SciMem: Scientific Reasoning with Structured Memory for Materials Design

Anonymous Authors[1]

Abstract

Large language models are increasingly used for scientific and engineering tasks, but domain adaptation often requires costly supervision or parameter updates. Memory-based methods offer a lightweight alternative by reusing past experience at inference time, which is especially attractive in materials science where experimental data are scarce and heterogeneous. Yet memory reuse in materials design is challenging because the effect of a condition depends on material class, target property, processing history, and measurement context, so naive accumulation can introduce irrelevant or conflicting experience. We propose *SciMem*, a structured memory framework for materials reasoning. SciMem represents experience as a *Graph of Experience*, whose nodes encode materials features and scientific effects and whose edges capture mechanistic dependencies. It further updates memory within disentangled scientific contexts such as mechanism family, material class, and target property. Experiments on property prediction and property-conditioned synthesis planning using Open Materials Guide show that SciMem improves reasoning performance and memory efficiency over prompting and unstructured-memory baselines.

1. Introduction

Recent advances in large language models (LLMs) have sparked growing interest in applying them to scientific and engineering domains such as mathematical reasoning, programming, materials design, drug discovery, and autonomous research (Trinh et al., 2024; Lee et al., 2025a; Gülmez, 2026; Lu et al., 2026; Li et al., 2026; Ahlawat et al., 2026). Their ability to generalize from large-scale pretraining makes them attractive for scientific tasks where explicit modeling is difficult (Alampara et al., 2026). However, adapting LLMs to specialized domains often still requires additional supervision, domain-adaptive fine-tuning, or curated labeled data, which can be costly or limited in practice (Luo et al., 2025a; Lu et al., 2025; Verma et al., 2025).

To address this challenge, recent studies have explored memory-based approaches that enhance LLM capabilities by reusing information from past trajectories (Liang et al., 2025; Cai et al., 2025; Yan et al., 2025; Kim et al., 2026a). These methods can improve performance without additional parameter updates, making them attractive for data-scarce domains such as materials science, where data acquisition is costly and experimental records are often scattered across heterogeneous formats (Wang et al., 2025; Miret & Krishnan, 2025; Kim et al., 2025).

However, applying memory-based methods to materials design introduces unique challenges. In benchmark domains such as mathematical reasoning or programming, prior solutions can often be reused through well-defined objectives, formal constraints, or executable feedback (Chervonyi et al., 2025; Hubert et al., 2026; Jimenez et al., 2024; Jain et al., 2025). In contrast, materials design depends on context-specific relationships between experimental conditions and material outcomes. Subtle changes in composition, precursors, processing, measurement environments, or characterization protocols can alter the underlying mechanism (Sun & David, 2025; Lee et al., 2025b; Nwabara et al., 2025). This makes it difficult to form generalizable priors across experimental contexts. Consequently, methods that simply accumulate and reuse condition-outcome relationships may fail to capture this complexity, making straightforward memory updates insufficient for materials reasoning.

Motivated by this limitation, we propose *SciMem*, a structured memory framework for scientific reasoning in materials design. SciMem goes beyond simple memory storage by explicitly modeling how individual features influence outcomes. Its first component is the *Graph of Experience (GoE)*, a structured memory representation that organizes experience in graph form. In GoE, each node encodes observed task features together with their associated scientific effects in an interpretable form. Edges represent mechanistic or relational dependencies between these effects, allow-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

ing the model to capture interactions between conditions and outcomes. By organizing experience around scientific effects, GoE encourages more fine-grained reasoning and helps the model reason over underlying mechanisms rather than superficial feature patterns.

The second component of SciMem is a disentangled memory update strategy for context-dependent materials design. Even identical experimental conditions can lead to different effects across materials or target properties, highlighting the need for context-specific memory. Building on GoE, SciMem organizes and updates experiences within independent units such as mechanism family, material class, and target property. This allows the model to isolate distinct mechanisms and reduce interference from unrelated experiences. As a result, memory updates can preserve relevant information while mitigating spurious correlations across heterogeneous data. This organization also improves memory efficiency by reducing redundant or conflicting updates across unrelated contexts.

