BEYOND CONSTRAINT COUNTS: A DIFFICULTY-AWARE VIEW OF INSTRUCTION FOLLOWING IN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models increasingly operate under complex user instructions, where violating explicit requirements can undermine otherwise useful outputs. Many instruction-following evaluations summarise compliance at the task level or by constraint count, potentially obscuring substantial variation among individual requirements. We study whether model-conditioned properties of constrained satisfaction provide a more informative characterisation of this variation. Across structurally graded tasks, we estimate individual difficulty using boundary slack, concentration among sampled successful outputs, and conditional perplexity. Compliance is consistently associated with all three properties, with a mean absolute correlation of $|\rho| \approx 0.55$, while prompt interventions designed to shift these properties improve compliance by approximately 6.5 percentage points on average. In multi-constraint settings, the same properties characterise interaction difficulty, where coupling can reduce constituent compliance by up to 15 points, while at fixed constraint counts, mean follow rates differ by > 50 points across sets stratified by measured constituent difficulty. We further test controlled extensions beyond feasible sharp constraints, showing that across low to high difficulty settings, infeasibility recognition and fuzzified properties increase by approximately 70.7% and change with a mean 53.7%, respectively. Together, these results provide evidence that difficulty-aware measurements capture variation in instruction following that raw constraint counts miss, motivating broader evaluation of difficulty-aware diagnostics and mitigation.

1 INTRODUCTION

Large language models (LLMs) are now used for complex workflows, where they operate under increasingly elaborate user instructions concerning formatting Liu et al. (2024), content Mao et al. (2025), and restrictions Ramírez et al. (2026). Here, task correctness is insufficient; an output may become unusable when instructions are violated Ouyang et al. (2022); Pyatkin et al. (2025). This risks user-perceived hallucinations and undermines model usefulness. Therefore, understanding instruction following (IF) failures is crucial for reliable LLM assistants Zhou et al. (2023).

Many prior works provide an important but incomplete view of this problem. Most have established that IF performance deteriorates under challenging conditions in specific tasks Wen et al. (2024); Zou et al. (2025), such as in agentic Qi et al. (2026), coding Chi et al. (2026), long-sequence Zhang et al. (2026), and multi-step reasoning tasks Qin et al. (2026); Li et al. (2025). These are valuable for identifying failures, but characterising the aggregate outcomes of IF tied to particular tasks risks obscuring substantial heterogeneity at the instruction level. Different instructions may impose very different demands, yet contribute to the same aggregate compliance. This makes it unclear why performance deteriorates as more instructions are introduced, and has negative implications for instruction tuning, which relies on these insights for data recipes.

To address this gap, we propose a cross-task, model-conditioned framework to isolate one aspect of instruction difficulty via a constrained-satisfaction formulation, with task-adapted measurements. First, we model individual instructions as numeric constraints to measure their difficulty using three properties: slack, the margin from the constraint boundary; solution concentration, the concentration

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

among sampled successful outputs; and conditional perplexity, the model-conditioned accessibility of feasible samples. Across tasks, IFR increases with slack ($\rho = 0.74$) and decreases with solution concentration ($\rho = -0.40$) and conditional perplexity ($\rho = -0.53$), with matched interventions improving IFR correspondingly. This suggests that instruction failures are likely associated with boundary proximity, concentrated successful generations, and lower conditional likelihood. Extending this analysis to instruction pairs, we compare expected additive marginal and joint solution concentration, revealing non-additive interactions: constraints may overlap or couple, respectively easing or increasing joint difficulty.

Building on both accounts, we find that constraint count alone is insufficient to characterise observed IF capacity. Despite an identical five constraint count, on difficulty-stratified pools, the task-averaged follow rate falls from about 85% on easy sets to 37% on medium and 12% on hard difficulty sets. Beyond the feasibility limit, difficulty also characterises IF contradiction recognition and satisfaction of constituent instructions. Here, we measure difficulty as the numeric separation between conflicting constraints from a common solution. From high to low difficulty, infeasibility recognition and target-count attainment increase by an average of $\sim 70.7\%$ and 91.73%, respectively. Finally, the difficulty account also establishes a controlled extension to fuzzy numeric instructions, with context-sensitive and graded boundaries. Boundary variability increases by 25.12% on average, while semantically matched contexts explain an average of 23.12% of the variance. Stronger fuzzy modifiers also increase individual difficulty: solution concentration rises by 104.70%, fuzzy slack falls by 44.79%, and conditional perplexity rises by 11.65%.

Our contributions are as follows:

1. We introduce a model-conditioned, constrained-satisfaction framework for instruction difficulty, using task-adapted measurements across three structurally graded tasks.
2. We provide controlled evidence that these difficulty measurements capture systematic variation in instruction-following performance beyond constraint count alone.
3. We test how the difficulty-aware account extends to several controlled instruction regimes, including individual, joint, limit, infeasible, and fuzzy regimes.

2 Instruction Following as Constrained Satisfaction

Feasibility We view an instruction as a constraint on the space of valid task outputs. Given task t, instance x, output y, and constraint c, let $v_t(y; x, c) \in \{0, 1\}$ denote a verifier indicating whether y satisfies c. Each constraint therefore induces a feasible region $F_t(x, c) = \{y : v_t(y; x, c) = 1\}$. For a set of constraints $C = \{c_1, \ldots, c_{|C|}\}$, exact instruction following requires satisfying all constituent c, with joint feasible region $F_t(x, C) = \bigcap_{c \in C} F_t(x, c)$. We denote exact satisfaction of C as $\phi_t(y; x, C) = \mathbf{1}[y \in F_t(x, C)] = \prod_{c \in C} v_t(y; x, c)$, with task-correct outputs (Appendix B.2). The constraint set is *feasible* when $F_t(x, C) \neq \varnothing$ and *infeasible* otherwise. In the feasible regime, model failure means producing an output outside $F_t(x, C)$; under infeasibility, no output can satisfy all instructions. We treat these cases separately below.

Model-Conditioned An LLM does not receive classical constraint functions nor is it a formal solver. Instead, a prompt $\pi_t(x, C)$ specifies the task and its constraints in natural language, and a model with parameters θ generates $Y \sim p_\theta(\cdot \mid \pi_t(x, C))$. Thus, the constrained-satisfaction formulation describes only the externally verifiable goal of generation. Whether a satisfying output exists is a property of the constraint set, whereas whether the model reliably generates one also depends on how its conditional output distribution is situated relative to the feasible region. **This distinction motivates our Monte Carlo analysis of instruction difficulty**.

2.1 Measuring Constraint Difficulty

Graded Task Spectrum. We consider three instruction-following tasks: Scheduling, MBPP, and OpenEnded, chosen to span a spectrum from highly to weakly structured output spaces. Scheduling is the most structured setting: each instance is a natural-language constraint-satisfaction problem whose exact feasible region can be computed. MBPP occupies an intermediate regime, requiring the model to implement Python functions. Its output space is substantially

broader, but remains constrained by program syntax. OPENENDED is the least structured setting and consists of open-ended question-answering prompts for which many substantively and linguistically distinct responses may be valid. This spectrum allows us to test whether the same measurements of instruction difficulty remain informative as the underlying output space becomes progressively less structured. The three tasks also provide a practical balance between such structural coverage and the cost of combinatorial sampling.

Constraint Construction and Verification. In SCHEDULING, constraints correspond to known mathematical boundaries, and verification follows directly from the relevant decision variables. For MBPP and OPENENDED, constraints are constructed to be atomic, countable, and verifiable properties of the generated output. Unless otherwise stated, each such instruction is treated as a separate constraint c with verifier $v_t(y; x, c)$ as defined in § 2. This common design reduces dependence on general-purpose model judges and gives the quantities introduced below a comparable interpretation across tasks. Full task, constraint, and verifier details are provided in Appendices A and B.

Generation Protocol. Because an LLM's output distribution cannot be enumerated directly, we estimate the quantities used in our analyses through Monte Carlo sampling. We evaluate $M = 3$ models under $\Omega = 3$ prompt templates, with $R = 10$ independent retries for each M-Ω condition, using decoding temperature $T = 0.8$. We sample $Y_{im\omega r} \sim p_{\theta_m,T}(\cdot \mid \pi_{t,\omega}(x_i, C_i))$, where $m = 1, \ldots, M$, $\omega = 1, \ldots, \Omega$, and $r = 1, \ldots, R$. Let ψ_t be an indicator for task-correctness (Appendix B.2). For m, we compute the instruction-following rate (IFR) as:

$$\mathrm{IFR}^{(m)} = \left(\sum_{i=1}^{N_t} \sum_{\omega=1}^{\Omega} \sum_{r=1}^{R} \phi_t(Y_{im\omega r}; x_i, C_i) \right) \Big/ \left(\sum_{i=1}^{N_t} \sum_{\omega=1}^{\Omega} \sum_{r=1}^{R} \psi_t(Y_{im\omega r}; x_i) \right). \tag{1}$$

For aggregate results across models, we report $\mathrm{IFR} = \frac{1}{M} \sum_{m=1}^{M} \mathrm{IFR}^{(m)}$. Thus, model-wise IFR averages over instances, prompt templates, and repeated generations, while the aggregate IFR additionally averages over models. The same generated samples form the empirical output distributions used to measure instruction difficulty. Appendix C provides full generation and prompt details.

3 WHAT MAKES AN INDIVIDUAL CONSTRAINT DIFFICULT?

IFR measures the outcome of difficulty, but does not identify its source. We therefore propose three properties that characterise constraints according to how reliably they are satisfied.

Boundary Proximity. Feasible outputs may differ in their distance to the constraint boundary, with outputs nearer the boundary being more fragile. For each task instance x_i and c, we pool generations across M, Ω, and R, retain task-correct feasible generations, and uniformly rarefy to $\mathcal{R}_{t,i}(c)$ with $|\mathcal{R}_{t,i}(c)| = n^\star = 20$. Let $s_t(y; x_i, c)$ denote the task-instance-normalised slack of feasible output y. We normalise slack as the relative distance to the boundary. We compute the constraint-level slack at $N_t = 20$ task instances as

$$\bar{s}_{t,i}(c) = \frac{1}{n^\star} \sum_{y \in \mathcal{R}_{t,i}(c)} s_t(y; x_i, c) \qquad \bar{s}_t(c) = \frac{1}{N_t} \sum_{i=1}^{N_t} \bar{s}_{t,i}(c). \tag{2}$$

Solution Space Concentration. We measure whether feasible outputs span many or a few modes using the Simpson Concentration Index (Simpsons for brevity), where higher values indicate that success concentrates on fewer distinct solutions. For fixed t, c, thresholding pairwise task-specific distance $d_t(y_{ij}, y_{ik})$ at τ_t (Appendix E.5) induces K_i solution modes $\{G_{ik}\}_{k=1}^{K_i}$. Let $n_{ik} = |G_{ik}|$, so $\sum_{k=1}^{K_i} n_{ik} = n^\star$. We compute the finite-sample-corrected Simpson concentration as

$$q_{t,i}(c) = \frac{\sum_{k=1}^{K_i} n_{ik}(n_{ik} - 1)}{n^\star(n^\star - 1)}, \qquad \bar{q}_t(c) = \frac{1}{N_t} \sum_{i=1}^{N_t} q_{t,i}(c). \tag{3}$$

Conditional Accessibility. $\bar{s}_t(c)$ and $\bar{q}_t(c)$ describe feasible outputs but not how probable they are under the model's conditional distribution. We therefore compute the response-token-normalized NLL, only descriptively termed "conditional accessibility", with prompt template tokens used as conditioning context. Higher values indicate that feasible outputs occupy lower-probability regions of the prompted distribution. Let $\bar{\ell}^{(m)}_{t,i}(c)$ denote the model-wise mean response-token-normalized NLL over feasible generations (Appendix E.8). The constraint-level conditional perplexity is

$$e_{t,i}(c) = \exp\left(\frac{1}{M}\sum_{m=1}^{M}\bar{\ell}^{(m)}_{t,i}(c)\right), \qquad \bar{e}_t(c) = \frac{1}{N_t}\sum_{i=1}^{N_t} e_{t,i}(c). \tag{4}$$

3.1 Difficulty Is Associated with Slack, Simpsons, and Perplexity

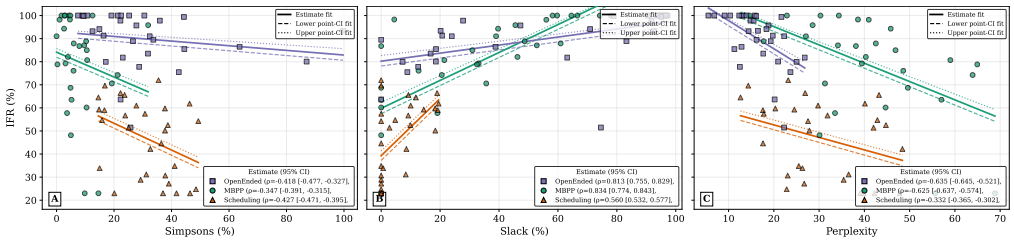


Figure 1: Observable properties of individual constraint difficulty.

Having defined three properties of constraint difficulty, we next ask whether they track observed IFR. Figure 1 relates each of these properties to IFR across OpenEnded, MBPP, and Scheduling. Each point represents an individual constraint. Lines show linear fits, while the legends report Spearman's rank correlation ρ with 95% confidence intervals. For brevity, we write $\bar{s}$, $\bar{q}$ and $\bar{e}$ when t and c are clear from context.

Together, the three panels reveal a consistent directional pattern across tasks: IFR decreases with solution concentration and conditional perplexity, and increases with slack. Slack shows the strongest and most consistent association, with positive correlations for OpenEnded ($\rho = 0.813$), MBPP ($\rho = 0.834$), and Scheduling ($\rho = 0.560$), indicating that constraints whose successful generations lie farther from the failure boundary are followed more reliably. Solution concentration is negatively associated with IFR across all tasks (OpenEnded: $\rho = -0.418$; MBPP: $\rho = -0.347$; Scheduling: $\rho = -0.427$), suggesting that constraints whose sampled successes occupy fewer solution modes tend to be harder to satisfy. Conditional perplexity provides a second negative signal (OpenEnded: $\rho = -0.635$; MBPP: $\rho = -0.625$; Scheduling: $\rho = -0.332$), indicating that higher perplexity among sampled feasible outputs is also associated with lower IFR.

Across tasks, higher IFR is associated with greater measured slack, lower concentration among sampled successful outputs, and lower conditional perplexity. These consistent patterns suggest that the three properties provide complementary signals associated with constraint difficulty.

3.2 Targeted Interventions on Difficulty Measures

Table 1: Mitigation interventions and cross-metric effects. Each intervention is evaluated on IFR and all three difficulty metrics. Bold entries denote the metric directly targeted by each intervention.

	OpenEnded			MBPP			Scheduling		
Δ IFR	+8.94 $^{+8.08}_{+9.82}$	+7.75 $^{+6.95}_{+8.56}$	+8.72 $^{+8.12}_{+9.33}$	+4.57 $^{+3.57}_{+5.54}$	+5.11 $^{+4.22}_{+6.03}$	+5.80 $^{+4.94}_{+6.67}$	+5.58 $^{+5.01}_{+6.17}$	+7.24 $^{+6.41}_{+8.04}$	+5.35 $^{+4.38}_{+6.34}$
Δ Slack	**+24.69** $^{**+23.08**}_{**+26.29**}$	+10.00 $^{+8.78}_{+11.22}$	+7.19 $^{+6.09}_{+8.30}$	**+10.54** $^{**+9.08**}_{**+11.90**}$	+3.99 $^{+3.04}_{+4.95}$	+3.18 $^{+2.18}_{+4.18}$	**+9.49** $^{**+8.24**}_{**+10.73**}$	+3.12 $^{+1.91}_{+4.33}$	+1.96 $^{+0.66}_{+3.26}$
Δ Concentration	-3.91 $^{-4.93}_{-2.89}$	**-11.87** $^{**-13.40**}_{**-10.30**}$	-2.76 $^{-3.94}_{-1.59}$	-2.50 $^{-3.46}_{-1.54}$	**-8.31** $^{**-9.62**}_{**-6.92**}$	-2.37 $^{-3.55}_{-1.20}$	-2.19 $^{-3.42}_{-0.96}$	**-6.81** $^{**-8.44**}_{**-5.14**}$	-1.93 $^{-2.99}_{-0.87}$
Δ Perplexity	-4.73 $^{-5.89}_{-3.57}$	-5.07 $^{-6.23}_{-3.91}$	**-14.92** $^{**-16.43**}_{**-13.48**}$	-1.36 $^{-2.58}_{-0.15}$	-1.95 $^{-2.97}_{-0.92}$	**-5.26** $^{**-6.92**}_{**-3.57**}$	-3.29 $^{-4.55}_{-2.03}$	-3.32 $^{-4.38}_{-2.26}$	**-9.32** $^{**-10.92**}_{**-7.71**}$

We next ask whether these quantities can move beyond descriptive statistics by testing whether prompt interventions, designed to shift each proposed difficulty measurement in its predicted direction, also improve IFR. Table 1 reports the outcome of such interventions, detailed in Appendix C.4.

