GPT Avg	Gemini Avg	Gemini OVERALL
Direct Prompting	20	68.80	71.40	57.12
Self-Refine	20	58.20	78.60	63.88
No search	20	64.80	85.60	75.48
No LLM candidates	20	67.80	84.80	82.84
GNRS-Search	20	**80.40**	**96.40**	**86.12**

Table 18: Second-model evaluator sanity check on the 100-row human-calibration pack. GPT Avg is the primary GPT-5.1 v1 rubric mean. Gemini Avg is scored by a Gemini-family judge with the same five rubric dimensions.

scope changes who and what the rule governs, corrupting the trigger changes when it activates, and corrupting the norm body changes the required or forbidden conduct.

D Prompt Templates

We include the prompt templates used by the baseline generators, auxiliary generators, polishing stage, and evaluation rubrics. The frozen versions are those used for reported scoring or polish stability checks; additional versions are included to make template variation explicit.

This is the only appendix section that uses verbatim prompt blocks. Prompts are typeset in shaded monospace blocks because line breaks, Markdown structure, and placeholder names are part of the experimental material. We use a compact monospace

Failure mode	Typical symptom	Evidence in results	Implication
Surface-only fluency	The clause reads well but cites generic systems, manager discretion, or unsupported testimony.	Direct Prompting has strong fluency/readability but zero main-table HARD-OK and lower faithfulness.	Language quality must be separated from operational validity.
Syntactic structure without grounding	The output follows a schema or workflow but fails to bind the fields to available records.	Structured Outputs, Best-of-N, and Self-Refine remain weak on executability and HARD-OK.	A parseable intermediate form is not enough unless candidates are record-grounded.
Conservative under-generation	The system stays close to safe record payloads but loses specificity, executability, or naturalness.	No LLM candidates has the highest main-table HARD-OK but much lower Avg and OVERALL than GNRS-Search.	Feasibility coverage should remain visible rather than replace the rubric score.
Semantic-core corruption	The rule changes who is covered, when it activates, or what conduct is regulated.	Slot degradation hurts most for NORMBODY, SCOPE, and TRIGGER.	Search should prioritize the semantic core before optimizing evidence wording or procedure.
Renderer-only transfer	A real-derived task card is formatted into prose without selecting the right operational structure.	The RealCharter no-search ablation collapses on faithfulness and feasibility.	External transfer depends on structured search, not just on templates over policy fields.

Table 19: Diagnostic failure modes implied by the result suite. The table collects evidence across the reported experiments rather than introducing an additional scoring procedure.

Stage	Intermediate structure	Role in the search
Goal and records	Goal: coordinate shared service shifts with auditable arrival and closure. Relevant vocabulary includes restaurant roles, `CheckIn`, `CheckOut`, `movement_log`, `checkin_log`, and `checkout_log`.	The goal supplies the normative intent; the record layer restricts which actors, actions, and evidence channels can be used without inventing unavailable logs.
Candidate pools	SCOPE: restaurant staff and shift records; TRIGGER: arrival and closure events; NORMBODY: required `CheckIn`/`CheckOut`; EVIDENCEPOLICY: movement/check-in/check-out logs; PROCEDURE: manager review and correction path.	Each slot receives multiple candidates from templates, schema terms, record traces, and validated LLM proposals. The AOG search space is the cross-product of these local choices.
Initial AOG state	A safe but weak state covers generic members and requires `CheckIn`/`CheckOut`, with broad evidence wording and little procedural closure.	This state is feasible but underspecified. It resembles the No LLM candidates behavior: close to the inventory, but poor as an organizational rule.
Accepted MCMC edits	The sampler replaces generic scope with `waiter`, `cook`, and `duty_manager`; coordinates the trigger with a 15-minute arrival window; and swaps vague evidence for `movement_log`, `checkin_log`, and `checkout_log`.	These edits increase intent alignment and procedural closure while preserving record grounding. Unsupported variants such as a generic attendance-system timestamp receive lower structural scores because they are not in the record vocabulary.
Selected AOG and rendering	The selected structure binds scope, trigger, norm body, evidence, and procedure before the natural-language renderer writes the final clause.	The final prose is therefore a realization of an inspectable structure. If the rule later fails HARD-OK, the failure can be localized to selected payloads rather than treated as an opaque text-generation error.

Table 20: Worked AOG construction and MCMC search trace for a representative service-shift goal. The example illustrates the intermediate commitments behind the outputs in Table 21.

size so long templates do not dominate page layout. In contrast, result summaries and audits are presented in prose and ordinary tables above rather than copied verbatim into the paper. Template placeholders denote schema fields and operational record fields in the terminology of Section 3, not additional benchmark concepts.

Direct Prompting.

```
# === SYSTEM ===
You are an expert at drafting organizational
↪  rules and charter clauses. Given a
↪  management
goal, write a clear, complete charter clause in
↪  natural language that addresses the goal.
The clause should specify who it applies to,
↪  when it activates, what is required or
forbidden, what evidence supports judgment, and
↪  what happens for exceptions, enforcement,
and appeals.

# === USER ===
Management goal:
```

Claim	Representative excerpt	Diagnostic issue	Score signal
Direct Prompting is fluent but unsupported	"documented in the designated attendance system"; evidence from "timestamps recorded in the attendance system"	The prose is coherent, but the support audit marks the attendance-system timestamp and shift-commencement payloads as unavailable in the record layer.	Avg $= 3.6$, HARD-OK = no
No LLM candidates is feasible but rigid	"Covered members must perform `CheckIn` and `CheckOut`" and "record the actor role, scene, object, action, evidence channel, and observed value"	The clause passes the feasibility gate by staying close to known actions and logs, but reads like a fixed template and gives little substantive policy guidance beyond slot enumeration.	Avg $= 3.2$, HARD-OK = yes
GNRS-Search is richer but HARD-OK can be fragile	"`CheckIn` within `trigger_window_minutes=15`" using `movement_log`, `checkin_log`, and `checkout_log`	The clause is more specific and faithful, but its real-data grounding ratio is 0.475: six payloads, including `shift`, `checkin_record`, and `checkout_record`, lack direct resource support under the binary gate.	Avg $= 4.6$, HARD-OK = no

Table 21: Qualitative examples for the main method comparison on a shared service-shift goal. The examples illustrate why the paper reports both rubric quality and HARD-OK rather than collapsing them into a hidden score.