We evaluate SciMem on two materials design tasks using the Open Materials Guide dataset (Kim et al., 2025). Property prediction asks the model to infer a target property from a candidate recipe, while property-conditioned synthesis planning asks the model to generate a synthesis procedure from a desired target property. Although materials synthesis planning and inverse materials design have recently gained increasing attention (Noh et al., 2026; Pan et al., 2026; Karpovich et al., 2024; Cheng et al., 2026; Kim et al., 2026b), property-conditioned synthesis planning remains less explored. This setting is practically important because it reflects the inverse-design problem faced in materials development. At the same time, it is inherently ill-posed, since multiple synthesis procedures can yield the same target property and useful supervision is limited. Property prediction therefore provides a useful auxiliary signal for screening generated procedures. Our experiments test whether structured memory improves both tasks beyond prompting and unstructured memory baselines.

- We propose the *Graph of Experience (GoE)*, a memory representation that encodes how materials features and experimental conditions influence target properties through interpretable mechanistic relations.
- We develop a disentangled memory update strategy that organizes experiences by scientifically meaningful contexts, reducing interference and improving memory efficiency across heterogeneous experimental data.
- We evaluate SciMem on property prediction and property-conditioned synthesis planning, a practically important yet less explored task, showing improvements over prompting and unstructured memory baselines.

2. Related Work

2.1. LLMs for Materials Design

LLMs are increasingly being applied to materials science, spanning property prediction, synthesis planning, literature-grounded assistance, and autonomous laboratories (Lei et al., 2024; Luo et al., 2025b; Bran et al., 2024; Choi et al., 2025; Lee et al., 2025a). However, most existing resources and benchmarks remain centered on static structure based prediction or knowledge centric evaluation, with limited support for process dependent reasoning. Resources such as Materials Project (Jain et al., 2013) and MPcules (Spotte-Smith et al., 2023) primarily provide computed properties of inorganic crystals and molecules, while benchmarks such as LLM4Mat-Bench focus largely on crystalline property prediction (Niyongabo Rubungo et al., 2025). Recent evaluations also emphasize materials knowledge and property prediction tasks (Wang et al., 2025). These works are foundational for evaluating domain knowledge and atomistic predictions, but they are less aligned with process sensitive materials design, where outcomes depend on synthesis procedures, processing conditions, and measurement context (Kim et al., 2025; Miret & Krishnan, 2025).

Recent synthesis oriented datasets and models increasingly encode recipes, operations, and process conditions rather than only static material representations (Kononova et al., 2019; Kim et al., 2025; Noh et al., 2026; Pan et al., 2026). This shift is important because many process-sensitive tasks are governed by coupled process–structure–property–performance relationships. These relationships span synthesis conditions, microstructure, interfaces, and device level contexts (Peng et al., 2023; Sood et al., 2021). Our work builds on this direction, focusing on property-conditioned synthesis planning, a relatively underexplored setting in which the goal is to infer synthesis procedures directly from a desired target property. Recent works such as Materealize have begun to connect property targeted design with synthesis recipes (Kim et al., 2026b). However, most synthesis planning formulations still start from a target material or structure (Noh et al., 2026; Pan et al., 2026). In contrast, property-conditioned synthesis planning starts from a desired target property and requires inferring synthesis choices that can realize it, making the relationship between experimental conditions and resulting properties central to the task.

2.2. Structured Reasoning and Materials Representations

Recent advances in LLM reasoning often rely on structured intermediate steps, beginning with sequential decomposition (Wei et al., 2022; Wang et al., 2022). Subsequent work has extended this paradigm to structured exploration (Yao et al., 2023; Besta et al., 2024), process control (Zhou et al.,

2022; Wang et al., 2023), and executable reasoning (Chen et al., 2022; Gao et al., 2023). However, these approaches rely on generic reasoning formats that do not explicitly encode domain-specific structure, limiting their ability to capture domain-specific dependencies in materials science.