For each task and intervention, we report $\Delta\mathrm{IFR}$ relative to its original prompt, along with relative changes in all three difficulty metrics to expose cross-metric effects. Bold entries denote the targeted metric. All three interventions improve IFR across tasks. Averaged across tasks, slack, diversity, and perplexity increase IFR by 6.36, 6.70, and 6.62 percentage points, respectively. They also shift their targeted metrics in the intended direction by an average of 14.91%, 9.00%, and 9.83%, respectively.

Table 2: **Left:** Matched controls for targeted difficulty interventions. **Right:** Held-out evaluation of metric-guided intervention selection.

(Left)	ΔIFR vs. original (target / control)			Target-specific
Intervention	OpenEnded	MBPP	Scheduling	gain
Slack	8.20/3.90	4.85/2.10	5.35/2.65	$+3.25_{4.90}^{1.65}$
Diversity	7.25/3.15	3.75/1.95	7.15/3.85	$+3.07_{4.78}^{1.28}$
Accessibility	7.55/3.70	6.80/3.25	4.10/1.95	$+3.18_{4.82}^{1.76}$

(Right) Selection policy	OpenEnded	MBPP	Scheduling	Mean
Generic control	+1.28	+2.18	+1.44	+1.63
Random intervention	+3.15	+2.64	+3.02	+2.94
Metric-guided intervention	+6.35	+5.98	+5.42	+5.92
Guided − random	+3.20	+3.34	+2.40	$+2.98_{4.67}^{1.37}$

To separate metric-specific targeting from generic benefits of additional prompting, we introduce a matched control for each intervention in Table 2 (**Left**), reporting IFR changes relative to the original prompt and, in the final column, the paired targeted-minus-matched effect. These controls broadly match the form of each intervention while removing its metric-specific targeting, detailed in Appendix C.4. The matched controls themselves improve IFR by an average of 2.94 points across tasks, indicating that generic guidance explain some of the improvement observed. However, the targeted variants retain an average advantage of 3.17 points over their respective controls, suggesting that metric-specific targeting contribute additional predictive value beyond generic guidance.

We next test whether the measured difficulty profile can prospectively guide mitigation in Table 2 (**Right**). We use development-set difficulty profiles to assign each constraint the targeted intervention best matched to its dominant difficulty, and evaluate this fixed policy on held-out instances. Table 2 compares this metric-guided assignment with a generic IFR control and random assignment over the same interventions, reporting $\Delta\,\mathrm{IFR}$. Implementation details are provided in Appendix C.4. All strategies improve held-out IFR, but metric-based selection performs best. Mean gains are 1.63 points for the generic control, 2.94 for random assignment, and 5.92 for development-set-based selection, indicating here that difficulty measurements may yield prospective mitigation information.

3.3 Interaction Difficulty

Constraint difficulty in isolation need not predict joint difficulty: constraints may interfere or be redundant when combined. We therefore ask whether the properties above can characterise **interaction difficulty** between constraints. We focus on pairwise interactions as an interpretable first step, avoiding the combinatorial growth and confounding of higher-order effects. We first define two IFR-related quantities on per-condition $\mathcal{H} = (x_i, m, \omega)$ basis, Constituent Marginal Shift (CMS) and Joint Interaction Penalty (JIP):

$$\mathrm{CMS}^{\mathcal{H}}_{\{i,j\}} = \frac{1}{2}\Big[(\mathrm{IFR}^{\mathcal{H}}_{i|ij} - \mathrm{IFR}^{\mathcal{H}}_{i}) + (\mathrm{IFR}^{\mathcal{H}}_{j|ij} - \mathrm{IFR}^{\mathcal{H}}_{j})\Big], \quad \mathrm{JIP}^{\mathcal{H}}_{ij} = \mathrm{IFR}^{\mathcal{H}}_{i|ij} \cdot \mathrm{IFR}^{\mathcal{H}}_{j|ij} - \mathrm{IFR}^{\mathcal{H}}_{ij}\,.$$

The final CMS and JIP are averaged over $\mathcal{H}$, with JIP Bessel-corrected via R. CMS measures the change in each constraint's IFR when moving from its individual prompt to the joint prompt. JIP measures dependence between constraints as the difference between the product of individual $\mathrm{IFR}_{i,j}$ and joint IFR_{ij}, both evaluated on the joint prompt. We use both quantities to define three interaction regimes using fixed tolerances, with mixed cases left unclassified, detailed in Appendix E.6. **Overlap**: positive CMS with negative JIP, indicating reinforcement and greater co-satisfaction. **Additive**: both CMS and JIP lie within their respective tolerances, indicating little measurable interaction under these diagnostics. **Coupled**: negative CMS with positive JIP, indicating additional difficulty under joint prompting.

We next ask whether these interaction regimes are reflected in solution-space concentration. Let S_i and S_j denote the feasible generations observed when c_i and c_j are imposed separately, and S_{ij} those under both constraints jointly. Using the solution-space concentration defined above, let $U_t(S)$ denote the corresponding diversity of feasible solution modes. We define the *interaction ratio* as

$$IR_{ij} = \frac{2U_t(S_{ij})}{U_t(S_i) + U_t(S_j)}. \tag{5}$$

$\mathrm{IR}_{ij} \approx 1$ suggests preservation of average marginal solution-mode diversity, detailed in Appendix E.6, with values below/above 1 denoting contraction/expansion. This allows us to test whether IR varies systematically across CMS–JIP regimes.

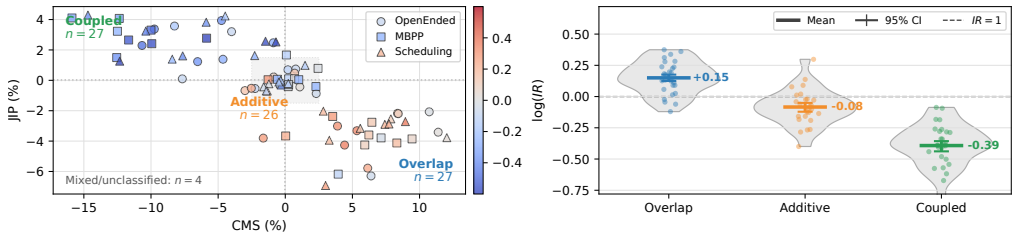


Figure 2: Constraint interaction regimes defined by CMS and JIP, compared to solution-space IR.

Figure 2 shows that IR tracks the ordering implied by the separately defined CMS–JIP regimes. In the left panel, overlapping pairs tend to exhibit positive $\log(\mathrm{IR})$, additive pairs cluster near $\log(\mathrm{IR}) = 0$, and coupled pairs have primarily negative $\log(\mathrm{IR})$. The right panel makes this separation clearer: mean $\log(\mathrm{IR})$ is approximately 0.15 for overlapping pairs, -0.08 for additive pairs, and -0.39 for coupled pairs. Although the distributions overlap, their central tendencies follow the predicted ordering. For these pairs, the agreement is not induced by the CMS–JIP regime definition, and thus provides evidence that solution concentration, although defined as an aspect of individual difficulty, also captures interaction difficulty here.

4 Instruction Capacity on Difficulty Stratified Pools

The maximum number of constraints that a model can satisfy is unlikely to be a function of count alone. Given our previous analysis, we next investigate how constraint difficulty properties relate to constraint capacity. We first define the individual difficulty score as $\mathcal{D}_t(c) = \frac{1}{3}[(1 - \tilde{s}_t(c)) + \tilde{q}_t(c) + \widetilde{\log} e_t(c)]$, where the difficulty terms are task-wise min–max normalised. Then, we provide a complementary view using IR interaction difficulty by sampling J distinct pre-identified subsets B_r, $|B_r| \in \{2\}$, allowing overlap but excluding duplicates. Let the constraint count be $n = \left|\cup_{r=1}^{J} B_r\right|$, with an additional singleton when required. Each B_r has a measured interaction type in $\{\text{overlap}, \text{additive}, \text{coupled}\}$, and subsets are sampled to match the targeted type. Let $n = 1$ be the singleton baseline, for which no pairwise interaction type is defined. Finally, constraints are grouped into difficulty terciles based on the above scores, reducing sensitivity to noise and control set construction, and enabling trend comparisons.

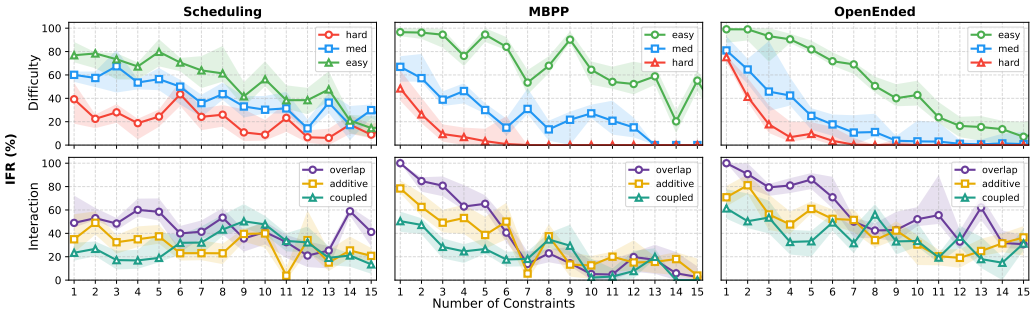


Figure 3: Constraint capacity plotted against difficulty and interaction.

Figure 3 shows how IF capacity varies with constraint count (1-15), stratified by individual (top) and interaction (bottom) difficulty. Across all tasks, IFR generally decreases as more constraints are imposed, but the rate of degradation differs substantially according to the difficulty of the constituent constraints. This separation is already visible at relatively small n. For example, at $n = 5$, SCHEDULING achieves approximately 80% IFR on easy constraints, compared with roughly 56% and 24% for medium and hard constraints. The corresponding differences are even larger for MBPP,

at approximately 94%, 30%, and 3%, and for OPENENDED, at approximately 82%, 25%, and 10%. Thus, sets containing the same number of constraints can differ by > 50 percentage points in follow rate across pools stratified by measured constituent difficulty.

The lower row provides a supporting view of the same capacity phenomenon through interaction type. IFR tends to decrease with constraint count for overlap, additive, and coupled sets alike. Overlapping constraints often retain somewhat higher IFR at smaller n, particularly in MBPP and OPENENDED, which is consistent with their greater degree of shared feasible behaviour. However, the separation between interaction regimes is substantially less systematic than the above row: the curves frequently cross, and their ordering changes with both task and n. In SCHEDULING, in particular, no single interaction regime dominates across the range. This noise may partly reflect the composition of pairwise interaction types into larger constraint sets, where the assigned type may not capture all interactions or higher-order effects.

The IFR at a given count varies substantially across individual-difficulty terciles, showing that count alone does not determine observed follow rate. Interaction-type differences are less systematic and should be treated as possible evidence of compositional effects. Overall, the results suggest a difficulty-aware view of instruction-following capacity. A statement such as "the model can follow N at most constraints" is incomplete unless the difficulty of those constraints is also specified.

5 DIFFICULTY BEYOND FEASIBILITY

The preceding sections have considered instruction difficulty within the feasible regime. However, when $F_t(x, C) = \varnothing$, infeasible constraints may still differ in severity. We thus characterise infeasible difficulty using an infeasibility gap (IG), defined jointly over paired constraints to measure how far conflicting constraints are from admitting a common solution. Larger amounts correspond to greater separation. As before, we construct infeasible instances at terciles to examine broad trends in the relationship between IG and model behaviour while reducing noise.

Table 3: Infeasibility recognition by difficulty and mitigation. Values are percentages.

Task	IG	Baseline			Indirect Reason			Direct Elicitation		
		Det.	Viol.	Corr.	Det.	Viol.	Corr.	Det.	Viol.	Corr.
SCHEDULING	Low	21.33	20.67	58.00	26.33	17.33	56.33	39.67	15.67	44.67
	Medium	23.33	16.34	60.33	26.33	15.33	58.33	40.33	11.33	48.33
	High	37.56	22.62	39.82	42.73	17.73	39.55	51.82	15.91	32.27
MBPP	Low	0.81	96.58	2.61	2.34	92.42	5.24	11.65	77.57	10.78
	Medium	1.86	95.32	2.82	2.57	92.23	5.20	14.74	75.12	10.14
	High	2.17	94.86	2.97	3.54	90.97	5.49	18.24	68.47	13.29
OPEN-ENDED	Low	0.67	94.18	5.15	1.12	93.49	5.39	8.13	74.93	16.94
	Medium	1.21	92.73	6.06	3.73	90.72	5.55	11.77	72.68	15.55
	High	1.21	92.12	6.67	2.53	91.66	5.81	18.28	69.76	11.96

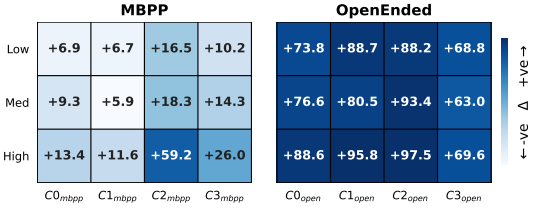


Figure 4: Infeasible Target-Count Attainment.

Table 3 summarises responses to infeasible constraint sets across tasks, difficulty levels, and mitigation conditions. Outcomes are classified as explicit detection, silent violations, endless correction loops. In SCHEDULING, detection increases with IG by 76.1% under the baseline prompt, 62.3% under indirect reasoning, and 30.6% under direct elicitation, indicating that more infeasible constraints are easier to identify. In contrast, the relationship is much weaker for MBPP and OPENENDED, where both baseline detection remains below 2.3%. One possible hypothesis is that both tasks do not naturally induce structured reasoning over constraints. Consistent with this explanation, with direct elicitation, detection increases with IG by 56.6% for MBPP and 124.8% for OPENENDED.

Recognition, however, captures only one response to infeasibility: models may still continue to pursue a requirement after encountering an infeasible constraint. We therefore also examine the Target-

Count Attainment (TCA), which measures attainment of the designated count target. The TCA is undefined for Scheduling. Figure 4 shows the TCA across IG levels and constraints for OPENENDED and MBPP: high-IG infeasibilities have TCA that are, on average, 10.02% and 173.45% higher than their low-IG counterparts, respectively. Thus, increasing IG also increases TCA. However, the TCA in MBPP is about $5\times$ lower on average compared to OPENENDED, likely due to conflicting constraints being less immediately apparent in structured tasks. Overall, the evaluated infeasible pairs show that IG difficulty is associated with differences in recognition and TCA, although the form and strength of this relationship depend on the task and prompting condition.

6 DIFFICULTY ON FUZZY NUMERIC CONSTRAINTS

Many instructions do not possess a single sharp constraint boundary, and are instead expressed through degree modifiers and adverbs. As a controlled extension toward fuzziness, we study whether the account of difficulty developed for conventional constraints extends to fuzzy settings for MBPP and OPENENDED, while still focusing only on countable output properties.

Table 4: Soft Constraint Boundary Spread and Difficulty.