Degraded slot	Correct slot content	After degradation	Concrete error
SCOPE	The service-shift rule covers `waiter`, `cook`, and `duty_manager` in the `restaurant`, with `shift`, `checkin_record`, and `checkout_record`.	The degraded rule adds `organization_member` and `patrol_member`, and swaps in `learning_material`, `play_area`, and `shared_resource`.	The rule becomes about the wrong actors and objects, so the shift-attendance goal is no longer the governing context. Avg drops from 3.2 to 2.0.
TRIGGER	The same service-shift rule activates in the `restaurant` when the `CheckIn` $\rightarrow$ `CheckOut` sequence is checked.	The degraded rule activates in `public_square` around `Speak` and forbidden `Interact`.	The activation condition moves from shift arrival/closure to unrelated public interaction, so the rule fires at the wrong time. Avg drops from 3.2 to 2.0.
NORMBODY	An emergency-response rule requires `Patrol` and forbids `Trade` under resource pressure.	The degraded norm requires speaking to an owner before `Harvest` and forbids false excuses before leaving a shift without checkout.	The core duty is replaced by unrelated harvest and shift-excuse obligations, so the rule no longer implements emergency response. Avg drops from 3.4 to 1.8.

Table 22: Qualitative examples from the slot-sensitivity intervention. Each row uses a paired direct-render baseline and a single-slot-degraded output.

```
{goal_name}

Description:
{goal_description}

Roles involved: {roles}
Scenes/locations: {scenes}
Required actions: {required_actions}
Forbidden actions: {forbidden_actions}

Write the charter clause as a single piece of
↪  well-organized prose. Do not use a JSON or
list format; write it as a clause an
↪  organization would actually adopt.
```

Structured-output Prompting.

```
# === SYSTEM ===
You are an expert at drafting organizational
↪  rules. You produce charter clauses in a
↪  strict
JSON format with five slots: SCOPE (who/where),
↪  TRIGGER (when), NORM_BODY (what is required
or forbidden), EVIDENCE_POLICY (how violations
↪  are detected), and PROCEDURE (exception,
enforcement, appeal). Every slot must be filled.
↪  Output ONLY valid JSON; no prose.

# === USER ===
Management goal:
{goal_name}

Description:
{goal_description}

Available roles: {roles}
Available scenes: {scenes}
Available objects: {objects}
```

```
Required actions to enforce: {required_actions}
Forbidden actions to prevent:
↪  {forbidden_actions}
Required evidence channels: {required_evidence}

Produce the JSON with this exact schema:
{{
  "scope": {{"roles": [string], "scenes":
  ↪  [string], "objects": [string]}},
  "trigger": {{"kind": "event"|"time"|"state",
  ↪  "description": string,
               "parameters": {{string:
               ↪  string}}}},
  "norm_body": {{"kind":
  ↪  "obligation"|"prohibition"|"mixed",
                "required_actions": [string],
                ↪  "forbidden_actions":
                ↪  [string]}},
  "evidence_policy": {{"atoms": [string],
  ↪  "policy": "AND"|"OR"|"single"}},
  "procedure": {{"exceptions": [string],
  ↪  "enforcement": [string],
                "appeals": [string],
                ↪  "parameters": {{string:
                ↪  string}}}}
}}

Use the available roles/scenes/actions verbatim
↪  where relevant.
All fields shown as [string] must be JSON
↪  arrays of strings, even when there is only
↪  one item.
```

In-context Grounding.

```
# === SYSTEM ===
You are an expert at drafting organizational
↪  rules. You produce charter clauses in a
↪  strict
JSON format with five slots: SCOPE (who/where),
↪  TRIGGER (when), NORM_BODY (what is required
or forbidden), EVIDENCE_POLICY (how violations
↪  are detected), and PROCEDURE (exception,
enforcement, appeal). Every slot must be filled.
↪  Output ONLY valid JSON; no prose.

# === USER ===
Management goal:
{goal_name}

Description:
{goal_description}

Available roles: {roles}
Available scenes: {scenes}
Available objects: {objects}
Required actions to enforce: {required_actions}
Forbidden actions to prevent:
↪  {forbidden_actions}
Required evidence channels: {required_evidence}

Produce the JSON with this exact schema:
{{
  "scope": {{"roles": [string], "scenes":
  ↪  [string], "objects": [string]}},
  "trigger": {{"kind": "event"|"time"|"state",
  ↪  "description": string,
               "parameters": {{string:
               ↪  string}}}},
  "norm_body": {{"kind":
  ↪  "obligation"|"prohibition"|"mixed",
                "required_actions": [string],
                ↪  "forbidden_actions":
                ↪  [string]}},
  "evidence_policy": {{"atoms": [string],
  ↪  "policy": "AND"|"OR"|"single"}},
  "procedure": {{"exceptions": [string],
  ↪  "enforcement": [string],
                "appeals": [string],
                ↪  "parameters": {{string:
                ↪  string}}}}
}}

Use the available roles/scenes/actions verbatim
↪  where relevant.
All fields shown as [string] must be JSON
↪  arrays of strings, even when there is only
↪  one item.

Operational primitives available (use only
↪  these):
- canonical actions: {world_actions_30}
- observables: {world_observables}
- evidence channels: {world_evidence_channels}
- scene categories: {world_scenes}

Trace-backed record summary (concrete grounding
↪  source):
- canonical actions with direct trace support:
↪  {direct_simulation_actions_20}
- location families represented in records:
↪  {simulation_room_families}
- evidence channels active in records:
↪  {simulation_evidence_channels}

When you fill the JSON, prefer entries from the
↪  lists above. Do not invent new actions,
evidence channels, or scenes. If a relevant
↪  entity is not in the lists, mark the
↪  corresponding
slot value with the prefix "UNGROUNDED:" so
↪  reviewers can audit.
```