In parallel, materials science has developed graph based representations for synthesis processes. Action graphs (Mysore et al., 2017) represent synthesis text by modeling materials, operations, and conditions as nodes and relations. Related representations have also been adapted to specific domains, such as flow graphs for battery synthesis procedures (Kuniyoshi et al., 2020). More recently, MatPROV (Tsuruta & Kumagai, 2025) represents synthesis workflows as provenance graphs, capturing causal relationships among precursors, intermediate states, and final products. These studies show that structured synthesis knowledge is useful for information extraction and downstream prediction. However, existing structured representations are primarily used for data organization and retrieval, and remain largely disconnected from the reasoning process of LLMs.

2.3. Memory-Enhanced Reasoning in LLMs

Memory-based approaches for LLMs have gained increasing attention as a way to improve reasoning during inference. Early work treated memory mainly as external context, using retrieval, trajectory reuse, or continual accumulation of past experience (Lewis et al., 2020; Borgeaud et al., 2022; Wang et al., 2022; Reed et al., 2022). Subsequent studies introduced reflection based methods that distill prior attempts into reusable guidance (Park et al., 2023; Shinn et al., 2023; Madaan et al., 2023). More recent optimization based approaches make experience reuse more stable and consistent (Liang et al., 2025; Cai et al., 2025). These methods are attractive when labeled data are limited because they can improve reasoning by reusing past experience, reducing the need for additional supervision or costly parameter updates.

However, as memory grows, simply accumulating trajectories or reflections can introduce context overhead, redundancy, and negative transfer. Recent work suggests that more abstract memory representations can improve transferability over raw traces (Kim et al., 2026a). In materials design, this limitation is not only about memory size but also about contextual specificity, since the usefulness of an experience depends on the materials class, target property, process, and measurement condition. This motivates SciMem to store experiences as structured scientific effects and update them within context-specific memory units.

3. Method

3.1. Task Formulation

We formulate materials design as two coupled forward and inverse tasks. The inverse task generate a synthesis procedure for a desired property, and the forward task predicts properties to screen generated procedures.

We define a recipe as a complete specification of a synthesis experiment. Formally, a recipe is denoted as $x = (x^{\text{mat}}, x^{\text{proc}})$, where x^{mat} encodes structured experimental context (e.g., materials, precursors, equipment, and characterization methods), and x^{proc} denotes the synthesis procedure. Let y denote the measured target property associated with recipe x.

Materials design is highly context-dependent. To provide a concrete experimental reference, we pair each query with a single baseline synthesis experiment sampled from the training set that shares the same material class and target property with query. This baseline consists of a recipe x_b and its measured property y_b, which provides a concrete experimental reference.

In property prediction, the query is a full recipe x_q, and the model predicts

$$\hat{y} = \text{LLM}(\{x_b, y_b\}, x_q).$$

In property-conditioned synthesis planning, the query provides partial context x_q^{mat} together with a desired target property $\bar{y}$, and the model predicts a synthesis procedure

$$\hat{x}_q^{\text{proc}} = \text{LLM}(\{x_b, y_b\}, x_q^{\text{mat}}, \bar{y}).$$

Unlike property prediction, synthesis planning is inherently a one-to-many problem, as multiple procedures may satisfy the same target property. This distinction motivates the reward design in Section 3.2.

3.2. SciMem

Overview. Figure 1 shows the overall training pipeline. We learn a separate memory for each task while keeping the base LLM frozen. Let M^{pred} denote the memory for property prediction and M^{plan} denote the memory for synthesis planning. A property prediction training prompt is $\mathtt{P}^{\text{pred}} = (\{x_b, y_b\}, x_q)$ with measured property y_q. A synthesis planning training prompt is $\mathtt{P}^{\text{plan}} = (\{x_b, y_b\}, x_q^{\text{mat}}, \bar{y})$ with an observed synthesis procedure x_q^{proc}. The observed procedure is informative, but it is not treated as uniquely correct because it represents only one feasible solution among multiple plausible procedures. Detailed prompts are provided in Figure 10 and 11.

