SciMem: Scientific Reasoning with Structured Memory for Materials Design

Anonymous ACL submission

Abstract

Large language models are useful for materials design, but adapting them to specialized experimental domains often requires costly supervision or parameter updates. Memory-based adaptation can reuse prior experience, yet naive accumulation is brittle because synthesis effects depend on material class, target property, processing history, and measurement context. We introduce *SciMem*, a structured memory framework that keeps the pretrained LLM frozen while distilling rollout feedback into a *Graph of Experience*, which links experimental features, scientific effects, and mechanistic dependencies within disentangled scientific contexts. Across property prediction and property-conditioned synthesis planning on the Open Materials Guide and an in-house QD-LED dataset, SciMem improves task performance, mechanistic validity, and memory efficiency over prompting and unstructured-memory baselines. Wet-lab validation on QD-LED further demonstrates that SciMem can generate experimentally actionable synthesis plans, highlighting structured memory as a practical direction for LLM-based scientific reasoning and autonomous materials discovery.

1 Introduction

LLMs are increasingly being applied to scientific and engineering tasks, from materials design and drug discovery to autonomous research (Lee et al., 2025a; Lu et al., 2026; Li et al., 2026; Ahlawat et al., 2026). Their broad pretraining is useful when task-specific rules, simulators, or mechanistic models are difficult to construct (Alampara et al., 2026). However, effective domain adaptation often still depends on additional supervision, domain-adaptive fine-tuning, or curated labeled data, which can be costly or scarce (Luo et al., 2025a; Lu et al., 2025; Verma et al., 2025).

To address this challenge, recent studies have explored memory-based approaches that improve LLMs by accumulating past experience (Liang et al., 2025; Cai et al., 2025; Yan et al., 2025; Kim et al., 2026a). By avoiding parameter updates, these methods are well suited to materials science, where data are costly and experimental records are heterogeneous (Wang et al., 2025; Miret and Krishnan, 2025; Kim et al., 2025).

However, memory-based methods face distinct challenges in materials design. Unlike mathematics (Chervonyi et al., 2025; Hubert et al., 2026) or programming benchmarks (Jimenez et al., 2024; Jain et al., 2025), where past solutions can often be reused under explicit objectives and verifiable feedback, materials design involves context-dependent dependencies between experimental settings and material behavior. Suabtle changes in composition, precursors, processing, or measurement can alter the underlying mechanism (Sun and David, 2025; Lee et al., 2025b; Nwabara et al., 2025), making simple condition–outcome memories insufficient for generalizable scientific reasoning.

Motivated by these challenges, we propose *SciMem*, a structured memory framework for scientific reasoning in materials design. SciMem encodes condition-dependent relationships that govern material behaviors and properties. The framework consists of two key components: the *Graph of Experience (GoE)* and a disentangled memory update strategy.

GoE organizes past experiences as an interpretable graph of scientific effects and their dependencies, rather than relying on superficial condition–outcome correlations. This effect-based representation enables more detailed mechanistic reasoning across experimental settings.

To further support context-dependent materials reasoning, SciMem incorporates a disentangled memory update strategy that organizes experiences into scientifically coherent memory units, thereby reducing interference across heterogeneous experimental settings.

We evaluate SciMem on property prediction

and property-conditioned synthesis planning using the Open Materials Guide (OMG) dataset (Kim et al., 2025). As an inverse materials design task, property-conditioned synthesis planning has attracted increasing attention for accelerating materials discovery (Noh et al., 2026; Pan et al., 2026; Karpovich et al., 2024; Cheng et al., 2026; Kim et al., 2026b), but remains ill-posed because multiple procedures can satisfy similar target properties under sparse or incomplete supervision. We therefore use predicted properties to filter generated synthesis candidates.

Beyond public benchmark evaluation, we further validate SciMem through wet-lab application in an industrial quantum-dot light-emitting diode (QD-LED) synthesis setting. SciMem proposes novel synthesis procedures that achieve the best observed laboratory performance observed in our experimental campaign, demonstrating its practical value for real-world materials discovery.

2 Related Work

2.1 LLMs for Materials Design

LLMs have been applied to materials science tasks such as property prediction (Lei et al., 2024; Lee et al., 2025a), synthesis planning (Noh et al., 2026; Pan et al., 2026), literature-assisted research (Bran et al., 2024), and autonomous labs (Choi et al., 2025; Luo et al., 2025b). However, many resources and benchmarks still emphasize static structures or general materials knowledge, providing limited support for synthesis conditions, processing histories, and measurement contexts (Jain et al., 2013; Spotte-Smith et al., 2023; Niyongabo Rubungo et al., 2025; Wang et al., 2025; Kim et al., 2025). This is particularly limiting for industrial materials design, where behavior depends on coupled process–structure–property relationships (Peng et al., 2023) and device-level factors such as interfaces and fabrication conditions (Sood et al., 2021).

Recent datasets and models incorporate synthesis recipes and processing conditions (Kononova et al., 2019; Kim et al., 2025; Noh et al., 2026; Pan et al., 2026), while inverse design methods generate materials or synthesis procedures from desired objectives (Noh et al., 2026; Pan et al., 2026; Karpovich et al., 2024; Cheng et al., 2026; Kim et al., 2026b). Nevertheless, many methods start from predefined materials or structures, or generate synthesis text without modeling how experimental conditions affect target properties. In contrast, our approach infers synthesis choices directly from desired properties, requiring reasoning over conditions, mechanisms, and outcomes. Although prior work has represented synthesis knowledge with action graphs (Mysore et al., 2017; Kuniyoshi et al., 2020) and provenance graphs (Tsuruta and Kumagai, 2025), these representations are mainly used for extraction or organization. SciMem instead uses structured memory updated from past experimental experience for materials reasoning.

2.2 Memory-Enhanced Reasoning in LLMs

Memory-based approaches improve LLM reasoning by updating memory from past experience without additional parameter updates. Prior work has explored retrieval-based memory (Lewis et al., 2020; Borgeaud et al., 2022), trajectory reuse (Wang et al., 2022), continual experience accumulation (Reed et al., 2022), reflection-based guidance (Park et al., 2023; Shinn et al., 2023; Madaan et al., 2023), and optimization-based methods (Liang et al., 2025; Cai et al., 2025). These approaches are especially attractive in scientific domains where labeled data, expert supervision, and experimental feedback are costly.

In materials design, however, the usefulness of past experience depends strongly on context, including the material system, synthesis route, target property, processing condition, measurement protocol, and device configuration. Simply accumulating trajectories, reflections, or condition–outcome pairs can therefore introduce redundancy, context overhead, and negative transfer as memory grows. Although abstract memory representations can improve transferability over raw traces (Kim et al., 2026a), materials reasoning also requires preserving the scientific context in which an experience is valid. SciMem addresses this need by storing experiences as structured scientific effects and updating them within context-specific memory units, enabling more interpretable and mechanism-aware reasoning.

3 Method

3.1 Task Formulation

We consider two complementary scientific reasoning tasks: property prediction and property-conditioned synthesis planning.

A complete materials recipe is denoted by

$$x = \left(x^{\mathrm{mat}}, x^{\mathrm{proc}}\right), \tag{1}$$

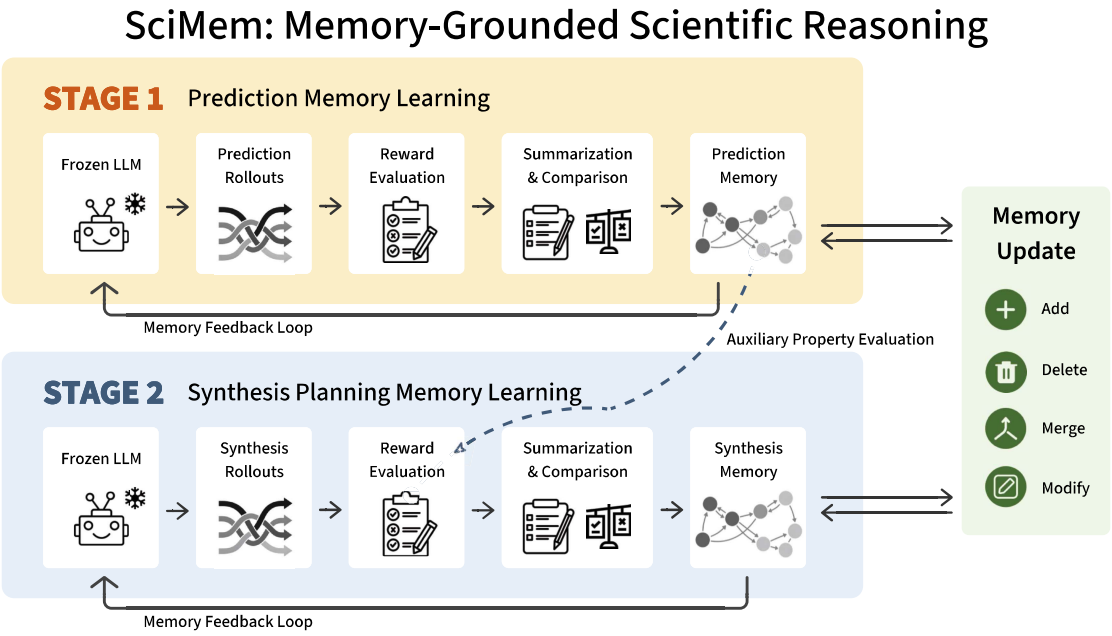

Figure 1: Overview of SciMem. For each query, grouped candidate generations are evaluated and converted into Graph-of-Experience through editable operations. At inference time, the accumulated memory is serialized into the prompt context to support memory-grounded scientific reasoning.

where x^{mat} represents structured experimental context, including materials, precursors, equipment, and characterization methods, and x^{proc} denotes the synthesis procedure.

Scientific reasoning often builds on relevant prior cases and experimental context. Following the common practice of adapting prior examples to new problems, SciMem retrieves a single reference synthesis example as contextual support. We define the reference for query q as

$$\mathcal{R}_q = (x_r, y_r), \tag{2}$$

where $x_r = (x_r^{\text{mat}}, x_r^{\text{proc}})$ denotes the reference recipe and y_r its measured property.

Let f_{θ^*} denote a pretrained LLM whose parameters remain frozen throughout training. We use y for measured properties, $\hat{y}$ for model predictions, and $\bar{y}$ for desired target properties.

In property prediction, the model predicts a target property $\hat{y}$ from a complete recipe:

$$\hat{y} = f_{\theta^*}(\mathcal{R}_q, x_q), \tag{3}$$

where $x_q = (x_q^{\text{mat}}, x_q^{\text{proc}})$ denotes the query recipe.

In synthesis planning, the model generates a synthesis procedure conditioned on partial experimental context and a desired property:

$$\hat{x}_q^{\text{proc}} = f_{\theta^*}\left(\mathcal{R}_q, x_q^{\text{mat}}, \bar{y}\right). \tag{4}$$

Unlike property prediction, synthesis planning admits multiple valid procedural solutions, making exact output matching insufficient for experience accumulation.

3.2 SciMem

Overview. Figure 1 illustrates the overall SciMem framework. SciMem learns editable external memories while keeping the pretrained LLM frozen.

SciMem maintains separate memories for property prediction and synthesis planning, denoted by M^{pred} and M^{plan}, respectively. Training proceeds in two stages. In Stage 1, SciMem learns M^{pred} from property-prediction tasks and then freezes it. In Stage 2, SciMem learns M^{plan} from synthesis-planning tasks using both planning feedback and auxiliary property signals computed with the frozen prediction memory.

Rather than directly updating model parameters, SciMem constructs memory through iterative consolidation of evaluated model responses. The resulting memory is represented using Graph-of-Experience (GoE), a structured textual representation that organizes reusable reasoning abstractions through localized transformations and their dependencies.

Training. In this work, we use the term training phase to refer to memory construction, while the

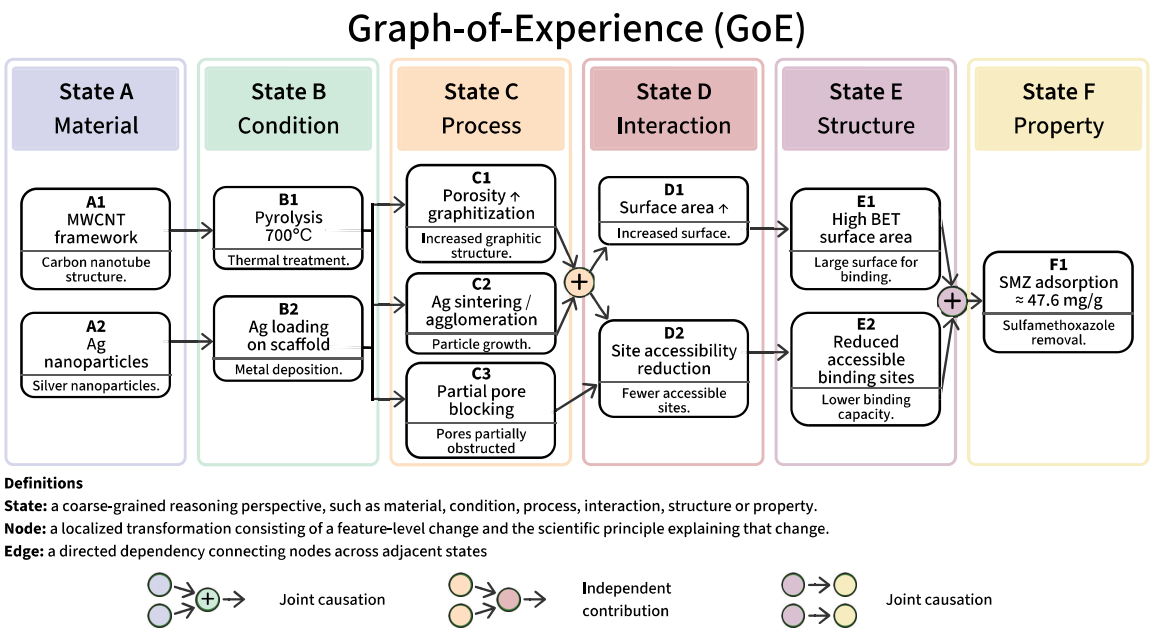

Figure 2: Illustration of the Graph-of-Experience (GoE) representation used in SciMem. Nodes represent state-specific reasoning transformations, while edges capture dependencies across material context, synthesis conditions, process implications, causal interactions, structural characteristics, and target-property outcomes.

backbone LLM parameters remain frozen.

For each task $\tau \in \{\text{pred}, \text{plan}\}$, let P_q^τ denote the task prompt for query q, and let M^τ denote the corresponding task memory. SciMem samples G candidate responses, and each response is represented as $o_{q,g}^\tau$:

$$o_{q,g}^\tau \sim f_{\theta^*}(P_q^\tau, M^\tau), \qquad g \in \{1, \ldots, G\}. \quad (5)$$

Each response contains a task output together with a rationale describing the associated scientific reasoning. The rationale provides an explicit explanation of intermediate reasoning steps, enabling SciMem to abstract procedural and causal patterns beyond final task outputs.

The property prediction reward is derived by comparing the predicted property $\hat{y}_{q,g}$ with the measured query property y_q. The synthesis planning reward captures both procedural plausibility and target consistency. For procedural plausibility, the LLM judge evaluates the generated procedure $\hat{x}_{q,g}^{\text{proc}}$ using the observed procedure x_q^{proc} as a reference trajectory rather than an exact target. Target consistency is scored by comparing the target property $\bar{y}$ with the property predicted from $\hat{x}_{q,g}^{\text{proc}}$ and x_q^{mat} using the Stage 1 prediction memory M^{pred}.

The resulting reward signals guide subsequent GoE-based summarization, allowing SciMem to emphasize reasoning trajectories associated with accurate property predictions in property prediction and scientifically plausible, target-consistent procedure in synthesis planning.

Groups of GoE-based summaries are jointly analyzed to identify scientific principles and procedural patterns that consistently correlate with successful outcomes, while discouraging reasoning patterns associated with poor evaluations. The resulting comparative patterns are abstracted into reusable GoE-based insights. The extracted insights are consolidated into the task memory, allowing reusable scientific abstractions to accumulate within a structured memory space.

At each memory-consolidation step t, the extracted insights are accumulated into the task memory with add, modify, merge, and delete operations:

$$M_{t+1}^\tau = \text{Update}\big(M_t^\tau, \Delta M_t^\tau\big). \quad (6)$$

Repeating this process progressively consolidates evaluated reasoning experience into reusable GoE memories.

Graph-of-Experience. Graph-of-Experience (GoE) is a structured memory representation designed to organize reusable reasoning abstractions. Free-form memory often collapses procedural context, intermediate outcomes, and explanatory signals within unstructured text. As a result, reusable reasoning patterns are obscured, limiting their retrieval and composition. GoE addresses this problem by representing reasoning dependencies in a

structured form.