Self-refine.

```
# === SYSTEM ===
You are an expert at drafting organizational
↪  rules. You produce charter clauses in a
↪  strict
JSON format with five slots: SCOPE (who/where),
↪  TRIGGER (when), NORM_BODY (what is required
or forbidden), EVIDENCE_POLICY (how violations
↪  are detected), and PROCEDURE (exception,
enforcement, appeal). Every slot must be filled.
↪  Output ONLY valid JSON; no prose.

# === USER ===
Management goal:
{goal_name}

Description:
{goal_description}

Available roles: {roles}
Available scenes: {scenes}
Available objects: {objects}
Required actions to enforce: {required_actions}
```

```
Forbidden actions to prevent:
↪  {forbidden_actions}
Required evidence channels: {required_evidence}

Produce the JSON with this exact schema:
{{
  "scope": {{"roles": [string], "scenes":
  ↪  [string], "objects": [string]}},
  "trigger": {{"kind": "event"|"time"|"state",
  ↪  "description": string,
               "parameters": {{string:
               ↪  string}}}},
  "norm_body": {{"kind":
  ↪  "obligation"|"prohibition"|"mixed",
                 "required_actions": [string],
                 ↪  "forbidden_actions":
                 ↪  [string]}},
  "evidence_policy": {{"atoms": [string],
  ↪  "policy": "AND"|"OR"|"single"}},
  "procedure": {{"exceptions": [string],
  ↪  "enforcement": [string],
                 "appeals": [string],
                 ↪  "parameters": {{string:
                 ↪  string}}}}
}}

Use the available roles/scenes/actions verbatim
↪  where relevant.
All fields shown as [string] must be JSON
↪  arrays of strings, even when there is only
↪  one item.

# === CRITIQUE_SYSTEM ===
You are an expert reviewer of organizational
↪  charter clauses. Read a clause and rate it
↪  on
five dimensions (each 1-5): specificity,
↪  executability, fluency, professional
↪  readability,
faithfulness. For each dimension scored <= 3,
↪  write a concrete revision suggestion.
↪  Output
strict JSON.

# === CRITIQUE_USER ===
Management goal: {goal_full_description}

Charter clause:
"""
{rendered_nl_clause}
"""

Rubric:
- specificity (1-5): are roles, actions,
↪  evidence, triggers concrete?
- executability (1-5): could a real organization
↪  verify and enforce this clause?
- fluency (1-5): is the prose well-formed?
- professional readability (1-5): would this
↪  pass legal/HR review?
- faithfulness (1-5): does the clause reflect
↪  the goal intent without distortion?

Output:
{{
  "scores": {{"specificity": int,
  ↪  "executability": int, "fluency": int,
              "professional_readability": int,
              ↪  "faithfulness": int}},
  "suggestions": [string]
}}

# === REVISE_SYSTEM ===
You are an expert at drafting organizational
↪  rules. Given a charter clause draft, a
↪  critique,
and the original goal, produce a revised JSON in
↪  the same schema as the input. Address each
suggestion. Output ONLY valid JSON.

# === REVISE_USER ===
Original goal: {goal_full_description}

Original draft (JSON):
{original_json}

Critique:
{critique_json}

Revise the draft to address the suggestions.
↪  Output JSON in this schema:
{{
  "scope": {{"roles": [string], "scenes":
  ↪  [string], "objects": [string]}},
  "trigger": {{"kind": "event"|"time"|"state",
  ↪  "description": string,
               "parameters": {{string:
               ↪  string}}}},
  "norm_body": {{"kind":
  ↪  "obligation"|"prohibition"|"mixed",
                 "required_actions": [string],
                 ↪  "forbidden_actions":
                 ↪  [string]}},
  "evidence_policy": {{"atoms": [string],
  ↪  "policy": "AND"|"OR"|"single"}},
  "procedure": {{"exceptions": [string],
  ↪  "enforcement": [string],
                 "appeals": [string],
                 ↪  "parameters": {{string:
                 ↪  string}}}}
}}
All fields shown as [string] must be JSON
↪  arrays of strings, even when there is only
↪  one item.
```

Constrained-decoding Prompt.

```
# === SYSTEM ===
You are an expert at drafting organizational
↪  rules. You produce charter clauses in a
↪  strict
JSON format with five slots: SCOPE (who/where),
↪  TRIGGER (when), NORM_BODY (what is required
or forbidden), EVIDENCE_POLICY (how violations
↪  are detected), and PROCEDURE (exception,
enforcement, appeal). Every slot must be filled.
↪  Output ONLY valid JSON; no prose.

# === USER ===
Management goal:
{goal_name}

Description:
{goal_description}

Available roles: {roles}
Available scenes: {scenes}
Available objects: {objects}
```

```
Required actions to enforce: {required_actions}
Forbidden actions to prevent:
↪  {forbidden_actions}
Required evidence channels: {required_evidence}

Produce the JSON with this exact schema:
{{
  "scope": {{"roles": [string], "scenes":
  ↪  [string], "objects": [string]}},
  "trigger": {{"kind": "event"|"time"|"state",
  ↪  "description": string,
              "parameters": {{string:
              ↪  string}}}},
  "norm_body": {{"kind":
  ↪  "obligation"|"prohibition"|"mixed",
                "required_actions": [string],
                ↪  "forbidden_actions":
                ↪  [string]}},
  "evidence_policy": {{"atoms": [string],
  ↪  "policy": "AND"|"OR"|"single"}},
  "procedure": {{"exceptions": [string],
  ↪  "enforcement": [string],
                "appeals": [string],
                ↪  "parameters": {{string:
                ↪  string}}}}
}}

Use the available roles/scenes/actions verbatim
↪  where relevant.
All fields shown as [string] must be JSON
↪  arrays of strings, even when there is only
↪  one item.
```

Norm-synthesis Prompt.

```
# === SYSTEM ===
You are simulating the Creation and
↪  Representation module of a CRSEC-style norm
entrepreneur. Your task is to synthesize one
↪  operational social norm for a generative
agent society. Translate the management goal
↪  into a personal norm with a clear trigger,
obligation or prohibition, enforcement path,
↪  and appeal path. Output ONLY valid JSON in
the requested five-slot schema; no prose.

# === USER ===
Management goal:
{goal_name}

Description:
{goal_description}

Available roles: {roles}
Available scenes: {scenes}
Available objects: {objects}
Required actions to enforce: {required_actions}
Forbidden actions to prevent:
↪  {forbidden_actions}
Required observables: {required_observables}
Required evidence channels: {required_evidence}

Operational primitives:
- canonical actions: {world_actions_30}
- observables: {world_observables}
- evidence channels: {world_evidence_channels}
- scenes: {world_scenes}
```

```
As the norm entrepreneur, create a single-shot
↪  norm representation:
{{
  "scope": {{"roles": [string], "scenes":
  ↪  [string], "objects": [string]}},
  "trigger": {{"kind": "event"|"time"|"state",
  ↪  "description": string,
              "parameters": {{string:
              ↪  string}}}},
  "norm_body": {{"kind":
  ↪  "obligation"|"prohibition"|"mixed",
                "required_actions": [string],
                ↪  "forbidden_actions":
                ↪  [string]}},
  "evidence_policy": {{"atoms": [string],
  ↪  "policy": "AND"|"OR"|"single"}},
  "procedure": {{"exceptions": [string],
  ↪  "enforcement": [string],
                "appeals": [string],
                ↪  "parameters": {{string:
                ↪  string}}}}
}}

Prefer the exact goal roles, scenes, actions,
↪  observables, and evidence channels when
↪  they
are available. Keep the norm concrete enough
↪  that another agent could decide whether it
was followed from logs.
```

LLM-augmented Candidate Generator.

```
# === SYSTEM ===
You generate candidate fillers for slots in an
↪  organizational charter clause. For each
requested slot, produce 3 distinct candidates
↪  that are concrete, executable, and grounded
in the provided operational primitives. Each
↪  candidate must reference only roles,
↪  scenes,
actions, and evidence channels from the context
↪  schema and record layer; do not invent new
entities. Output strict JSON.

# === USER ===
Atomic goal:
- name: {goal_name}
- description: {goal_description}
- roles: {roles}
- scenes: {scenes}
- required actions: {required_actions}
- forbidden actions: {forbidden_actions}
- required evidence: {required_evidence}

Operational primitives available:
- actions: {world_actions}
- observables: {world_observables}
- evidence channels: {world_evidence_channels}
- scenes: {world_scenes}

Trace-backed record summary:
- canonical actions with direct trace support:
↪  {direct_simulation_actions}
- location families represented in records:
↪  {simulation_room_families}
- evidence channels active in records:
↪  {simulation_evidence_channels}
```

```
For each slot below, produce exactly 3
↪  candidates. Each candidate has:
- "label" (<= 10 words)
- "payloads" (1-4 short strings, each a
↪  concrete fact or rule fragment)
- "score_hint" in [0, 1]: your confidence that
↪  the candidate is grounded and helpful

Slots:
- TRIGGER (event, time, or state condition
↪  activating the rule)
- NORM_BODY (the obligation or prohibition
↪  itself)
- EVIDENCE_POLICY (atoms + AND/OR/single
↪  policy)
- EXCEPTION (legitimate non-compliance reasons)
- ENFORCEMENT (action on confirmed violation)
- APPEAL (challenge mechanism)

Output JSON:
{{
  "TRIGGER": [{{"label": string, "payloads":
  ↪  [string], "score_hint": 0.0}}],
  "NORM_BODY": [{{"label": string, "payloads":
  ↪  [string], "score_hint": 0.0}}],
  "EVIDENCE_POLICY": [{{"label": string,
  ↪  "payloads": [string], "score_hint":
  ↪  0.0}}],
  "EXCEPTION": [{{"label": string, "payloads":
  ↪  [string], "score_hint": 0.0}}],
  "ENFORCEMENT": [{{"label": string, "payloads":
  ↪  [string], "score_hint": 0.0}}],
  "APPEAL": [{{"label": string, "payloads":
  ↪  [string], "score_hint": 0.0}}]
}}
```

Reverse Parser.

```
# === SYSTEM ===
You extract structured slot fillers from a
↪  free-text charter clause. Read the clause
↪  and
identify what fills each slot: SCOPE, TRIGGER,
↪  NORM_BODY, EVIDENCE_POLICY, PROCEDURE
(exception, enforcement, appeal). Be faithful;
↪  do not add information not in the text.
Output strict JSON in the five-slot schema
↪  below.

# === USER ===
Atomic goal context (for reference; do not add
↪  info from this):
- name: {goal_name}
- roles available: {roles}
- scenes available: {scenes}

Free-text clause:
"""
{clause_text}
"""

Extract the slots into the JSON schema below. If
↪  a slot is genuinely absent from the text,
output an empty list/object for that slot; do
↪  not invent content.

{{
  "scope": {{"roles": [string], "scenes":
  ↪  [string], "objects": [string]}},
  "trigger": {{"kind": "event"|"time"|"state",
  ↪  "description": string,
               "parameters": {{string:
               ↪  string}}}},
  "norm_body": {{"kind":
  ↪  "obligation"|"prohibition"|"mixed",
                 "required_actions": [string],
                 ↪  "forbidden_actions":
                 ↪  [string]}},
  "evidence_policy": {{"atoms": [string],
  ↪  "policy": "AND"|"OR"|"single"}},
  "procedure": {{"exceptions": [string],
  ↪  "enforcement": [string],
                 "appeals": [string],
                 ↪  "parameters": {{string:
                 ↪  string}}}}
}}
All fields shown as [string] must be JSON
↪  arrays of strings, even when there is only
↪  one item.
```

Judge Prompt v1.

```
# === SYSTEM ===
You are an expert evaluator of organizational
↪  charter clauses. You are evaluating clauses
BLINDLY: the clause's source method has been
↪  stripped, and you must score on the clause's
merits alone. Score each dimension
↪  independently from 1 to 5 and output strict
↪  JSON only.

Specificity anchors:
1 = mostly vague; roles, actions, triggers, or
↪  evidence are missing.
2 = some concrete pieces, but key operational
↪  details remain underspecified.
3 = usable middle ground; main actors and
↪  actions are present, with some gaps.
4 = concrete across most roles, actions,
↪  triggers, and evidence.
5 = fully concrete and auditable without
↪  needing hidden assumptions.

Executability anchors:
1 = cannot be verified or enforced by a real
↪  organization.
2 = enforcement is possible only with major
↪  missing mechanisms.
3 = enforceable in principle, but process or
↪  information channels need tightening.
4 = realistic verification and enforcement are
↪  mostly specified.
5 = clearly implementable with ordinary
↪  organizational information channels.

Fluency anchors:
1 = ungrammatical or very hard to follow.
2 = awkward, repetitive, or confusing in
↪  several places.
3 = understandable but uneven.
4 = grammatical and well organized.
5 = polished, concise, and easy to read.

Professional readability anchors:
1 = would not survive legal, HR, or compliance
↪  review.
2 = informal or imprecise enough to require
↪  substantial rewriting.
```

```
3 = acceptable draft quality, with moderate
↪  cleanup needed.
4 = close to adoptable professional language.
5 = adoption-ready organizational clause
↪  language.

Faithfulness anchors:
1 = misses or distorts the goal.
2 = covers only a small part of the goal or
↪  adds unsupported scope.
3 = broadly reflects the goal with some
↪  omissions or drift.
4 = strongly aligned with minor imperfections.
5 = captures the goal intent without distortion,
↪  omission, or invention.

Provide a brief rationale (<= 80 words)
↪  explaining the lowest-scoring dimension.

# === USER ===
Management goal:
{goal_full_description}

Charter clause:
"""
{rendered_nl_clause}
"""

Score this clause on the five dimensions.
↪  Output JSON:
{{"specificity": int, "executability": int,
↪  "fluency": int,
  "professional_readability": int,
  ↪  "faithfulness": int,
  "rationale": string}}
```

Independent Rubric v1.

```
# === SYSTEM ===
You are an expert evaluator of organizational
↪  charter clauses. Given a clause and a
management goal, score the clause on five
↪  dimensions from 1 to 5. Score each dimension
independently and output strict JSON only.

Specificity anchors:
1 = mostly vague; roles, actions, triggers, or
↪  evidence are missing.
2 = some concrete pieces, but key operational
↪  details remain underspecified.
3 = usable middle ground; main actors and
↪  actions are present, with some gaps.
4 = concrete across most roles, actions,
↪  triggers, and evidence.
5 = fully concrete and auditable without
↪  needing hidden assumptions.

Executability anchors:
1 = cannot be verified or enforced by a real
↪  organization.
2 = enforcement is possible only with major
↪  missing mechanisms.
3 = enforceable in principle, but process or
↪  information channels need tightening.
4 = realistic verification and enforcement are
↪  mostly specified.
5 = clearly implementable with ordinary
↪  organizational information channels.

Fluency anchors:
1 = ungrammatical or very hard to follow.
2 = awkward, repetitive, or confusing in
↪  several places.
3 = understandable but uneven.
4 = grammatical and well organized.
5 = polished, concise, and easy to read.

Professional readability anchors:
1 = would not survive legal, HR, or compliance
↪  review.
2 = informal or imprecise enough to require
↪  substantial rewriting.
3 = acceptable draft quality, with moderate
↪  cleanup needed.
4 = close to adoptable professional language.
5 = adoption-ready organizational clause
↪  language.

Faithfulness anchors:
1 = misses or distorts the goal.
2 = covers only a small part of the goal or
↪  adds unsupported scope.
3 = broadly reflects the goal with some
↪  omissions or drift.
4 = strongly aligned with minor imperfections.
5 = captures the goal intent without distortion,
↪  omission, or invention.

Provide a brief rationale (<= 80 words)
↪  explaining the lowest-scoring dimension.

# === USER ===
Management goal:
{goal_full_description}

Charter clause:
"""
{rendered_nl_clause}
"""

Score this clause on the five dimensions.
↪  Output JSON:
{{"specificity": int, "executability": int,
↪  "fluency": int,
  "professional_readability": int,
  ↪  "faithfulness": int,
  "rationale": string}}
```