Training proceeds in two stages. In Stage 1, we learn M^{pred} from property prediction prompts and then freeze it. In

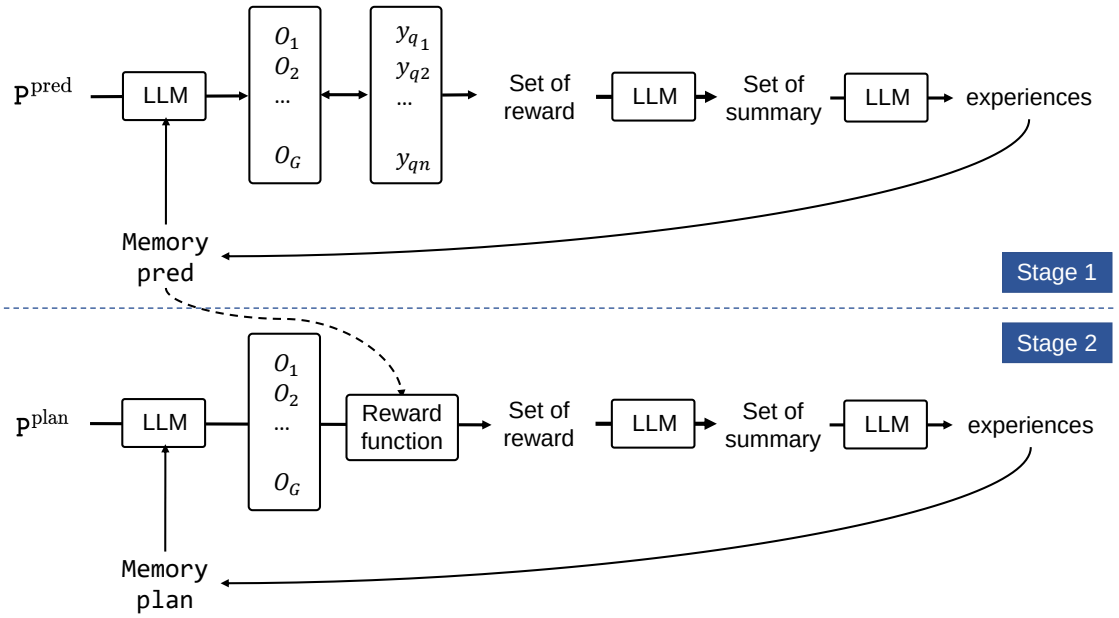

Figure 1. Overview of SciMem training pipeline. Training uses solved source examples to run grouped rollouts for property prediction and property-conditioned synthesis planning while keeping the base LLM frozen. Candidate outputs are scored, compared, summarized, and distilled into memory edits such as add, modify, merge, and delete. At inference time, the learned memory is serialized into the prompt context to support new queries without parameter updates.

Stage 2, we learn M^{plan} from synthesis planning prompts using rewards that combine compatibility with observed procedures and an auxiliary property signal computed with the M^{pred} . We keep the M^{pred} fixed during M^{plan} learning so that the auxiliary reward remains stable and does not depend on biases introduced by the evolving planning memory. All learning is performed in context space through memory updates rather than parameter updates.

Training. For a task $\tau \in \{\mathrm{pred}, \mathrm{plan}\}$, let $\mathtt{P}^{\tau}$ denote the task prompt and M^{τ} denote the current task memory. We sample a group of G rollout outputs

$$o_1^{\tau}, \ldots, o_G^{\tau} \sim f_\theta(\mathtt{P}^{\tau}, M^{\tau}),$$

where each o_g^{τ} contains both a reasoning trace and a final answer. We use task-specific parsers to extract the answer used for scoring:

$$\hat{y}_g = \pi_{\mathrm{pred}}(o_g^{\mathrm{pred}}), \qquad \hat{x}_{q,g}^{\mathrm{proc}} = \pi_{\mathrm{plan}}(o_g^{\mathrm{plan}}).$$

For property prediction, each rollout is scored against the measured query property:

$$r_g^{\mathrm{pred}} = R_{\mathrm{pred}}(\hat{y}_g, y_q; o_g^{\mathrm{pred}}).$$