GoE consists of textual nodes and edges. A node represents a localized transformation defined by (i) a feature-level change and (ii) the scientific principle explaining that change. Nodes are assigned to states, which provide coarse-grained perspectives over a materials reasoning process.

For materials reasoning, we instantiate GoE using six coarse-grained states: (A) material or precursor context, (B) controllable synthesis conditions, (C) immediate process implications, (D) key causal interactions, (E) derived structural characteristics, and (F) target-property outcomes. These states follow a typical progression from experimental setup to final material outcomes. The state decomposition is task-dependent and can be adapted to other domains requiring structured reasoning abstraction.

Let N_i^s denote the i-th node assigned to state s. Edges connect nodes across adjacent states and represent temporal, causal, and interaction dependencies. Figure 2 illustrates an example GoE with state-specific transformations and dependency structures across reasoning states. We use three recurring edge patterns:

$$\begin{aligned} \mathcal{E}_{\text{joint}} &: \{N_1^s, N_2^s\} \to N^{s+1}, \\ \mathcal{E}_{\text{independent}} &: N_1^s \to N^{s+1}, \qquad N_2^s \to N^{s+1}, \\ \mathcal{E}_{\text{parallel}} &: N_1^s \to N_1^{s+1}, \qquad N_2^s \to N_2^{s+1}. \end{aligned} \tag{7}$$

The joint pattern represents multiple factors acting together to produce a downstream effect. The independent pattern represents separate factors contributing to the same downstream effect. The parallel pattern represents distinct effects evolving independently across states. Together, these patterns compactly encode multi-factor scientific dependencies.

The accumulated GoE memory is injected directly into the prompting context at inference time. The memory provides a structured reasoning prior for memory-grounded generation.

4 Experimental Settings

We evaluate SciMem under both controlled benchmark settings and real-world laboratory validation. Benchmark evaluation measures property prediction and property-conditioned synthesis planning on public and anonymized materials datasets, while wet-lab validation evaluates whether memory-grounded synthesis planning transfers to practical QD-LED fabrication workflows.

4.1 Evaluation Datasets

We use two evaluation datasets: Open Materials Guide (OMG) benchmark (Kim et al., 2025; CC-BY-SA-4.0)[1] and an internal QD-LED dataset. For OMG, we retain instances containing aligned material context, multi-step procedures, and quantitative measured properties. The benchmark contains 1,495 training instances derived from papers published before 2024 and 159 test instances from papers published in 2024 or later. To study reasoning under more complex synthesis conditions, we further construct an anonymized QD-LED dataset with confidential experimental details removed. The dataset consists of 600 instances with a 9:1 train–test split. Additional details are provided in Appendices A and E.

4.2 Implementation Details

To prevent test-set leakage, we extracts memories only from the train split and never uses test instances as memory sources or reference examples.

The candidate pool is restricted to samples with the same x^{mat}. Train queries retrieve from the remaining training samples, while test queries retrieve only from train samples. SciMem samples one reference recipe from the top-7 candidates ranked by Jaccard similarity over x^{proc}.

We use Qwen3.5-27B (Qwen Team, 2026; Apache-2.0) as the backbone model for benchmark evaluation and compare four inference variants. *Baseline* directly queries the backbone LLM without additional reasoning support. *Prompting* augments the model with scientific reasoning guidance. *Unstructured Memory* accumulates experiences as free-form textual reflections, while *GoE* organizes accumulated experiences using structured representations. We sample five independent generations per test query and report mean evaluation scores.

Memory construction typically required 1–2 days on four H100 GPUs. Detailed prompting templates are provided in Appendix D, and detailed hyperparameters are reported in Table 6.

4.3 Evaluation Metrics

We evaluate both final answers and reasoning traces with GPT-5.4 as the LLM judge, accessed via the OpenAI API (OpenAI, 2026) with `reasoning.effort=xhigh`. Judge prompts and scoring rubrics are fixed across all methods and tasks, and all evaluation metrics use a 0–5

[1] https://huggingface.co/datasets/tattabio/OMG

Table 1: Main evaluation results for property prediction and *synthesis planning* on the OMG and QD-LED application. All scores are reported on a 0–5 scale.

Dataset	Method	Property Prediction		Synthesis Planning		
		Value Alignment ↑	Mechanistic Validity ↑	Feasibility & Specificity ↑	Observed Consistency ↑	Mechanistic Validity ↑
OMG	Baseline	3.33	2.32	**2.56**	2.86	2.78
	Prompting	3.36	2.35	2.50	2.82	2.80
	Unstructured Memory	3.56	2.69	2.55	**2.90**	2.84
	GoE (ours)	**3.69**	**2.95**	2.54	2.88	**2.89**
QD-LED	Baseline	4.12	2.75	2.45	2.55	2.50
	Unstructured Memory	4.13	2.71	**2.58**	2.66	2.64
	GoE (ours)	**4.57**	**2.79**	2.57	**2.70**	**2.69**

Table 2: Ablation of Graph-of-Experience (GoE) memory organization. All variants use the same GoE representation with different memory grouping strategies. All scores are reported on a 0–5 scale.

Method	Property Prediction		Synthesis Planning		
	Value Alignment ↑	Mechanistic Validity ↑	Feasibility & Specificity ↑	Observed Consistency ↑	Mechanistic Validity ↑
GoE (Global)	3.69	2.95	**2.54**	2.88	2.89
GoE (MF)*	3.72	**2.99**	2.53	2.86	**2.99**
GoE (MP)**	**3.74**	2.92	2.53	**2.91**	2.91

* MF: mechanism family. ** MP: matched material-property group.

scale. Detailed judge prompts are provided in Appendix G.

For property prediction, final outputs are scored using *Value Alignment*, which measures consistency between the predicted property $\hat{y}$ and the measured property y_q. The full definition is provided in Appendix B. For synthesis planning, procedural quality is evaluated using *Feasibility & Specificity* and *Observed Consistency*. Feasibility & Specificity measures whether generated procedures are experimentally coherent and executable, while Observed Consistency measures whether generated procedures can plausibly produce outcomes comparable to the observed procedure without relying on superficial textual overlap. For both tasks, reasoning traces are assessed using *Mechanistic Validity*.

Unlike conventional text-generation tasks, synthesis planning admits multiple experimentally valid procedures for the same desired property. Exact textual matching alone therefore provides an incomplete measure of scientific planning quality.

4.4 Wet-Lab Validation

We additionally validate SciMem in a real-world QD-LED synthesis workflow through in-house laboratory experiments.

Wet-lab candidate generation follows a separate resource-constrained inference setup with DeepSeek-V3.2(DeepSeek-AI, 2025; MIT) because real-world QD-LED fabrication workflows contain substantially more complex multi-layer procedural dependencies than the benchmark datasets, requiring a stronger backbone model. For each wet-lab experiment, the model generates six candidate synthesis procedures conditioned on the retrieved baseline and desired properties. Due to the cost and duration of wet-lab validation, one or two candidates are selected by domain researchers for physical execution.

Generated procedures are compared against reproduced baselines within the same experimental batch using device efficiency and operational lifetime measurements. This relative evaluation protocol controls for uncontrolled variation across fabrication batches.

For wet-lab validation, device efficiency and operational lifetime are used as experimental outcomes. Performance is reported as normalized relative improvement over the reproduced baseline, computed as $(y - y_{\mathrm{r}})/y_{\mathrm{r}}$, where y denotes the measured outcome of the generated procedure.

5 Experimental Results

5.1 Property Prediction and Synthesis Planning

Table 1 reports performance on property prediction and synthesis planning. Across tasks, memory-based methods generally outperform Prompting, in-

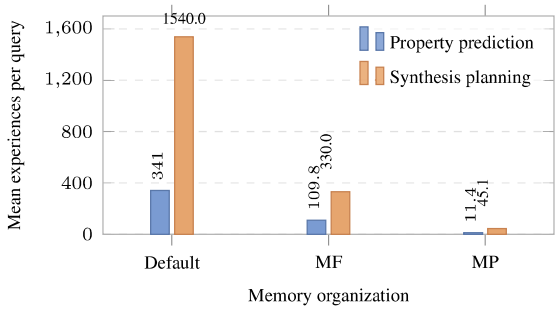

Figure 3: Active memory size under different GoE memory organizations. Bars show the mean number of experiences inserted into each query for property prediction and synthesis planning.

dicating that reusable reasoning patterns are better supported by accumulated memory than by static prompting alone. Unstructured memory yields less consistent gains, especially in Mechanistic Validity, whereas GoE Memory achieves the strongest Value Alignment and Mechanistic Validity on both OMG and QD-LED property prediction tasks.

For synthesis planning, GoE Memory achieves the strongest Observed Consistency and Mechanistic Validity scores on QD-LED while remaining competitive on Feasibility & Specificity. On OMG, the baseline occasionally attains slightly higher feasibility scores, likely because generic synthesis procedures can remain broadly plausible even without strong mechanistic grounding. GoE Memory, however, more consistently improves metrics evaluating causal and process-level consistency between generated procedures and experimentally observed outcomes.

5.2 Structured Memory

Figure 4 illustrates how memory structure affects reasoning reuse in property prediction. While unstructured memory can preserve broad scientific heuristics, it often overgeneralizes assumptions across unrelated synthesis settings.

In the Fe_3O_4/PBN example, both the baseline model and unstructured memory incorrectly predict particle growth by relying on a generic heuristic that high-energy ball milling increases aggregation and effective particle size. GoE Memory instead preserves the relevant dependency structure: the particles are pre-synthesized, no post-introduction high-temperature processing occurs, and particle growth is therefore constrained. This supports the inference that the final size remains close to the precursor size.

These results suggest that GoE improves reasoning reuse not simply by storing additional scientific knowledge, but by preserving context-dependent reasoning constraints that are often lost in unstructured reflection memory. Additional qualitative examples from the QD-LED benchmark exhibiting similar patterns are provided in Appendix F.

5.3 Memory Organization

Table 2 analyzes how memory organization shapes reasoning reuse. The default global setting exposes the model to reasoning trajectories collected across heterogeneous material systems. While this broader sharing increases retrieval noise, it can also encourage analogical reasoning across material contexts.

Material-family (MF) organization instead preserves mechanistically related reasoning trajectories within the same material family while filtering unrelated material systems. This maintains shared mechanistic constraints while exposing the model to diverse property-level interpretations of similar synthesis processes, which may contribute to improved Mechanistic Validity. Material-property (MP) organization further restricts retrieval to instances sharing both material family and target-property context, yielding the strongest retrieval locality and the highest Observed Consistency scores.

Figure 3 further shows that MF and MP organization dramatically reduce the number of active memories retrieved per query. For synthesis planning, the mean active memory size decreases from 1540.0 in the global setting to 330.0 under MF organization and 45.1 under MP organization, while reasoning quality remains stable or slightly improves. These results suggest that memory organization controls the granularity of reasoning reuse by determining which constraints are broadly shared, selectively preserved, or tightly localized. Additional MF and MP memory organization details are provided in Appendix C.

5.4 Wet-Lab Validation

Figure 5 presents wet-lab validation results for generated synthesis procedures. Most evaluated candidates achieve positive relative improvements in either device efficiency or operational lifetime, although several remain neutral or underperform relative to the reproduced baseline. Because synthesis planning is inherently open-ended and experimentally uncertain, such outcomes may partially reflect exploratory candidate selection beyond well-

(a) Memory for property prediction

Unstructured memory. *"When predicting particle size for oxide based composites, high energy ball milling can promote aggregation or cold welding rather than fragmentation. In such cases, the effective particle size can increase relative to the initial oxide nanoparticle size."*

GoE (ours). *"When pre-synthesized oxide nanoparticles are used as the precursor (A_1), and the subsequent synthesis only involves mechanical mixing or ball milling followed by low temperature drying (B_1), the oxide phase may experience dispersion or partial aggregation (C_1), but it does not undergo high temperature grain growth or Ostwald ripening (C_2). This creates a constraint where the initial precursor morphology remains the dominant structural determinant (E_1), so the final particle size should remain close to the precursor size unless the process introduces strong sintering or chemical reformation of the oxide phase (F_1)."*

(b) Property prediction query for Fe_3O_4/PBN particle size

Query. Predict `particle_size` for Fe_3O_4/PBN, where pre-synthesized Fe_3O_4 nanoparticles with average diameter $\sim$ 25 nm are ball-milled with PBN at 550 rpm for 6 h and dried at 80 °C. **Ground truth**:25 nm.

Method	Prediction	Reasoning trace excerpt
Baseline	58 nm	*"The high-energy ball-milling step at 550 rpm for 6 h can induce strong interparticle collisions and mechanical deformation. Such processing may promote aggregation of Fe_3O_4 within the PBN matrix during mixing and drying. Therefore, the final particle size is expected to be substantially larger than the nominal precursor size."*
Unstructured memory	40 nm	*"The query uses pre-synthesized Fe_3O_4 nanoparticles that are incorporated into a PBN scaffold by high-energy ball milling. Prior experience suggests that mechanical processing can increase the effective particle size through aggregation or cold welding rather than reducing hard oxide particles by fragmentation. Thus, the prediction should be adjusted upward from the initial feedstock size."*
GoE	25 nm	*"The query explicitly provides pre-synthesized Fe_3O_4 nanoparticles with an average diameter of $\sim$ 25 nm. After these nanoparticles are introduced, the recipe includes ball milling and drying at 80 °C but no high-temperature step that would drive grain coarsening or re-nucleation of the iron oxide phase. The particle size is therefore expected to remain close to the precursor size."*

Figure 4: Qualitative effect of memory structure on Fe_3O_4/PBN particle-size prediction. GoE preserves the query-specific constraint that pre-synthesized Fe_3O_4 nanoparticles are not exposed to post-introduction high-temperature coarsening, yielding the correct prediction.

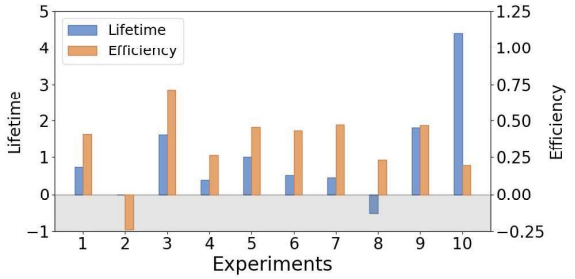

Figure 5: Wet-lab validation results for generated synthesis procedures. Bars show normalized relative improvement over reproduced references for device efficiency and operational lifetime within the same fabrication batch.

established synthesis regimes.

Overall, the wet-lab results provide empirical evidence that memory-grounded synthesis planning can generate transferable synthesis reasoning under real laboratory conditions.

6 Conclusion

We presented SciMem, a structured memory framework for LLM-based materials reasoning. SciMem represents prior experiences as a Graph of Experience (GoE), capturing explicit scientific states and mechanistic dependencies while organizing memory by scientific context. Across property prediction, synthesis planning, and wet-lab validation, SciMem improves mechanistic reasoning over prompting-only and unstructured-memory baselines. Our results show that scientific memory is inherently conditional: weakly related experiences should be filtered, while mechanistically coupled experiences benefit from shared accumulation. These findings highlight structured memory as a promising direction for scientific reasoning and autonomous discovery.

Limitations

Generalization to Device-Level Tasks. The internal QD-LED data and complete experimental records cannot be publicly released due to confidentiality constraints. To provide a reproducible evaluation, we report the performance of the proposed method on the publicly available OMG dataset. Nevertheless, OMG focuses on isolated material-property prediction, whereas the real-world QD-LED application involves device-level outcomes shaped by material interactions, processing conditions, and experimental protocols. Thus, the public benchmark does not fully capture the complexity of the deployment setting. Further validation is needed to assess whether the proposed method generalizes across broader material systems and device-level datasets.

LLM Dependency. SciMem relies on the backbone LLM for memory extraction, interpretation of structured context, and reasoning over retrieved memories. Our experiments use a backbone model selected based on the available computational resources. Since different LLMs can exhibit different instruction-following behavior, reasoning patterns, retrieval behavior, and structured output reliability, the performance and robustness of SciMem may vary across model families. Evaluation with a wider range of LLM backbones would further characterize SciMem's robustness and portability across model families.

Memory Reliability. Memory updates based on previous predictions and feedback may allow incorrect entries or noisy feedback to accumulate over time. The memory may also overgeneralize rare observations. In our experiments, domain experts inspected sampled memory entries for validation. However, such a non-automated process does not scale well to large memory stores. Future work could automate memory validation by incorporating uncertainty estimates and cross-checking memory entries against scientific literature.

References

Dhruv Ahlawat, Vaibhav Mishra, Somaditya Singh, Mohd Zaki, Vaibhav Bihani, Hargun Singh Grover, Biswajit Mishra, Santiago Miret, Mausam, and N. M. Anoop Krishnan. 2026. A family of large language models for materials research with insights into model adaptability in continued pretraining. *Nature Machine Intelligence*, 8:435–448.

Nawaf Alampara, Anagha Aneesh, Martiño Ríos-García, Adrian Mirza, Mara Schilling-Wilhelmi, Ali Asghar Aghajani, Meiling Sun, Gordan Prastalo, and Kevin Maik Jablonka. 2026. General-purpose models for the chemical sciences: LLMs and beyond. *Chemical Reviews*, 126(4):2484–2549.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. 2024. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535. Published version of ChemCrow arXiv:2304.05376.

Yuzheng Cai, Siqi Cai, Yuchen Shi, Zihan Xu, Lichao Chen, Yulei Qin, Xiaoyu Tan, Gang Li, Zongyi Li, Haojia Lin, Yong Mao, Ke Li, and Xing Sun. 2025. Training-free group relative policy optimization. *arXiv preprint arXiv:2510.08191*.

Mouyang Cheng, Chu-Liang Fu, Ryotaro Okabe, Abhijatmedhi Chotrattanapituk, Artittaya Boonkird, Nguyen Tuan Hung, and Mingda Li. 2026. Artificial intelligence-driven approaches for materials design and discovery. *Nature Materials*, 25(2):174–190.

Yuri Chervonyi, Trieu H. Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang H. Nguyen, Marcelo Menegali, Junehyuk Jung, Junsu Kim, Vikas Verma, Quoc V. Le, and Thang Luong. 2025. Gold-medalist performance in solving olympiad geometry with alphageometry2. *Journal of Machine Learning Research*, 26(241):1–39.

So Eun Choi, Miyoung Jang, Sohee Yoon, Sanghyun Yoo, Jooyeon Ahn, Minho Kim, Ho-Gyeong Kim, Yebin Jung, Seongeon Park, Young-Seok Kim, and 1 others. 2025. LLM-driven synthesis planning for quantum dot materials development. *Journal of Chemical Information and Modeling*, 65(6):2748–2758.

DeepSeek-AI. 2025. Deepseek-v3.2: Pushing the frontier of open large language models.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Nino Vieillard, Aleksei Andreev, Chloé Crepy, Ana Sofia Pinto Oliveira, Olivier Bachem, Yukie Ono, Léonard Hussenot, and 1 others. 2026. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613. Published online 2025; version of record in 2026.

Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin A. Persson. 2013. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002.

Kush Jain, Gabriel Synnaeve, and Baptiste Roziere. 2025. Testgeneval: A real world unit test generation and test completion benchmark. In *International Conference on Learning Representations*.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*.

Christopher Karpovich, Elton Pan, and Elsa A. Olivetti. 2024. Deep reinforcement learning for inverse inorganic materials design. *npj Computational Materials*, 10(1).

Heegyu Kim, Taeyang Jeon, Seungtaek Choi, Ji Hoon Hong, Dong Won Jeon, Sung Beom Cho, Ga-Yeon Baek, Gyeong-Won Kwak, Dong-Hee Lee, Sun-Jin Choi, Jisu Bae, Chihoon Lee, Yunseo Kim, Jinsung Park, and Hyunsouk Cho. 2025. Towards fully-automated materials discovery via large-scale synthesis dataset and expert-level LLM-as-a-judge. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 1302–1312. Also available as arXiv:2502.16457.

Kangsan Kim, Minki Kang, Taeil Kim, Yanlai Yang, Mengye Ren, and Sung Ju Hwang. 2026a. Memory transfer learning: How memories are transferred across domains in coding agents. *arXiv preprint arXiv:2604.14004*.

Seongmin Kim, Jaehwan Choi, Kunik Jang, Junkil Park, Varinia Bernales, Alán Aspuru-Guzik, and Yousung Jung. 2026b. Materealize: a multi-agent deliberation system for end-to-end material design and synthesis. *arXiv preprint arXiv:2601.15743*.

Olga Kononova, Haoyan Huo, Tanjin He, Ziqin Rong, Tiago Botari, Wenhao Sun, Vahe Tshitoyan, and Gerbrand Ceder. 2019. Text-mined dataset of inorganic materials synthesis recipes. *Scientific Data*, 6:203.

Fusataka Kuniyoshi, Kohei Makino, Jun Ozawa, and Makoto Miwa. 2020. Annotating and extracting synthesis process of all-solid-state batteries from scientific literature. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1941–1950.

Hyun Lee, Sohee Yoon, Jinwoo Park, Sue In Chae, Seongeon Park, Jooyeon Ahn, Yebin Jung, Youjung Chung, Hogeun Chang, Sujin Park, Myeonginn Kang, Jina Kim, Ho-Gyeong Kim, and Myeonghun Jeong. 2025a. Aligning reasoning LLMs for materials discovery with physics-aware rejection sampling. *arXiv preprint arXiv:2509.00768*.

Sanghoon Lee, Kevin Cruse, Viktoriia Baibakova, Gerbrand Ceder, and Anubhav Jain. 2025b. Text-mined dataset of solid-state syntheses with impurity phases using large language model. *Scientific Data*, 12(1).

Ge Lei, Ronan Docherty, and Samuel J. Cooper. 2024. Materials science in the era of large language models: A perspective. *Digital Discovery*, 3(7):1257–1272.

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

Xinghang Li, Peiyan Li, Long Qian, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Xinlong Wang, Di Guo, Tao Kong, Hanbo Zhang, and 1 others. 2026. What matters in building vision–language–action models for generalist robots. *Nature Machine Intelligence*, 8:158–172.

Xuechen Liang, Meiling Tao, Yinghui Xia, Jianhui Wang, Kun Li, Yijin Wang, Yangfan He, Jingsong Yang, Tianyu Shi, Yuantao Wang, Miao Zhang, and Xueqian Wang. 2025. SAGE: Self-evolving agents with reflective and memory-augmented abilities. *Neurocomputing*, 647:130470.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. 2026. Towards end-to-end automation of ai research. *Nature*, 651:914–919.

Wei Lu, Rachel K. Luu, and Markus J. Buehler. 2025. Fine-tuning large language models for domain adaptation: exploration of training strategies, scaling, model merging and synergistic capabilities. *npj Computational Materials*, 11(84).

Junyu Luo, Xiao Luo, Xiusi Chen, Zhiping Xiao, Wei Ju, and Ming Zhang. 2025a. Semi-supervised fine-tuning for large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2795–2808.

Ziming Luo, Zonglin Yang, Zexin Xu, Wei Yang, and Xinya Du. 2025b. LLM4SR: A survey on large language models for scientific research. *arXiv preprint arXiv:2501.04306*.

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

Santiago Miret and N. M. Anoop Krishnan. 2025. Enabling large language models for real-world materials discovery. *Nature Machine Intelligence*, 7:991–998.

Sheshera Mysore, Edward Kim, Emma Strubell, Ao Liu, Haw-Shiuan Chang, Srikrishna Kompella, Kevin Huang, Andrew McCallum, and Elsa Olivetti. 2017. Automatically extracting action graphs from materials science synthesis procedures. *arXiv preprint arXiv:1711.06872*.

Andre Niyongabo Rubungo, Kangming Li, Jason Hattrick-Simpers, and Adji Bousso Dieng. 2025. LLM4Mat-Bench: Benchmarking large language models for materials property prediction. *Machine Learning: Science and Technology*, 6(2):020501.

Heewoong Noh, Gyoung S. Na, Namkyeong Lee, and Chanyoung Park. 2026. Msp-llm: A unified large language model framework for complete material synthesis planning. *arXiv preprint arXiv:2602.07543*.

Uzoma Nwabara, Kunran Yang, Akshay Talekar, Varinia Bernales, Jorge González, Stuart Miller, and Jinfeng Wu. 2025. High throughput computational and experimental methods for accelerated electrochemical materials discovery. *Journal of Materials Chemistry A*, 13(32):26041–26066.

OpenAI. 2026. GPT-5.4 Model. https://developers.openai.com/api/docs/models/gpt-5.4. Accessed: 2026-05-25.

Elton Pan, Soonhyoung Kwon, Sulin Liu, Mingrou Xie, Alexander J. Hoffman, Yifei Duan, Thorben Prein, Killian Sheriff, Yuriy Roman-Leshkov, Manuel Moliner, Rafael Gomez-Bombarelli, and Elsa A. Olivetti. 2026. DiffSyn: A generative diffusion approach to materials synthesis planning. *Nature Computational Science*, 6:404–416.

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

Zhengxing Peng, Natalie Stingelin, Harald Ade, and Jasper J. Michels. 2023. A materials physics perspective on structure–processing–function relations in blends of organic semiconductors. *Nature Reviews Materials*, 8:439–455.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, and 1 others. 2022. A generalist agent. *arXiv preprint arXiv:2205.06175*.

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

Aditya Sood, Andrey D. Poletayev, Daniel A. Cogswell, Peter M. Csernica, J. Tyler Mefford, Dimitrios Fraggedakis, Michael F. Toney, Aaron M. Lindenberg, Martin Z. Bazant, and William C. Chueh. 2021. Electrochemical ion insertion from the atomic to the device scale. *Nature Reviews Materials*, 6:847–867.

Evan Walter Clark Spotte-Smith, Orion Archer Cohen, Samuel M. Blau, Jason M. Munro, Ruoxi Yang, Rishabh D. Guha, Hetal D. Patel, Sudarshan Vijay, Patrick Huck, Ryan Kingsbury, and 1 others. 2023. A database of molecular properties integrated in the materials project. *Digital Discovery*, 2(6):1862–1882.

Wenhao Sun and Nicholas David. 2025. A critical reflection on attempts to machine-learn materials synthesis insights from text-mined literature recipes. *Faraday Discussions*, 256:614–638.

Hirofumi Tsuruta and Masaya Kumagai. 2025. MatPROV: A provenance graph dataset of material synthesis extracted from scientific literature. *arXiv preprint arXiv:2509.01042*.

Amit K. Verma, Zhisong Zhang, Junwon Seo, Robin Kuo, Runbo Jiang, Emma Strubell, and Anthony D. Rollett. 2025. Text mining for process–structure–properties relationships in metals. *Integrating Materials and Manufacturing Innovation*. Published-title metadata matched to DOI; earlier draft title: Structured Extraction of Process Structure Properties Relationships in Materials Science.

Hongchen Wang, Kangming Li, Scott Ramsay, Yao Fehlis, Edward Kim, and Jason Hattrick-Simpers. 2025. Evaluating the performance and robustness of llms in materials science q&a and property predictions. *Digital Discovery*, 4:1612–1624.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schütze, Volker Tresp, and Yunpu Ma. 2025. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*.

A OMG Dataset Filtering and Normalization

We retain OMG records that satisfy seven criteria: non-null chemical formula; an explicit quantitative functional property; recipe completeness sufficient for *synthesis planning*; at least one precursor or starting material; at least three meaningful synthesis stages; at least one explicit process condition such as temperature, time, pH, or concentration; and unambiguous recipe–property alignment.

We then normalize property names and units and normalize material-class labels. Mechanism-family labels are assigned by giving every candidate data sample, together with the four mechanism-family definitions in Figure 6, to GPT-5.4-xhigh and selecting one of `surface_interface`, `transport_percolation`, `electronic_band_structure`, or `bulk_phase_microstructure`. Records whose governing mechanism cannot be determined unambiguously from the synthesis-property evidence are filtered out, so the MF memory organization is defined only for examples with an identifiable dominant mechanism.

For matched material-property (MP) memory organization, we keep only normalized material/property pairs with more than 20 property-level training instances. Smaller groups are removed because they provide too little repeated evidence for stable, memory updates. The retained train–test combined MF and MP distributions are reported in Tables 4 and 5. The temporal split is applied after filtering and normalization, using papers published before 2024 for training and papers published in 2024 or later for testing. Only the training split is used to construct the memory.

B Programmatic Target Value Alignment

For property prediction, Target Value Alignment compares the model prediction $\hat{y}$ with the ground-truth query property y_q after canonical parsing and unit normalization. The parser accepts scalar values, explicit ranges such as a–b or "a to b", uncertainty expressions such as $x \pm d$, and approximate forms. If the prediction cannot be parsed or normalized, or if the normalized units do not match, the score is set to 0. In all cases, the metric is mapped to the same 0–5 range as the judge-scored metrics; this is why the formulas below carry a leading factor of 5.

B.1 Scalar Case

When both $\hat{y}$ and y_q are scalar after normalization, we compare their transformed scalar values directly. Let $\tilde{\hat{y}}$ and $\tilde{y}_q$ denote the normalized quantities. For strictly positive properties, we compare them in $\log_{10}$ space so that multiplicative error is treated more naturally. Otherwise, we compare them in a linear space normalized by the scale of y_q.

The scalar score is

$$s_{\text{scalar}} = 5 \exp\left(- \left(\frac{|\tilde{\hat{y}} - \tilde{y}_q|}{\tau_c} \right)^2 \right).$$

This form assigns score 5 to a perfect match and decays smoothly as the normalized scalar error increases.

B.2 Range-Aware Case

When either $\hat{y}$ or y_q is expressed as a range, we convert both into normalized intervals. For compactness, let $\hat{I} = \tilde{I}_{\hat{y}}$ and $I_q = \tilde{I}_q$. Let $c(\cdot)$ and $w(\cdot)$ denote interval center and width, respectively. We define the normalized center and width errors as

$$d_c = \frac{|c(\hat{I}) - c(I_q)|}{\tau_c},$$

$$d_w = \frac{\left[w(\hat{I}) - w(I_q) \right]_+}{\tau_w},$$

where $[x]_+ = \max(0, x)$. The range-aware score is then

$$s_{\text{range}} = 5 \exp\left(-d_c^2 - d_w^2 \right).$$

The center term rewards predictions whose interval is correctly located. The width term penalizes predictions that are broader than the ground-truth interval, preventing the model from receiving an artificially high score by outputting a very wide interval that trivially overlaps the target.

B.3 Unified Implementation

The implementation can be written as a single interval-based equation by treating a scalar quantity as a zero-width interval. In that view, the scalar case is simply the special case of the range-aware metric with $w(\tilde{I}_q) = w(\tilde{I}_{\hat{y}}) = 0$. We present the two cases separately here only for clarity.

C Memory Organization Details

Materials-design experience is highly context dependent: similar synthesis conditions can induce different effects depending on the material class, target property, and governing mechanism. SciMem therefore uses a disentangled memory update strategy that organizes GoE experiences into independent context units rather than updating a single undifferentiated memory. Table 3 summarizes the resulting memory organizations, and Tables 4 and 5 show the train–test combined data distributions for the mechanism-family (MF) and matched material-property (MP) groups. Counts are computed after combining the training and test splits and expanding records with multiple extracted properties.

For MF memory, we define four mechanism families corresponding to surface/interface control, transport/percolation, electronic band-structure control, and bulk phase/microstructure control. Each candidate sample is assigned to one of these four labels by GPT-5.4-xhigh using the definitions in Figure 6. Samples whose mechanism cannot be resolved unambiguously from the available synthesis-property evidence are excluded, so each retained MF group has a clear dominant mechanism.

For MP memory, each memory is tied to an exact normalized material/property pair. We therefore filter out pairs with 30 or fewer property-level training instances before training MP memories. These small cells would yield too few rollout comparisons and memory-update opportunities to accumulate scientific experience, making the group-specific memory more sensitive to noise than to a stable material-property pattern. After this thresholding, the retained train–test combined MP distribution contains 1,654 property-level instances, as shown in Table 5.

The MF labels are defined by broad mechanism regimes, while MP labels combine normalized material and property labels. Figures 6–9 list the full textual descriptions used for these labels.

Table 3: Active GoE experience statistics for the memory organizations.

Task	Memory	# Groups	Total Exp.	Mean
Property prediction	Default	1	341	341.0
Property prediction	MF	4	439	109.8
Property prediction	MP	41	466	11.4
Synthesis planning	Default	1	1,540	1,540.0
Synthesis planning	MF	4	1,320	330.0
Synthesis planning	MP	20	903	45.1

Table 4: Mechanism-family (MF) data distribution after GPT-5.4-xhigh assignment and filtering of ambiguous mechanism cases.

Mechanism family	Count	Share
surface_interface	1,035	62.6%
electronic_band_structure	334	20.2%
bulk_phase_microstructure	195	11.8%
transport_percolation	90	5.4%
Total	1,654	100.0%

Table 5: Material-property (MP) data distribution after filtering out groups with 30 or fewer property-level training instances. Counts are property-level instances.

Material group	Property group	Count	Share
coordination_polymer	product_yield	128	7.7%
polymer	product_yield	112	6.8%
tio2_based_oxide	band_gap	110	6.7%
iron_oxide	adsorption_capacity	88	5.3%
iron_oxide	saturation_magnetization	81	4.9%
zno_based_oxide	band_gap	75	4.5%
iron_oxide	particle_size	59	3.6%
iron_oxide	specific_surface_area	52	3.1%
amorphous_carbon	specific_surface_area	50	3.0%
chalcogenide_semiconductor	band_gap	49	3.0%
amorphous_carbon	product_yield	49	3.0%
supercapacitor_electrode_material	gravimetric_specific_capacitance	46	2.8%
metal_organic_framework	product_yield	44	2.7%
inorganic_salt	product_yield	44	2.7%
heterogeneous_catalyst_material	product_yield	42	2.5%
polymer_electrolyte	ionic_conductivity	40	2.4%
tio2_based_oxide	specific_surface_area	39	2.4%
solid_electrolyte	ionic_conductivity	39	2.4%
silicate	adsorption_capacity	39	2.4%
zno_based_oxide	crystallite_size	38	2.3%
iron_oxide	product_yield	37	2.2%
non_oxide_ceramic	hardness	36	2.2%
metal_nanoparticle	particle_size	36	2.2%
metal_organic_framework	specific_surface_area	35	2.1%
tio2_based_oxide	particle_size	35	2.1%
polymer	specific_surface_area	33	2.0%
polymer	number_average_molecular_weight	33	2.0%
silicate	specific_surface_area	33	2.0%
metal_nanoparticle	product_yield	31	1.9%
adsorbent_material	adsorption_capacity	31	1.9%
amorphous_carbon	gravimetric_specific_capacitance	30	1.8%
graphene_based_carbon	product_yield	30	1.8%
perovskite	band_gap	30	1.8%
Total	-	1,654	100.0%

Mechanism-family descriptions for MF memory

`surface_interface`: performance is governed mainly by surface area, active-site accessibility, adsorption-desorption, and interfacial reaction chemistry at exposed surfaces.

`transport_percolation`: performance is governed mainly by electron or ion transport continuity, percolation pathways, interface connectivity, and transport bottlenecks.

`electronic_band_structure`: performance is governed mainly by band alignment, carrier energetics, trap-mediated recombination behavior, and electronic structure constraints.

`bulk_phase_microstructure`: performance is governed mainly by bulk composition and phase state, crystallinity, grain or domain structure, densification, and microstructural organization.

Figure 6: Full mechanism-family descriptions used for MF memory organization.

Property-name descriptions for MP memory

`adsorption_capacity`: uptake capacity of a target adsorbate under stated test conditions, commonly reported as qe or qmax for dyes, metal ions, oxyanions, antibiotics, or other dissolved species and often conditioned on pH, temperature, and isotherm regime.

`band_gap`: optical or electronic band-gap energy in eV, usually inferred from absorption, photoluminescence, or Tauc-type analysis for semiconducting oxides, chalcogenides, and perovskites where phase, dopants, and defects shift carrier energetics.

`crystallite_size`: average coherent crystalline domain size, typically extracted from XRD peak broadening or phase-resolved diffraction analysis and reflecting calcination or growth driven grain coarsening rather than the full particle dimension.

`gravimetric_specific_capacitance`: charge-storage capacitance normalized by active-material mass under stated current density, scan rate, electrolyte, and voltage window, capturing electric-double-layer and/or pseudocapacitive contributions.

`hardness`: indentation-derived resistance to plastic deformation, usually measured by nanoindentation or Vickers-type testing on dense non-oxide ceramic coatings or bulk phases under specified load and microstructural state.

`ionic_conductivity`: ionic transport conductivity of a solid or polymer electrolyte under stated temperature, salt loading, humidity, or phase regime, governed by mobile-ion concentration, dissociation, and connectivity of conduction pathways.

`number_average_molecular_weight`: number-average molar mass Mn of the synthesized polymer population, typically determined by GPC/SEC or closely related calibration methods and used to track chain-growth extent across polymerization conditions.

`particle_size`: observed or inferred particle dimension of the synthesized particulate product, usually reported as an average or range from SEM, TEM, or diffraction-based estimates for the final nanoparticles or agglomerates.

`product_yield`: recovered amount of the intended product after the stated synthesis and purification sequence, usually expressed as isolated percent yield but sometimes reported as wt%, recovered mass, or output per batch for a named product state.

`saturation_magnetization`: maximum magnetization reached under sufficiently high applied field, typically from VSM or SQUID measurements of iron-oxide-rich magnetic materials and strongly affected by phase purity, coating thickness, and nonmagnetic dilution.

`specific_surface_area`: accessible surface area per unit mass, typically BET-derived from gas adsorption and used across porous carbons, MOFs, oxides, polymers, and silicates to quantify exposed pore and surface accessibility.

Figure 7: Full property-name descriptions used for the property dimension of MP memory organization.

Material-name descriptions for MP memory (1/2)

`adsorbent_material`: engineered sorbent systems intentionally synthesized for contaminant uptake, often combining porous carbon, layered hydroxides, oxides, biochar, or silica-rich components through hydrothermal, solution, or hybrid routes to maximize accessible binding sites.

`amorphous_carbon`: non-graphitic or weakly ordered carbon materials produced mainly by pyrolysis, templating, or hybrid carbonization of biomass or polymer precursors, often heteroatom-doped and pore-engineered for high accessible surface area and electrochemical response.

`chalcogenide_semiconductor`: sulfide- or selenide-dominant semiconductors such as MoS2, WS2, CZTS, and related alloys or nanosheets prepared by vapor, solution, hydrothermal, or hybrid routes where composition, layer thickness, and defect density control the electronic structure.

`coordination_polymer`: extended metal-ligand coordination compounds and polymeric complexes assembled from solution, self-assembly, or solvothermal chemistry, typically emphasizing discrete-to-network complex formation where isolated synthesis yield is the main tracked outcome.

`graphene_based_carbon`: graphene, graphene oxide, reduced graphene oxide, graphene quantum dots, and few-layer graphene derivatives produced by exfoliation, oxidation-reduction, electrochemical, or hybrid biomass-derived routes with highly exposed sp2-carbon-rich surfaces.

`heterogeneous_catalyst_material`: solid catalyst materials and supported nanocatalysts prepared for recoverable catalytic reactions, frequently combining metals, oxides, carbons, or organosilicon supports so that accessible surface sites and dispersion dominate behavior.

`inorganic_salt`: salt- or ionic-compound-dominant functional materials, including lanthanide fluorides, triazolium or perchlorate salts, surfactant salts, and related crystalline ionic products that are usually tracked by isolated synthesis yield.

`iron_oxide`: Fe-oxide-dominant particles, composites, and magnetic nanomaterials synthesized from iron salts by solution, hydrothermal, hybrid, sonochemical, or pyrolytic routes and commonly tuned for magnetization, adsorption, particle size, and accessible surface area.

`metal_nanoparticle`: metallic nanoparticle or nanowire systems based on Ag, Au, Pt, Pd, Cu, Ni, and related metals, formed through reduction, deposition, biological, or hybrid routes where nucleation, stabilization, and support interactions set particle dimensions and recovery.

`metal_organic_framework`: porous crystalline MOF systems built from metal nodes and organic linkers via solvothermal, hydrothermal, microwave, or hybrid synthesis, commonly optimized for isolated yield together with BET-accessible porosity.

Figure 8: Full material-name descriptions used for the material dimension of MP memory organization, part 1 of 2.

Material-name descriptions for MP memory (2/2)

`non_oxide_ceramic`: carbide, nitride, boride, carbonitride, and diamond-like ceramic phases or coatings formed by high-temperature solid-state or vapor-deposition processing, where phase constitution and dense microstructure govern hardness.

`perovskite`: perovskite-structured halide or oxide materials such as cesium, lead, tin, ferrite, or niobate systems synthesized by solution or hybrid routes where stoichiometry, dopants, and phase purity tune the band structure.

`polymer`: organic polymeric materials spanning linear, block, dendritic, porous, and conjugated polymers prepared mainly by solution, hybrid, or mechanochemical polymerization or functionalization routes, often tracked by isolated yield, Mn, and sometimes porosity.

`polymer_electrolyte`: polymer-host electrolyte matrices containing dissolved salts, plasticizers, or inorganic fillers, typically cast or hybrid-processed into films or gels where segmental mobility, salt dissociation, and percolated ion pathways control ionic conductivity.

`silicate`: silica-, silicate-, or zeolite-dominant porous inorganic frameworks synthesized by hydrothermal or solution templating, ion exchange, or surface-functionalization routes, usually targeting adsorption capacity and BET surface area through pore architecture and surface chemistry.

`solid_electrolyte`: inorganic solid-state ion conductors such as thiophosphates, NASICON-type oxides, borohydrides, and oxynitrides produced by mechanochemical, solid-state, or assisted solution routes where densification and continuous ion-conduction pathways govern ionic conductivity.

`supercapacitor_electrode_material`: electrode-active materials for supercapacitors, including sulfides, selenides, carbons, and hybrid architectures grown on foams or composite supports so that redox-accessible area and coupled electron-ion transport jointly determine capacitance.

`tio2_based_oxide`: TiO2-dominant oxide materials including mixed-phase, doped, decorated, nanotubular, film, and composite titania systems prepared by solution, hydrothermal, vapor, or hybrid methods, with phase balance and defect chemistry controlling band gap, size, and surface area.

`zno_based_oxide`: ZnO-dominant oxides and ZnO-based composites, films, or nanoparticles synthesized by solution, hydrothermal, biological, electrochemical, or vapor routes where dopants, crystallite growth, and morphology tune band gap and crystalline domain size.

Figure 9: Full material-name descriptions used for the material dimension of MP memory organization, part 2 of 2.

D Prompt Templates

This section lists the prompt templates used to instantiate the experimental pipeline described in Sections 3 and 4. Figure 10 shows the scientific reasoning guidance used by the Prompting baseline and by memory-augmented variants whenever explicit domain grounding is required. The guidance asks the model to justify synthesis and design choices using real materials-science mechanisms, concrete process variables, and physically meaningful causal explanations.

The task-level inference prompts preserve the common experimental setting used throughout the paper: each query is conditioned on a reference recipe, material and property descriptors, and explicit scientific reasoning constraints. Figure 11 shows the property-prediction prompt, which asks the model to infer a structured target-property object from a full synthesis record. Figure 12 shows the property-conditioned synthesis-planning prompt, which asks the model to design a synthesis procedure for a requested target property while remaining within the provided material-system constraints. Long recipe record blocks are abbreviated in these displayed examples; the actual prompts use complete records.

The trajectory summarization stage optionally inserts a scientific target prompt to control how much task-specific context is used when converting rollouts into scientific experience. Figure 13 shows the default, mechanism family specific, and material-property-specific variants. These variants correspond to the memory organization schemes studied in Table 2: the default prompt keeps only general scientific validity constraints, the mechanism-family prompt grounds the summary in a broad process regime, and the material-property prompt further conditions the summary on the material and target-property pair.

For property prediction, the unstructured-memory baseline uses a four-step prompt sequence. Figure 14 shows the inference prompt that exposes learned experiences at test time. Figure 15 summarizes model trajectories against the ground-truth property, Figure 16 extracts scientific experience edits from those summaries, and Figure 17 applies the resulting memory updates.

The Graph-of-Experience version uses the same property-prediction inference setting but imposes an explicit A–F causal structure during summarization and extraction. Figure 18 shows the GoE inference prompt. Figures 19 and 20 define the GoE trajectory-summary prompt, including node types, edge rules, and dominant path extraction. Figures 21 and 22 define the experience-extraction prompt that turns GoE summaries into add, modify, merge, or delete operations, and Figure 23 shows the final update prompt.

For *synthesis planning*, the unstructured-memory prompt sequence mirrors the property-prediction pipeline while changing the output target from a property value to a synthesis procedure. Figure 24 shows the inference prompt for proposing a recipe under memory guidance. Figure 25 summarizes inverse-design rollouts with respect to the desired property and observed recipe, Figure 26 extracts planning experiences, and Figure 27 updates the planning memory.

Finally, the GoE synthesis-planning prompts use the same inverse-design task but encode the generated recipe, intermediate implications, interactions, and target-property outcome as a structured graph. Figure 28 shows the inference prompt. Figures 29 and 30 define the GoE summary prompt for inverse design. Figures 31 and 32 define the corresponding experience-extraction prompt, and Figure 33 shows the update prompt used to revise the synthesis-planning memory. The figures below are ordered to follow this pipeline.

Scientific Reasoning Guidance

Grounding. All reasoning must be grounded in real materials-science principles, including thermodynamics, kinetics, defect chemistry, crystallography, surface and interface phenomena, transport processes, and precursor chemistry.

Mechanistic justification. The model must justify each synthesis modification by referencing the physical or chemical mechanisms affected, explain why the modification leads to the desired property change, and use real synthesis parameters such as temperature, solvent, pH, precursor concentration, annealing duration, oxygen partial pressure, atmosphere, and mixing rate. The reasoning must remain mechanistically correct and applicable to real materials systems.

Prohibited shortcuts. The model must not introduce symbolic mathematical variables, perform algebraic manipulation, treat process parameters as abstract mathematical variables, perform analytic optimization, skip scientific explanations for synthesis changes, or rely on superficial keyword matching without causal, physics-based justification.

Figure 10: Scientific Reasoning Guidance used in the Prompting condition.

Property Prediction Prompt

Task context. You are a world-class materials-science assistant for synthesis-to-property reasoning across catalysts, batteries, semiconductors, ceramics, polymers, porous materials, composites, and nanomaterials.

You are given: (1) one reference recipe with a measured property value, and (2) one query recipe with the same material/property context but without the target property value. **Your role:** Predict the target property of the query recipe by reasoning from the reference recipe and the query recipe.

Mechanism Family Description. `bulk_phase_microstructure`: performance is governed mainly by bulk composition and phase state, crystallinity, grain or domain structure, densification, and microstructural organization.

Material Description. `iron_oxide`: Fe-oxide-dominant particles, composites, and magnetic nanomaterials synthesized from iron salts by solution, hydrothermal, hybrid, sonochemical, or pyrolytic routes and commonly tuned for magnetization, adsorption, particle size, and accessible surface area.

Target Property Description. `adsorption_capacity`: uptake capacity of a target adsorbate under stated test conditions, commonly reported as q_e or $q_{\max}$ for dyes, metal ions, oxyanions, antibiotics, or other dissolved species and often conditioned on pH, temperature, and isotherm regime.

Abbreviated prompt record block. The actual prompt includes the complete reference and query records; here we show the beginning and end of each record and omit the middle with . . . .

> **Reference.** Recipe: materials include waste sludge-based $ZnCl_2$ activated carbon, $Fe(NO_3)_3{\cdot}9H_2O$, $Mg(NO_3)_2{\cdot}6H_2O$, phosphate/nitrate/sulfate/carbonate salts, HCl, and NaOH. Synthesis procedure: disperse SBAC by ultrasonication, mix Fe and Mg nitrate solutions, agitate at $60\,^{\circ}$C, adjust pH to 10 ± 0.5, reflux at $90\,^{\circ}$C for 18 h, wash, and dry at $85\,^{\circ}$C. . . . Class: `bulk_phase_microstructure`, `iron_oxide`, `adsorption_capacity`. Property: 110 mg/g, maximum adsorption capacity for phosphate.
>
> **Query.** Target recipe: materials include $FeCl_2{\cdot}4H_2O$, hexamethylenetetramine, $NaNO_3$, MWCNTs, $AgNO_3$, Viscum album extract, $NaBH_4$, and ferrocene. Synthesis procedure includes γ-Fe_2O_3 nanoparticle synthesis, MWCNT pyrolysis from waste tire, Ag nanoparticle biosynthesis, and assembly of γ-Fe_2O_3/MWCNT/Ag nanocomposite particles. . . . Class: `bulk_phase_microstructure`, `iron_oxide`, `adsorption_capacity`. Property unit: mg/g; measurement context: sulfamethazine removal.

Task. Predict the most plausible structured `target_property` object for the synthesized material from the synthesis record. Reason in a single coherent path from the provided materials, synthesis conditions, intermediate implications, and likely resulting behavior to the final target property. Do not replace the property with another label, even if related labels seem plausible. The explanation must remain concise, physically meaningful, internally consistent, and directly supported by the input context.

Final output format (JSON only).

```
{
"predicted_property_value_with_unit": "",
"rationale": ""
}
```

Example output shape.

```
{
"property_value": "180-190 F/g at low scan rate",
"rationale": "The input record primarily controls interfacial area and accessibility
through co-precursor hydrothermal growth and a moderate post-anneal, so the focal
property should be interpreted through the declared mechanism lens rather than through
an unrelated property family."
}
```

Figure 11: Property-prediction prompt. The displayed prompt preserves the task context, scientific conditioning labels, reasoning requirements, task definition, and output schema. Long reference and query recipe records are abbreviated with . . . ; the actual prompt uses complete records.

Synthesis Planning Prompt

Task context. You are a world-class materials-science assistant for inverse design across catalysts, batteries, semiconductors, ceramics, polymers, porous materials, composites, and nanomaterials.

You are given: a reference recipe and one specified query material system with target property. **Your task:** Design the most plausible `## Synthesis Procedure` that can achieve the target property while remaining consistent with the provided material-system constraints. **Your role:** translate the target property into a mechanism-aware synthesis strategy, propose a realistic and actionable procedure with concrete processing variables, and explicitly note key risks and uncertainties instead of making unsupported guarantees.

Mechanism Family Description. `bulk_phase_microstructure`: performance is governed mainly by bulk composition and phase state, crystallinity, grain or domain structure, densification, and microstructural organization.

Material Description. `iron_oxide`: Fe-oxide-dominant particles, composites, and magnetic nanomaterials synthesized from iron salts by solution, hydrothermal, hybrid, sonochemical, or pyrolytic routes and commonly tuned for magnetization, adsorption, particle size, and accessible surface area.

Property Description. `adsorption_capacity`: uptake capacity of a target adsorbate under stated test conditions, commonly reported as q_e or $q_{\max}$ for dyes, metal ions, oxyanions, antibiotics, or other dissolved species.

Abbreviated prompt record block. The actual prompt includes the complete reference and query records; here we show representative fields and omit the middle with . . . .

> **Reference.** Materials include $FeSO_4{\cdot}7H_2O$, $FeCl_3{\cdot}6H_2O$, tea waste, distilled water, ammonium hydroxide, and N_2. Procedure: dissolve iron salts in water, heat to 80 °C, add ammonium hydroxide and tea waste, react for 30 min under N_2, wash, magnetically separate, and dry at 60 °C. . . . Property: 75.76 mg/g maximum adsorption capacity for Cr(VI).
>
> **Query.** Materials include $FeCl_2{\cdot}4H_2O$, hexamethylenetetramine, $NaNO_3$, MWCNTs, $AgNO_3$, Viscum album extract, $NaBH_4$, and ferrocene; available equipment includes an N_2 atmosphere oven, ultrasonic bath, and quartz tube. . . . Target property: 47.61 mg/g adsorption capacity for sulfamethazine removal.

Task. Design only the `## Synthesis Procedure` for a material that achieves the target property value specified in the query recipe. The procedure should maximize the likelihood of achieving the requested target while remaining experimentally coherent.

Final output format (JSON only).

```
{
"synthesis_procedure": "## Synthesis Procedure
n1. ...
n2. ..."
}
```

Example output shape.

```
{
"synthesis_procedure": "## Synthesis Procedure
n1. Dissolve the provided precursors in the specified solvent.
n2. Apply the route-defining growth, mixing, or thermal step using concrete temperature,
time, and atmosphere conditions.
n3. Wash, separate, dry, and collect the final material for downstream evaluation."
}
```

Figure 12: Property-conditioned synthesis-planning prompt. The displayed prompt preserves the task context, scientific conditioning labels, inverse-design requirements, task definition, and output schema. Long reference and query recipe records are abbreviated with . . . ; the actual prompt uses complete records.

Specific Target Prompts

Default: The summary must remain concise, physically meaningful, internally consistent, and directly supported by the input context.

Mechanism-family-specific target prompt (`mf`): The summary must remain concise, physically meaningful, internally consistent, and directly supported by the input context. The task must focus on the defining characteristics of the mechanism family. It must explicitly identify the family-specific constraints, sensitivities, and dominant mechanisms, and explain how these features affect synthesis conditions and lead to specific scientific consequences in phase stability, growth behavior, defect formation, interfacial chemistry, and transport. Any proposed synthesis modification must therefore be justified in the context of that mechanism family, not as a generic optimization rule.

Material-property-specific target prompt (`mat_prop`): The summary must remain concise, physically meaningful, internally consistent, and directly supported by the input context. The task must refine its reasoning by jointly using `mechanism_family`, `material_name`, and `property_name`. It must not apply mechanism-family knowledge as a generic template. Instead, it must determine how the specific material composition, crystal chemistry, defect chemistry, phase behavior, and interfacial characteristics associated with `material_name` modify the relevant mechanisms, and how the target `property_name` further selects which synthesis constraints and process variables matter most. The response must therefore explain how the combination of `mechanism_family`, `material_name`, and `property_name` leads to specific synthesis sensitivities and scientifically meaningful outcomes, including effects on phase formation, crystallinity, defect population, surface/interface chemistry, and transport behavior. Any proposed synthesis modification must be justified using this combined context, not by broad mechanism-level reasoning alone.justification.

Figure 13: Specific scientific target prompts used during trajectory summarization to control the level of scientific grounding. The `default` prompt enforces basic scientific validity, `mf` adds mechanism-family-level grounding, and `mat_prop` further conditions the summary on mechanism family, material identity, and target property for more precise experience extraction and refinement.

Unstructured memory inference prompt for property prediction

Please solve the problem:
`{problem}`
When solving problems, you MUST first carefully read and understand the helpful instructions and experiences:
`{experiences}`

Figure 14: Unstructured memory inference prompt for property prediction. The `problem` field specifies the full task, including the problem definition and required output, while the `experiences` field provides accumulated knowledge from previous iterations. By conditioning the model on both the task and prior experience, the prompt encourages the generation of reasoning trajectories that leverage past knowledge and align with the target objective.

Unstructured memory summary prompt for property prediction

You are summarizing a property-prediction process for a materials synthesis task.
A trajectory and a ground-truth property are provided. Use the ground-truth property as the evaluation reference.
Do the following:
Summarize each step briefly.
Identify any errors or detours relative to the ground-truth property.
Rewrite the overall process as a Chain of Experience.
`{specific}`
`<trajectory> {trajectory} </trajectory>`
`<ground-truth property> {gt} </ground-truth property>`
Output only the step-by-step summary and the unstructured memory .

Figure 15: Unstructured memory summary prompt for property prediction. This prompt summarizes the LLM trajectory into a unstructured memory for later experience extraction and update. The `specific` field inserts the selected scientific target prompt described in Figure 13, allowing the summary to emphasize the appropriate level of scientific grounding for downstream experience refinement. The `trajectory` field contains the original task prompt together with the model response, while the `gt` field provides the ground-truth property value used as the evaluation reference.

Unstructured memory extract prompt for property prediction

You are given problem-solving attempts from an agent predicting structured target properties from synthesis recipes.
Your task is to extract and update generalizable experiences for recipe-to-property prediction.
Prefer merging over adding. Only add a new experience if it is clearly distinct and cannot be merged.
For each experience, choose one:
merge: combine overlapping or similar experiences (default choice)
modify: adjust or generalize if needed
add: only if clearly new
delete: remove if redundant or not useful
<trajectories> {trajectories} </trajectories>
<experience> {experiences} </experience>
Return a JSON list:

```
[
  {
    "option": "modify|add|merge|delete",
    "experience_id": "EXP_...",
    "experience": "When an aromatic precursor is acid-catalyzed at
        moderate temperature, crosslinking improves yield, while low
        catalysis or high temperature increases degradation.",
    "reason": "<why this was modified or added or merged or deleted>"
  }
]
```

Figure 16: Unstructured memory extraction prompt for property prediction. This prompt is used to extract and update generalizable experiences from summarized reasoning trajectories. The `trajectories` field provides the summaries generated in the previous step, which encode step-by-step reasoning grounded in the task and evaluation signal. The `experience` field contains the current memory accumulated so far. Based on these inputs, the model refines the experience set by selecting appropriate update operations (merge, modify, add, or delete), enabling iterative improvement and consolidation of reusable knowledge.

Unstructured memory update prompt for property prediction

You are consolidating experience-based knowledge for recipe-to-target-property prediction.
Your task is to refine experiences by removing redundancy and clarifying causal decision points.

[REQUIREMENTS]
- Keep each experience concise (<= 100 words) and generalizable - Write each as a single continuous reasoning flow - Merge experiences only when their causal logic is substantively overlapping - Maintain consistency in how the target property is interpreted - Remove placeholder or non-informative text (e.g., "...") - Merge redundant experiences and modify incomplete or unclear ones - Keep outputs compact; avoid unnecessary verbosity

---

[UPDATE MODES]
For each experience choose EXACTLY one:
- modify: refine incomplete, unclear, or overly specific experiences - merge: combine experiences with overlapping causal logic into one generalized experience - delete: remove redundant, low-value, or fully subsumed experiences
Rules:
- Prefer merge over modify when overlap exists - Prefer delete over keeping weak or duplicate experiences - Do NOT create new experiences; only refine existing ones - If two experiences express similar reasoning, you MUST merge them

---

[LIBRARY CONTROL]
- The library must remain small, non-redundant, and high-quality - Do not keep multiple experiences with similar logic - Favor fewer, more general experiences over many specific ones

---

[INPUTS]
<existing_experiences> {experiences} </existing_experiences>
<suggested_updates> {all_group_adv} </suggested_updates>

[OUTPUT]
Return JSON list only:

```
[
  {
    "option": "modify|merge|delete",
    "experience_id": "EXP_...",
    "experience": "<concise causal reasoning sentence>",
    "reason": "<why this was modified or merged or deleted>"
  }
]
```

Figure 17: Unstructured memory update prompt for property prediction. This prompt performs consolidation and refinement of the memory after candidate updates have been proposed. The `experiences` field provides the current memory, while the `all_group_adv` field contains the update suggestions generated in the previous extraction step. Using these inputs, the model removes redundancy, merges overlapping experiences, and improves clarity and generality, resulting in a compact and high-quality memory for subsequent iterations.

GoE inference prompt for property prediction

Please solve the problem:
`{problem}`
When solving problems, you MUST first carefully read and understand the helpful instructions and experiences:
`{experiences}`

Figure 18: GoE inference prompt for property prediction. This prompt is used to generate LLM responses that will later be transformed into Graph-of-Experiment (GoE) representations. The `problem` field contains the full task prompt, including the property prediction objective, while the `experiences` field provides the accumulated memory from prior iterations. By conditioning the model on both the task and existing memory, the prompt encourages the generation of solution trajectories that reflect prior experience and align with the property prediction objective.

GoE summary prompt for property prediction (1/2)

You are summarizing a property-prediction rollout for a broad materials-synthesis task.
The model produced a trajectory and a ground-truth structured target-property object. Treat the ground-truth property as the primary grading signal.
Please summarize the trajectory step-by-step:
1. For each step, describe what concrete synthesis cue, scientific assumption, or mechanistic judgment was used, and which experience influenced that step.
2. Using the ground-truth property object as the primary grading signal, identify detours, errors, or backtracking and explain why they may have occurred.
3. Convert the reasoning trace into a Graph-of-Experiment (GoE) that explicitly encodes branching, convergence, competition, and co-required pathways.

[GoE NODE TYPES]
Graph nodes represent the major elements of the synthesis-to-property chain and MUST use the following six node classes.
When serializing a path, write A and B as separate consecutive nodes. Do not collapse them into a single merged label.
A. Material Composition Definition: chemical composition or starting-material choices used in synthesis. Examples: - precursor composition - dopant element - support material - coating or shell composition
B. Synthesis Condition Definition: experimental conditions or process settings applied during synthesis or post-processing. Examples: - temperature - pressure - precursor ratio - pH - annealing time - atmosphere
C. Intermediate Outcome Definition: partial or local synthesis outcomes that emerge under the chosen composition and conditions. Examples: - phase formation - grain growth - impurity formation - nucleation density - partial reduction or oxidation - coating coverage
D. Interaction Between Outcomes Definition: relationships among intermediate outcomes that influence one another, compete, or must co-occur. Examples: - phase interaction - competing reaction - defect compensation - pore blocking versus surface-area gain - coupled grain growth and densification
E. Derived Characteristics Definition: structure or property mediators that connect intermediate outcomes to the final target property. Examples: - microstructure - grain-boundary property - defect density - interface quality - accessible surface area - transport-network continuity - band alignment
F. Target Property Definition: the final measured materials property to be predicted. Examples: - ionic conductivity - catalytic activity - adsorption capacity - mechanical strength - band gap

Figure 19: GoE summary prompt for property prediction, part 1 of 2. This part defines the summarization task and the six GoE node types used to structure synthesis-to-property reasoning.

GoE summary prompt for property prediction (2/2)

[EDGE RULES]
A → B: Write material-composition choices first and synthesis-condition choices second. If multiple A nodes feed the same B regime, keep the A nodes separate and state how the chosen condition acts on them.
B → C: Use the explicit condition nodes to infer the most probable intermediate outcomes.
C → D: Use D nodes to encode the dominant interaction, tradeoff, competition, or co-required balance among intermediate outcomes. Do not skip D. If the interaction evidence is weak, use D to state the key constraint that still connects C to E.
D → E: Translate those interactions or constraints into derived characteristics such as microstructure, defect density, interface quality, surface accessibility, transport continuity, or band alignment.
E → F: Use derived characteristics to explain the final target property.
Branching, merging, conditional paths, and uncertainty-bearing paths MUST be preserved and described explicitly.

CASE 1 - Independent Divergent Mechanisms
Example pattern: A1 → B1 → C1 → D1 → E1 → F A2 → B2 → C2 → D2 → E2 → F
- C1 and C2 are independent - They produce different downstream consequences
Label format: (path1 independent of path2)
CASE 2 - Independent Convergent Mechanisms
Example pattern: A1 → B1 → C1 → D1 → E1 → F A2 → B2 → C2 → D2 → E2 → F
- E1 and E2 independently contribute to the same target property
Label format: (E1, E2 → F)
CASE 3 - Co-Required Mechanisms
Example pattern: A1 → B1 → C1 → D1 → E1 → F A2 → B2 → C2 → D1 → E1 → F
- The combined effects are required - Neither path alone is sufficient
Label format: (jointly required)

[GoE REQUIREMENTS]
GoE MUST contain:
1. GoE Node List - List all A, B, C, D, E, and F nodes extracted from the reasoning trace.
2. GoE Edge List - Specify all directed edges. - Explicitly label divergent, convergent, and jointly required cases where applicable.
3. Dominant GoE Path Extraction - Identify the most important causal routes from composition and condition nodes to the target property. - Output each as: Path#: A patterns → B patterns → C patterns → D patterns → E patterns → F
4. Mechanistic Interpretation - Explain why each dominant path is causally important. - Note where the model's reasoning matched or missed the ground-truth property.
5. Optional Experiment Candidates - Suggest targeted experiments that would directly test the uncertain or disputed GoE edges.
`{specific}`
`<trajectory> {trajectory} </trajectory>`
`<ground-truth property> {gt} </ground-truth property>`
Only return the step-by-step trajectory summary and the resulting GoE analysis.

Figure 20: GoE summary prompt for property prediction, part 2 of 2. This part defines directed edge rules, branching and convergence cases, required GoE outputs, and input fields. The `trajectory` field contains the full task prompt together with the model-generated response, and the `gt` field provides the ground-truth property used as the primary evaluation signal. The `specific` field inserts the scientific target prompt, conditioning the summarization toward the desired level of scientific grounding.

GoE extract prompt for property prediction (1/2)

You are given a set of problem-solving attempts made by an agent system that predicts target properties from synthesis recipes.
Your task is to extract and refine generalizable experiences for recipe-to-target-property prediction under a given mechanism family.

[CORE OBJECTIVE]
Produce experiences that:
- follow the A–F causal structure internally - but are expressed as a natural, continuous reasoning flow in fluent text
DO NOT output symbolic chains such as:
A → B → C → D → E → F
Instead, embed (A), (B), (C), (D), (E), and (F) naturally inside sentences.

[EXPERIENCE STRUCTURE (MANDATORY, BUT NATURAL IN FORM)]
Each experience MUST include all six components:
- (A) Material composition - (B) Synthesis condition - (C) Intermediate outcome - (D) Interaction between outcomes - (E) Derived characteristics - (F) Target property
Requirements:
- A and B must appear as distinct causal inputs, not merged into one label - All six components must be explicitly represented in the reasoning - Use inline labels (A)–(F) for clarity - The experience must read like a causal explanation, not like a graph or a template

---

[WRITING STYLE REQUIREMENTS]
- Start from concrete cues in the recipe or material context, especially (A) and (B) - Explain what those cues imply mechanistically through (C) - Describe how cooperative, competing, limiting, or jointly required effects shape (D) - Show which structural, compositional, or functional state emerges as (E) - Conclude with the resulting target property as (F)
Use causal language such as:
- "this suggests" - "under these conditions" - "this leads to" - "however if" - "by contrast" - "therefore"
Avoid:
- symbolic arrows - bullet-style logic chains inside the experience text - fragmented phrases - empty fillers or placeholder text such as "..."
Each experience should be concise, generalizable, and no longer than 100 words.

[EXPERIENCE IDS]
Experience IDs are arbitrary library labels. They are NOT GoE nodes and MUST NOT be interpreted as A, B, C, D, E, or F.
All experience IDs MUST use the prefix:
EXP_

[UPDATE MODES]
For each experience choose EXACTLY one mode:
MODE: modify
Use when the experience is directionally correct and still contains a distinct usable insight, but needs improvement, such as:
- incomplete or weak causal linkage across A–F - overly narrow scope that should be generalized - insufficient mechanistic grounding - wording redundancy or minor cleanup - branch conditions that need to be made explicit
Use modify by default when the experience is salvageable and not clearly new.
Do NOT use modify if:
- the experience is fundamentally redundant with another and should be merged - the core logic is unsound, unsupported, or non-informative and should be deleted

---

Figure 21: GoE experience extraction prompt for property prediction, part 1 of 2. This part defines the extraction objective, required A–F experience structure, writing style, experience ID convention, and the `modify` update mode.

GoE extract prompt for property prediction (2/2)

MODE: add
Use only under a high threshold.
Choose add only if ALL of the following are true:
1. the experience expresses a causal pathway not already covered by any existing experience 2. its prediction logic cannot be captured by modifying an existing experience 3. its core mechanism cannot be represented as a branch within a merged experience 4. it adds distinct value for predicting the target property
Valid reasons to add include:
- a genuinely new causal pathway - a missing interaction pattern at (D) - a missing derived characteristic at (E) - a distinct target-property signature under a different regime
Do NOT use add for:
- paraphrases of existing logic - narrower variants of an existing regime - condition tweaks that can be expressed with if / when / unless - repeated target-property signatures driven by the same causal logic
If there is any substantial overlap with existing experiences, prefer modify or merge instead of add.

MODE: merge
Use when two or more experiences share substantially overlapping causal logic, even if they differ in:
- example material or precursor - exact operating window - wording emphasis - which part of the same mechanism they highlight
Choose merge whenever separate experiences would remain partially redundant.
A valid merge must:
- preserve meaningful differences through explicit if / when / unless conditions - rewrite the experiences into one coherent causal flow - remove repeated claims, placeholder text, and low-information phrasing - keep only distinctions that change prediction logic or applicability - remain concise and not become a concatenated summary
When in doubt between add and merge, prefer merge if the target-property reasoning is materially the same.
Do NOT use merge if:
- the experiences represent different mechanisms - the governing regime changes the prediction logic in a way that cannot be captured as a branch

---

MODE: delete
Use when the experience should not remain in the library because it is:
- empty, placeholder-like, or non-informative - fully subsumed by another experience after merge - mechanistically incorrect, misleading, or unsupported - too vague to support transferable prediction even after attempted refinement
Delete aggressively when an experience adds no unique predictive leverage.
Do NOT use delete if:
- the experience contains recoverable insight that can be preserved by modify or merge

---

[CONSERVATIVE UPDATE POLICY]
Apply the most conservative valid mode:
- prefer modify over add - prefer merge over keeping overlapping experiences separate - prefer delete over retaining empty or non-actionable text
Do not preserve multiple experiences that express the same target-property reasoning with only superficial variation.

[REQUIRED INTERNAL DECISION FACTORS]
Before updating any experience, you must internally consider:
1. relevance regime 2. applicability conditions and explicit exclusion conditions 3. trigger features in the recipe 4. trigger features in the active material or evaluation setup 5. transferable patterns across material systems
These factors must be reflected in the final experience text whenever supported by the evidence.

[OUTPUT FORMAT]
Return JSON list only:

```
[
  {
    "option": "modify|add|merge|delete",
    "experience_id": "EXP_...",
    "experience": "When a recipe combines an aromatic polyphenol such as phloroglucinol (A) with a solid-state pyrolysis
          step at 160-200 (*@$^\circ$@*)C (B), this suggests that acid-catalyzed dehydration and polycondensation (C) can
          proceed efficiently at relatively low temperatures; under these conditions, the synergy between catalyst activity
          and thermal activation promotes rapid crosslink formation over volatile evolution (D), leading to early formation
          of an insoluble amorphous carbon network (E) and consequently high mass retention and yield (~80%) (F); however,
          if catalyst loading is negligible, this cooperative effect weakens and delays network formation, and if
          temperature exceeds ~250 (*@$^\circ$@*)C, increased degradation and volatilization reduce yield.",
    "reason": "<why this was modified or added or merged or deleted>"
  }
]
```

Note that your updated experiences may not need to cover all two options. Only using one type of updates is also very good.
<trajectories> {trajectories} </trajectories>
<experience> {experiences} </experience>

Figure 22: GoE experience extraction prompt for property prediction, part 2 of 2. This part defines `add`, `merge`, and `delete` modes, the conservative update policy, required decision factors, output schema, and input fields. The `trajectories` field provides the GoE summaries generated in the previous step, where reasoning is explicitly structured as causal relationships among nodes (A)–(F) and their connecting edges. The `experiences` field contains the current memory of accumulated experiences. Using these inputs, the model generates updated experiences that follow the same A–F causal structure, expressed as natural continuous reasoning.

GoE update prompt for property prediction

You are consolidating GoE-based experiences for recipe-to-target-property prediction.
Produce a compact revision plan that removes redundancy and clarifies causal decision points.
Requirements:
1. Keep each revised experience concise (<= 100 words) while retaining essential detail and generalizability.
2. Preserve the explicit causal chain format: A → B → C → D → E → F.
3. Merge experiences only when their underlying causal logic is substantively overlapping.
4. Preserve branch-specific applicability using explicit if, when, or unless conditions, along with the prediction heuristic.
5. Maintain consistency in branch logic and target-property interpretation.
6. Remove placeholder or non-informative text (e.g., "...") and ensure all content is meaningful.
7. Merge redundant or highly similar experiences, and modify those that are incomplete or insufficiently specified.
8. Prevent experiences from becoming overly long during merging or modification; keep them concise and focused on decision-relevant details.
Inputs:
<existing_experiences> {experiences} </existing_experiences>
<suggested_updates> {all_group_adv} </suggested_updates>
Return JSON list only:

```
[
  {
    "option": "modify|merge|delete",
    "experience_id": "EXP_...",
    "experience": "When a recipe combines an aromatic polyphenol such as
    phloroglucinol (A) with a solid-state pyrolysis step at 160-200 °C (B), this
    suggests that acid-catalyzed dehydration and polycondensation (C) can proceed
    efficiently at relatively low temperatures; under these conditions, the synergy
    between catalyst activity and thermal activation promotes rapid crosslink
    formation over volatile evolution (D), leading to early formation of an insoluble
    amorphous carbon network (E) and consequently high mass retention and yield (~80%)
     (F); however, if catalyst loading is negligible, this cooperative effect weakens
    and delays network formation, and if temperature exceeds ~250 °C, increased
    degradation and volatilization reduce yield.",
    "reason": "<why this was modified or merged or deleted>"
  }
]
```

Figure 23: GoE experience update prompt for property prediction. This prompt consolidates and refines the accumulated experiences based on previously suggested update directions. The `experiences` field contains the current memory of experiences collected so far, while the `all_group_adv` field provides the candidate update actions proposed by the model in the previous extraction step. Using these inputs, the model removes redundancy, merges overlapping causal patterns, and clarifies decision-relevant relationships while preserving the explicit A–F causal structure (composition to target property). The resulting updates produce a compact, consistent, and high-quality memory that supports improved causal reasoning and generalization in subsequent iterations.

Unstructured memory inference prompt for synthesis planning

Please solve the problem:
`{problem}`
When solving problems, you MUST first carefully read and understand the helpful instructions and experiences:
`{experiences}`

Figure 24: Unstructured memory inference prompt for inverse material design. This prompt is used to generate LLM responses for target-property-to-recipe inverse design, which will later be utilized for updating memory. The `problem` field contains the full task prompt, including the target property specification and design objective, while the `experiences` field provides the accumulated memory from prior iterations. By conditioning the model on both the design goal and existing memory, the prompt encourages the generation of synthesis recipes that align with the target property and reflect prior experience, forming suitable inputs for subsequent summarization and experience refinement.

Unstructured memory summary prompt for synthesis planning

You are summarizing an inverse-design process for a materials synthesis task.
A trajectory, a predicted (or judged) property outcome, and a reference recipe are provided. Use target alignment as the main evaluation signal.
Do the following:
Summarize each step briefly.
Identify which steps helped or harmed the target alignment.
Rewrite the process as a Chain of Experience (CoE).
Provide the final generated recipe.
`{specific}`
`<trajectory> {trajectory} </trajectory>`
`<prediction_or_judgment> {prediction} </prediction_or_judgment>`
`<ground-truth property> {gt} </ground-truth property>`
Output only the step-by-step summary, the experiences, and the final recipe.

Figure 25: Unstructured memory summary prompt for inverse material design. This prompt summarizes the inverse-design trajectory into a unstructured memory for subsequent experience extraction and refinement. The `trajectory` field contains the full task prompt together with the LLM-generated response, corresponding to the newly proposed synthesis recipe. The `prediction` field provides the predicted target-property value for the generated recipe, serving as the primary signal for evaluating alignment with the design objective. The `gt` field contains the ground-truth synthesis recipe used as a reference. The `specific` field inserts the scientific target prompt described earlier, conditioning the summary to reflect the desired level of scientific grounding. Using these inputs, the model produces a step-by-step summary and a expriences that highlights decision-making quality and supports downstream experience extraction and updating.

Unstructured memory extract prompt for synthesis planning

You are given trajectories from an agent generating synthesis recipes to match target properties.
Your task is to extract and update generalizable experiences for inverse design.
Prefer merging over adding. If experiences share similar design logic, merge them.
For each experience, choose one:
merge: combine overlapping experiences (default)
modify: refine if needed
add: only if clearly new and not mergeable
delete: remove if redundant or not useful
Keep experiences concise and general.
Return a JSON list:

```
[
  {
    "option": "modify|add|merge|delete",
    "experience_id": "EXP_...",
    "experience": "When rapid nucleation is followed by short heating, small grains
     form, while longer heating or impurities lead to larger grains.",
  }
]
```

```
<trajectories> {trajectories} </trajectories>
<experience> {experiences} </experience>
```

Figure 26: Unstructured memory extraction prompt for inverse material design. This prompt extracts and updates generalizable experiences from summarized inverse-design trajectories. The `trajectories` field contains the summaries generated in the previous step, where each trajectory reflects the task prompt and the corresponding LLM-generated synthesis recipe along with its evaluation signal. The `experience` field provides the current memory of accumulated experiences. Using these inputs, the model identifies reusable design patterns and refines them through update operations (`merge`, `modify`, `add`, or `delete`), enabling iterative improvement and consolidation of knowledge for target-property-to-recipe generation.

Unstructured memory update prompt for synthesis planning

You are consolidating experiences for target-property-to-recipe inverse design.
Your task is to reduce redundancy and improve clarity.
Prefer merging over modifying. If experiences share similar logic, merge them.
For each experience, choose one:
merge: combine overlapping experiences (default)
modify: refine if needed
delete: remove if redundant or not useful
Keep experiences concise and general.
`<existing_experiences> {experiences} </existing_experiences>`
`<suggested_updates> {all_group_adv} </suggested_updates>`
Return a JSON list:

```
[
  {
    "option": "modify|merge|delete",
    "experience_id": "EXP_...",
    "experience": "When rapid nucleation is followed by short heating, small grains
     form, while longer heating or impurities lead to larger grains.",
    "reason": "<why this was modified or merged or deleted>"
  }
]
```

Figure 27: Unstructured memory update prompt for inverse material design. This prompt consolidates and refines the accumulated experiences for target-property-to-recipe inverse design. The `experiences` field contains the current memory of experiences collected across iterations, while the `all_group_adv` field provides the candidate update directions proposed in the previous extraction step. Using these inputs, the model removes redundant or overlapping experiences, merges similar design patterns, and improves clarity and generality. The resulting updated memory forms a compact and consistent knowledge base that supports more effective and transferable inverse material design in subsequent iterations.

GoE inference prompt for synthesis planning

Please solve the problem:
`{problem}`
When solving problems, you MUST first carefully read and understand the helpful instructions and experiences:
`{experiences}`

Figure 28: GoE inference prompt for inverse material design. This prompt is used to generate LLM responses for target-property-to-recipe inverse design, which will later be analyzed using Graph-of-Experiment (GoE) representations. The `problem` field contains the full task prompt, including the desired target property specification and design objective, while the `experiences` field provides the accumulated memory from prior iterations. By conditioning the model on both the design goal and existing memory, the prompt encourages the generation of synthesis recipes that reflect prior experience and align with the target property.

GoE summary prompt for synthesis planning (1/2)

You are summarizing an inverse-design rollout for a broad materials-synthesis task.
The model produced a trajectory and a predicted or judged target-property outcome for the generated recipe. Treat target alignment as the primary grading signal.
Please summarize the trajectory step-by-step:
1. For each step, describe what concrete recipe-design action, mechanism choice, or control-variable decision was taken, and which experience influenced that step.
2. Using the predicted or judged property alignment as the primary grading signal, identify which steps likely improved versus harmed the design objective, and explain why.
3. Convert the reasoning trace into a Graph-of-Experiment (GoE) that explicitly encodes branching, convergence, competition, and co-required pathways.
4. End with the final generated recipe in concise form.

[GoE NODE TYPES]
Graph nodes represent the major elements of the synthesis-to-property chain and MUST use the following six node classes.
When serializing a path, write A and B as separate consecutive nodes. Do not collapse them into a single merged label.
A. Material Composition Definition: chemical composition or starting-material choices used in synthesis. Examples: - precursor composition - dopant element - support material - coating or shell composition
B. Synthesis Condition Definition: experimental conditions or process settings applied during synthesis or post-processing. Examples: - temperature - pressure - precursor ratio - pH - annealing time - atmosphere
C. Intermediate Outcome Definition: partial or local synthesis outcomes that emerge under the chosen composition and conditions. Examples: - phase formation - grain growth - impurity formation - nucleation density - partial reduction or oxidation - coating coverage
D. Interaction Between Outcomes Definition: relationships among intermediate outcomes that influence one another, compete, or must co-occur. Examples: - phase interaction - competing reaction - defect compensation - pore blocking versus surface-area gain - coupled grain growth and densification
E. Derived Characteristics Definition: structure or property mediators that connect intermediate outcomes to the final target property. Examples: - microstructure - grain-boundary property - defect density - interface quality - accessible surface area - transport-network continuity - band alignment
F. Target Property Definition: the final measured property or target-property outcome that the designed recipe is trying to achieve. Examples: - ionic conductivity - catalytic activity - adsorption capacity - mechanical strength - band gap

Figure 29: GoE summary prompt for inverse material design, part 1 of 2. This part defines the inverse-design summarization task and the six GoE node types used to structure target-property-to-recipe reasoning.

GoE summary prompt for synthesis planning (2/2)

[EDGE RULES]
A → B: Write material-composition choices first and synthesis-condition choices second. If multiple A nodes feed the same B regime, keep the A nodes separate and state how the chosen condition acts on them.
B → C: Use the explicit condition nodes to infer the most probable intermediate outcomes of the designed recipe.
C → D: Use D nodes to encode the dominant interaction, tradeoff, competition, or co-required balance among intermediate outcomes. Do not skip D. If the interaction evidence is weak, use D to state the key constraint that still connects C to E.
D → E: Translate those interactions or constraints into derived characteristics such as microstructure, defect density, interface quality, surface accessibility, transport continuity, or band alignment.
E → F: Use derived characteristics to explain how the generated recipe is expected to move the target property.
Branching, merging, conditional paths, and uncertainty-bearing paths MUST be preserved and described explicitly.

[MECHANISTIC COMPOSITION RULES]
CASE 1 - Independent Divergent Mechanisms
Example pattern: A1 → B1 → C1 → D1 → E1 → F A2 → B2 → C2 → D2 → E2 → F
- C1 and C2 are independent - They produce different downstream consequences
Label format: (path1 independent of path2)
CASE 2 - Independent Convergent Mechanisms
Example pattern: A1 → B1 → C1 → D1 → E1 → F A2 → B2 → C2 → D2 → E2 → F
- E1 and E2 independently contribute to the same target property
Label format: (E1, E2 → F)
CASE 3 - Co-Required Mechanisms
Example pattern: A1 → B1 → C1 → D1 → E1 → F A2 → B2 → C2 → D1 → E1 → F
- The combined effects are required - Neither path alone is sufficient
Label format: (jointly required)

[GoE REQUIREMENTS]
GoE MUST contain:
1. GoE Node List - List all A, B, C, D, E, and F nodes extracted from the reasoning trace.
2. GoE Edge List - Specify all directed edges. - Explicitly label divergent, convergent, and jointly required cases where applicable.
3. Dominant GoE Path Extraction - Identify the most important causal routes from recipe choices to the target property. - Output each as: Path#: A patterns → B patterns → C patterns → D patterns → E patterns → F
4. Mechanistic Interpretation - Explain why each dominant path is causally important. - Note where the model's recipe design was strong, weak, or internally inconsistent.
5. Optional Experiment Candidates - Suggest experiments that would directly test the uncertain or disputed GoE edges.
`{specific}`
`<trajectory> {trajectory} </trajectory>`
`<prediction_or_judgment> {prediction} </prediction_or_judgment>`
`<GT_recipe> {gt} </GT_recipe>`
Only return the step-by-step trajectory summary, the GoE analysis, and the final generated recipe.

Figure 30: GoE summary prompt for inverse material design, part 2 of 2. This part defines directed edge rules, mechanistic composition cases, required GoE outputs, and input fields. The `trajectory` field contains the full task prompt together with the LLM-generated synthesis recipe, while the `prediction` field provides the predicted target-property value of the generated recipe, serving as the primary signal for evaluating alignment with the design objective. The `gt` field contains the ground-truth synthesis recipe that achieves the target property. The `specific` field inserts the scientific target prompt to control the level of domain grounding.

GoE extract prompt for synthesis planning (1/2)

An agent system is provided with a set of experiences and has attempted to generate synthesis recipes across diverse materials systems to achieve a target property specification.
Each trajectory summary includes a predicted or judged target-property outcome.
Your goal is to extract generalizable, actionable experiences that improve target-property-to-recipe inverse design.
1. Make generalizable experiences for the task (`target_property_to_recipe`), mechanism family, and target property.
2. Update Existing Experiences

[IMPORTANT NOTE ABOUT EXPERIENCE IDs]
Experience IDs (for example EXP_01, EXP_12, EXP_A7) are arbitrary labels that identify experiences in the experience library.
They are NOT GoE graph nodes and MUST NOT be interpreted as A, B, C, D, E, or F nodes.
To prevent confusion, ALL experience IDs MUST use the prefix EXP_.

[UPDATE MODES]
For each experience you handle, choose EXACTLY one mode:
MODE: modify
Use when the experience is directionally correct but incomplete, too specific, insufficiently grounded, or missing important applicability conditions.
MODE: add
Use when the GoE analysis reveals:
- a new design pathway - a missing interaction node - a missing derived characteristic - a missing control-variable pattern - or a missing target-property signature
Use add only when the new logic is not already covered by an existing experience and cannot be cleanly absorbed by modification or merging.
MODE: merge
Use when two or more experiences are substantively overlapping in causal logic, control logic, applicability, or design implication, even if they differ in wording, examples, or minor operating details.
A valid merge should:
- combine overlapping experiences into one more general and transferable experience - preserve meaningful branch-specific differences using explicit if, when, or unless conditions - remove redundancy rather than concatenate repeated content - stay concise and focused on decision-relevant logic
Prefer merge over add when multiple experiences express essentially the same design pattern.
MODE: delete
Use when an experience should not remain in the library because it is:
- redundant and fully covered by another experience - too vague or non-informative to be useful - a placeholder, fragment, or low-information restatement - mechanistically unsupported, misleading, or inconsistent with the target-property objective - no longer necessary after its meaningful content has been preserved elsewhere through modification or merging
Prefer delete over retaining low-value or duplicate experiences.

[EXPERIENCE LIBRARY CONTROL]
The experience library should remain compact, non-redundant, and useful for future inverse-design reasoning.
Therefore:
- do not preserve multiple experiences that express the same design logic with only superficial variation - prefer merge instead of keeping near-duplicates separate - prefer modify instead of add when an existing experience can absorb the new insight - prefer delete for placeholder-like, low-information, or fully subsumed experiences - avoid creating unnecessary new experiences that do not add distinct design value
The goal is to improve coverage and transferability without allowing the library to grow through redundancy.

Figure 31: GoE experience extraction prompt for inverse material design, part 1 of 2. This part defines the extraction objective, experience ID convention, update modes, and experience-library control policy.

GoE extract prompt for synthesis planning (2/2)

[REQUIRED METADATA BEFORE UPDATING EACH EXPERIENCE]
For every updated experience, you MUST internally consider:
1. Relevance regime 2. Applicability conditions and explicit exclusion conditions 3. Trigger features in the target request 4. Trigger features in the candidate recipe 5. Transferable design patterns across material systems
These factors MUST be incorporated into the final experience text.

[EXPERIENCE STRUCTURE REQUIREMENT]
Each updated experience MUST follow the GoE-compatible causal form:
A → B → C → D → E → F
Where:
- A = Material Composition - B = Synthesis Condition - C = Intermediate Outcome - D = Interaction Between Outcomes - E = Derived Characteristics - F = Target Property
Additional requirements:
- A and B must be written as separate clauses, not merged into one label. - All six node classes must appear in every experience. - Applicability must be branch-specific and concrete. Use explicit if, when, unless, or regime statements instead of generic claims. - Conditions must be operationally specific whenever the GoE supports them: precursor identity, ratio, solvent system, atmosphere, temperature window, time window, pH, post-treatment, loading, or thickness. - Avoid generic advice such as "increase temperature" or "improve morphology" without a specific regime and mechanism-grounded reason. - The experience should explicitly encode divergent, convergent, or jointly required effects when the GoE supports them.

[OUTPUT FORMAT]
Return JSON list only:

```
[
  {
    "option": "modify",
    "modified_from": "EXP_2",
    "experience_id": "EXP_2",
    "experience": "When a Zn-Ce oxide system with a Zn:Ce ratio of ~0.3:0.7 using nitrate precursors (A) is synthesized via microwave-assisted
      hydrothermal treatment followed by short annealing at ~450°C (B), this promotes high nucleation density with wurtzite phase formation (C);
      under these conditions, rapid nucleation competes with grain growth and grain boundary migration (D), leading to kinetically arrested grain
      growth with a low-defect microstructure (E) and resulting in crystallite sizes within the 34-44 nm range (F); however, residual nitrates or
      increased thermal budget shift the system toward enhanced diffusion and grain coarsening, exceeding the target size.",
  }
]
```

Note that your updated experiences may not need to cover all two options. Only using one type of updates is also very good.
`<trajectories> {trajectories} </trajectories>`
`<experience> {experiences} </experience>`

Figure 32: GoE experience extraction prompt for inverse material design, part 2 of 2. This part defines required metadata, the GoE-compatible A–F experience structure, output schema, and input fields. The `trajectories` field contains the GoE summaries generated in the previous step, where each trajectory encodes the relationship between the task prompt, the generated synthesis recipe, and its predicted target-property outcome. The `experiences` field provides the current memory of accumulated experiences. Using these inputs, the model identifies reusable causal design patterns expressed in the A–F structure (composition to target property) and updates them through `modify`, `add`, `merge`, or `delete` operations.

GoE update prompt for synthesis planning

You are consolidating experience-based knowledge for target-property-to-recipe inverse design.
Produce a compact revision plan that removes redundancy and sharpens causal control points.
Requirements:
1. Keep each revised experience concise (<= 120 words) while retaining sufficient mechanistic and control detail for decision-making.
2. Express each experience as a continuous, experience-driven reasoning flow (Chain of Experience).
3. Merge experiences only when their underlying reasoning and control logic are substantively overlapping.
4. Preserve branch-specific applicability using explicit if, when, or unless conditions when necessary.
5. Maintain consistency in how the design target is interpreted and addressed.
6. Remove placeholder or non-informative text (e.g., "...") and ensure all content is meaningful.
7. Merge redundant or highly similar experiences, and modify those that are incomplete, underspecified, or unclear.
8. Prevent experiences from becoming overly long during merging or modification; keep them concise and focused on decision-relevant control variables.
Inputs:
`<existing_experiences> {experiences} </existing_experiences>`
`<suggested_updates> {all_group_adv} </suggested_updates>`
Guidelines for rewriting experiences:
- Start from concrete cues in the target requirement or recipe context - Explain how those cues inform design decisions based on prior experience - Explicitly link control variables (composition, ratios, solvent, temperature, time, atmosphere, pH, post-treatment, etc.) to expected outcomes - Highlight key decision points, tradeoffs, competing effects, and limiting factors - Describe how the reasoning progresses toward achieving the target property - Keep the reasoning sequential, causal, and transferable across material systems - Use condition-specific logic only when necessary to distinguish regimes - Avoid any reference to predefined structures, nodes, staged templates, or symbolic chain formats - Do not use expressions such as A → B → ...

[OUTPUT FORMAT]
Return JSON list only:

```
[
  {
    "option": "modify|merge|delete",
    "experience_id": "EXP_...",
    "experience": "When a Zn-Ce oxide system with a Zn:Ce ratio of ~0.3:0.7 using
    nitrate precursors (A) is synthesized via microwave-assisted hydrothermal
    treatment followed by short annealing at ~450°C (B), this promotes high nucleation
     density with wurtzite phase formation (C); under these conditions, rapid
    nucleation competes with grain growth and grain boundary migration (D), leading to
     kinetically arrested grain growth with a low-defect microstructure (E) and
    resulting in crystallite sizes within the 34-44 nm range (F); however, residual
    nitrates or increased thermal budget shift the system toward enhanced diffusion
    and grain coarsening, exceeding the target size.",
    "reason": "<why this was modified or merged or deleted>"
  }
]
```

Figure 33: GoE experience update prompt for inverse material design. This prompt consolidates and refines the accumulated experiences for target-property-to-recipe inverse design after candidate updates have been proposed. The `experiences` field contains the current memory of experiences collected across iterations, while the `all_group_adv` field provides the update suggestions generated in the previous extraction step. Using these inputs, the model removes redundancy, merges overlapping causal patterns, and sharpens decision-relevant reasoning while preserving clarity and generality. The resulting updates produce a compact and consistent memory that supports improved causal reasoning and more effective inverse material design in subsequent iterations.

E QD-LED Dataset

The QD-LED dataset is an in-house wet-lab dataset for device-level materials design. Each record describes a complete quantum-dot light-emitting diode experiment, including the multilayer device stack, experimental equipment, layer-by-layer synthesis and fabrication procedure, characterization protocol, and measured device performance. The device stack consists of a transparent anode, hole-injection and hole-transport layers, a quantum-dot emissive layer, an electron-transport layer (ETL), and a metal cathode. The recipe covers substrate preparation, functional organic/inorganic layer deposition, quantum-dot emissive-layer coating, ETL synthesis and deposition, cathode evaporation, encapsulation, and device-level measurement.

The dataset contains two device-level target properties. *EQEmax* is the maximum external quantum efficiency extracted from the J–V–L/EQE sweep of an encapsulated QD-LED. *Lifetime(T95)* is the operational lifetime measured as the time for luminance to decrease to 95% of its initial value under constant-current aging. These targets depend on coupled device mechanisms, including carrier injection through the transport layers, charge balance, QD/ETL interfacial passivation, non-radiative recombination, morphology, and device stability.

In property prediction, the model receives the device context and a complete synthesis/process recipe, and predicts either *EQEmax* or *Lifetime(T95)*. In *synthesis planning*, the model receives the device context and a desired target-property condition, and generates a complete device synthesis and fabrication procedure expected to realize that target. While our standardized evaluation scores each target property separately for comparability, the practical device-design objective is multi-property-conditioned synthesis planning: the model should ultimately propose a recipe that jointly satisfies device-level targets such as high *EQEmax* and long *Lifetime(T95)*.

QD-LED instantiation differs from OMG and many open-source materials-design benchmarks, which primarily expose material-level records, isolated synthesis recipes, or target properties measured for individual materials rather than complete device outcomes. The QD-LED task is device-level property-conditioned synthesis planning: the desired output is an executable full-device synthesis and processing recipe conditioned on target device metrics after full-stack fabrication and characterization. This formulation is industrially important because materials-development decisions are ultimately judged by finished-device performance, not by isolated material properties alone. It is also substantially more challenging, since *EQEmax* and *Lifetime(T95)* are often coupled through trade-offs: process changes that improve charge injection and efficiency can affect interfacial stability, while recipes that improve lifetime can alter transport, morphology, or recombination balance. Effective inverse design must therefore reason over coupled device-level objectives under sparse labels, where each wet-lab label requires a complete fabrication and measurement cycle.

Because the records are proprietary, we report only aggregate evaluation results in the main paper. For security and confidentiality reasons, all QD-LED-related results and examples released with the paper are anonymized before disclosure. Figure 34 shows one anonymized QD-LED prompt record, illustrating the full-device recipe, measured property of reference recipe, and held-out query target used for evaluation. The in-house workflow uses the same structured-memory training and inference protocol as the public benchmark, with memories built from the QD-LED training split and evaluated on held-out QD-LED records.

F Additional Qualitative Example

Figure 35 provides an additional example of how learned memory improves property prediction. The baseline over-anchors to a superficial analogy with mechanical disruption and predicts high recovery. In contrast, memory-augmented reasoning identifies the relevant biological bottleneck. PHB is intracellular, and mechanical grinding without lysis chemistry can leave substantial polymer trapped in biomass debris. This supports the main observation in Section 5 that memory improves calibration by moving the model from generic process analogy toward the mechanism controlling the queried property.

Figure 36 shows a complementary QD-LED example from the in-house device dataset. Here, the model must weigh competing device-level effects: higher Mg content and lower ETL conductivity can penalize charge transport, while reduced precursor water, lower interfacial trap density, and smoother morphology can stabilize the QD/ETL interface. The baseline and unstructured memory over-penalize the conductivity drop, whereas GoE

An anonymized QD-LED property-prediction sample

Task. Given a reference QD-LED recipe with measured *EQEmax* and a query recipe under the same material/property context, predict the query `target_property` with a mechanism-grounded rationale.

Shared device context. Patterned ITO / PEDOT:PSS / PF8Cz / fixed green InP/ZnSe/ZnS QD emissive layer / inorganic oxide ETL / Al cathode. The target property is *EQEmax*, extracted from the encapsulated-device J–V–L/EQE sweep in nitrogen.

Recipe component	Reference record	Query record
Equipment and setup	Nitrogen glovebox, UV–ozone cleaner, programmable spin coater, leveled hot plates, controlled base-injection setup, sonication/centrifugation tools, high-vacuum thermal evaporator, and calibrated device-measurement station.	Same equipment and measurement setup.
Anode	Patterned ITO glass, cleaned by detergent/water/acetone/isopropanol ultrasonication, dried under N_2, and UV–ozone treated before coating.	Same substrate preparation and surface activation.
Hole-injection layer	PEDOT:PSS AI4083, filtered, spin coated from 45 μL at 4000 rpm, and baked at 150 °C for 30 min; measured thickness 39 $\pm$ 2 nm.	Same layer recipe and measured context.
Hole-transport layer	PF8Cz, 8 mg/mL in chlorobenzene, filtered, spin coated at 3000 rpm, and annealed at 120 °C for 30 min; measured thickness 29 $\pm$ 2 nm.	Same layer recipe and measured context.
QD emissive layer	Green InP/ZnSe/ZnS QDs, 20 mg/mL in n-octane, filtered, spin coated at 2000 rpm, and annealed at 80 °C for 7 min; measured thickness 23 $\pm$ 1 nm.	Same layer recipe and measured context.
Inorganic oxide ETL	$Zn_{0.830}Mg_{0.143}Li_{0.026}O$ from metal-acetate/LiCl precursors in DMSO with TMAH base injection, colloid aging, antisolvent precipitation, washing, redispersion to 21.3 mg/mL, spin coating at 1923 rpm, and 75 °C annealing; measured thickness 25.3 nm.	$Zn_{0.816}Mg_{0.151}Li_{0.033}O$ from the corresponding precursor mixture with TMAH base injection, colloid aging, antisolvent precipitation, washing, redispersion to 24.5 mg/mL, spin coating at 2390 rpm, and 55–61 °C annealing; measured thickness 28.2 nm.
Cathode and encapsulation	Aluminum cathode, 100 nm, thermally evaporated through a shadow mask after transfer to high vacuum and encapsulated under nitrogen.	Same cathode and encapsulation procedure.
Characterization and property	J–V–L/EQE measured on the encapsulated device in nitrogen; measured *EQEmax*: 22.65%.	Same characterization protocol; query *EQEmax* is hidden during prediction and used only for evaluation. Ground truth: 23.31%.

Figure 34: An anonymized QD-LED sample extracted from the in-house property-prediction prompt records. The example shows the full layer-wise recipe exposed to the model, including substrate preparation, functional-layer deposition, quantum-dot emissive-layer coating, ETL synthesis and deposition, cathode formation, encapsulation, and device-level characterization. The query target value is withheld at inference time and used only for evaluation.

preserves the conditional pathway in which interface passivation dominates the transport penalty. This supports the QD-LED results in Table 1, where structured memory provides a clearer benefit in the more entangled device-level setting.

G LLM-as-a-Judge Prompt

The LLM-as-a-judge prompt is designed to separate answer quality from reasoning quality and to make the evaluation target explicit for each task. For property prediction, numerical target-value alignment is computed programmatically after parsing and unit normalization, so the judge is asked only to assess whether the reasoning trace is grounded in the given record and mechanistically valid. For *synthesis planning*, where many procedures can plausibly satisfy the same target, the judge scores both the parsed synthesis procedure and the reasoning trace.

The prompt also prevents the observed synthesis procedure from being treated as a unique string-level answer. Instead, for synthesis planning it is provided as evidence for the likely outcome state, and the judge is instructed to reward chemically plausible alternative routes when they converge to a similar phase, composition, defect, morphology, interface, or loading regime. Figure 37 summarizes the judge payload and metric schema, while Figures 38–40 provide the task-specific judge prompts and scoring anchors.

G.1 Shared-Context QD-LED Memories

Figure 41 shows a memory from property prediction task in QD-LED. The memory explicitly links EQE_{max} and operational lifetime (T95) with scientific context: Zn-based ETL transport, CBM alignment, morphology, defect chemistry, passivation, and roughness jointly determine whether a device improves efficiency, lifetime, or both.

Property prediction with PHB product yield without lysis buffer

Query. Predict `product_yield` for PHB recovery from *Bacillus* biomass by grinding without lysis buffer. **Ground truth is** 34%**.**

Method	Prediction	Reasoning trace excerpt
Baseline	**83%**	*"The query recipe employs a similar mechanochemical disruption strategy to breach the surface interface of bacterial cell walls for PHB polymer release. The inclusion of optimized grinding intervals indicates a targeted approach to maximize surface disruption efficiency."*
Unstructured memory	**65%**	*"The absence of a lysis buffer limits polymer release from cell walls, preventing the near-quantitative recovery seen in the chemical reference. Complex biological extraction processes often suffer handling losses and inefficient cell disruption."*
GoE	**47%**	*"Unlike the chemical driving force in the reference, the query relies solely on physical cell wall disruption. Mechanical grinding of Gram-positive bacteria without surfactants or chemical lysing agents is inherently inefficient, often leaving significant intracellular product unreleased."*

Figure 35: Additional qualitative example for the effect of experience memory. Memory shifts the reasoning from a superficial analogy with mechanical disruption to the biological recovery bottleneck caused by intracellular PHB and the absence of lysis chemistry.

(a) Memory for QD-LED property prediction

Unstructured memory. *"In ZnMgO ETLs for QDLEDs, increasing Mg content widens the bandgap and raises the conduction band minimum, which can create an electron-injection barrier. Higher Mg can improve lattice stability and reduce oxygen vacancies, but excessive Mg reduces conductivity and accelerates Joule-heating degradation pathways."*

GoE. *"When a water-controlled ZnMgO ETL uses reduced precursor water content (A_1) together with higher Mg content (A_2), hydroxyl incorporation and oxygen-vacancy formation are suppressed (C_1). Although the wider-bandgap ETL can reduce electron conductivity (C_2), the interaction between lower trap density and smoother morphology stabilizes the QD/ETL interface (E_1), so the operational lifetime can increase when passivation dominates the transport penalty (F_1)."*

(b) QD-LED lifetime prediction query

Query. Predict `Lifetime(T95)` for a QD-LED. The reference ETL is $Zn_{0.854}Mg_{0.146}O$ with 1619 ppm precursor water, conductivity 6.69×10^{-5} S/cm, and trap-density proxy 8.48×10^{17} cm^{-3}, with measured T95 of 2560 h. The query ETL is $Zn_{0.823}Mg_{0.177}O$ with 466 ppm precursor water, conductivity 1.67×10^{-5} S/cm, smoother morphology, and trap-density proxy 3.41×10^{17} cm^{-3}. **Ground truth is 3404 h.**

Method	Prediction	Reasoning trace excerpt
Baseline	**2100–2300 h**	*"The Mg content increases from 14.6% to 17.7%, widening the bandgap and shifting the conduction band minimum upward. The much lower conductivity creates an electron-injection bottleneck, so the query lifetime should be lower than the reference despite some reduction in trap density."*
Unstructured memory	**950–1100 h**	*"Prior experience indicates that excessive Mg doping can reduce conductivity and induce Joule-heating degradation. The query conductivity is much lower than the reference, so the transport penalty is expected to dominate the device lifetime even though the trap-density proxy is reduced."*
GoE	**2680–2750 h**	*"The query combines higher Mg content with substantially lower precursor water, which suppresses hydroxyl-related defects and reduces the trap-density proxy from 8.48×10^{17} to 3.41×10^{17} cm^{-3}. Although conductivity decreases, smoother morphology and stronger interface passivation reduce non-radiative recombination and chemical degradation, supporting a longer T95 than the reference."*

Figure 36: Qualitative effect of memory structure on an in-house QD-LED lifetime prediction query. The baseline over-penalizes the lower ETL conductivity and predicts a lifetime below the measured reference. Unstructured memory retrieves a relevant Mg–conductivity degradation prior, but over-applies it to the query. GoE stores the experience conditionally, helping the model account for reduced precursor water, lower interfacial trap density, and smoother morphology as stabilizing factors that move the prediction toward the measured high lifetime.

LLM-as-a-judge payload and metric schema

Judge call structure. Each request uses `temperature=0` and contains a task-specific system prompt plus a user payload with: `TASK`, `query_id`, `sample_id`, `mechanism_family`, `material_name`, `property_name`, `query_property_unit`, `BASELINE CONTEXT`, `QUERY CONTEXT`, the observed target output, `MODEL PARSED OUTPUT`, `MODEL REASONING TRACE`, and optionally `RAW FINAL JSON SEGMENT`.

Metric targets.

Task	Metric	Scored evidence and definition
Property prediction	*Value Alignment*	Programmatic score outside the judge. Compares the parsed prediction with the observed property after value parsing, interval handling, and unit normalization.
Property prediction	*Mechanistic Validity*	Judge-scored reasoning metric. Uses only `MODEL REASONING TRACE`; measures whether the reasoning identifies the dominant physical lever, gets the sign and relative importance right, and links recipe differences to the target property through a valid causal chain.
Synthesis planning	*Feasibility & Specificity*	Judge-scored output metric. Uses only `MODEL PARSED OUTPUT`; measures whether the generated procedure is experimentally coherent and route-definingly specific enough to execute.
Synthesis planning	*Observed Consistency*	Judge-scored output metric. Uses only `MODEL PARSED OUTPUT`; measures whether the procedure plausibly converges to a chemically and physically similar target-relevant outcome state as the observed procedure, without rewarding surface procedural overlap.
Synthesis planning	*Mechanistic Validity*	Judge-scored reasoning metric. Uses only `MODEL REASONING TRACE`; evaluates chemically specific causal reasoning from processing choices to outcome-relevant structure or mechanism.

Structured responses. `recipe_to_prop` returns a reasoning subscore, flags, a reasoning rationale, and an overall verdict; the programmatic evaluator appends *Value Alignment*. `prop_to_recipe` returns output subscores, a reasoning subscore, flags, output and reasoning rationales, and an overall verdict.

Flag vocabulary. `wrong_property`, `wrong_units`, `large_gt_mismatch`, `physically_implausible`, `constraint_violation`, `introduced_unlisted_material`, `introduced_unlisted_equipment`, `underspecified_procedure`, `mechanism_handwave`, `hallucinated_facts`, `internal_contradiction`, `overconfident_precision`, and `good_alternative_noncanonical`.

Figure 37: Judge payload structure and metric schema. The following figures show the task-specific system prompts used to define the scoring behavior.

Judge system prompt for property prediction

You are a strict but fair expert materials-science evaluator. Evaluate one candidate answer for the `recipe_to_prop` task. Score only the candidate answer; do not reward verbosity.

Numeric target-value alignment is computed programmatically outside the judge. You should score only reasoning quality on a 0–5 integer scale for:

1. `mechanistic_validity`

Use only `MODEL REASONING TRACE` when scoring this reasoning metric. Do not let the final numeric prediction rescue weak reasoning, and do not penalize otherwise strong reasoning merely because the final numeric value is off; numeric accuracy is scored separately by *Value Alignment*. Also emit flags when the final answer appears to use the wrong property, wrong units, a physically absurd value, unsupported hallucinations, internal contradictions, or unsupported precision.

Be strict about unsupported assumptions and shallow mechanism language.

For `mechanistic_validity`, focus on whether the reasoning identifies the dominant physical lever(s) for the target property, gets the sign of effect right, distinguishes dominant from secondary effects, and links recipe differences to those lever(s) with physically valid logic. Reward chemically specific causal chains from recipe differences to property change, not post hoc stories or laundry lists of generic effects. Do not give high `mechanistic_validity` scores to generic statements like "lighter support", "more porosity", or "better conductivity" unless they are clearly the dominant lever, correctly signed, and grounded in the record.

Suggested anchors for `mechanistic_validity`.

0: irrelevant, self-contradictory, or materially incompatible with the dominant lever.
1: weak and mostly generic; little trustworthy causal value.
2: partially relevant but shallow, missing dominance/sign, or mixing correct and incorrect logic.
3: directionally solid dominant-lever reasoning with noticeable gaps or genericity.
4: strong, specific, mostly complete dominant-lever explanation with only minor gaps.
5: exceptional, precise, record-grounded dominant-lever explanation; should be rare.

A perfect 5 should be relatively rare. Keep rationales concise and specific.

Figure 38: System prompt for judging property-prediction reasoning. The final numeric property value is evaluated separately by the programmatic Value Alignment metric.

Judge system prompt for synthesis planning (1/2)

You are a strict but fair expert materials-science evaluator. Evaluate one candidate answer for the `prop_to_recipe` task. The observed recipe is evidence for the likely outcome state, not a unique gold answer. A non-matching alternative recipe can still score highly if it is experimentally coherent, respects the material-system constraints, and plausibly converges to a chemically/physically similar target-relevant outcome regime.

Use a 0–5 integer scale for every subscore. Score these metrics:

Output metrics:
1. *Feasibility & Specificity*
2. *Observed Consistency*

Reasoning metric:
3. *Mechanistic Validity*

Be strict: a generic but plausible recipe is not enough for a high score. For output metrics, use only `MODEL PARSED OUTPUT`. Do not rescue missing procedural detail from `MODEL REASONING TRACE`. For the reasoning metric, use only `MODEL REASONING TRACE`. Do not infer a better mechanism than the model actually articulated.

For *Feasibility & Specificity*, judge local executability, not similarity to the observed procedure or target success. High scores require a chemically coherent route with sufficient route-defining operational detail, such as step order, key reagents/ratios or loading window, phase-forming treatment, material temperature/-time/atmosphere where relevant, and necessary route-defining workup. A recipe can be target-ambitious yet still score low on *Feasibility & Specificity* if it is underspecified or internally incoherent.

Suggested anchors for *Feasibility & Specificity*.

0: broken, contradictory, or chemically incoherent.
1: severe underspecification or major operational incoherence.
2: broadly plausible but missing major route-defining detail needed to execute reliably.
3: executable in outline, but with notable missing detail or minor inconsistency.
4: strong and actionable from the final parsed procedure alone, with only small omissions.
5: exceptionally clear, route-defining, self-consistent, and directly executable; rare.

Figure 39: System prompt for judging synthesis-planning outputs, part 1. This part defines the score scale, evidence boundary, and Feasibility & Specificity anchors.

Judge system prompt for synthesis planning (2/2)

For *Observed Consistency*, judge the likely outcome state, not surface procedural overlap with the observed recipe. Use the observed recipe only as evidence for what result state the validated route is likely to produce. Reward alternative recipes when they plausibly converge to a similar target-relevant regime, such as similar dominant phase or phase family, composition/loading window, oxidation-state regime, defect regime, microstructure or morphology, porosity regime, support-interface structure, or active-site family.

Lower *Observed Consistency* when the candidate route would likely land in a materially different phase space, incompatible chemistry, or an outcome state unlikely to realize the same target-relevant functionality. Do not reward mere property wording, generic optimization language, or textual similarity to the observed recipe. Do not judge by procedural similarity to the observed recipe. Many valid alternatives may exist.

Do not use separate metric scores for constraint adherence, risk handling, or nonhallucination. Express those concerns via flags instead. Treat unlisted materials or equipment as severe only when they are critical and route-defining; routine workup items alone should not force failing scores. Do not penalize a candidate merely for choosing one reasonable reading of the provided conditions. Only penalize when it materially reinterprets the given conditions, adds new conditions, or relies on unsupported assumptions that change the experimental setup or target-property logic.

For *Mechanistic Validity*, focus on whether the reasoning explains a physically valid causal chain from the proposed recipe changes to intermediate chemistry/structure and then to the likely target-relevant outcome regime or dominant target-property lever. Reward reasoning that identifies dominant lever(s), sign of effect, and major tradeoffs or competing pathways where relevant. Penalize generic slogans like "better conductivity", "more porosity", or "higher loading" when they are not translated into a chemically specific pathway for this material system.

Suggested anchors for *Mechanistic Validity*.

0: irrelevant, internally contradictory, or mechanistically incompatible with the system.
1: generic slogans with little chemically specific causal content.
2: partially correct but shallow, missing sign/dominance, or weakly connected to the actual chemistry.
3: physically plausible and directionally meaningful, but still missing important causal detail or tradeoffs.
4: strong, chemically specific causal chain with only modest gaps.
5: exceptional, dominant-lever-focused, chemically precise causal account; rare.

If the recipe is generic or mechanistically disconnected from the target, *Observed Consistency* and *Mechanistic Validity* should be low. A perfect 5 should be rare. Keep rationales concise and specific.

Figure 40: System prompt for judging synthesis-planning outputs, part 2. This part defines Observed Consistency, flag handling, and Mechanistic Validity anchors.

GoE from QD-LED property prediction task

"When a QDLED uses a Zn-based ETL (A) whose design varies electronic transport metrics such as conductivity, mobility, and CBM alignment together with material phase behavior and interfacial energetics (B), this suggests that trap-density or conductivity proxies alone can be misleading because smoother or thicker ETL films may decouple measured trap proxies from operational lifetime while also changing charge injection and recombination pathways (C). Under these conditions, the interaction among transport, morphology, defect chemistry, passivation, and roughness determines whether efficiency gains or lifetime losses dominate the device response (D), leading to a derived design regime where $EQE_{\max}$ and T95 must be evaluated jointly rather than as independent targets (E), and consequently the target properties of maximum external quantum efficiency and operational lifetime require balanced optimization instead of single-proxy maximization (F)."

Figure 41: QD-LED memory learned with shared-context memory updates from the single-target training set. Although each training sample is annotated with a single target property, the learned memory captures a physically multi-target design constraint linking $EQE_{\max}$ and operational lifetime (T95) through shared transport, morphology, and interface mechanisms.

Configuration	**Value**
Memory configuration	
Max stored memories	3000
Max retrieved memories	512
Extraction epochs	1
Stage 1: Prediction	
Rollouts	8
Temperature	0.7
Top-p	0.8
Max tokens	16384
Top-k	0.20
min-p	0.0
Repetition penalty	1.5
Enable thinking	False
Stage 2: Synthesis planning Rollout	
Rollouts	8
Temperature	0.7
Top-p	0.8
Max tokens	16384
Top-k	0.20
min-p	0.0
Repetition penalty	1.5
Enable thinking	False
Stage 2: Reward Evaluation	
Rollouts for Property Reward	1
Temperature for Property Reward	0.6
Top-p for Property Reward	0.95
Max tokens for Property Reward	16384
Top-k for Property Reward	0.20
min-p for Property Reward	0.0
Repetition penalty for Property Reward	1.0
Enable thinking for Property Reward	False

Table 6: Hyperparameters used for memory configuration and stage-wise generation in the prediction and property-conditioned synthesis planning stages.