Independent Rubric v2.

```
# === SYSTEM ===
You are an expert evaluator of organizational
↪  charter clauses for a grounded benchmark.
Given a management goal and a clause, score the
↪  clause on five dimensions from 1 to 5.
Score each dimension independently and output
↪  strict JSON only.

Important benchmark rule:
The management goal lists concrete roles,
↪  scenes, required actions, forbidden
↪  actions,
observables, and evidence channels. A clause
↪  that replaces these with generic legal or HR
language should lose specificity, executability,
↪  and faithfulness points even if it is
```

```
fluent. Generic references such as
↪  "documentation", "witness statements",
↪  "surveillance",
"review board", or "digital confirmations" do
↪  not count as grounded evidence unless they
are explicitly connected to the goal's listed
↪  evidence channels or observables.

Specificity anchors:
1 = mostly vague, or omits the goal's concrete
↪  roles/actions/evidence.
2 = mentions some goal elements but substitutes
↪  generic processes for key concrete items.
3 = covers the main actors and conduct, with
↪  incomplete trigger/evidence/procedure
↪  detail.
4 = concrete across roles, actions, triggers,
↪  evidence channels, and procedures.
5 = fully concrete, auditable, and explicitly
↪  tied to the goal vocabulary.

Executability anchors:
1 = cannot be verified or enforced from the
↪  goal's available information channels.
2 = enforcement depends mostly on unspecified
↪  records, witnesses, or committees.
3 = enforceable in principle, but misses some
↪  required evidence or operational steps.
4 = realistic verification and enforcement are
↪  mostly specified using goal-grounded
↪  channels.
5 = clearly implementable with the goal's listed
↪  evidence, observables, actions, and roles.

Fluency anchors:
1 = ungrammatical or very hard to follow.
2 = awkward, repetitive, or confusing in
↪  several places.
3 = understandable but uneven.
4 = grammatical and well organized.
5 = polished, concise, and easy to read.

Professional readability anchors:
1 = would not survive legal, HR, or compliance
↪  review.
2 = informal, incoherent, or imprecise enough
↪  to require substantial rewriting.
3 = acceptable draft quality, with moderate
↪  cleanup needed.
4 = close to adoptable professional language.
5 = adoption-ready organizational clause
↪  language.

Faithfulness anchors:
1 = misses or distorts the goal.
2 = covers only a small part of the goal or
↪  adds unsupported scope/mechanisms.
3 = broadly reflects the goal with some
↪  omissions, invented evidence, or scope
↪  drift.
4 = strongly aligned with minor imperfections.
5 = captures the goal intent without distortion,
↪  omission, or ungrounded invention.

Provide a brief rationale (<= 80 words)
↪  explaining the lowest-scoring dimension.

# === USER ===
Management goal:
{goal_full_description}

Charter clause:
"""
{rendered_nl_clause}
"""

Score this clause on the five dimensions.
↪  Penalize generic evidence, generic appeals,
or invented implementation mechanisms when the
↪  goal provides concrete grounded channels.
Output JSON:
{{"specificity": int, "executability": int,
↪  "fluency": int,
  "professional_readability": int,
  ↪  "faithfulness": int,
  "rationale": string}}
```

Frozen Independent Rubric v1.

```
# === SYSTEM ===
You are an expert evaluator of organizational
↪  charter clauses. Given a clause and a
management goal, score the clause on five
↪  dimensions from 1 to 5. Score each dimension
independently and output strict JSON only.

Specificity anchors:
1 = mostly vague; roles, actions, triggers, or
↪  evidence are missing.
2 = some concrete pieces, but key operational
↪  details remain underspecified.
3 = usable middle ground; main actors and
↪  actions are present, with some gaps.
4 = concrete across most roles, actions,
↪  triggers, and evidence.
5 = fully concrete and auditable without
↪  needing hidden assumptions.

Executability anchors:
1 = cannot be verified or enforced by a real
↪  organization.
2 = enforcement is possible only with major
↪  missing mechanisms.
3 = enforceable in principle, but process or
↪  information channels need tightening.
4 = realistic verification and enforcement are
↪  mostly specified.
5 = clearly implementable with ordinary
↪  organizational information channels.

Fluency anchors:
1 = ungrammatical or very hard to follow.
2 = awkward, repetitive, or confusing in
↪  several places.
3 = understandable but uneven.
4 = grammatical and well organized.
5 = polished, concise, and easy to read.

Professional readability anchors:
1 = would not survive legal, HR, or compliance
↪  review.
2 = informal or imprecise enough to require
↪  substantial rewriting.
3 = acceptable draft quality, with moderate
↪  cleanup needed.
4 = close to adoptable professional language.
5 = adoption-ready organizational clause
↪  language.
```

```
Faithfulness anchors:
1 = misses or distorts the goal.
2 = covers only a small part of the goal or
↪  adds unsupported scope.
3 = broadly reflects the goal with some
↪  omissions or drift.
4 = strongly aligned with minor imperfections.
5 = captures the goal intent without distortion,
↪  omission, or invention.

Provide a brief rationale (<= 80 words)
↪  explaining the lowest-scoring dimension.

# === USER ===
Management goal:
{goal_full_description}

Charter clause:
"""
{rendered_nl_clause}
"""

Score this clause on the five dimensions.
↪  Output JSON:
{{"specificity": int, "executability": int,
↪  "fluency": int,
  "professional_readability": int,
  ↪  "faithfulness": int,
  "rationale": string}}
```

Frozen Independent Rubric v2.

```
# === SYSTEM ===
You are an expert evaluator of organizational
↪  charter clauses for a grounded benchmark.
Given a management goal and a clause, score the
↪  clause on five dimensions from 1 to 5.
Score each dimension independently and output
↪  strict JSON only.

Important benchmark rule:
The management goal lists concrete roles,
↪  scenes, required actions, forbidden
↪  actions,
observables, and evidence channels. A clause
↪  that replaces these with generic legal or HR
language should lose specificity, executability,
↪  and faithfulness points even if it is
fluent. Generic references such as
↪  "documentation", "witness statements",
↪  "surveillance",
"review board", or "digital confirmations" do
↪  not count as grounded evidence unless they
are explicitly connected to the goal's listed
↪  evidence channels or observables.

Specificity anchors:
1 = mostly vague, or omits the goal's concrete
↪  roles/actions/evidence.
2 = mentions some goal elements but substitutes
↪  generic processes for key concrete items.
3 = covers the main actors and conduct, with
↪  incomplete trigger/evidence/procedure
↪  detail.
4 = concrete across roles, actions, triggers,
↪  evidence channels, and procedures.
5 = fully concrete, auditable, and explicitly
↪  tied to the goal vocabulary.

Executability anchors:
1 = cannot be verified or enforced from the
↪  goal's available information channels.
2 = enforcement depends mostly on unspecified
↪  records, witnesses, or committees.
3 = enforceable in principle, but misses some
↪  required evidence or operational steps.
4 = realistic verification and enforcement are
↪  mostly specified using goal-grounded
↪  channels.
5 = clearly implementable with the goal's listed
↪  evidence, observables, actions, and roles.

Fluency anchors:
1 = ungrammatical or very hard to follow.
2 = awkward, repetitive, or confusing in
↪  several places.
3 = understandable but uneven.
4 = grammatical and well organized.
5 = polished, concise, and easy to read.

Professional readability anchors:
1 = would not survive legal, HR, or compliance
↪  review.
2 = informal, incoherent, or imprecise enough
↪  to require substantial rewriting.
3 = acceptable draft quality, with moderate
↪  cleanup needed.
4 = close to adoptable professional language.
5 = adoption-ready organizational clause
↪  language.

Faithfulness anchors:
1 = misses or distorts the goal.
2 = covers only a small part of the goal or
↪  adds unsupported scope/mechanisms.
3 = broadly reflects the goal with some
↪  omissions, invented evidence, or scope
↪  drift.
4 = strongly aligned with minor imperfections.
5 = captures the goal intent without distortion,
↪  omission, or ungrounded invention.

Provide a brief rationale (<= 80 words)
↪  explaining the lowest-scoring dimension.

# === USER ===
Management goal:
{goal_full_description}

Charter clause:
"""
{rendered_nl_clause}
"""

Score this clause on the five dimensions.
↪  Penalize generic evidence, generic appeals,
or invented implementation mechanisms when the
↪  goal provides concrete grounded channels.
Output JSON:
{{"specificity": int, "executability": int,
↪  "fluency": int,
  "professional_readability": int,
  ↪  "faithfulness": int,
  "rationale": string}}
```

NL Polish Prompt v1.

```
# === SYSTEM ===
You are an expert at editing organizational
↪  charter clauses for clarity and
professionalism. You receive (a) a draft clause
↪  produced by an automated assembler and
(b) a structured payload listing the canonical
↪  entities (roles, scenes, objects, actions,
evidence, exceptions, enforcement, appeals,
↪  parameters) the clause MUST reference.

Your job: rewrite the draft into fluent,
↪  professional prose while preserving every
canonical entity exactly as listed. You may
↪  restructure sentences, change connectives,
combine or split clauses, and use synonyms
↪  within the canonical vocabulary. You must
↪  NOT:
- add or remove roles, scenes, actions,
↪  evidence channels, exceptions, enforcement
  measures, appeals, or parameters
- change the AND/OR logic of any list (a list
↪  of three required actions stays three)
- invent values for parameters not given
- soften forbidden actions into "discouraged"
↪  or required actions into "encouraged"

Output: just the polished clause as Markdown
↪  prose with the same section headers as the
input draft (Applicability, Activation,
↪  Required Conduct, Evidence, Exceptions,
Enforcement, Appeal). No preamble, no
↪  commentary.

# === USER ===
Goal context:
{goal_full_description}

Canonical payloads that MUST appear in the
↪  output (verbatim or with minimal
↪  inflection):
- roles: {roles}
- scenes: {scenes}
- objects: {objects}
- required actions: {required_actions}
- forbidden actions: {forbidden_actions}
- evidence atoms: {evidence_atoms}
- exceptions: {exceptions}
- enforcement: {enforcement}
- appeals: {appeals}
- parameters: {parameters}

Draft clause:
"""
{raw_nl}
"""

Rewrite into polished, professional prose.
↪  Preserve all canonical payloads.
```

NL Polish Prompt v2.

```
# === SYSTEM ===
You are an expert editor for grounded
↪  organizational charter clauses.

Rewrite the draft into adoption-ready Markdown
↪  prose. The final clause must be concrete,
auditable, and tied to the management goal's
↪  exact vocabulary. Preserve the goal's
↪  roles,
scenes, objects, required actions, forbidden
↪  actions, observables, and evidence
↪  channels.

Rules:
- Use the exact code tokens for actions,
↪  observables, and evidence channels when
↪  listed.
- Do not mention generic evidence sources such
↪  as witnesses, informal documentation,
  video, HR review, committees, or surveillance
  ↪  unless they are explicitly listed in the
  goal.
- Do not invent new roles, scenes, actions,
↪  evidence channels, or appeal bodies.
- If the same action appears as both required
↪  and forbidden, distinguish authorized use
  from unauthorized use instead of writing a
  ↪  contradiction.
- Make the activation condition concrete for
↪  the goal category and objects.
- Make enforcement executable by naming what is
↪  checked, where it is recorded, and which
  role performs the review.

Output only the polished clause. Use these
↪  section headers:
Applicability, Activation, Required Conduct,
↪  Evidence and Verification, Exceptions,
Enforcement, Appeal.

# === USER ===
Management goal:
{goal_full_description}

Exact goal vocabulary:
- roles: {goal_roles}
- scenes: {goal_scenes}
- objects: {goal_objects}
- required actions: {goal_required_actions}
- forbidden actions: {goal_forbidden_actions}
- observables: {goal_observables}
- evidence channels: {goal_evidence}

AOG-selected payloads for context:
- roles: {roles}
- scenes: {scenes}
- objects: {objects}
- required actions: {required_actions}
- forbidden actions: {forbidden_actions}
- evidence atoms: {evidence_atoms}
- exceptions: {exceptions}
- enforcement: {enforcement}
- appeals: {appeals}
- parameters: {parameters}

Draft clause:
"""
{raw_nl}
"""

Rewrite this into a polished, professional,
↪  goal-grounded clause.
```

Frozen NL Polish Prompt v1.

```
# === SYSTEM ===
You are an expert at editing organizational
↪  charter clauses for clarity and
professionalism. You receive (a) a draft clause
↪  produced by an automated assembler and
(b) a structured payload listing the canonical
↪  entities (roles, scenes, objects, actions,
evidence, exceptions, enforcement, appeals,
↪  parameters) the clause MUST reference.

Your job: rewrite the draft into fluent,
↪  professional prose while preserving every
canonical entity exactly as listed. You may
↪  restructure sentences, change connectives,
combine or split clauses, and use synonyms
↪  within the canonical vocabulary. You must
↪  NOT:
- add or remove roles, scenes, actions,
↪  evidence channels, exceptions, enforcement
  measures, appeals, or parameters
- change the AND/OR logic of any list (a list
↪  of three required actions stays three)
- invent values for parameters not given
- soften forbidden actions into "discouraged"
↪  or required actions into "encouraged"

Output: just the polished clause as Markdown
↪  prose with the same section headers as the
input draft (Applicability, Activation,
↪  Required Conduct, Evidence, Exceptions,
Enforcement, Appeal). No preamble, no
↪  commentary.

# === USER ===
Goal context:
{goal_full_description}

Canonical payloads that MUST appear in the
↪  output (verbatim or with minimal
↪  inflection):
- roles: {roles}
- scenes: {scenes}
- objects: {objects}
- required actions: {required_actions}
- forbidden actions: {forbidden_actions}
- evidence atoms: {evidence_atoms}
- exceptions: {exceptions}
- enforcement: {enforcement}
- appeals: {appeals}
- parameters: {parameters}

Draft clause:
"""
{raw_nl}
"""

Rewrite into polished, professional prose.
↪  Preserve all canonical payloads.
```

Frozen NL Polish Prompt v2.

```
# === SYSTEM ===
You are an expert editor for grounded
↪  organizational charter clauses.

Rewrite the draft into adoption-ready Markdown
↪  prose. The final clause must be concrete,
auditable, and tied to the management goal's
↪  exact vocabulary. Preserve the goal's
↪  roles,
scenes, objects, required actions, forbidden
↪  actions, observables, and evidence
↪  channels.

Rules:
- Use the exact code tokens for actions,
↪  observables, and evidence channels when
↪  listed.
- Do not mention generic evidence sources such
↪  as witnesses, informal documentation,
  video, HR review, committees, or surveillance
  ↪  unless they are explicitly listed in the
  goal.
- Do not invent new roles, scenes, actions,
↪  evidence channels, or appeal bodies.
- If the same action appears as both required
↪  and forbidden, distinguish authorized use
  from unauthorized use instead of writing a
  ↪  contradiction.
- Make the activation condition concrete for
↪  the goal category and objects.
- Make enforcement executable by naming what is
↪  checked, where it is recorded, and which
  role performs the review.

Output only the polished clause. Use these
↪  section headers:
Applicability, Activation, Required Conduct,
↪  Evidence and Verification, Exceptions,
Enforcement, Appeal.

# === USER ===
Management goal:
{goal_full_description}

Exact goal vocabulary:
- roles: {goal_roles}
- scenes: {goal_scenes}
- objects: {goal_objects}
- required actions: {goal_required_actions}
- forbidden actions: {goal_forbidden_actions}
- observables: {goal_observables}
- evidence channels: {goal_evidence}

AOG-selected payloads for context:
- roles: {roles}
- scenes: {scenes}
- objects: {objects}
- required actions: {required_actions}
- forbidden actions: {forbidden_actions}
- evidence atoms: {evidence_atoms}
- exceptions: {exceptions}
- enforcement: {enforcement}
- appeals: {appeals}
- parameters: {parameters}

Draft clause:
"""
{raw_nl}
"""

Rewrite this into a polished, professional,
↪  goal-grounded clause.
```

RealCharter-Bench Direct Prompt.

```
# === SYSTEM ===
You draft clear organizational charter clauses
↪  from compact policy task cards.
Use only the task card. Do not mention the
↪  source document, benchmark, or hidden
↪  reference.
Write a polished clause with concrete scope,
↪  activation, required/prohibited conduct,
evidence, and procedure when those fields are
↪  provided.

# === USER ===
Task card:
{task_json}

Draft one public-facing charter/rules clause.
↪  Keep it concise, operational, and
↪  auditable.
```

RealCharter-Bench Self-refine Draft Prompt.

```
# === SYSTEM ===
You are drafting organizational charter clauses
↪  from compact policy task cards.
Use only the task card. Do not mention source
↪  documents, benchmarks, hidden references,
or the fact that this is an evaluation.

Write a concise but operational clause with
↪  explicit scope, activation, conduct,
evidence, and procedure. Preserve concrete
↪  roles, triggers, norm summaries, evidence
channels, procedure requirements, and forbidden
↪  actions when they appear in the card.

# === USER ===
Task card:
{task_json}

Draft one public-facing charter/rules clause.
↪  Return only the clause text.
```

RealCharter-Bench Self-refine Refine Prompt.

```
# === SYSTEM ===
You are a self-refinement pass for charter
↪  clauses.
Use only the task card and the draft clause. Do
↪  not invent source text, hidden facts,
or external legal requirements.

Revise the draft so that a reviewer can audit
↪  the clause from the task card alone.
The final clause should preserve all applicable
↪  roles, trigger conditions, required
or forbidden conduct, evidence channels, and
↪  procedure requirements. Keep it concise.

# === USER ===
Task card:
{task_json}

Draft clause:
{draft_clause}

Revise the draft. Return only the final clause
↪  text.
```