For synthesis planning, direct comparison to a single observed procedure is insufficient because multiple procedures may satisfy the same target property. We therefore use a composite reward:

$$r_g^{\mathrm{plan}} = \lambda_{\mathrm{obs}} R_{\mathrm{obs}}(\hat{x}_{q,g}^{\mathrm{proc}}, x_q^{\mathrm{proc}}; o_g^{\mathrm{plan}}) + \lambda_{\mathrm{prop}} R_{\mathrm{prop}}(\tilde{y}_g, \bar{y}).$$

The auxiliary property signal is computed by forming the generated full recipe

$$x_g = (x_q^{\mathrm{mat}}, \hat{x}_{q,g}^{\mathrm{proc}})$$

and predicting its property with the frozen Stage 1 prediction memory:

$$\tilde{y}_g = \pi_{\mathrm{pred}}\left(f_\theta(\{x_b, y_b\}, x_g, M^{\mathrm{pred}})\right).$$

This rewards generated procedures that are predicted to achieve the desired property, even when they differ from the observed procedure.

Instead of converting rollout rewards into parameter gradients, we distill them into explicit memory updates. For each rollout group, we compare higher and lower reward candidates and extract the conditions, mechanism claims, and procedural choices that explain the reward gap. These extracted insights are written as free-form text for the unstructured memory baseline and in the GoE format for SciMem. Let ΔM_t denote the set of proposed edits collected at training step t. We update memory as

$$M^{t+1} = \mathrm{Update}(M^t, \Delta M_t),$$

where ΔM_t consists of add, modify, merge, and delete operations proposed from relative comparisons within the rollout groups. Repeating this rollout, score, extract, and update cycle turns task-specific rewards into scientific experience.

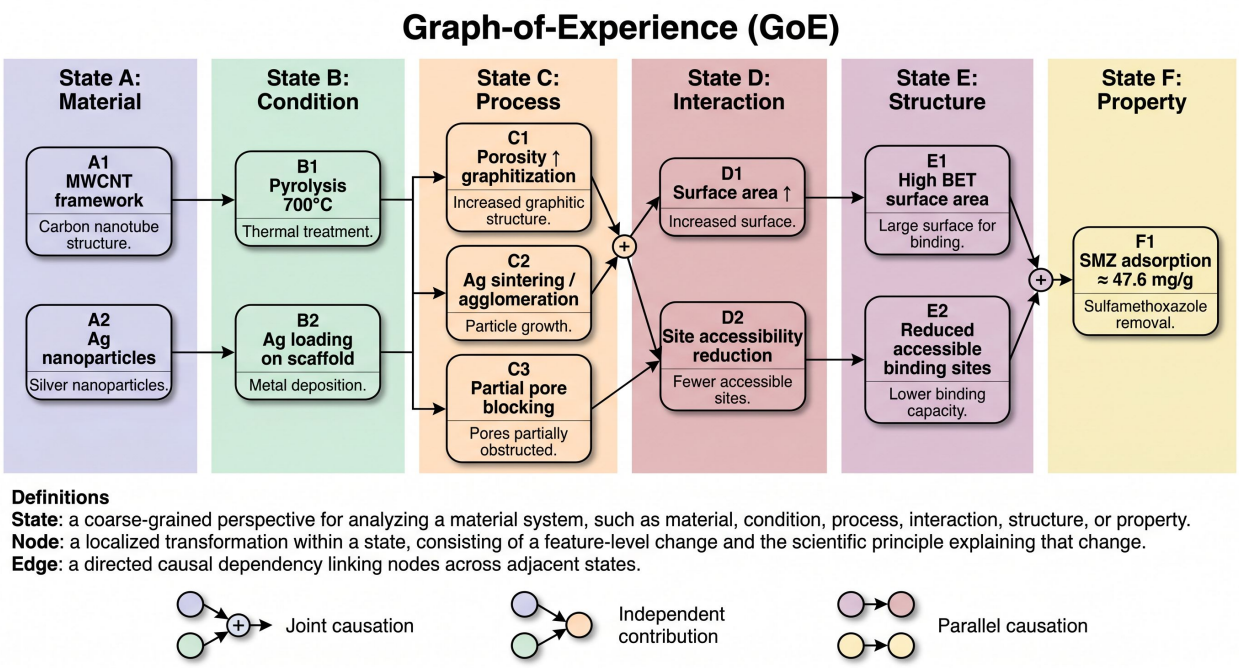

Figure 2. Graph-of-Experience (GoE) representation. The upper panel shows a materials GoE with state-specific nodes and directed dependencies across the A–F states: material or precursor context, synthesis conditions, immediate process implications, causal interactions, derived characteristics, and target-property outcomes. The example illustrates joint, independent, and parallel causal patterns; the legend defines states, nodes, and edges.

Graph-of-Experience. Graph-of-Experience (GoE) is a structured memory representation with nodes and edges. It captures how material features and experimental conditions affect target properties through scientific effects. Each node represents a localized transformation, defined by a feature-level change and the scientific principle explaining it. Nodes are assigned to states, which provide coarse perspectives for analyzing a materials system.

We define six states for materials design: (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 inputs to final performance and are informed by structured representations for materials synthesis and provenance modeling (Mysore et al., 2017; Kuniyoshi et al., 2020; Tsuruta & Kumagai, 2025). Figure 2 shows an example GoE with state-specific nodes, localized transformations, and causal dependencies. The state definition is task-specific and can be adapted to other domains.

Edges connect nodes across adjacent states and capture temporal and causal dependencies. Let N_i^t denote the i-th node assigned to state t. We use three recurring edge patterns:

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

The joint pattern represents factors that act together to produce a downstream effect. The independent pattern represents factors that separately contribute to the same effect. The parallel pattern represents separate factors that propagate into distinct downstream effects. Together, these patterns compactly encode multi-factor causal structure.

Each rollout is converted into a GoE using the query, generated output, and reward signal. SciMem then compares higher- and lower-reward rollout GoEs to identify feature-level changes, scientific principles, and dependencies that explain the reward gap. The resulting insights are stored as localized transformations, with nodes encoding feature changes and edges encoding temporal, causal, and interaction dependencies.

GoE entries are stored as structured text and incorporated directly into the prompting context at inference time. They are not executed as symbolic graphs. Instead, they provide a mechanism-aware scaffold that helps the model reason over intermediate scientific effects rather than surface-level feature patterns.

Disentangled Memory Update Strategy. Experience reuse in materials design is highly context dependent. Similar synthesis choices can induce different effects depending on the material class, target property, governing mechanism, and measurement condition. A single global GoE memory can therefore grow quickly and introduce redundancy or negative transfer across unrelated contexts. SciMem addresses this by organizing GoE experiences into scientifically meaningful groups and training a separate memory within each group.

We study two grouping schemes. Mechanism Family (MF) assigns each example to one of four dominant mechanism regimes: surface/interface control, transport/percolation, electronic band-structure control, or bulk phase/microstructure control. Ambiguous cases are excluded so that each MF memory captures a coherent synthesis-property mechanism. Material-Property (MP) defines more local memories using normalized material-class and target-property pairs. We retain only pairs with more than 30 instances to ensure enough rollout comparisons for stable memory updates.

At inference time, each query is routed to the matching MF or MP memory bank instead of the full global memory. This reduces active context while preserving scientifically relevant experience. Appendix C provides the group definitions, filtering rules, and statistics.

4. Experiments

4.1. Setup

We instantiate property prediction and property-conditioned synthesis planning on a filtered subset of the Open Materials Guide dataset (Kim et al., 2025), which spans diverse compositions, synthesis conditions, target properties, and application domains.

Property-conditioned synthesis planning requires aligned material context, multi-step procedures, and quantitative property measurements, so only a subset of records is suitable. Each retained example contains sufficient material context, a concrete synthesis procedure, and a measured target property, enabling both forward prediction from recipe to property and inverse planning from target property to procedure. