How Prompt Structural Framing and Cognitive Scaffolding Influence Performance in Generative AI Design?

Yitian Huang [1] **Caishun Chen** [2 3] **Jian Cheng Wong** [2] **Yew-Soon Ong** [1 3]

[1]*College of Computing and Data Science, Nanyang Technological University, Singapore 639798* [2]*Institute of High Performance Computing, A*STAR, Singapore 138632* [3]*Centre for Frontier AI Research, A*STAR, Singapore 138632*. Correspondence to: Yitian Huang d240013@e.ntu.edu.sg.

1. Introduction

Text-to-3D generative models enable rapid structural prototyping via natural language. However, while prompting strategies heavily influence logical reasoning and text-to-image generation, their impact on generative AI design systems remains relatively unexplored. This abstract presents an empirical study to evaluate the effects of *prompt structural framing* and *cognitive scaffolding* with a *closed-loop generative design framework*, where generated outputs are continuously evaluated and fed back into the system to refine results. We investigate how prompt architectures affect text-to-3D generation by iteratively testing diverse prompt structures and reasoning strategies.

2. Related Work

Recent frameworks like LLM2TEA [1] demonstrate agentic AI's efficacy in closed-loop generative design. However, in text-to-3D instantiation, models like Shap-E [2] offer rapid prototyping but lack inherent physical awareness. Consequently, methodologies to control physical parameters like the drag coefficient (C_D) during generation remain lacking.

Concurrently, while existing prompt taxonomies benchmark complex tasks [3], classify aesthetic modifiers [4], and assess structural formatting in coding [5], their application to quantifiable physical performance in text-to-3D engineering remains unmapped. This study addresses this lack of physical awareness by evaluating how prompt architectures dictate generative physical and semantic outcomes.

3. Methodology

Our methodology utilizes a closed-loop generative design framework through a three-phase pipeline (Figure 1). First, *LLM prompts* are constructed using structural framing and cognitive scaffolding to direct an LLM to generate *3D prompts*. Second, Shap-E executes these to synthesize 3D geometries for physical and semantic evaluation. Finally, evaluation results guide LLM prompt refinement for subsequent rounds. Configuration details are in Appendix A.

3.1 Prompt Taxonomy and Formats

To systematically formulate the initial LLM prompts, we define 20 baseline strategies by combining five prompting strategies with four output formats (the detailed matrix is provided in Appendix B):

- **Structural Framing (User-Based):** Isolates the provided task context via *Zero-Shot Directives* [6], *Few-Shot Exemplars* [7], or *Explicit-Constraints* [5].
- **Cognitive Scaffolding (Model-Based):** Triggers specialized internal reasoning via *Role-Conditioned Personas* [8] or *Chain-of-Thought (CoT)* sequential logic [9].
- **Output Prompt Format:** These taxonomies are strictly formatted into *Plain Text, Markdown, YAML,* or *JSON* to test machine-readability impacts.

For each of the 20 combinations, $N = 100$ geometry models are generated via Shap-E. A surrogate model (MAE= 0.029) calculates the C_D, while a 3D classifier evaluates semantic fidelity.

3.2 Evolutionary Refinement Scenarios

As illustrated in the pipeline flowchart (Figure 1), the initial evaluation scores (C_D and semantics) are fed back into the LLM to autonomously refine its 3D prompt generation across subsequent rounds. We test two specific scenarios:

1. **Single-Objective:** The LLM is instructed to minimize C_D.
2. **Multi-Objective:** The LLM navigates the Pareto front [10], attempting to simultaneously minimize C_D and maximize the semantic classifier score.

See Appendix C for prompt exemplars, and Appendix D for prompts for refinement.

4. Results and Discussion

4.1 Empirical Validation of Prompt Influence

We statistically verified that prompt architecture causally dictates the diffusion model's physical outputs, overcoming generative stochasticity. Round 1 was executed 5 times (20 combinations $\times$ 20 prompts), yielding 2,000 generated models. A Kruskal-Wallis H test [11] on these C_D scores confirmed prompt taxonomy globally governs aerodynamic performance ($p = 1.45 \times 10^{-6}$).

A Two-Way ANOVA on rank-transformed C_D scores [12] revealed cognitive scaffolding *Type* is a dominant factor ($F = 6.72$, $p = 2.2 \times 10^{-5}$), whereas *Format* alone is insignificant ($p = 0.1467$). Crucially, a significant Type-Format interaction ($F = 2.65$, $p = 0.00158$) demonstrates machine-readable formats (e.g., JSON) do not inherently improve designs. Instead, they act as structural catalysts, enforcing physics-based reasoning generated by advanced taxonomies.

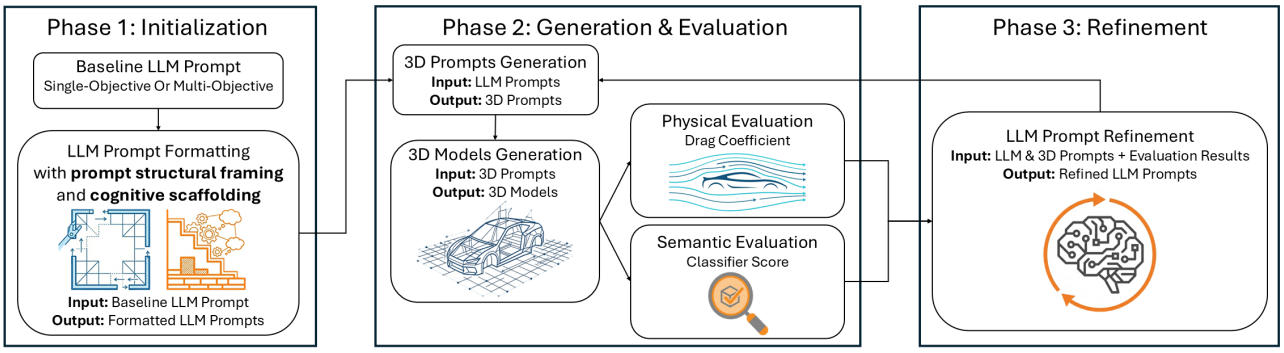


Fig. 1: Pipeline flowchart of the closed-loop generative design framework.

4.2 *Optimal Strategies and Pareto Yield*

Next, we evaluated strategies using **median** C_D and **Pass Rate** (models achieving semantic classifier score > 0.6). Table 1 compares peak Round 1 performance against the **Zero-Shot Plain Text** baseline. Full dataset: Appendix E.

Table 1: Peak Strategy Performance ($N = 100$ samples per condition)

Scenario & Strategy	**Med.** C_D	**Pass Rate**
Single-Objective		
Baseline (Zero-Shot Text)	0.2081	55%
Winner (Few-Shot JSON)	0.1915	55%
Multi-Objective		
Baseline (Zero-Shot Text)	0.2081	53%
Winner (Role-Conditioned Text)	0.1954	70%

Single-Objective: Constrained exclusively to minimize C_D, **Few-Shot JSON** achieved a statistically significant reduction in median C_D versus the baseline (Mann-Whitney [13] $p = 0.0086$).

Multi-Objective: To identify the optimal Pareto strategy balancing aerodynamics and semantic fidelity, we computed an aggregated robust Z-score ($Z_{total} = Z_{pass\_rate} - Z_{drag}$) [14]. **Role-Conditioned Text** maximized this metric, significantly increasing viable geometry Pass Rate from 53% to 70% (χ^2 $p = 0.020$) over the baseline. Although its 6.1% median C_D reduction was not statistically significant ($p = 0.200$), it proved optimal by substantially boosting viable yield without compromising aerodynamics.

4.3 *Discovering Generative Limits: Objective Trade-offs and Diminishing Returns*

Trajectory analysis reveals non-linear optimization with sharp objective trade-offs (Figure 2). Peak C_D minimization universally occurs in Round 2, demonstrating the LLM's spatial-physical feedback processing capability. However, this severely costs semantic fidelity: as median C_D improves, Pass Rate drops precipitously.

By Round 3, optimization plateaus, yielding marginal C_D rebounds and flat Pass Rates. This slight regression may indicate "prompt drift" or the agent reaching its spatial-reasoning capacity limits. While the agent optimizes a single physical metric short-term, balancing multidimensional constraints degrades semantic fidelity. Thus, additional rounds yield diminishing returns without constraint-buffering.

Appendix F (Figure A1) visually illustrates this trade-off: a Round 1 model maintains semantic fidelity but exhibits higher C_D, while its Round 2 refinement reduces C_D but fails the structural classifier.

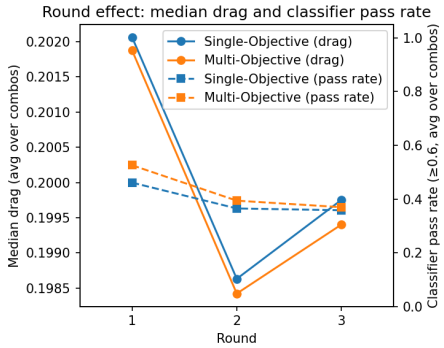


Fig. 2: Prompt drift: later-round performance degradation.

5. Conclusion

Prompt architectures significantly dictate text-to-3D generation viability. Structured formats and multi-objective cognitive scaffolding act as algorithmic regularizers, successfully navigating the Pareto front to maximize the yield of semantically viable designs without compromising baseline aerodynamics. However, pushing for deeper aerodynamic optimization through extended iterative cycles triggers severe semantic degradation and destructive prompt drift. Future work will explore memory-buffering techniques to stabilize these autonomous feedback loops.

Acknowledgments

This project was supported by Nanyang Technological University under the URECA Undergraduate Research Programme.

Declaration on the Use of LLM

LLMs were used as general-purpose tools to improve clarity and language during the writing process. All research design, experiments, analysis, and conclusions were conducted and verified by the authors, who take full responsibility for the content.

References

[1] M. Wong, J. Liu, T. Rios, S. Menzel, and Y. S. Ong. LLM2TEA: An Agentic AI Designer for Discovery With Generative Evolutionary Multitasking. *IEEE Computational Intelligence Magazine*, 20(4):42–55, 2025.

[2] H. Jun and A. Nichol. Shap-E: Generating Conditional 3D Implicit Functions. *arXiv preprint arXiv:2305.02463*, 2023.

[3] S.K.K. Santu and D. Feng. Teler: A general taxonomy of LLM prompts for benchmarking complex tasks. *arXiv preprint arXiv:2305.11430*, 2023.

[4] J. Oppenlaender. A taxonomy of prompt modifiers for text-to-image generation. *Behaviour & Information Technology*, 43(15):3763–3776, 2024.

[5] J. He, M. Rungta, D. Koleczek, A. Sekhon, F.X. Wang, and S. Hasan. Does prompt formatting have any impact on LLM performance? *arXiv preprint arXiv:2411.10541*, 2024.

[6] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, D. Amodei, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[8] A. Kong, S. Zhao, H. Chen, Q. Li, Y. Qin, R. Sun, and X. Dong. Better zero-shot reasoning with role-play prompting. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4099–4113, 2024.

[9] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[10] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[11] William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.

[12] William J Conover and Ronald L Iman. Rank transformations as a bridge between parametric and nonparametric statistics. *The American Statistician*, 35(3):124–129, 1981.

[13] Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

[14] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013.

[15] OpenAI. GPT-4 Technical Report, 2024.

[16] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[17] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *International Conference on Machine Learning*, pages 19730–19742. PMLR, 2023.

[18] Dongxu Li, Junnan Li, Hung Le, Guangsen Wang, Silvio Savarese, and Steven CH Hoi. LAVIS: A library for language-vision intelligence. *arXiv preprint arXiv:2209.09019*, 2022.

Appendix A. Experimental Configuration

The generative pipeline was executed using the following architectural components and hyperparameters to ensure rigorous reproducibility:

- **Large Language Model (LLM):** The iterative prompt refinement was driven by **GPT-4o-mini** [15] (accessed via an OpenAI-compatible API). Generation queries were executed with a batch size of 10 and a maximum of 3 retries to handle API volatility while analyzing historical performance data.
- **Text-to-3D Generator: Shap-E** [2] was utilized to instantiate the physical 3D geometries, implemented via the Hugging Face `diffusers` library (`ShapEPipeline`). Inference parameters were strictly standardized across all generations:

64 inference steps, a guidance scale of 15, and an output frame size of 256.

- **Aerodynamic Surrogate Evaluator:** A pre-trained two-branch Convolutional Neural Network (CNN) surrogate was employed. It utilizes a ResNeXt-101-32×8d backbone [16] (TorchVision) to process rendered depth and normal images (384×384) of the 3D meshes, fused via cross-attention to a fully connected regression head (size 128). This surrogate calculates the expected drag coefficient (C_D) at a configured freestream velocity of 20 (Mean Absolute Error: 0.029).

- **Semantic Classifier (Visual Quality):** Topological fidelity was evaluated using **BLIP-2** [17] for image-text matching, implemented via the LAVIS library [18] (`blip2_image_text_matching`, COCO model type). The classifier processes multi-view renderings of the 3D mesh, scoring the visual match against the target label ("A car"). A strict threshold of ≥ 0.6 (aggregated from the maximum view score) defined the topological pass rate used in our Pareto yield calculations.

- **Software Frameworks:** The overarching automated environment was constructed utilizing PyTorch, `omegaconf` for configuration, `transformers`, and `trimesh` for 3D boundary processing and rendering.

Appendix B. Detailed Matrix of Prompt Taxonomy and Formats

To systematically formulate the baseline LLM prompts, we defined 20 baseline strategies by combining five taxonomic types with four output formats.

2.1 Prompt Taxonomy

Structural Framing (User-Based) These categories isolate the syntax and reference data provided:

- **Zero-Shot Directive:** Baseline task-oriented instructions [6].

- **Few-Shot Exemplar:** Replicating patterns from successful prior designs to simulate evolutionary crossover [7].

- **Constraint-Explicit:** Enforcing machine-readability and physical boundaries through rigid output instructions [5].

Cognitive Scaffolding (Model-Based) These categories attempt to trigger specialized internal reasoning pathways:

- **Role-Conditioned (Persona):** Shifting the model's latent space toward engineering-specific tokens [8].

- **Chain-of-Thought (CoT):** Explicitly activating the model's sequential reasoning capabilities to calculate physical implications before finalizing geometry [9].

2.2 Prompt Formats

- **Plain Text:** Standard natural language formatting.

- **Markdown:** Hierarchical structure using headers and lists.

- **YAML:** Data-serialization format focused on readability.

- **JSON:** Rigid key-value structure for machine-readability.

Appendix C. Prompt Architecture Exemplars

This section provides exact instances of applying structural framing and cognitive scaffolding to format LLM prompts. Below are representative frameworks of how instructions were passed to the LLM for 3D prompt generation.

Exemplar 1: Single-Objective Baseline (Zero-Shot Text)

> *Generate {batchsize} 3D car design prompts with the objective of minimizing the drag coefficient. Step 1: Each prompt must start with the phrase 'A car in the shape of' and end with a full stop. Step 2: Predict the car drag coefficient (0-1) to exactly four decimal places for each generated 3D car object, ensuring the values reflect the goal of minimizing drag. Step 3: Append your predicted drag coefficient immediately before the prompt text. Show only the combined outputs and do not show your intermediate steps.*

Exemplar 2: Single-Objective (Few-shot JSON)

```
{
  "task": "
     few_shot_3d_car_prompt_generation",
  "batchsize": "{batchsize}",
  "objective": "Minimize the drag
     coefficient",
  "examples": [
    {"output": "0.1905 A car in the
       shape of an arrow."}
  ],
  "requirements": [
    "Strictly follow the example format
       for a 3D car object.",
    "Prompt must start with 'A car in
       the shape of' and end with '.'",
    "Predict drag coefficient between 0
       and 1 (exactly 4 decimal places,
       optimized for low drag).",
```

```
    "Append coefficient before the
        prompt text.",
    "No intermediate outputs."
  ]
}
```

Exemplar 3: Single-Objective (Constraint-Explicit Markdown)

```
### Strict Boundary Conditions (3D Car
    Object)
You are required to output {batchsize}
    lines of data for 3D car objects.
* Boundary 1 (Objective): Designs must
    minimize the drag coefficient.
* Boundary 2 (Prompt): Must start with "
    A car in the shape of" and end with a
     full stop.
* Boundary 3 (Value): 0.0000 <= Cd <=
    1.0000 (Exactly 4 decimal places).
* Boundary 4 (Ordering): You must append
     the Cd before the prompt.
* Boundary 5 (Syntax): Text must be: `[
    Cd value] A car in the shape of [
    descriptor].`
* Boundary 6 (Visibility): Intermediate
    steps must not be rendered. Output
    final list only.
```

Exemplar 4: Multi-Objective Baseline (Zero-Shot Text)

Generate {batchsize} 3D car design prompts with the dual objective of minimizing the drag coefficient and maximizing the classifier score. Step 1: Each prompt must start with the phrase 'A car in the shape of' and end with a full stop. Step 2: Predict the car drag coefficient (0-1, 4 decimal places). Step 3: Predict the classifier score (0-1, 4 decimal places), representing how recognizable the object is. Step 4: Append both the predicted drag coefficient and the classifier score before the prompt text, separated by a space. Show only the final combined outputs and do not show your intermediate steps.

Exemplar 5: Multi-Objective (Role-Conditioned Text)

Act as a senior automotive aerodynamics engineer and computer vision specialist. Your task is to conceptualize {batchsize} 3D car objects that minimize drag while maximizing classifier recognition. For each concept, write a prompt starting with 'A car in the shape of' and ending with a full stop. Predict the drag coefficient (0-1, 4 decimals) and the classifier score (0-1, 4 decimals). You must append both predicted values before the prompt. Present your final report as a simple list of '[Cd] [Score] [Prompt]', omitting all your intermediate steps.

Exemplar 6: Multi-Objective (Chain-of-Thought YAML)

```
reasoning_engine: "Chain-of-Thought"
batch_size: "{batchsize}"
objectives:
  - "Minimize drag coefficient"
  - "Maximize classifier score"
internal_steps:
  step_1: "Draft 3D car object prompt: '
      A car in the shape of [structure].'
      "
  step_2: "Predict drag coefficient and
      classifier score (both 0-1, exactly
       4 decimals)."
  step_3: "Append both values before the
       prompt."
output_control:
  instruction: "Display only the final
      combined strings."
  format: "${Cd} ${Score} ${Prompt}"
  suppress_thought_process: true
```

Appendix D. Refinement Prompts

The following are the prompts used to refine LLM prompts with Single-Objective or Multi-Objective. They are passed to the LLM along with the current round LLM prompts and historical results, including 3D prompts, real C_D, and semantic classifier scores (added for Multi-Objective only):

1. **Single-Objective:**

   You refine the instruction (LLM prompt) used to generate 3D car design prompts. Given the current LLM prompt and the 3D prompts that were generated with their drag results (lower is better), output a new, improved LLM prompt that would lead to lower-drag designs. You MUST preserve the exact same structure as the current prompt: it must still instruct the model to generate a batch (use {batchsize} if present), to predict drag coefficient (and classifier score if present), and to output each line in the format [Cd] [Score] [Prompt]. Refine only the aerodynamic/design guidance wording. Output only the refined full LLM prompt text, no preamble or explanation.

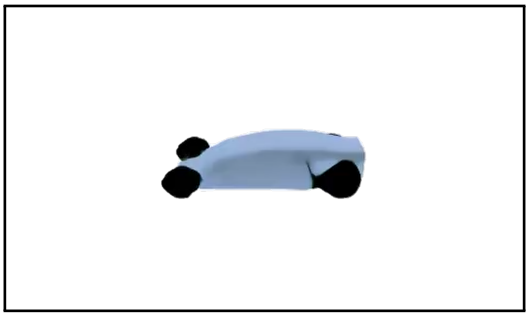
2. **Multi-Objective:**

   You refine the instruction (LLM prompt) used to generate 3D car design prompts. Given the current LLM prompt and the 3D prompts with their drag (lower is better) and classifier score (higher is better), output a new, improved LLM prompt that would lead to designs with both lower drag and higher classifier score. You

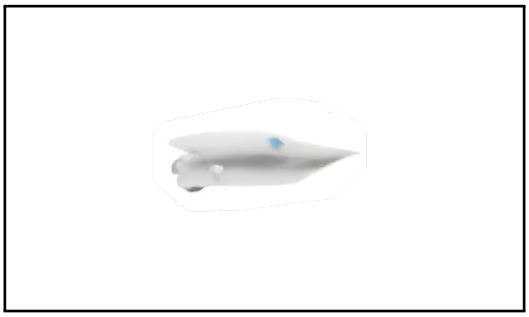
MUST preserve the exact same structure as the current prompt: it must still instruct the model to generate a batch (use {batchsize} if present), to predict drag coefficient and classifier score (0-1, 4 decimals), and to output each line in the format [Cd] [Score] [Prompt]. Refine only the aerodynamic/design guidance wording. Output only the refined full LLM prompt text, no preamble or explanation.

Appendix E. Comprehensive Generational Performance Data

To adhere to the page limits of the main abstract, only peak performance strategies were detailed in Table 1. The comprehensive, 60-result empirical dataset (derived from 100 generation samples per prompt configuration) capturing the generative latent space mapped across all three evolutionary rounds is presented in Table A1 for single-objective refinement and Table A2 for multi-objective refinement.



(a) Round 1: High Classifier Score (Pass, 0.85), Relatively High Drag (0.2106).



(b) Round 2: Low Classifier Score (Fail, 0.30), Relatively Low Drag (0.1897).

Fig. A1: Comparison of a single generation sequence demonstrating the objective trade-off. The Round 2 model (bottom) is a direct iterative refinement of the Round 1 model (top), showing drag reduction at the cost of structural viability.

Appendix F. Evolution of Generated Geometry: Round 1 vs. Round 2

Figure A1 illustrates the structural degradation that occurs when the generative design agent over-optimizes for aerodynamic performance. Both models represent the exact same prompt combination and generation lineage, highlighting the specific geometric sacrifices made during the second round of refinement to achieve a lower drag coefficient.

Table A1: Single-Objective Refinement Results: Exhaustive Generational Data (N = 100 samples per condition)

Taxonomy & Format	**R1 Med** C_D	**R1 Pass %**	**R2 Med** C_D	**R2 Pass %**	**R3 Med** C_D	**R3 Pass %**
Zero-Shot (Plain Text)	0.2081	50%	0.1961	44%	0.1969	50%
Zero-Shot (Markdown)	0.1979	48%	0.1956	43%	0.1948	32%
Zero-Shot (YAML)	0.1967	46%	0.1998	33%	0.1988	36%
Zero-Shot (JSON)	0.2030	52%	0.1997	39%	0.1961	41%
Few-Shot (Plain Text)	0.1959	40%	0.1947	38%	0.2023	40%
Few-Shot (Markdown)	0.2000	47%	0.2020	23%	0.1990	37%
Few-Shot (YAML)	0.2001	47%	0.2031	44%	0.2012	36%
Few-Shot (JSON)	0.1915	55%	0.1991	40%	0.1948	50%
Explicit-Constraint (Plain Text)	0.2005	49%	0.2027	34%	0.2068	33%
Explicit-Constraint (Markdown)	0.2003	49%	0.1976	42%	0.1973	32%
Explicit-Constraint (YAML)	0.2046	43%	0.1991	32%	0.2014	29%
Explicit-Constraint (JSON)	0.2181	48%	0.2021	32%	0.2021	32%
Role-Conditioned (Plain Text)	0.1954	42%	0.1965	31%	0.2058	28%
Role-Conditioned (Markdown)	0.1997	40%	0.1929	34%	0.2041	29%
Role-Conditioned (YAML)	0.1992	48%	0.1997	50%	0.1982	42%
Role-Conditioned (JSON)	0.2113	42%	0.1947	34%	0.2071	32%
Chain-of-Thought (Plain Text)	0.2086	37%	0.1974	45%	0.1946	40%
Chain-of-Thought (Markdown)	0.2047	51%	0.1975	37%	0.1991	30%
Chain-of-Thought (YAML)	0.2016	51%	0.1995	33%	0.2005	44%
Chain-of-Thought (JSON)	0.2018	46%	0.2032	28%	0.1961	33%

Table A2: Multi-Objective Refinement Results: Exhaustive Generational Data (N = 100 samples per condition)

Taxonomy & Format	**R1 Med** C_D	**R1 Pass %**	**R2 Med** C_D	**R2 Pass %**	**R3 Med** C_D	**R3 Pass %**
Zero-Shot (Plain Text)	0.2081	53%	0.1999	44%	0.2036	32%
Zero-Shot (Markdown)	0.1979	45%	0.1995	32%	0.2004	29%
Zero-Shot (YAML)	0.1967	48%	0.2022	32%	0.1998	26%
Zero-Shot (JSON)	0.2030	47%	0.1958	42%	0.2021	33%
Few-Shot (Plain Text)	0.1971	48%	0.1967	43%	0.1928	39%
Few-Shot (Markdown)	0.2000	53%	0.1891	53%	0.1974	45%
Few-Shot (YAML)	0.2001	55%	0.1987	43%	0.2030	37%
Few-Shot (JSON)	0.1915	63%	0.1998	45%	0.1962	37%
Explicit-Constraint (Plain Text)	0.2005	47%	0.2018	22%	0.2072	34%
Explicit-Constraint (Markdown)	0.2029	59%	0.1883	47%	0.1953	49%
Explicit-Constraint (YAML)	0.2046	59%	0.2023	39%	0.1960	31%
Explicit-Constraint (JSON)	0.2181	64%	0.2022	37%	0.2005	40%
Role-Conditioned (Plain Text)	0.1954	70%	0.1985	36%	0.1924	48%
Role-Conditioned (Markdown)	0.1997	56%	0.2006	36%	0.2092	26%
Role-Conditioned (YAML)	0.1992	55%	0.1944	47%	0.1992	47%
Role-Conditioned (JSON)	0.2113	49%	0.1997	41%	0.1914	37%
Chain-of-Thought (Plain Text)	0.2086	45%	0.1993	38%	0.1986	45%
Chain-of-Thought (Markdown)	0.2051	47%	0.1973	40%	0.1995	37%
Chain-of-Thought (YAML)	0.2016	43%	0.1996	31%	0.2021	35%
Chain-of-Thought (JSON)	0.2018	46%	0.1991	47%	0.1949	31%