	Standard Deviation (SD)			R^2		MBPP			OpenEnded		
pair_id	MBPP	OpenEnded	cr_id	MBPP	OpenEnded	$\bar{q}_f$	$\bar{s}_f$	$\bar{e}_f$	$\bar{q}_f$	$\bar{s}_f$	$\bar{e}_f$
P_0^0	$34.22^{30.58}_{38.93}$	$23.57^{23.20}_{29.82}$	C_{e}^0	$0.138^{0.116}_{0.153}$	$0.257^{0.238}_{0.288}$	$0.106^{0.100}_{0.124}$	$0.349^{0.343}_{0.398}$	$1.510^{1.208}_{1.540}$	$0.258^{0.223}_{0.270}$	$0.116^{0.115}_{0.139}$	$1.970^{1.909}_{2.250}$
P_1^0	$5.39^{4.33}_{5.55}$	$45.17^{38.92}_{48.33}$	C_{m}^0	$0.070^{0.065}_{0.084}$	$0.200^{0.171}_{0.208}$	$0.234^{0.215}_{0.285}$	$0.328^{0.317}_{0.381}$	$1.735^{1.514}_{2.020}$	$0.314^{0.248}_{0.339}$	$0.118^{0.102}_{0.135}$	$2.060^{2.027}_{2.410}$
P_2^0	$43.71^{37.78}_{47.40}$	$37.42^{36.37}_{46.20}$	C_{h}^0	$0.058^{0.055}_{0.064}$	$0.177^{0.158}_{0.203}$	$0.383^{0.287}_{0.389}$	$0.264^{0.207}_{0.276}$	$2.168^{1.737}_{2.198}$	$0.456^{0.374}_{0.485}$	$0.065^{0.057}_{0.069}$	$2.219^{1.972}_{2.420}$
P_0^1	$7.60^{5.62}_{7.85}$	$13.11^{11.82}_{14.63}$	C_{e}^1	$0.353^{0.339}_{0.402}$	$0.177^{0.134}_{0.185}$	$0.025^{0.021}_{0.027}$	$0.819^{0.639}_{0.880}$	$1.402^{1.269}_{1.492}$	$0.143^{0.132}_{0.154}$	$0.856^{0.720}_{0.872}$	$1.902^{1.799}_{2.253}$
P_1^1	$16.10^{15.42}_{17.89}$	$45.06^{38.74}_{45.77}$	C_{m}^1	$0.272^{0.233}_{0.288}$	$0.216^{0.207}_{0.259}$	$0.167^{0.154}_{0.201}$	$0.628^{0.500}_{0.655}$	$1.433^{1.402}_{1.760}$	$0.208^{0.186}_{0.224}$	$0.505^{0.413}_{0.524}$	$1.932^{1.852}_{2.252}$
P_2^1	$12.33^{10.63}_{14.01}$	$34.48^{30.75}_{38.71}$	C_{h}^1	$0.236^{0.198}_{0.245}$	$0.150^{0.125}_{0.164}$	$0.250^{0.217}_{0.257}$	$0.351^{0.315}_{0.375}$	$1.490^{1.289}_{1.715}$	$0.080^{0.064}_{0.086}$	$0.529^{0.433}_{0.547}$	$1.982^{1.905}_{2.419}$
P_0^2	$18.89^{16.77}_{21.75}$	$48.06^{43.46}_{54.54}$	C_{e}^2	$0.189^{0.177}_{0.228}$	$0.356^{0.311}_{0.405}$	$0.022^{0.021}_{0.024}$	$0.729^{0.694}_{0.882}$	$1.420^{1.333}_{1.758}$	$0.118^{0.105}_{0.137}$	$0.358^{0.337}_{0.422}$	$1.909^{1.636}_{1.942}$
P_1^2	$40.37^{33.61}_{41.56}$	$10.86^{9.57}_{12.82}$	C_{m}^2	$0.202^{0.174}_{0.211}$	$0.392^{0.355}_{0.443}$	$0.062^{0.050}_{0.067}$	$0.706^{0.644}_{0.821}$	$1.480^{1.318}_{1.603}$	$0.070^{0.060}_{0.079}$	$0.361^{0.263}_{0.370}$	$2.024^{1.680}_{2.080}$
P_2^2	$12.44^{11.95}_{14.26}$	$3.42^{3.27}_{3.86}$	C_{h}^2	$0.176^{0.161}_{0.190}$	$0.543^{0.442}_{0.579}$	$0.171^{0.136}_{0.176}$	$0.398^{0.391}_{0.495}$	$1.527^{1.471}_{1.807}$	$0.036^{0.028}_{0.037}$	$0.176^{0.170}_{0.209}$	$1.904^{1.791}_{2.140}$

Table 4 first tests whether these constraints exhibit the variability expected of fuzzy boundaries. Its **left** block compares output-property distributions under fuzzy and conventional constraint pairs. SD increases in all 18 comparisons by 3.42%–48.06%. The **middle** block further stratifies fuzzy outputs by semantically similar task instances. These strata explain, on average, 18.82% of the variance in MBPP and 27.42% in OPENENDED. The **right** block tests whether the individual difficulty properties vary as expected with modifier intensity: pooled across tasks-constraints, $\bar{q}_f$ increases by 104.70%, $\bar{s}_f$ decreases by 44.79%, and $\bar{e}_f$ increases by 11.65%. The small change in $\bar{e}_f$ likely reflects the flexibility of fuzzy constraints in accommodating rare generations. Overall, these fuzzy constraints show broadly compatible trends versus conventional counterparts, with greater dispersion and context dependence.

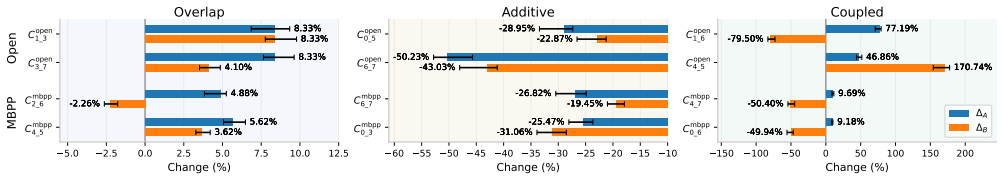


Figure 5: Soft constraint interaction regimes using constituent-level satisfaction changes.

we next test whether fuzzy constraint pairs retain the three interaction regimes identified in Section 3.3, reporting the output-properties of these pairs in Figure 5. Let $\mu_k(C)$ denote the measured value of the output property of c_k. For a pair $\{c_i, c_j\}$, we define the relative constituent-level change as $\Delta_k = 100[\mu_k(\{c_i, c_j\}) - \mu_k(\{c_k\})]/\mu_k(\{c_k\})$, where $k \in \{i, j\}$ and the blue and orange lollipops report Δ_i and Δ_j. Overlapping pairs show small changes (-2.26%-8.33%) and low intra-pair discrepancies (0-7.14 points). Additive pairs show larger, roughly symmetric changes

(−50.23%–−19.45%; 5.59–7.37-point discrepancies), whereas coupled pairs are strongly asymmetric, often shifting in opposite directions (59.12–156.69-point discrepancies). Overall, these fuzzy pairs provide qualitative evidence that the hard-constraint interaction taxonomy remains informative here, with graded and directional shifts.

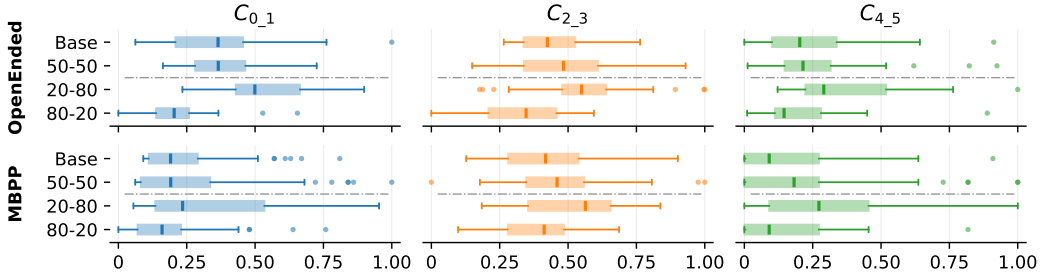


Figure 6: Infeasible Behaviour in Soft Constraints.

Finally, Figure 6 considers the fuzzy analogue of infeasibility: paired constraints that push the same output property in opposite directions. Because soft constraints can be satisfied to intermediate degrees, exact infeasibility is not appropriately defined. BASE specifies no priority, while the other conditions serve as controls by making priorities explicit in the prompt. The 50:50 control provides an equal-priority reference, while the asymmetric conditions serve as allocation-shift controls. Across tasks and constraints, the baseline and 50:50 distributions generally have similar medians and overlapping interquartile ranges, while asymmetric controls confirm that allocation steers in the expected directions, but is neither perfectly proportional nor uniform across contexts. Models appear to treat these infeasible fuzzy constraints as approximately equally important by producing a compromise, paralleling their tendency to seek plausible partial solutions under conventional infeasibility. Taken together, these results provide a controlled extension of difficulty, suggesting that the difficulty framework remains informative under this form of fuzziness.

7 DISCUSSION

Difficulty-Aware We show that a common constrained-satisfaction framing exposes recurring, predictive dimensions of instruction difficulty across structurally graded output spaces, even though their empirical measurements are task-adapted. This supports a difficulty-aware view of instruction following: raw constraint count alone is likely insufficient to characterise capacity, since equally sized instruction sets can differ substantially in follow rate. Within this scope, difficulty-aware evaluations could improve benchmark design, failure analysis, and mitigation by distinguishing constraints that are permissive, restrictive, low-probability, or strongly interacting.

Limitations Our analyses are limited to the models, tasks, and constraints studied and may not generalise to multi-turn settings. Unlike our atomic, verifiable constraints, real instructions may be implicit, hierarchical, or distributed across turns. The proposed measures and taxonomies depend on finite sampling and may be sensitive to prompting, decoding, and model-based judgments. Moreover, soft-constraint satisfaction relies partly on model-based judgments. Future work should explore settings where constraints emerge and change throughout long-horizon workflows.

Conclusion We introduced a model-conditioned, constrained-satisfaction framework for studying instruction difficulty across three structurally graded tasks. Within the evaluated setting, boundary slack, solution concentration, and conditional accessibility consistently track IFR, while targeted interventions improve compliance. Equally sized constraint sets can exhibit substantially different follow rates when stratified by measured difficulty, while pairwise interactions reveal coupling and overlap regimes. Extensions to infeasible and fuzzy domains further show where difficulty-aware characterisations remain informative. Together, these results provide evidence that difficulty-aware measurements capture systematic variation in instruction following that aggregate compliance and raw constraint counts can obscure, motivating broader evaluation of such measurements for benchmark design, failure analysis, and mitigation.

AI USE STATEMENT

In this work, we used generative AI tools for model-judge evaluations for fuzzy slack, coding purpose-built functions for specific experimental needs, typographical checks, and writing refinements. We have not used generative AI tools for ideation, background research, direction, problem formulation, code architecture, and hypothesis. Mathematical proofs and transcribing aids are not applicable to this work. All AI-assisted outputs were manually reviewed. Code functions underwent adversarial and unit testing, especially for critical components. Model-judge evaluations were sampled and sanity-checked, while typographical and writing refinements were compared against the original and manually revised. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies instruction-following behaviour and does not involve human participants, private information, or personally identifiable data. All data instances are either synthetic or publicly available. We use existing datasets and models subject to their applicable licenses and attribution requirements. The evaluated constraints concern benign and mechanically measurable output properties. Nevertheless, a better understanding of instruction following has dual-use implications: it may support more reliable assistants and improved recognition of conflicting requirements, but it could also be used to strengthen model compliance in undesirable applications. The experiments required repeated model inference, incurring environmental costs. Thus, we limited these costs by evaluating a bounded model suite and reusing fixed task instances where possible.

REPRODUCIBILITY STATEMENT

We provide the experimental details needed to reproduce our analyses throughout the main text and appendices. Appendix A describes the construction and selection of the Scheduling, MBPP, and OpenEnded task datasets, including the fixed base instances reused across experimental conditions. Appendix B specifies the constraint families, numerical instantiations, verifier implementations, and experimental constraint pools used for the individual-difficulty, interaction, capacity, infeasibility, and soft-constraint analyses. Appendix C documents the evaluated model suite, prompt templates, chat configuration, and sampling and decoding procedure. The corresponding source code is made available as supplementary material.

REFERENCES

Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, et al. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Wayne Chi, Valerie Chen, Ryan Shar, Aditya Mittal, Jenny Liang, Wei-Lin Chiang, Anastasios Nikolas Angelopoulos, Ion Stoica, Graham Neubig, Ameet Talwalkar, and Chris Donahue. EDIT-bench: Evaluating LLM abilities to perform real-world instructed code edits. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=FtL9eEmU6v.

Xiaomin Li, Zhou Yu, Zhiwei Zhang, Xupeng Chen, Ziji Zhang, Yingying Zhuang, Narayanan Sadagopan, and Anurag Beniwal. When thinking fails: The pitfalls of reasoning for instruction-following in LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=w5uUvxp81b.

Michael Xieyang Liu, Frederick Liu, Alexander J. Fiannaca, Terry Koo, Lucas Dixon, Michael Terry, and Carrie J. Cai. "we need structured output": Towards user-centered constraints on large language model output. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, CHI EA '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703317. doi: 10.1145/3613905.3650756. URL https://doi.org/10.1145/3613905.3650756.

Yuetian Mao, Junjie He, and Chunyang Chen. From prompts to templates: A systematic prompt template analysis for real-world llmapps. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, FSE Companion '25, pp. 75–86, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712760. doi: 10.1145/3696630.3728533. URL https://doi.org/10.1145/3696630.3728533.

Meta AI. Llama 3.1 8b. Hugging Face model repository, 2024. URL https://huggingface.co/meta-llama/Llama-3.1-8B. meta-llama/Llama-3.1-8B.

Mistral AI. Mistral small 3.2. Mistral AI API, June 2025. URL https://docs.mistral.ai/models/mistral-small-3-2-25-06. Model version 25.06; API model identifier: mistral-small-2506.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=yfYgwjj5F8.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Amy Xin, Youfeng Liu, Bin Xu, Lei Hou, and Juanzi Li. AGENTIF: Benchmarking large language models instruction following ability in agentic scenarios. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=FLiMxTkIeu.

Yulei Qin, Gang Li, Zongyi Li, Zihan Xu, Yuchen Shi, Zhekai Lin, Xiao Cui, Ke Li, and Xing Sun. Incentivizing reasoning for advanced instruction-following of large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=vRVfgcoeIl.

Guillem Ramírez, Alexandra Birch, and Ivan Titov. Controlling what you share: Assessing language model adherence to privacy preferences. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 40024–40049, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1990. URL https://aclanthology.org/2026.findings-acl.1990/.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. Benchmarking complex instruction-following with multiple constraints composition. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Wei Zhang, Zhenhong Zhou, Kun Wang, Junfeng Fang, Rongwu Xu, Yuanhe Zhang, Rui Wang, Ge Zhang, Xinfeng Li, Li Sun, Lingjuan Lyu, Yang Liu, and Sen Su. LIFEBENCH: Evaluating length instruction following in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=dndqkXKY6W.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Tao Zou, Xinghua Zhang, Haiyang Yu, Minzheng Wang, Fei Huang, and Yongbin Li. EIFBENCH: Extremely complex instruction following benchmark for large language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20930–20953,

Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1059. URL https://aclanthology.org/2025.emnlp-main.1059/.

A Task and Datasets Construction

We construct three base task datasets spanning different levels of output structure: Scheduling, MBPP, and OpenEnded. Each dataset contains 20 task instances, for a total of 60 base instances. The same underlying task instances are reused throughout our experiments to hold task content fixed. Experimental conditions may alter the prompt realization, add or modify constraints, or otherwise change the instruction-following setting; these transformations are described separately in the corresponding appendices. Here, we describe only the construction of the underlying base tasks.

When comparing constraint difficulty, constraint interactions, apparent instruction-following capacity, infeasible requirements, or soft instructions, we reuse these underlying task instances rather than constructing a new task dataset for each experiment.

This distinction is important. Reusing the base tasks does not imply that every experimental prompt is identical. Prompt templates, explicit instructions, constraint sets, and intervention-specific context may change according to the experimental condition. Those manipulations constitute the instruction-following treatment rather than the task itself. Holding the base instances fixed reduces task-content variation across conditions and makes comparisons more directly attributable to changes in the imposed constraints and prompting conditions.

The task instances and corresponding construction code will be made available in the accompanying repository. Dataset redistribution and reuse are subject to the applicable inherited licensing conditions of their upstream sources.

A.1 Scheduling

Scheduling consists of synthetically generated linear feasibility problems expressed in natural language. Each instance asks the model to return one assignment to a set of decision variables that satisfies all stated variable bounds and linear requirements. There is no optimization objective in the base task: the objective is simply to identify a feasible point. We generate two-dimensional linear programs of the form $\mathbf{u} \in \mathbb{R}^2$, subject to variable bounds and a small collection of linear inequalities. For each instance, both variables have lower bound zero. Their upper bounds are sampled independently as integers between 0 and 18. Each problem can contain several inequality or equality linear constraints.

This construction yields small but non-trivial two-variable feasibility regions whose boundaries and exact feasibility can be evaluated directly using linear programming. The low dimensionality is intentional: it makes the underlying geometry interpretable while still permitting substantial variation in the position, orientation, and intersection of constraint boundaries.

The numerical problem is separated from its natural-language presentation. An LP instance can be rendered through several domain templates while retaining the same underlying mathematical form. We use three templates (factory production, staffing allocation, and transportation planning) to alter the semantic interpretation of the variables and requirements without changing the basic feasibility task.

The narration explicitly introduces the decision variables, states their bounds, lists the requirements in natural language, and asks the model to return one feasible assignment in terms of the decision variables. Variable names and descriptions are configurable, allowing instances to use semantically meaningful labels rather than generic mathematical symbols.

A representative Scheduling instance is:

```
A factory must decide how much of each product to make.

Decision variables:
  - Let A denote the number of chairs to produce.
  - Let B denote the number of tables to produce.

Variable bounds:
  - A must be between 0 and 3.
  - B must be between 0 and 9.

Requirements:
  - contract_requirement: 2 A + 3 B must be at least 7.

Find one production plan in the form of decision variables that
satisfies all requirements and bounds.
```

The Scheduling task therefore provides the most structured setting in our evaluation. Unlike the other two tasks, candidate outputs can be mapped directly to decision-variable assignments and checked against an explicitly represented feasible region.

A.2 MBPP

For the code-generation setting, we use a subset of the sanitized test split of the Mostly Basic Python Problems (MBPP) benchmark. We conduct a pilot screening before fixing the final task set at 20 problems. The screening is performed manually. We retain tasks that satisfy the following properties:

1. The specification is not unusually difficult to be solved.
2. The problem does not depend on unusual input/output conventions, external interaction, or other incidental execution requirements.
3. The specification is not ambiguous.
4. The problem admits multiple plausible implementations rather than strongly prescribing a single surface-form solution.

The final criterion is particularly useful for our setting because our analyses concern the distribution of valid model outputs rather than reproduction of a particular reference implementation. We therefore prefer tasks whose semantic requirements determine correctness while still leaving meaningful freedom in implementation strategy and code form. The resulting 20 problems were human verified before use in the main experiments.

The base MBPP prompt contains only the problem description and required function interface, together with ordinary output-formatting instructions. In particular, generated implementations are required to appear inside Python Markdown code fences. We do not provide few-shot demonstrations in the base task. Demonstrations are introduced only in experiments where they form an explicit intervention or experimental condition.

A representative MBPP instance is:

```
Write a function to find the shared elements from the given two lists.

- Implementation must be fenced in its own ` ```python ` and ` ``` ` Markdown fences.
- Implementations must follow the input signature:
`similar_elements((3, 4, 5, 6),(5, 7, 4, 10))`
`similar_elements((1, 2, 3, 4),(5, 4, 3, 7))`
`similar_elements((11, 12, 14, 13),(17, 15, 14, 13))`
```

This task occupies an intermediate position in our task spectrum. The space of possible programs is substantially broader than the explicitly parameterised Scheduling space, but Python syntax and the required function behaviour still impose substantial structure on valid outputs.

A.3 OpenEnded

OpenEnded consists of 20 open-ended natural-language questions. Their style and coverage are loosely inspired by a mixture of established open-ended question-answering benchmarks, after which new questions are written and curated for this study. We use human filtering to retain prompts that genuinely support open-ended responses. In particular, selection prioritises:

Open-endedness. A question should permit multiple substantively valid answers, organizations, and phrasings, rather than reduce to a single short factual span.

Answerability. The question should contain enough information to admit a useful response without requiring unavailable private context.

Topic diversity. The final set should cover varied everyday, explanatory, conceptual, and practical subject matter, rather than concentrate on a narrow semantic domain.

Response diversity. Different valid answers should be able to emphasize different considerations or organize the explanation in different ways while still addressing the task.

Clarity. Prompts with material ambiguity about what is being asked are excluded.

These criteria are intended to create a comparatively unconstrained output space while maintaining an identifiable base task. They also reduce the chance that measured output diversity is driven primarily by malformed or underspecified questions.

A representative OpenEnded instance is:

```
Explain what makes a team successful?
```

Unlike Scheduling and MBPP, such prompts impose relatively little intrinsic surface-form structure. A valid response may vary substantially in length, organisation, examples, vocabulary, and substantive emphasis. OpenEnded therefore serves as the least structured end of our task spectrum.

B Constraints and Verifiers Details

This appendix describes the construction and verification of the constraints used throughout the experiments. The three tasks differ substantially in the structure of their output spaces, and we therefore implement task-specific verifiers while retaining a common design principle: whenever possible, an instruction is represented by an atomic, mechanically measurable property of the generated output. This allows a verifier to be evaluated without relying on a general-purpose model-based judge for the hard-constraint experiments. We first describe the constraint-design criteria and task-specific verifier implementations, then give the concrete constraint inventories used for the individual difficulty, interaction, capacity, infeasibility, and soft-constraint analyses.

B.1 Constraint design principles

Constraints were selected by balancing six considerations.

Deterministically Verifiable. A constraint must admit to an unambiguous operational rule. This is stronger than merely being verifiable. For example, avoiding the use of conjunctions in English is verifiable, but the set of conjunctions is not ambiguous. In contrast, an instruction such as "use less than 250 words" maps directly to a verifiable and fixed rule.

1. **Atomic.** A constraint should correspond to one separately measurable output property. Constraints are avoided when they can be decomposed into constituent requirements. Atomicity is important because the analysis assign difficulty, interaction, and satisfaction measurements to individual constraints.
2. **Tractable.** Verification must remain computationally manageable relative to generation. We therefore favour properties that can be evaluated by arithmetic, regular expressions, tokenization, or syntax-tree inspection, rather than constraints requiring expensive semantic search or another unrestricted generation procedure.

3. **Relevant.** Constraints are chosen from properties naturally expressed in the output space of the corresponding task. Python coding, for example, admits a very large number of constraints, but only a small subset are useful for studying code-generation in MBPP. The same principle applies for OpenEnded and Scheduling constraints.

4. **Practical.** The base constraint templates are selected to resemble requirements that can plausibly occur in real-world use, such as limiting answer length, restricting code structure, requesting formatting, or specifying numerical bounds. Some experiments deliberately instantiate these templates at extreme values in order to vary difficulty or induce infeasibility. Those extreme instantiations should not be interpreted as claiming that every numerical threshold is itself representative of ordinary user requests.

5. **Numerically Graded.** A constraint should be associated with an underlying numeric measurement. This permits multiple boundary instantiations of the same semantic requirement and allows the margin to the constraint boundary to be measured. For example, the template "Keep below _ lines of code" can be instantiated at several values while leaving the underlying output property unchanged. This property is particularly useful for analysis where gradations are required.

B.2 Task Correctness and Constraint Satisfaction

For all feasible experiments, only task-correct outputs are accepted before verifying constraints. For MBPP, base-task correctness is defined by whether the solution passes all unit tests supplied for the problem. Scheduling is a special case where the base-task correctness coincides with returning a feasible decision-variable assignment. For OpenEnded, we first apply deterministic validity checks to exclude degenerate outputs: responses must be non-null and non-empty, contain at least four alphabetic word tokens, and have at least 25% alphabetic characters. For responses with eight or more tokens, no lower-cased token may account for more than 80% of all tokens. A binary, independent judge, "mistral-large-3", then determines whether the response meaningfully and coherently addresses the underlying question, regardless of constraints, preventing malformed outputs from distorting the estimated constraint geometry. For infeasible experiments, explicit contradiction detections are evaluated directly as recognition outcomes and are not excluded for failing to produce the ordinary task output.

Constraint verification is kept conceptually separate from correctness on the underlying base task. This distinction prevents task failure from being silently interpreted as failure of every explicit instruction. Conditioning on task correctness is therefore necessary to isolate the instruction-following phenomenon studied here; otherwise, failures of the base task would be conflated with failures of the imposed constraint.

Experimental Implications Because IFR is defined conditional on base-task correctness, changes in conditional compliance could in principle coincide with changes in the probability of solving the underlying task. We therefore additionally examine whether the principal experimental manipulations materially alter the task-correctness rate (TCR). For each intervention family ι, we compute $\Delta\mathrm{TCR}_{t,\iota} = 100\,(\mathrm{TCR}_{t,\iota} - \mathrm{TCR}_{t,0})$, where 0 denotes the original unmodified prompt. For numerical constraint severity, we analogously compare the strictest and loosest instantiations within each constraint family, holding the underlying task and constraint type fixed.

Uncertainty is estimated with the same hierarchical bootstrap used for the main IFR analyses. Unlike the difficulty measurements, TCR is computed from all generated records and does not use successful-output rarefaction. To distinguish a small effect from a merely non-significant one, we use a practical-equivalence region of ± 2.5 percentage points, matching the rate-scale tolerance used elsewhere in the paper. We regard a TCR contrast as practically small when its 95% bootstrap interval lies wholly inside this region.

Table 5 shows that the evaluated prompt interventions produce little corresponding movement in base-task correctness. Across the nine task-specific intervention contrasts, point estimates range from -0.43 to $+0.29$ percentage points, and every 95% interval remains within the ± 2.5-point practical-equivalence region. Cross-task mean changes are similarly close to zero: $+0.07$, -0.09, and -0.05 points for the slack, diversity, and accessibility interventions, respectively.

Table 5: Task-correctness stability under intervention and constraint-severity changes. Entries are changes in TCR relative to the corresponding reference. For the first three rows, the reference is the original prompt; for the final row, it is the loosest numerical instantiation of the same constraint.

Manipulation	OpenEnded	MBPP	Scheduling	Mean
Slack intervention	$+0.23\ ^{-0.99}_{+1.45}$	$-0.16\ ^{-1.18}_{+0.86}$	$+0.13\ ^{-1.21}_{+1.47}$	$+0.07\ ^{-0.83}_{+0.97}$
Diversity intervention	$-0.35\ ^{-1.46}_{+0.76}$	$+0.29\ ^{-0.68}_{+1.26}$	$-0.22\ ^{-1.29}_{+0.85}$	$-0.09\ ^{-0.79}_{+0.60}$
Accessibility intervention	$+0.01\ ^{-1.42}_{+1.44}$	$-0.43\ ^{-1.42}_{+0.55}$	$+0.27\ ^{-1.13}_{+1.68}$	$-0.05\ ^{-0.89}_{+0.79}$
Strictest vs. loosest constraint	$-0.54\ ^{-1.77}_{+0.69}$	$-0.34\ ^{-1.55}_{+0.87}$	$-0.17\ ^{-1.14}_{+0.81}$	$-0.35\ ^{-1.02}_{+0.32}$

Increasing numerical constraint severity yields the same qualitative result. Moving from the loosest to the strictest boundary produces a small negative TCR change in all three tasks, with a cross-task mean of -0.35 percentage points (95% CI $[-1.02, 0.32]$). Thus, within the evaluated manipulations, the substantially larger changes observed in conditional IFR are not accompanied by comparably large changes in the probability of solving the underlying task. This indicates that post-treatment selection through task correctness is small on the scale of the intervention and severity manipulations studied here.

B.3 Task-specific verifier implementations

Scheduling instances request a numerical assignment to named decision variables. The prompt instructs the model to return its solution in a structured format, extracted using regex. Once an assignment $\hat{\mathbf{u}}$ has been extracted, each mathematical requirement is evaluated directly. For a linear inequality $a^\top \mathbf{u} \leq b$, the numerical verifier accepts $\hat{\mathbf{u}}$ when $a^\top \hat{\mathbf{u}} \leq b + \epsilon$, and for $a^\top \mathbf{u} \geq b$ it accepts when $a^\top \hat{\mathbf{u}} \geq b - \epsilon$. Variable bounds are evaluated in the same manner.

MBPP outputs are Python programs. The base prompt requires the implementation to appear in a Python Markdown code fence, after which the implementation is extracted and inspected using Python's (`ast`) and `tokenize` modules. Both modules are used over regex because they understand Python's actual syntax and structure, reliably distinguishing code from strings, comments, identifiers, and expressions, making verification much more robust.

OpenEnded responses are evaluated with a combination of regular expressions and NLTK tokenizers. NLTK tokenization supplies word- and sentence-level segmentation for constraints defined over words or sentences. Regex is used for lexical properties such as punctuation counts and explicit marker phrases. For a hard count constraint, satisfaction follows the comparison expressed in the instruction. Thus, "less than" and "below" induce strict upper bounds, "more than" induces a strict lower bound, "at least" includes the boundary, and interval constraints test both stated endpoints.

Verifiers are tested during pilot experiments before the main generation sweeps. Unit tests were written for representative examples, together with adversarial cases. These pilot tests were used to validate the verification logic before fixing the experimental constraint sets.

B.4 Individual-Difficulty Constraints

For MBPP and OpenEnded, the numbers inside braces below denote distinct numerical instantiations of the same constraint template. The MBPP individual-difficulty constraints are:

```
Keep below {30,14,10,6,3} lines of code.
Keep below {81,60,48,29,17} characters per line.
Keep below {7,6,5,4,3} levels of indentation.
Keep below {7,6,5,4,3} assignment operators (the "=" operator).
Keep below {7,6,5,4,3} tuples, lists, sets, dicts in totality.
Keep below {5,4,3,2,1} for/while loops in totality.
```

Each brace-delimited set therefore represents five constraints sharing the same measurement but placing the boundary at different locations. The OpenEnded individual-difficulty constraints are:

```
Keep below {20, 15, 10, 7, 4} fullstops.
Keep below {20, 15, 10, 7, 4} commas.
Keep below {800, 600, 400, 250, 80} words.
Keep below {20, 15, 10, 7, 4} exclamation marks.
Keep below {20, 15, 10, 7, 4} question marks.
Keep below {20, 15, 10, 7, 4} semicolons.
```

For Scheduling, the output property is already a linear function of the decision variables, so difficulty is varied by changing the right-hand side of an explicitly represented linear constraint. The constraint families and five boundary instantiations are shown in Table 6. The percentage labels index the five nominal boundary settings used in the experiment; the numerical values below are the actual verifier boundaries.

LHS	Sense	100%	80%	60%	40%	25%
$u_1 + 2u_2$	$\leq$	40	32	24	16	10
$2u_1 + u_2$	$\leq$	40	32	24	16	10
$2u_1 + 3u_2$	$\leq$	60	48	36	24	15
$3u_1 + 2u_2$	$\leq$	60	48	36	24	15
$u_1 + 3u_2$	$\leq$	50	40	30	20	12.5
$3u_1 + u_2$	$\leq$	50	40	30	20	12.5

Table 6: Scheduling constraint boundaries used for the individual-constraint analysis.

B.5 Pairwise-interaction constituent constraints

For MBPP and OpenEnded we begin with fifteen constituent constraints and form the required pairs with distinct constraints. The interaction analysis is performed over pre-identified pairs to ensure tractability, with constituents listed in Table 7.

Table 7: Constraint details for Interaction Experiments

Index	OpenEnded	MBPP
0	Keep below 9 fullstops.	Keep below 10 lines of code.
1	Keep below 9 commas.	Keep below 4 levels of indentation.
2	Keep below 350 words.	Keep below 67 characters for each line of code.
3	Keep below 5 exclaimation marks.	Keep below 5 assignment operators (the "=" operator).
4	Keep below 5 question marks.	Keep below 6 tuples, lists, sets, dicts in totality.
5	Keep below 5 semicolons.	Keep below 4 loops ("for" and "while" together).
6	Keep below 9 sentences.	Keep below 4 "if" statements.
7	Use the word "and" less than 9 times.	Keep below 4 "and" and "or" statements together.
8	Keep below 3 paragraphs.	Keep below 4 lines of comments.
9	Keep below 4 bullet points (in the form of a +).	Keep below 3 list/set/dict comprehension.
10	Keep below 7 apostrophes (').	Keep below 2 try clause with 1 excepts.
11	Keep below 5 pairs of parentheses.	Keep below 2 lambdas.
12	Keep below 4 en dashes (—).	Keep below 3 return statement(s).
13	Keep below 4 pairs of double quotation marks (" ").	Keep below 3 function defs (func def).
14	Keep below 5 bolded words (using **word**).	Keep below 3 magic constants (a number or str).

For Scheduling, the geometric relation between two linear constraints can be controlled directly. To make the directional relation explicit, write every inequality in the canonical form $\tilde{a}^\top \mathbf{u} \geq \tilde{b}$. An original constraint $a^\top \mathbf{u} \geq b$ is unchanged, while a constraint $a^\top \mathbf{u} \leq b$ is multiplied by -1. For two canonicalized constraints, define the oriented normal alignment

$$\kappa(a_1, a_2) = \frac{\tilde{a}_1^\top \tilde{a}_2}{\|\tilde{a}_1\|_2 \|\tilde{a}_2\|_2}.$$

Values near $+1$ indicate similarly oriented half-spaces, values near zero indicate approximately orthogonal requirements, and values near -1 indicate oppositely oriented half-spaces.

To obtain balanced coverage of the geometric regimes considered in the main text, we categorise pairs according to their oriented normal alignment: $\kappa \geq 0.5$, $-0.5 < \kappa < 0.5$, and $\kappa \leq -0.5$. These strata are used only for internal pair selection and balancing. Based on κ the final design contains three approximately equal sets. Within this stratification, pairs are selected to preserve broad coverage of the available alignment values. Different pairs may share the same alignment because κ depends only on the relative orientations of the canonicalized constraint normals.

B.6 Constraint Pools Capacity experiment

MBPP and OpenEnded each use separate easy, medium, and hard pools containing exactly 15 available constraints. Crucially, these pools were assigned solely from the previously defined quantitative difficulty score, rather than by "pre-expected" difficulty. Constraint sets are sampled using Python's `random` library with a fixed seed. For a target set size $n_{\text{inst}} \in \{1, \ldots, 15\}$, the procedure samples $16 - n_{\text{inst}}$ distinct sets from the $\binom{15}{n_{\text{inst}}}$ possible combinations. Each sampled combination receives a unique identifier. The decreasing sampling schedule gives wider coverage at small set sizes, where many qualitatively different combinations are available, while avoiding a large combinatorial expansion at high set sizes. The underlying pool construction itself is defined over the 15 available constraints. Some constraints may appear compound, such as those grouping lists and tuples, or `for` and `while` loops. These are kept at this level because they represent closely related constructs and are already as atomic as practicable for the evaluation setting. Further decomposition would produce overly narrow constraints.

B.7 Hard infeasible constraints

Each entry introduces jointly infeasible requirements at easy, medium, and hard severity levels. IDs prefixed by `E`, `M`, and `H` denote the easy, medium, and hard groups, respectively. In one constraint, each constituent clause may be straightforward to verify locally, while their simultaneous requirements can make the full set impossible.

B.7.1 Scheduling Infeasibility Construction

Scheduling permits infeasibility to be planted analytically rather than through a fixed list of surface-form constraints. A compromise point $\mathbf{u}^\star$ is first sampled, together with a random unit direction a, where $\|a\|_2 = 1$. Let $\xi = a^\top \mathbf{u}^\star$. For a target infeasibility level $\zeta > 0$, the generator adds the contradictory pair $a^\top \mathbf{u} \le \xi - \zeta$ and $a^\top \mathbf{u} \ge \xi + \zeta$. No $\mathbf{u}$ can satisfy both constraints. Since a has unit norm, the perpendicular separation of the two parallel hyperplanes is exactly 2ζ. The experimental records therefore store `target_infeasibility` $= s$ and `contradiction_gap` $= 2\zeta$. Moreover, the minimum possible worst-case violation of the contradictory pair is exactly ζ, attained at the central hyperplane $a^\top \mathbf{u} = \xi$. Additional support constraints are constructed to be satisfied by $\mathbf{u}^\star$ and therefore do not increase the planted infeasibility. The empirical severity distributions used in the experiments are summarised in Table 10:

As implied by construction, the observed contradiction gap is approximately twice the target infeasibility at every severity level. The sampled compromise points remain broadly centered near the origin, with mean locations

$$(-0.03249, -0.2519), \qquad (-0.406, 0.1135), \qquad (-0.4315, -0.3887)$$

for easy, medium, and hard instances respectively, and coordinate-wise standard deviations of approximately three. The sampled unit directions are likewise approximately centred around zero. Consequently, severity is driven principally by the planted separation 2ζ, rather than a systematic change in the location or orientation of the contradictory constraints.

B.8 Infeasible constraints used for constituent target-count attainment

We also study how strongly models continue to pursue individual constituent requirements after proceeding with an infeasible prompt, listed in Table 11. These constraints differ from the infeasible families used for the recognition analysis. Scheduling is not included in the constituent target-count attainment experiment. The Constraints are grouped into E, M, H terciles to indicate the value of the Infeasibility Gap (IG).

B.9 Fuzzy Constraints

Fuzzy constraints retain numeric and countable output properties but replaces fixed boundaries with degree-modified natural-language requirements. This permits us to compare a hard boundary against a fuzzier description of the same general output property. The fuzzy conditions below therefore do not change the observable being measured; they change how its desired region is expressed to the

Table 8: Constraint capacity pools for OpenEnded and MBPP datasets at easy, medium, and hard difficulty levels.

OpenEnded easy pool	**OpenEnded medium pool**	**OpenEnded hard pool**
Use at most 14 fullstops.	Use at most 11 fullstops.	Use at most 8 fullstops.
Use at most 14 commas.	Use at most 11 commas.	Use at most 8 commas.
Keep to at most 750 words.	Keep to at most 600 words.	Keep to at most 450 words.
Use at most 14 sentences.	Use at most 11 sentences.	Use at most 8 sentences.
Use at most 5 paragraphs.	Use at most 4 paragraphs.	Use at most 3 paragraphs.
Use at most 8 bullet points (in the form of a +).	Use at most 6 bullet points (in the form of a +).	Use at most 4 bullet points (in the form of a +).
Bold at most 10 words using `**word**`.	Bold at most 7 words using `**word**`.	Bold at most 4 words using `**word**`.
Italicize at most 10 words using `_word_`.	Italicize at most 7 words using `_word_`.	Italicize at most 4 words using `_word_`.
Use at most 4 question marks.	Use at most 3 question marks.	Use at most 2 question marks.
Use at most 6 pairs of parentheses.	Use at most 4 pairs of parentheses.	Use at most 2 pairs of parentheses.
Use at most 4 semicolons.	Use at most 3 semicolons.	Use at most 2 semicolons.
Use at most 6 en dashes (in the form of three normal dashes `---`).	Use at most 4 en dashes (in the form of three normal dashes `---`).	Use at most 2 en dashes (in the form of three normal dashes `---`).
Use at most 8 apostrophes (`'`).	Use at most 6 apostrophes (`'`).	Use at most 4 apostrophes (`'`).
Use at most 7 pairs of double quotation marks (`" "`).	Use at most 5 pairs of double quotation marks (`" "`).	Use at most 3 pairs of double quotation marks (`" "`).
Use at most 6 exclamation marks.	Use at most 4 exclamation marks.	Use at most 2 exclamation marks.

MBPP easy pool	**MBPP medium pool**	**MBPP hard pool**
Do not exceed 20 lines of code.	Do not exceed 18 lines of code.	Do not exceed 16 lines of code.
Do not exceed 6 levels of indentation.	Do not exceed 5 levels of indentation.	Do not exceed 4 levels of indentation.
Use at most 5 return statements.	Use at most 4 return statements.	Use at most 3 return statements.
Include at most 6 lines of comments; keep them short and concise.	Include at most 5 lines of comments; keep them short and concise.	Include at most 4 lines of comments; keep them short and concise.
Use at most 3 lambdas.	Use at most 2 lambdas.	Use at most 1 lambda.
Use at most 4 for/while loops.	Use at most 3 for/while loops.	Use at most 2 for/while loops.
Define at most 4 functions (`def`).	Define at most 3 functions (`def`).	Define at most 2 functions (`def`).
Define at most 8 magic constants (a number or str).	Define at most 6 magic constants (a number or str).	Define at most 4 magic constants (a number or str).
Use at most 7 lists, sets, or dicts in aggregate totality (sum).	Use at most 6 lists, sets, or dicts in aggregate totality (sum).	Use at most 5 lists, sets, or dicts in aggregate totality (sum).
Use at most 4 list/set/dict comprehensions.	Use at most 3 list/set/dict comprehensions.	Use at most 2 list/set/dict comprehensions.
Each line of code should not exceed 88 characters.	Each line of code should not exceed 80 characters.	Each line of code should not exceed 72 characters.
Use at most 5 if statements.	Use at most 4 if statements.	Use at most 3 if statements.
Use at most 4 except clauses.	Use at most 3 except clauses.	Use the `del` keyword at most 1 time.
Use at most 4 semicolons.	Use at most 3 semicolons.	Use at most 2 semicolons.
Include at most 3 docstrings; keep them short and concise.	Include at most 2 docstrings; keep them short and concise.	Include at most 1 docstring; keep it short and concise.

Table 9: Infeasible constraints

Index	$E_{OpenEnded}$	$M_{OpenEnded}$	$H_{OpenEnded}$
0	Use less than 10 words and more than 9 sentences.	Use less than 10 words and more than 13 sentences.	Use less than 10 words and more than 18 sentences.
1	Use less than 10 words and more than 9 paragraphs.	Use less than 10 words and more than 13 paragraphs.	Use less than 10 words and more than 18 paragraphs.
2	Use more than 9 sentences. Every sentence contains fewer than 19 words. Each sentence after the first contains more words than the sentence before it. The last sentence contains fewer than twice as many words as the first sentence.	Use more than 9 sentences. Every sentence contains fewer than 15 words. Each sentence after the first contains more words than the sentence before it. The last sentence contains fewer than twice as many words as the first sentence.	Use more than 9 sentences. Every sentence contains fewer than 10 words. Each sentence after the first contains more words than the sentence before it. The last sentence contains fewer than twice as many words as the first sentence.
3	Use more than 4 sentences. The first sentence contains more than 1 comma. Each sentence after the first contains more commas than the sentence before it. The whole passage contains fewer than 20 commas.	Use more than 4 sentences. The first sentence contains more than 1 comma. Each sentence after the first contains more commas than the sentence before it. The whole passage contains fewer than 16 commas.	Use more than 4 sentences. The first sentence contains more than 1 comma. Each sentence after the first contains more commas than the sentence before it. The whole passage contains fewer than 11 commas.
4	Use more than 3 sentences. Each sentence after the first shares more than 5 distinct word types with the sentence immediately before it. No word type occurs in more than 2 sentences. Every sentence that is neither first nor last contains fewer than 12 words.	Use more than 3 sentences. Each sentence after the first shares more than 6 distinct word types with the sentence immediately before it. No word type occurs in more than 2 sentences. Every sentence that is neither first nor last contains fewer than 10 words.	Use more than 3 sentences. Each sentence after the first shares more than 8 distinct word types with the sentence immediately before it. No word type occurs in more than 2 sentences. Every sentence that is neither first nor last contains fewer than 9 words.

Index	E_{MBPP}	M_{MBPP}	H_{MBPP}
0	Use more than 9 top-level assignment statements. Every assignment right-hand side contains fewer than 19 AST expression nodes. Each assignment after the first contains more AST expression nodes in its right-hand side than the assignment before it. The last assignment right-hand side contains fewer than twice as many AST expression nodes as the first.	Use more than 9 top-level assignment statements. Every assignment right-hand side contains fewer than 15 AST expression nodes. Each assignment after the first contains more AST expression nodes in its right-hand side than the assignment before it. The last assignment right-hand side contains fewer than twice as many AST expression nodes as the first.	Use more than 9 top-level assignment statements. Every assignment right-hand side contains fewer than 10 AST expression nodes. Each assignment after the first contains more AST expression nodes in its right-hand side than the assignment before it. The last assignment right-hand side contains fewer than twice as many AST expression nodes as the first.
1	Use more than 4 top-level assignment statements. The first assignment statement contains more than 1 comma token. Each assignment after the first contains more comma tokens than the assignment before it. Across all top-level assignment statements, use fewer than 20 comma tokens.	Use more than 4 top-level assignment statements. The first assignment statement contains more than 1 comma token. Each assignment after the first contains more comma tokens than the assignment before it. Across all top-level assignment statements, use fewer than 16 comma tokens.	Use more than 4 top-level assignment statements. The first assignment statement contains more than 1 comma token. Each assignment after the first contains more comma tokens than the assignment before it. Across all top-level assignment statements, use fewer than 11 comma tokens.
2	Use more than 3 top-level assignment statements. Each assignment right-hand side after the first shares more than 5 distinct string literal values with the immediately preceding assignment right-hand side. No string literal value occurs in more than 2 assignment right-hand sides. Every assignment that is neither first nor last contains fewer than 12 distinct string literal values in its right-hand side.	Use more than 3 top-level assignment statements. Each assignment right-hand side after the first shares more than 6 distinct string literal values with the immediately preceding assignment right-hand side. No string literal value occurs in more than 2 assignment right-hand sides. Every assignment that is neither first nor last contains fewer than 10 distinct string literal values in its right-hand side.	Use more than 3 top-level assignment statements. Each assignment right-hand side after the first shares more than 8 distinct string literal values with the immediately preceding assignment right-hand side. No string literal value occurs in more than 2 assignment right-hand sides. Every assignment that is neither first nor last contains fewer than 9 distinct string literal values in its right-hand side.
3	Use more than 4 top-level if statements. The first if body contains more than 1 direct assignment statement. Each later if body contains more direct assignment statements than the preceding if body. Across all of these if bodies, use fewer than 20 direct assignment statements.	Use more than 4 top-level if statements. The first if body contains more than 1 direct assignment statement. Each later if body contains more direct assignment statements than the preceding if body. Across all of these if bodies, use fewer than 16 direct assignment statements.	Use more than 4 top-level if statements. The first if body contains more than 1 direct assignment statement. Each later if body contains more direct assignment statements than the preceding if body. Across all of these if bodies, use fewer than 11 direct assignment statements.

Table 10: Summary statistics for target infeasibility and contradiction gap across difficulty levels.

	Easy	Medium	Hard
Mean target infeasibility	0.2998	2.246	9.774
SD target infeasibility	0.1169	0.429	1.359
Median target infeasibility	0.2979	2.272	9.777
IQR target infeasibility	[0.1933,0.3972]	[1.871,2.620]	[8.573,10.85]
Range target infeasibility	[0.1029,0.4970]	[1.505,2.990]	[7.625,12.42]
Mean contradiction gap	0.5997	4.492	19.55
SD contradiction gap	0.2337	0.858	2.718

Table 11: Constraints for Target-Count Attainment Experiments.

Index	OpenEnded-E	OpenEnded-M	OpenEnded-H
0	Use more than 20 fullstops. Keep below 15 fullstops. Maintain writing quality and readability.	Use more than 30 fullstops. Keep below 15 fullstops. Maintain writing quality and readability.	Use more than 40 fullstops. Keep below 15 fullstops. Maintain writing quality and readability.
1	Use more than 20 commas. Keep below 15 commas. Maintain writing quality and readability.	Use more than 30 commas. Keep below 15 commas. Maintain writing quality and readability.	Use more than 40 commas. Keep below 15 commas. Maintain writing quality and readability.
2	Use more than 20 semicolons. Keep below 15 semicolons. Maintain writing quality and readability.	Use more than 30 semicolons. Keep below 15 semicolons. Maintain writing quality and readability.	Use more than 40 semicolons. Keep below 15 semicolons. Maintain writing quality and readability.
3	Use more than 20 question marks. Keep below 15 question marks. Maintain writing quality and readability.	Use more than 30 question marks. Keep below 15 question marks. Maintain writing quality and readability.	Use more than 40 question marks. Keep below 15 question marks. Maintain writing quality and readability.

Index	MBPP-E	MBPP-M	MBPP-H
0	Use more than 19 loops. Strictly keep below 20 lines of code. Semicolons are strictly prohibited. Ensure code quality and correctness.	Use more than 26 loops. Strictly keep below 20 lines of code. Semicolons are strictly prohibited. Ensure code quality and correctness.	Use more than 32 loops. Strictly keep below 20 lines of code. Semicolons are strictly prohibited. Ensure code quality and correctness.
1	Use more than 19 if-else statements. Strictly keep below 20 lines of code. Semicolons are strictly prohibited. Ensure code quality and correctness.	Use more than 26 if-else statements. Strictly keep below 20 lines of code. Semicolons are strictly prohibited. Ensure code quality and correctness.	Use more than 32 if-else statements. Strictly keep below 20 lines of code. Semicolons are strictly prohibited. Ensure code quality and correctness.
2	Use more than 19 case clauses in a match. Strictly keep below 20 lines of code. Semicolons are strictly prohibited. Ensure code quality and correctness.	Use more than 26 case clauses in a match. Strictly keep below 20 lines of code. Semicolons are strictly prohibited. Ensure code quality and correctness.	Use more than 32 case clauses in a match. Strictly keep below 20 lines of code. Semicolons are strictly prohibited. Ensure code quality and correctness.
3	Use more than 19 except clauses in a try. Strictly keep below 20 lines of code. Semicolons are strictly prohibited. Ensure code quality and correctness.	Use more than 26 except clauses in a try. Strictly keep below 20 lines of code. Semicolons are strictly prohibited. Ensure code quality and correctness.	Use more than 32 except clauses in a try. Strictly keep below 20 lines of code. Semicolons are strictly prohibited. Ensure code quality and correctness.

model. The hard variants are checked by the deterministic verifiers described above. The underlying numeric measurements are also retained for the fuzzy variants. All constraints are listed in Table 12.

B.10 Fuzzy Interaction Constraints

Eight fuzzy constituent constraints are defined for each task. As in the hard interaction experiment, all $\binom{8}{2} = 28$ candidate pairs can be formed from the constituent set. The main text figure reports selected empirically identified examples of the three interaction regimes rather than displaying all 28 pairs. The labels report the empirical interaction categories used in the main-text figure; they are not assigned from the literal wording alone. All constraints are listed in Table 13.

B.11 Fuzzy Infeasible Constraints

Unlike the conventional setting, neither direction defines a sharp boundary, so a response can compromise between the two. The construction uses fixed, symmetric pairs rather than manually curating pair-specific compromises. All constraints are listed in Table 14.

Thus the contradictory soft pairs are hard-coded from constituents $(0, 1)$, $(2, 3)$, and $(4, 5)$ for each task. This construction has two advantages. First, the opposed directions are defined before observ-

Table 12: Fuzzy Constraint details

Index	OpenEnded	MBPP
1	Use less than 200 words.	Use less than 7 lines.
2	Use less than 350 words.	Use less than 13 lines.
3	Use less than 500 words.	Use less than 18 lines.
4	Keep the number of word somewhat moderately short	Keep the number of lines somewhat moderately short
5	Keep the number of word short	Keep the number of lines short
6	Keep the number of word extremely short	Keep the number of lines extremely short
7	Use less than 10 commas.	Use less than 6 lists, tuples, sets, and dicts in totality.
8	Use less than 17 commas.	Use less than 9 lists, tuples, sets, and dicts in totality.
9	Use less than 25 commas.	Use less than 12 lists, tuples, sets, and dicts in totality.
10	Keep the number of commas somewhat moderately low	Keep the number of lists, tuples, sets, and dicts in totality somewhat moderately low.
11	Keep the number of commas low	Keep the number of lists, tuples, sets, and dicts in totality low.
12	Keep the number of commas extremely very low	Keep the number of lists, tuples, sets, and dicts in totality extremely very low.
13	Use less than 7 periods.	Use less than 4 if-else blocks and for/while loops in totality.
14	Use less than 12 periods.	Use less than 8 if-else blocks and for/while loops in totality.
15	Use less than 17 periods.	Use less than 12 if-else blocks and for/while loops in totality.
16	Keep the number of periods somewhat moderately low	Keep the number of if-else blocks and for/while loops in totality somewhat moderately low.
17	Keep the number of periods low	Keep the number of if-else blocks and for/while loops in totality low.
18	Keep the number of periods extremely very low	Keep the number of if-else blocks and for/while loops in totality extremely very low.

Table 13: Constraint details for Fuzzy Interaction Experiments

Index	OpenEnded	MBPP
0	Use less words.	Keep the number of lines short.
1	Use less sentences.	Use less if-else block and for/while loops in totality.
2	Use less commas.	Keep the number of lists, tuples, sets, and dicts in totality low.
3	Use less semicolons.	Keep the max level indentation small.
4	Use less periods.	Keep each line of code short.
5	Keep each sentence short.	Use less comprehension expressions.
6	Use less conjunctions (and, but, or, nor, for, yet, so).	Use less magic constants.
7	Use less determiners (the, a, this, that, some, every, each, no).	Use less assignment operators (the '=' operator).

ing model behaviour, so no pair-specific model output is used to curate a desired trade-off. Second, the two constituents retain parallel wording and differ primarily in the requested direction, reducing incidental lexical differences between the two sides of the trade-off. The baseline condition presents both constituents without an explicit priority. The control conditions attach the same weight syntax to each constituent and vary only the numerical allocation. For example, the OpenEnded 0_1 pair under an 80:20 allocation is presented as "- Use less words. (80% Weightage) - Use more words. (20% Weightage)".

The corresponding 20:80 condition reverses the numerical weights, while the 50:50 condition assigns equal weight to both. Because weighting is implemented by mechanically affixing the requested percentage to the original constituent line, no additional explanatory text, model-generated compromise, or hand-written pair-specific guidance is introduced. This yields a curation-free control in which the principal manipulated variable is the stated relative priority of the two opposing soft constraints.

C Models and Generation Method

We evaluate instruction-tuned models drawn from three model families: "mistral-small-2506", "gemma-4-26B-A4B-it", and "llama31-8b-instruct". The suite was selected to balance three considerations: benchmark performance, architectural diversity, and the computational cost. Our goal is not to identify the strongest available model in isolation. Rather, we require models that are sufficiently capable for our analysis to be meaningful, while remaining tractable and manageable enough to support the large number of repeated generations required by our analyses.

Mistral-small-2506 is the 24B instruction-tuned Mistral Small 3.2 release from June 2025 Mistral AI (2025). Gemma-4-26B-A4B-it is a sparse mixture-of-experts model with approximately 26B total and 3.8B active parameters, reducing active computation relative to similarly sized dense mod-

Table 14: Fuzzy Constraint details for Contradictory Infeasibility Experiments

Index	OpenEnded	MBPP
0	Use less words.	Use less lines of code.
1	Use more words.	Use more lines of code.
0_1	- Use less words. - Use more words.	- Use less lines of code. - Use more lines of code.
2	Keep each sentence short.	Keep each line of code short.
3	Keep each sentence long.	Keep each line of code long.
2_3	- Keep each sentence short. - Keep each sentence long.	- Keep each line of code short. - Keep each line of code long.
4	Use less sentences.	Use less if-else blocks, for/while loops, and comprehension expressions in totality.
5	Use more sentences.	Use more if-else blocks, for/while loops, and comprehension expressions in totality.
4_5	- Use less sentences. - Use more sentences.	- Use less if-else blocks, for/while loops, and comprehension expressions in totality. - Use more if-else blocks, for/while loops, and comprehension expressions in totality.

els Abd et al. (2026). Llama31-8b-instruct is the 8B instruction-tuned Llama 3.1 model Meta AI (2024), providing a smaller and cheaper inference regime. Together, the models span dense and sparse architectures from 8B to roughly 26B parameters, varying model family and computational scale while maintaining tractable inference.

C.1 Chat Interface and Prompt Construction

Each model is queried through its native chat template. Where a model or serving endpoint provides a default system or helpfulness prompt, that native default is retained. All models are evaluated in their standard conversation configuration. No separate thinking or reasoning mode is enabled.

For every experiment, we use 3 semantically equivalent prompt templates. Recall that base task and formatting requirements are held fixed across experimental conditions. By aggregating uniformly across different prompt templates, we aim to reduce the risk of lexical confounders with reported measurements. The three templates for OpenEnded and MBPP are given in Table 15. Braced terms denote instance-specific fields substituted at runtime.

For Scheduling, a single instance can be presented through several different natural-language question templates without changing the underlying mathematical problem. Each question template is composed of an introductory scenario, a naming convention for the decision variables, sentence templates for linear constraints and variable bounds, and formatting requirements. The LP instance supplies the numerical content, while the selected template determines how that content is verbalized.

The first source of variation is the scenario. Each template gives the LP a domain-specific interpretation. A template may begin with either one of the three choices:

```
1. A factory must decide how much of each product to make.
2. A manager must assign hours across several workers or tasks.
3. A logistics planner must decide how much to ship along several routes.
```

The scenario determines how the decision variables are referred by assigning them domain-specific roles and units. Consequently, the same numerical variable can acquire a different semantic meaning depending on the selected template. After the variables have been given names, each algebraic constraint is converted into a sentence where the direction of the inequality is written in its textual version. Individual story templates can change semantic meaning while preserving exactly the same mathematical structure. Once the scenario, variables, bounds, and constraints have been translated, the pieces are assembled into a fixed conceptual order from scenario introduction, meaning of the decision variables, variable bounds, linear requirements and to final feasibility question.

Additional instructions introduced by an experimental condition are inserted at the interface between the base task and the experimental constraint block. Depending on the condition, these instructions are prefixed or appended to the constraint text while leaving the base-task requirements unchanged.

Table 15: Model PromptTemplates

Task	Template	Prompt
OpenEnded	1	{qns} When writing your answer, strictly follow these requirements: {constraint}
OpenEnded	2	{qns} Your response *must* follow the rules below: {constraint}
OpenEnded	3	{qns} Ensure that your output follows the listed requirements below: {constraint}
MBPP	1	{qns} You *must* follow the instructions below: - The code must be fenced in ```python and ``` markdown fence. - {constraint} - `{fn_name}` MUST be the solution function name, with signature: `{signature}`.
MBPP	2	{qns} Must-follow requirements: - Function name must be `{fn_name}` with exact input signature: `{signature}` - Must be wrapped in ```python and ``` markdown fences - {constraint}
MBPP	3	{qns} - Strictly *ONE* single solution only. - `{fn_name}` MUST be the solution function name, with signature: `{signature}`. - The solution *must* be placed in ```python and ``` markdown fences. - {constraint}

This includes the prompt-level interventions used. Preference instructions, for example, are expressed as natural-language prefixes.

C.2 Sampling and Decoding

All generations use temperature $T = 0.8$. to ensure distributional coverage when sampling. We do not impose any additional experiment-specific decoding controls. The maximum generation length is 2000 output tokens, which significantly exceeds the typical length of our observed sequences. Generation terminates when either the model's native end-of-sequence token is emitted or the prescribed token limit is reached. Inference is performed using a mixture of local model execution and hosted API calls. Generation failures are handled using a backoff-and-retry protocol. This procedure was sufficient to recover the generation failures encountered in our experiments.

C.3 Rarefaction of Feasible Generations

Difficulty properties are defined conditional on successful generation. For each x_i and c, we pool generations across $M = 3$ models, $\Omega = 3$ prompt templates, and $R = 10$ independent retries, yielding $M\Omega R = 90$. From the task-correct generations satisfying c, we uniformly sample without replacement $n^\star = 20$. Conditions with fewer than 20 observations were excluded; however, all conditions satisfied this threshold. The same rarefied sample is used to estimate slack, solution-space concentration, and conditional accessibility. Dataset-level constraint measurements are obtained by first computing the corresponding metric separately for each of the $N_t = 20$ task instances and then taking their unweighted arithmetic mean. Each dataset-level difficulty measurement for one exact constraint is based on $20 \times 20 = 400$ retained feasible generations, with each of the 20 task instances contributing equally. Appendix E.1 uses a separate rarefaction protocol for out-of-sample validation.

C.4 TARGETED INTERVENTIONS

Section 3.2 evaluates whether directly steering generation in the direction predicted by each individual-difficulty property improves instruction following. We use the same individual-constraint inventory as Section 3.1, detailed in Appendix B.4, with the following changes. Slack interventions insert one preference sentence immediately before the experimental constraint, telling the model to "prefer" staying below 50%-80% of the stated boundary. Diversity interventions encourage solutions outside the dominant feasible mode without altering task constraints. We identify the dominant mode from control generations, select a representative example, and ask the model for one valid solution that is structurally distinct from it. The accessibility intervention uses three few-shot examples, chosen for low response-token-normalised NLL, appended after the template. The control prompt is simply the original unmodified template. An intervention is matched exactly to its corresponding control. Generation protocols follow as elsewhere in the paper and in Section C. For intervention $\iota \in \mathcal{I}$, the reported IFR effect for task t is $\Delta\mathrm{IFR}_{t,\iota} = 100\,(\mathrm{IFR}_{t,\iota} - \mathrm{IFR}_{t,0})$, where 0 denotes the control. Finally, we compute the paired relative changes:

$$\Delta s_{t,\iota} = 100\,\frac{s_{t,\iota} - s_{t,0}}{s_{t,0}}, \qquad \Delta q_{t,\iota} = 100\,\frac{q_{t,\iota} - q_{t,0}}{q_{t,0}}, \qquad \Delta e_{t,\iota} = 100\,\frac{e_{t,\iota} - e_{t,0}}{e_{t,0}}.$$

C.4.1 MATCHED CONTROLS FOR TARGETED INTERVENTIONS

For the **slack** intervention, the targeted condition is unchanged from Table 1. The matched control adds a similarly sized generic reminder to satisfy the stated requirements, without mentioning boundaries or target values. For the **diversity** intervention, the targeted condition is also unchanged from Table 1. The matched control uses the same instruction and single-example format, but supplies a feasible example sampled without regard to solution mode and matched approximately in response length. Thus, the conditions differ only in whether example selection uses measured solution concentration. For **perplexity**, building on Table 1, we construct a metric-selected variant using three successful examples with low response-token-normalised negative log-likelihood. The matched control uses three successful examples sampled without regard to likelihood, while matching demonstration count, formatting, and approximate total length. This variant is used only for the matched-control and held-out-selection analyses and does not alter the intervention in Table 1.

For task t and intervention family $\iota \in \mathcal{I}$, we separately compute the change in IFR relative to the original prompt,

$$\Delta\mathrm{IFR}^{\mathrm{tar}}_{t,\iota} = 100\big(\mathrm{IFR}^{\mathrm{tar}}_{t,\iota} - \mathrm{IFR}_{t,0}\big), \qquad \Delta\mathrm{IFR}^{\mathrm{ctl}}_{t,\iota} = 100\big(\mathrm{IFR}^{\mathrm{ctl}}_{t,\iota} - \mathrm{IFR}_{t,0}\big),$$

where 0 denotes the original unmodified prompt. The metric-specific gain reported in Table 2 is the paired difference

$$\Gamma_{t,\iota} = 100\big(\mathrm{IFR}^{\mathrm{tar}}_{t,\iota} - \mathrm{IFR}^{\mathrm{ctl}}_{t,\iota}\big).$$

Confidence intervals follows Appendix D. The same resampled base-instance indices are reused across the original, targeted, and matched-control conditions so that intervention differences remain paired. Additional instructions are inserted at the same prompt location used for the interventions in Table 1. Generation procedures follow Appendix C.

C.4.2 HELD-OUT DIFFICULTY-GUIDED INTERVENTION SELECTION

We test whether difficulty measurements can prospectively guide intervention choice. For each task, the $N_t = 20$ base instances are split equally into fixed development and held-out sets before intervention construction. Development instances are used only to estimate difficulty and construct intervention-specific examples; held-out instances are used only for evaluation. Because the three metrics differ in scale and direction, we convert their development-set values to within-task empirical ranks. in $[0, 1]$. For c, we define adverse scores

$$a_s(c) = 1 - r_s(c), \qquad a_q(c) = r_q(c), \qquad a_e(c) = r_e(c),$$

where r_s, r_q, and r_e are the within-task ranks of slack, concentration, and conditional perplexity, respectively. The largest adverse score determines the corresponding difficulty-matched intervention. Ties are resolved by a fixed ordering. Any generation-dependent prompt material is constructed from development data only, and the selected intervention is frozen before held-out evaluation.

We compare this policy with the **original** prompt, a **generic** non-metric compliance reminder, and a **random-intervention** policy that assigns the same three intervention families approximately evenly within each task without using difficulty measurements. Held-out $\Delta\mathrm{IFR}$ is measured relative to the original prompt. The principal comparison is

$$\Gamma_t^{\mathrm{select}} = 100\big(\mathrm{IFR}_t^{\mathrm{guided}} - \mathrm{IFR}_t^{\mathrm{random}}\big),$$

which tests whether the difficulty measurements improve intervention selection beyond choosing from the same intervention set without metric information. Uncertainty is estimated following Appendix D, restricted to held-out instances. The development-set measurements, intervention selections, and random-policy assignments are treated as fixed design quantities during the bootstrap.

D Hierarchical-Bootstrap Confidence Intervals

We quantify uncertainty using $B = 2000$ hierarchical-bootstrap replicates. Models, prompt templates, tasks, constraint inventories, interventions, and interaction pairs are treated as fixed design factors, so intervals generalise over task instances and stochastic generations conditional on the evaluated design. Within each task, we resample the $N_t = 20$ base instances with replacement, using the same sampled indices across all constraints and conditions to preserve matched comparisons. For IFR-based quantities, we additionally resample the $R = 10$ generations within each fixed model–template–condition cell. Bootstrap multiplicities are shared wherever the same generation contributes to multiple statistics or interaction pairs, preserving observed dependence.

For difficulty quantities, each selected (x_i, c) resamples $n^\star = 20$ records from its retained rarefied feasible set $\mathcal{R}_{t,i}(c)$. The same resample is used for slack, Simpson concentration, and conditional perplexity, preserving the conditional-success design and equal instance weighting. Point estimates remain those from the original unresampled data and without-replacement rarefaction.

Each replicate recomputes the statistics reported in the corresponding analysis. For § 3.1, this includes constraint-level IFR, difficulty measures, Spearman's ρ, and OLS R^2. For § 3.2, matched intervention and control observations are resampled jointly and we recompute IFR differences and targeted-metric changes. For § 3.3, we recompute constituent and joint IFRs, CMS, JIP, $\log(\mathrm{IR})$, and regime-wise means, while preserving dependence between constituent indicators. Interaction tolerances, displayed pairs, and regime assignments remain fixed from the original analysis.

For § 4, generation multiplicities are shared across all constituent constraints of each sampled set, preserving within-output dependence; follow rates are then recomputed by task, constraint count, and difficulty or interaction condition. Capacity sets, difficulty terciles, and interaction assignments remain fixed. For § 6, we apply the same instance- and generation-level resampling but omit feasible-set rarefaction, recomputing fuzzy statistics from task-correct generations. Matched conventional–fuzzy conditions share instance resamples, and model-judge fuzzy-slack scores travel with their corresponding generations. For Figure 5, marginal streams shared across pairs are resampled once and reused when recomputing $\mu_k(\{c_k\})$, $\mu_k(\{c_i, c_j\})$, and Δ_k for $k \in \{i, j\}$, as well as the intra-pair discrepancies; displayed pairs and labels remain fixed.

All reported 95% confidence intervals are pointwise percentile intervals from the $(0.025, 0.975)$ quantiles of the bootstrap distribution. Cross-task summaries use the same unweighted task average as the main analysis, with tasks themselves not resampled. We use a fixed recorded random seed and do not regularise undefined ratios or logarithms; any undefined replicate counts are reported with the corresponding interval.

E Metric Validation and Details

E.1 Pilot Validation of Individual Difficulty Properties

Out-of-Sample Validation The individual-difficulty measurements are computed from feasible model generations, so relating them to IFR risks shared-sample bias. If IFR is estimated from the same generations, associations may reflect sampling variation or conditioning on common success events rather than a reproducible relationship with IFR. A separate concern is that tightening the numerical boundary of a fixed output property $g(y) < b$ affects not only satisfaction but also the

distribution of feasible outputs, so observed relationships with IFR may partly reflect the boundary manipulation itself. We therefore introduce three validation procedures to separate the estimated difficulty measurements from predicted outcomes. First, difficulty measures and IFR are estimated from disjoint Monte Carlo samples. Second, we test prediction of IFR at a fixed numerical constraint, ruling out identification from boundary tightening. Finally, we test cross-model transfer by using difficulty estimated from one model to predict failures in another.

Sampling Budgets To avoid shared-sample bias while fixing the rarefaction size, we generate two independent Monte Carlo streams for each task instance x_i and constraint c. Stream A samples under the same model and prompt-template conditions until $n^\star = 20$ task-correct generations satisfying c are obtained, up to stream $B_{\max} = 180$. Difficulty estimates are excluded from validation if Stream A reaches $B_{\max}$ before $n^\star$; however, this did not occur in our experiments. These outputs are used to estimate slack, solution-space concentration, and conditional accessibility. An independent stream B of size $B_{\max}$ estimates the corresponding held-out instruction-following rate, ensuring that no generation contributes to both difficulty and IFR estimates.

Disjoint Estimation Difficulty properties are estimated from $n^\star = 20$ feasible generations from Stream A. Stream B generations are pooled separately and used to estimate the corresponding held-out satisfaction rate. We repeat the paired streaming procedure across multiple seeds to improve statistical independence and robustness. The results are shown in Table 16

Table 16: Disjoint-sample associations between difficulty properties and held-out IFR.

Difficulty property	OpenEnded	MBPP	Scheduling
Simpson concentration	-0.351	-0.324	-0.421
Slack	0.791	0.837	0.673
Perplexity	-0.650	-0.549	-0.305

Macro-averaged correlations were -0.365 for Simpson concentration, 0.767 for slack, and -0.5013 for perplexity. Thus, no generation used to construct a difficulty measurement contributes to the corresponding IFR estimate. Difficulty remains a conditional property of successful outputs, but those successful outputs are statistically distinct from the generations used to estimate satisfaction probability.

Boundary Control We test whether the difficulty properties predict satisfaction when the numerical constraint boundary is held fixed. Instance-level Simpson concentration, slack, and conditional perplexity are estimated from Stream A and used to predict generation-level satisfaction in independent Stream B via logistic regression. Let $v_{icm\omega r} \in \{0, 1\}$ indicate whether a Stream-B generation satisfies constraint c, and let $z(\cdot)$ denote standardisation. We fit

$$\operatorname{logit} \Pr(v_{icm\omega r} = 1) = \alpha_c + \gamma_m + \delta_\omega + \beta_q z(\bar{q}_{ic}) + \beta_s z(\bar{s}_{ic}) + \beta_e z(\log \bar{e}_{ic}).$$

Here, α_c is an intercept for each exact constraint and numerical boundary, while γ_m and δ_ω capture model and prompt-template effects. Thus, difficulty coefficients are identified only from variation across task instances under the same numerical requirement. We evaluate using leave-one-task-instance-out cross-validation, reporting AUROC and Brier score from out-of-fold predictions.

Table 17: Fixed-boundary logistic regressions predicting satisfaction of an independent generation. Values are odds ratios for a one-standard-deviation increase in each predictor.

Predictor	Separate OR	Joint OR
Simpson concentration	0.78	0.85
Slack	1.22	1.19
Log perplexity	0.84	0.90

In Table 17, the separate models show the expected directions: greater slack increases satisfaction odds, whereas greater concentration and log perplexity reduce them. These directions persist in the joint model, with smaller magnitudes due to shared explanatory information.

Table 18: Held-out predictive performance of the fixed-boundary models.

Model	AUROC	Brier score
Fixed effects only	0.612	0.256
+ Simpson concentration	0.640	0.229
+ Slack	0.661	0.234
+ Log perplexity	0.635	0.201
+ All three properties	0.695	0.192

In Table 18, adding all three properties increases held-out AUROC from 0.612 to 0.695 and reduces Brier score from 0.256 to 0.192. Thus, difficulty properties predict independent satisfaction even within an identical numerical constraint, indicating that their association with IFR is not explained solely by boundary tightness.

Cross-Model Transfer For each held-out model m, we estimate difficulty using only Stream A generations from the remaining source models. Feasible generations are pooled across source models and prompt templates, rarefied to $n^\star = 20$, and aggregated into source-only estimates of slack, concentration, and perplexity. We then fit, separately by task, an ordinary least-squares mapping from $[\bar{s}_{-m}(c), \bar{q}_{-m}(c), \log \bar{e}_{-m}(c)]$ to source-model Stream-B IFR. All standardisation and fitting use source models only; the fitted mapping is applied without refitting to the held-out model's independently measured Stream-B IFR. Predictions are clipped to $[0, 100]\%$, and the procedure is repeated with each model held out in turn.

Table 19: Leave-one-model-out transfer performance. MAE is measured in IFR percentage points.

Held-out target model	Predicted–observed IFR ρ	IFR MAE	Out-of-sample R^2
Mistral Small 2506	+0.676	7.7	0.26
Gemma 4 26B-A4B-IT	+0.691	8.3	0.31
Llama 3.1 8B Instruct	+0.638	11.4	0.22

Table 19 shows positive predicted–observed IFR correlations for every held-out model. Macro-averaged ρ, MAE, and out-of-sample R^2 are 0.66, 9.1 percentage points, and 0.26, respectively, indicating that the difficulty measurements contain information that transfers across model families, rather than reflecting only model-specific sampling artifacts.

E.1.1 Success Prediction With Difficulty Beyond Baseline IFR

The preceding analyses test whether properties of successful generations predict IFR on an independent generation stream. The same sampling procedure also yields a direct empirical measure of difficulty: the rate at which satisfying generations are obtained. Using the disjoint Stream-A/Stream-B construction above, let A_{ic} denote the number of task-correct Stream-A generations required to obtain 20 satisfying generations, and define $\widehat{p}^{A}_{ic} = n^\star / A_{ic}$. We test whether the proposed difficulty properties provide predictive information beyond this baseline. We retain the fixed-boundary prediction setting above and compare three nested logistic models, where $v_{icm\omega r} \in 0, 1$ indicates whether independent Stream-B generation r from m and prompt template ω satisfies c on x_i:

$$\mathcal{M}_0: \quad \operatorname{logit} \Pr(v_{icm\omega r} = 1) = \alpha_c + \gamma_m + \delta_\omega,$$

$$\mathcal{M}_1: \quad \operatorname{logit} \Pr(v_{icm\omega r} = 1) = \alpha_c + \gamma_m + \delta_\omega + \beta_p z(\widehat{p}^{A}_{ic}),$$

$$\mathcal{M}_2: \quad \operatorname{logit} \Pr(v_{icm\omega r} = 1) = \alpha_c + \gamma_m + \delta_\omega + \beta_p z(\widehat{p}^{A}_{ic}) + \beta_q z(\bar{q}_{ic}) + \beta_s z(\bar{s}_{ic}) + \beta_e z(\log \bar{e}_{ic}).$$

$\mathcal{M}_0$ contains the fixed effects from the preceding analysis, $\mathcal{M}_1$ adds the independently measured Stream-A success-rate proxy, and $\mathcal{M}_2$ additionally includes the three proposed difficulty measurements. All predictors are computed from Stream A, with standardisation based only on the training fold; no Stream-B outcome contributes to any predictor. As above, each exact constraint receives its own intercept α_c. The primary comparison is therefore $\mathcal{M}_2$ versus $\mathcal{M}_1$.

Table 20: Held-out prediction after controlling for $\widehat{p}_{ic}^{A}$. Values are AUROC (↑).

Predictors	OpenEnded	MBPP	Scheduling
Fixed effects	$0.638^{0.595}_{0.647}$	$0.617^{0.605}_{0.658}$	$0.623^{0.606}_{0.659}$
$+\,\widehat{p}^{A}$	$0.664^{0.648}_{0.685}$	$0.634^{0.598}_{0.644}$	$0.647^{0.613}_{0.672}$
$+\,\widehat{p}^{A} + \bar{q} + \bar{s} + \log \bar{e}$	$\mathbf{0.720}^{0.703}_{0.746}$	$\mathbf{0.690}^{0.684}_{0.720}$	$\mathbf{0.697}^{0.684}_{0.721}$

We use the same task-instance-level held-out procedure as in the preceding fixed-boundary analysis. Uncertainty estimates follow Appendix D. Table 20 shows that $\widehat{p}_{ic}^{A}$ is itself informative, increasing mean AUROC by 0.022 over the fixed-effects model. Adding the three difficulty measurements yields a further average gain of 8.33%, indicating predictive information beyond this empirical success-rate baseline.

Instruction difficulty may also transfer across models, so we compare the metric-based transfer in Appendix E.1 with direct transfer of source-model IFR under the same leave-one-model-out construction. Using Stream-B samples, define the source-model prediction as

$$\widehat{\mathrm{IFR}}^{\mathrm{src}}_{-m}(c) = \frac{1}{M-1} \sum_{m' \neq m} \mathrm{IFR}^{B}_{m'}(c).$$

For the metric-based predictor, we use the previously defined source-model difficulty vector

$$[\bar{s}_{-m}(c), \bar{q}_{-m}(c), \log \bar{e}_{-m}(c)]$$

and the same source-only fitted mapping described above. This measures whether properties of successful generations provide transferable information beyond source-model success alone.

Both predictors are evaluated against the same independently measured Stream-B IFR of the held-out target model. We report Spearman correlation ρ between predicted and observed constraint-level IFR, mean absolute error (MAE) in IFR percentage points, and out-of-sample R^2. Constraint-level predictions are pooled across tasks only after the task-specific quantities have been constructed. Uncertainty estimates follow Appendix D.

Table 21 compares direct transfer of source-model IFR with the difficulty-based predictor. Source-model IFR is already informative, indicating that instruction difficulty is partly shared across models, with macro-averaged $\rho = 0.605$, MAE of 10.5 percentage points, and out-of-sample $R^2 = 0.18$. The difficulty-based predictor improves these values to 0.668, 9.1, and 0.26, respectively, and outperforms direct IFR for every held-out model across all three metrics. The gains, particularly in calibration and R^2, suggest that properties of successful generations capture transferable information beyond the shared ordering of constraints reflected in source-model success rates.

Table 21: Leave-one-model-out transfer compared with direct source-model IFR. Higher ρ and R^2 and lower MAE are better. Difficulty-metric results correspond to the existing transfer analysis.

Held-out target model	Transfer signal	ρ	MAE	R^2
Mistral Small 2506	Direct source IFR	$0.612^{0.580}_{0.721}$	$9.1^{7.3}_{9.3}$	$0.18^{0.11}_{0.22}$
	Difficulty metrics	$0.676^{0.627}_{0.751}$	$7.7^{7.3}_{10.3}$	$0.26^{0.24}_{0.33}$
Gemma 4 26B-A4B-IT	Direct source IFR	$0.628^{0.586}_{0.689}$	$9.8^{8.2}_{10.9}$	$0.22^{0.15}_{0.24}$
	Difficulty metrics	$0.691^{0.623}_{0.739}$	$8.3^{7.8}_{9.8}$	$0.31^{0.23}_{0.32}$
Llama 3.1 8B Instruct	Direct source IFR	$0.574^{0.559}_{0.693}$	$12.7^{11.7}_{13.9}$	$0.14^{0.11}_{0.22}$
	Difficulty metrics	$0.638^{0.577}_{0.682}$	$11.4^{10.1}_{12.9}$	$0.22^{0.15}_{0.26}$

E.2 Transfer Beyond Empirical Source-Model IFR

We test whether the proposed difficulty measurements provide cross-model predictive information beyond empirical source-model IFR. For each held-out target model m, let $\bar{f}_{-m}(c)$ denote the mean IFR of c across the source m. We compare direct transfer of $\bar{f}_{-m}(c)$, a ridge-calibrated IFR-only predictor,

$$\widehat{f}^{\mathrm{IFR}}_{m}(c) = \beta_0 + \beta_1 \bar{f}_{-m}(c),$$

Table 22: Cross-model prediction under matched fitting. The calibrated IFR baseline and augmented model use identical fitting and source-only model-selection procedures.

Held-out target	Predictor	ρ	MAE ↓	R^2
Mistral Small 2506	Direct source IFR	0.613	8.8	0.17
	Calibrated source IFR	0.613	8.2	0.22
	Source IFR + difficulty metrics	**0.678**	**6.9**	**0.28**
Gemma 4 26B-A4B-IT	Direct source IFR	0.626	9.6	0.22
	Calibrated source IFR	0.625	9.2	0.27
	Source IFR + difficulty metrics	**0.681**	**8.1**	**0.34**
Llama 3.1 8B Instruct	Direct source IFR	0.568	12.8	0.12
	Calibrated source IFR	0.569	12.1	0.15
	Source IFR + difficulty metrics	**0.622**	**11.2**	**0.21**
Macro mean: Direct source IFR		0.602	10.4	0.17
Macro mean: Calibrated source IFR		0.602	9.8	0.21
Macro mean: Source IFR + difficulty metrics		**0.660**	**8.7**	**0.28**

and an augmented predictor,

$$\widehat{f}_m^{\text{full}}(c) = \beta_0 + \beta_1 \bar{f}_{-m}(c) + \beta_2 \bar{s}_{-m}(c) + \beta_3 \bar{q}_{-m}(c) + \beta_4 \log \bar{e}_{-m}(c).$$

Both fitted predictors use ridge regression with predictor standardisation and regularisation selected from source data only; the target model contributes neither predictors nor outcomes to fitting or model selection. They therefore receive matched fitting and use the same source-generation budget. We evaluate predictions against independently measured held-out target-model IFR using Spearman correlation ρ, mean absolute error (MAE, in IFR percentage points), and out-of-sample R^2. Uncertainty estimates follow Appendix D. Hierarchical bootstrap replicates recompute preprocessing, hyperparameter selection, and fitting using source data only.

Table 22 separates gains due to fitting from gains associated with the difficulty measurements. Calibration alone improves MAE and R^2, while adding the difficulty measurements further improves all three metrics for every held-out model. Macro-averaged ρ increases from 0.602 to 0.660, MAE decreases from 9.8 to 8.7 points, and R^2 increases from 0.21 to 0.28. Thus, within the evaluated models, the difficulty measurements provide predictive information beyond comparably fitted empirical source-model IFR.

E.3 Controls for Difficulty-Guided Intervention Selection

We evaluate whether difficulty-guided intervention selection benefits from constraint-specific matching rather than only from choosing interventions with high average effectiveness. All policy construction uses the development partition, while reported IFR is measured only on the disjoint held-out partition. Let $\mathcal{I} = \{\text{slack}, \text{diversity}, \text{accessibility}\}$ denote the three targeted intervention families. We consider five policies. The generic control applies the common non-targeted prompt modification. The balanced-random policy assigns the three intervention families with equal expected frequency. The *best-fixed* policy selects

$$\iota^\star = \arg\max_{\iota \in \mathcal{I}} \Delta\text{IFR}_\iota^{\text{dev}}$$

on the development set and applies $\iota^\star$ to every held-out constraint. This measures the value of selecting a generally effective intervention without constraint-specific matching.

We also construct a frequency-matched permutation control. Let $g(c) \in \mathcal{I}$ denote the intervention selected by the difficulty-guided policy for constraint c, and let $n_\iota = \sum_c \mathbf{1}[g(c) = \iota]$ denote its intervention counts. For each permutation replicate b, the same multiset of assignments is randomly permuted across constraints,

$$g_{\text{perm}}^{(b)} = \sigma_b(\{g(c)\}_{c \in \mathcal{C}}),$$

so that

$$\sum_c \mathbf{1}[g_{\text{perm}}^{(b)}(c) = \iota] = n_\iota \qquad \forall \iota \in \mathcal{I}.$$

The permutation control thus has exactly the same intervention frequencies as the guided policy and differs only in which constraints receive each intervention. We average over $10{,}000$ permutations.

All policies are evaluated on the same held-out instances and generation budget. For comparisons with the best-fixed policy, development-set selection and held-out evaluation are recomputed within each hierarchical bootstrap replicate, so that uncertainty in selecting $\iota^\star$ is included. For the frequency-matched comparison, the observed guided-policy gain is compared with the permutation distribution obtained under assignment exchangeability.

Table 23: Held-out intervention selection with stronger policy controls. Values are changes in IFR, in percentage points, relative to the original prompt. The frequency-matched permutation preserves exactly the intervention frequencies selected by the metric-guided policy.

Selection policy	OpenEnded	MBPP	Scheduling	Mean
Generic control	+1.21	+1.94	+1.65	+1.60
Balanced random	+3.19	+2.71	+3.23	+3.04
Best fixed (development)	+4.51	+4.26	+3.42	+4.06
Frequency-matched permutation	+4.02	+3.83	+3.72	+3.86
Metric-guided	**+6.20**	**+5.95**	**+5.40**	**+5.85**
Guided − best fixed				$+1.79^{+1.16}_{+2.40}$
Guided − frequency-matched				$+1.99^{+1.36}_{+2.59}$

Table 23 separates the value of constraint-specific matching from two simpler explanations. Mean IFR gains are 4.06 points for best-fixed, 3.86 for frequency-matched permutation, and 5.85 for metric-guided selection. The metric-guided policy exceeds the best-fixed by 1.79 points and the frequency-matched permutation control by 1.99 points. These comparisons indicate that the measured difficulty profile is informative for matching interventions to individual constraints, beyond identifying a generally strong intervention or a favourable mixture of intervention frequencies.

E.4 Generalization to Unseen Constraint Families and External Instructions

We test whether the proposed difficulty measurements transfer to unseen constraint families within each task and to an independently constructed instruction-following benchmark

Leave-one-family-out evaluation. For each task, we hold out each of its six constraint families in turn, excluding all five numerical instantiations of that family when calibrating the difficulty-behaviour relationship and intervention-selection rule. For the held-out family, slack, solution concentration, and conditional perplexity are estimated on a calibration partition without using its IFR or intervention outcomes; IFR is evaluated on a disjoint test partition. Difficulty measurements are normalized using only the five non-held-out families. We report the Spearman correlation between the resulting adverse difficulty score

$$\mathcal{D}_t(c) = \frac{1}{3}\left[1 - \widetilde{s}(c) + \widetilde{q}(c) + \widetilde{\log e}(c)\right]$$

and held-out failure rate, and compare generic, random, and metric-guided intervention selection. Uncertainty is estimated by hierarchical bootstrap while preserving family exclusion.

External IF Evaluation. We additionally evaluate on IFEval, pre-selecting verifier-compatible instruction types without examining model outputs. We retain instructions for which the same slack, concentration, and conditional-perplexity definitions apply without an additional model-based verifier. Normalization and intervention-selection parameters are learned only from the three controlled tasks and frozen before IFEval evaluation.

As shown in Table 24, difficulty remains associated with failure for held-out families ($\rho = 0.52$ on average), while metric-guided intervention selection improves IFR by 4.98 points and exceeds random assignment by 2.11 points. On IFEval, the corresponding association is $\rho = 0.49$, and guided selection exceeds random assignment by 2.80 points. These results suggest that the measurements generalize beyond unseen instances or thresholds, retaining predictive and mitigation value for unseen constraint families and, in this controlled external evaluation, instructions from a separately constructed benchmark.

Table 24: Generalization to unseen constraint families and IFEval. Intervention columns report IFR changes (percentage points) relative to the original prompt; side-numbers denote 95% confidence intervals. Controlled-task rows aggregate six leave-one-family-out folds.

Evaluation	$\rho(\mathcal{D}, 1-\text{IFR})$	Generic	Random	Guided	Guided − Random
OpenEnded	$0.54_{0.62}^{0.46}$	$+1.40$	$+3.10$	$+5.22$	$+2.12_{2.66}^{1.59}$
MBPP	$0.55_{0.61}^{0.50}$	$+1.61$	$+2.74$	$+4.81$	$+2.07_{2.88}^{1.26}$
Scheduling	$0.47_{0.54}^{0.40}$	$+1.62$	$+2.76$	$+4.90$	$+2.13_{2.81}^{1.45}$
Family-holdout mean	$\mathbf{0.52_{0.56}^{0.49}}$	$\mathbf{+1.54}$	$\mathbf{+2.87}$	$\mathbf{+4.98}$	$\mathbf{+2.11_{2.41}^{1.80}}$
IFEval	$0.49_{0.55}^{0.43}$	$+1.40$	$+2.35$	$+5.15$	$+2.80_{3.54}^{2.06}$

E.5 Task-Specific Solution Distances and Duplicate Thresholds

The distance functions are chosen to reflect the natural notion of solution similarity in each output space.

1. **OpenEnded** responses are compared at the sentence level. We segment responses with the NLTK sentence tokenizer and encode sentences using "sentence-transformers/all-MiniLM-L6-v2" with L2-normalized embeddings. For each response pair, we compute pairwise cosine similarities and retain, for each sentence, its maximum similarity to any sentence in the other response. We average these maxima in both directions and take their harmonic mean as an order-invariant similarity score; distance is one minus this score. This captures paraphrasing, sentence reordering, and segmentation differences while allowing different response segments to match independently.
2. **MBPP** distance is computed from Python abstract syntax trees (ASTs). Each program is parsed into an ordered tree labelled by syntactic node type, omitting identifier names and literal values so that renaming alone does not distinguish solutions. We compute unit-cost tree edit distance, with insertions, deletions, and substitutions each counting as one edit, and normalise by the sum of the two tree sizes to obtain a value between zero and one. This representation emphasises program structure over surface-form differences. Because the analysis uses task-correct generations, all programs are parseable.
3. **Scheduling** outputs are numerical assignments in a two-dimensional decision space. Their distance is the ordinary Euclidean distance between the two returned assignments. This directly reflects the geometry of the scheduling feasible region and therefore does not require an additional representation or learned similarity model.

Threshold Calibration and Solution Modes Each task uses a task-specific threshold τ_t, with outputs linked when their pairwise distance is strictly below τ_t. We calibrate thresholds on held-out output pairs labelled as duplicate or distinct solutions, then fix them for all reported analyses; calibration examples are excluded from experimental results. The resulting thresholds for OpenEnded, MBPP, and Scheduling are 0.3, 10^{-3}, and 10^{-5}, respectively. We define solution modes as connected components of the graph whose edges satisfy $d_t(y_j, y_k) < \tau_t$, and use their cardinalities as frequencies in the Simpson concentration calculation defined in the main text.

E.6 Pairwise Interaction Metrics and Regime Assignment

For each constraint pair (c_i, c_j), we evaluate three prompting conditions: c_i alone, c_j alone, and the joint condition c_i, c_j. Let S_i, S_j, and S_{ij} denote the corresponding retained task-correct feasible generations. For each task instance, outputs are rarefied and partitioned into solution modes using d_t and τ_t from Appendix E.5. For condition $a \in i, j, ij$, let $K_\ell^{(a)}$ denote the number of solution modes for task instance ℓ, and $n_{\ell k}^{(a)}$ the number of retained outputs assigned to mode k.

The finite-sample Simpson concentration is

$$q_{t,\ell}(S_a) = \frac{\sum_{k=1}^{K_\ell^{(a)}} n_{\ell k}^{(a)} \left(n_{\ell k}^{(a)} - 1 \right)}{n^\star (n^\star - 1)}.$$

Higher $q_{t,\ell}$ indicates greater concentration. We use the corresponding Simpson diversity $U_{t,\ell}(S_a) = 1 - q_{t,\ell}(S_a)$, where larger values indicate greater solution-mode diversity. We average diversity uniformly over task instances:

$$U_t(S_a) = \frac{1}{N_t} \sum_{\ell=1}^{N_t} U_{t,\ell}(S_a).$$

Because U_t is a normalized per-condition index, we use the equal-weight mean rather than the sum of marginal diversities:

$$U_t^{\text{marg}}(i,j) = \frac{U_t(S_i) + U_t(S_j)}{2}.$$

We then define the interaction ratio as

$$IR_{ij} = \frac{U_t(S_{ij})}{U_t^{\text{marg}}(i,j)} = \frac{2U_t(S_{ij})}{U_t(S_i) + U_t(S_j)}.$$

Thus, $IR_{ij} = 1$ indicates preservation of average marginal solution-mode diversity, while values below/above 1 indicate contraction/expansion under joint prompting. We use IR only as a secondary solution-space diagnostic; interaction regimes are defined independently using the CMS and JIP introduced in Section 3.3.

Interaction Regime Assignment We assign interaction regimes using fixed practical-equivalence tolerances. With rates on the $[0,1]$ scale, we set $\epsilon_{\text{CMS}} = 0.025$ and $\epsilon_{\text{JIP}} = 0.010$, corresponding to 2.5 and 1.0 percentage points. A pair is classified as *additive* when

$$|\text{CMS}_{ij}| \le \epsilon_{\text{CMS}} \quad \text{and} \quad |\text{JIP}_{ij}| \le \epsilon_{\text{JIP}},$$

with boundary values included. Outside this region, pairs are classified as *overlap* when $\text{CMS}_{ij} > 0$ and $\text{JIP}_{ij} < 0$, and as *coupled* when $\text{CMS}_{ij} < 0$ and $\text{JIP}_{ij} > 0$; all other directional combinations are retained as *mixed/unclassified*. The tolerances represent small effects rather than being optimised for downstream interaction-ratio separation. In pilot analysis, varying the CMS and JIP tolerances over approximately 2–3 and 0.75–1.5 percentage points, respectively, changed assignments only near regime boundaries and did not materially alter the qualitative ordering of regime-level interaction-ratio results. We therefore fixed the above values for the reported analyses.

E.7 Task-Specific Slack Normalization

For a constraint c with scalar output statistic $g_t(y; x_i)$, boundary b_c, and inequality direction $\eta_c \in \{-1, +1\}$, define the oriented margin

$$\text{mar}_{t,i,c}(y) = \eta_c\big(g_t(y; x_i) - b_c\big).$$

For MBPP and OpenEnded, the statistics used here depend only on the generated output, so we write $g_t(y; x_i) = g_t(y)$. Since the individual-difficulty constraints are upper bounds on non-negative count statistics,

$$Z_{t,i,c} = b_c, \qquad s_t(y; x_i, c) = \frac{b_c - g_t(y)}{b_c}.$$

For Scheduling, let $F_t^{(0)}(x_i)$ denote the feasible region induced by the underlying task instance x_i before the experimental constraint c is added. With $g_t(y; x_i) = a^\top \hat{\mathbf{u}}$, we define

$$Z_{t,i,c} = \max_{z \in F_t^{(0)}(x_i)} \eta_c(a^\top z - b_c).$$

Thus, $\text{mar}_{t,i,c}(y) \ge 0$ for every retained feasible output and $\text{mar}_{t,i,c}(y) = 0$ at the constraint boundary. Slack is the dimensionless relative margin

$$s_t(y; x_i, c) = \frac{\text{mar}_{t,i,c}(y)}{Z_{t,i,c}}.$$

The normalization $Z_{t,i,c}$ is task-specific. For MBPP and OpenEnded, the individual-difficulty constraints are upper bounds on non-negative count statistics, so

$$Z_{t,i,c} = b_c, \qquad s_t(y; x_i, c) = \frac{b_c - g_t(y)}{b_c}.$$

For Scheduling, where $g_t(y; x_i) = a^\top \hat{\mathbf{u}}$, we normalize by the largest attainable margin within the base feasible region $F_t^{(0)}(x_i)$ before imposing the experimental constraint:

$$Z_{t,i,c} = \max_{z \in F_t^{(0)}(x_i)} \eta_c\big(a^\top z - b_c\big)\,.$$

This quantity is computed directly by linear programming. The resulting slack is invariant to the units and positive rescaling of the linear constraint and remains well-defined when $b_c < 0$. In the degenerate case $Z_{t,i,c} = 0$, we set $s_t(y; x_i, c) = 0$. Constraint- and dataset-level slack are then obtained by the averaging in Equation 2, with no further normalization.

E.8 Model-Balanced Conditional Accessibility

Conditional accessibility measures how probable successful outputs are under the model that generated them. For a feasible response $y = (y_1, \ldots, y_{\mathrm{n_t oks(y)}})$ generated by model m, we compute response-token-normalized negative log-likelihood

$$\ell_m(y \mid x_i, c) = -\frac{1}{\mathrm{n_t oks(y)}} \sum_{r=1}^{\mathrm{n_t oks(y)}} \log p_m(y_r \mid \pi_t(x_i, c), y_{<r}),$$

where prompt and template tokens are used only as conditioning context. Token normalisation controls for response length while retaining the intended model-conditioned likelihood interpretation. Let $\mathcal{R}_{t,i}^{(m)}(c)$ denote the retained feasible generations produced by model m. We first average the response-token-normalised NLL within each model,

$$\bar{\ell}_{t,i}^{(m)}(c) = \frac{1}{\left|\mathcal{R}_{t,i}^{(m)}(c)\right|} \sum_{y \in \mathcal{R}_{t,i}^{(m)}(c)} \ell_m(y \mid x_i, c),$$

and then average equally across models before exponentiating:

$$e_{t,i}(c) = \exp\left(\frac{1}{M} \sum_{m=1}^{M} \bar{\ell}_{t,i}^{(m)}(c)\right), \qquad \bar{e}_t(c) = \frac{1}{N_t} \sum_{i=1}^{N_t} e_{t,i}(c).$$

Thus, each model contributes equally to conditional accessibility regardless of how many feasible generations it produces. In the realised rarefied samples, every model contributed at least one retained observation, so all model-wise means were defined

E.9 Infeasibility Gap and Target-Count Attainment

For the infeasibility gap, let $z_t(y)$ denote the scalar quantity providing the contradiction certificate, with the constituent requirements implying

$$z_t(y) \geq L(C), \qquad z_t(y) \leq U(C),$$

where $L(C) > U(C)$. We define the infeasibility gap as

$$\mathrm{IG}_t(C) = L(C) - U(C) > 0. \tag{16}$$

Thus, IG is the numerical separation between the incompatible bounds and is fixed by construction before model generation. For Scheduling, let $\hat{\mathbf{u}}$ denote the decision-variable assignment decoded from the generated output y. We then set

$$z_t(y) = a^\top \hat{\mathbf{u}}, \qquad L = \xi + \zeta, \qquad U = \xi - \zeta,$$

giving $\mathrm{IG} = 2\zeta$. For MBPP and OpenEnded, L is the deterministic lower bound implied by the constituent structural requirements and U the corresponding upper bound in the constructed contradiction. Absolute IG values are therefore interpreted within their task-specific units; the analyses use the fixed low, medium, and high terciles rather than comparing raw gaps across tasks.

For best-satisfaction rate, let $g_t(y) \in \mathbb{N}_0$ be its verified count and let the instruction require $g_t(y) > K$. Because counts are integer-valued, the smallest satisfying value is $b = K+1$. The per-generation attainment is

$$A_t(y; K) = \min\left\{1,\ \frac{g_t(y)}{K+1}\right\}. \tag{17}$$

For each experimental condition h, let K_h denote the count threshold K specified by that condition, and let $\mathcal{Y}^{\mathrm{tc}}_{t,h}$ denote the retained task-correct generations under condition h. This notation distinguishes these generations from the rarefied feasible sets $\mathcal{R}_{t,i}(c)$ used in the feasible-regime analyses. We report

$$\mathrm{TCA}_{t,h} = 100\,\frac{1}{\left|\mathcal{Y}^{\mathrm{tc}}_{t,h}\right|}\sum_{y\in\mathcal{Y}^{\mathrm{tc}}_{t,h}} A_t(y;K_h). \tag{18}$$

Aggregation otherwise follows the generation and task-correctness filtering used throughout the experiment. TCA is reported only for MBPP and OPENENDED, and is undefined for SCHEDULING.

F SCOPE AND INTERPRETATION OF DIFFICULTY MEASURES

Solution-space concentration. This term denotes concentration among observed successful model generations, not concentration or volume of the full mathematical feasible region. We do not sample uniformly from the feasible space. For each condition, we retain task-correct generations that satisfy the constraint, cluster those sampled generations into task-specific solution modes, and compute finite-sample Simpson concentration over the resulting empirical mode frequencies. The quantity is therefore model- and sampling-conditioned and may omit feasible solutions that the evaluated generators rarely or never produce.

Conditional accessibility. This term likewise denotes a model-conditioned likelihood proxy rather than global accessibility of the feasible space. It is computed from response-token NLL/perplexity of sampled feasible generations under the model that generated them, conditioned on the prompt. It measures how probable the observed successful outputs are under the evaluated prompted model distribution; it does not estimate the total probability mass, reachability, or accessibility of all mathematically feasible outputs.

Capacity The easy, medium, and hard capacity pools use numerically graded versions of related constraints, so this experiment is not intended as an independent test that the proposed difficulty measurements outperform raw numerical restrictiveness or baseline single-constraint IFR. Its scoped to test whether constraint count alone determines observed follow rate under a controlled variation in constituent difficulty. Independent evidence that slack, concentration, and conditional perplexity contain predictive information beyond the exact numerical boundary and an empirical baseline success rate is provided by the fixed-boundary and baseline-controlled analyses in Appendix E.1.

Empirical Scope and External Validity Our experimental design prioritises control and sampling depth over empirical breadth. Holding task content fixed enables cleaner comparisons across constraint manipulations and settings, while repeated generations provide the sampling depth needed to estimate the proposed distributional quantities. Expanding the number of instances and task families would provide additional coverage, but would not by itself support population-level claims over the broader space of tasks and instructions without an explicit sampling framework. Moreover, reliable estimation of distributional quantities requires repeated generations within each condition, so under a fixed compute budget, greater breadth trades directly against estimation precision and the strength of matched comparisons. We therefore prioritise structural diversity across task regimes and sampling depth within conditions rather than task-family count alone. Accordingly, we do not establish population-level effect sizes or universal generality across arbitrary domains or natural interactions. Instead, the same qualitative relationships recur across three task structures, multiple models and prompts, matched interventions, independent generation streams, and leave-one-model-out transfer. We interpret this as controlled evidence that difficulty-aware characterisation is reproducible across the evaluated settings and warrants broader external validation.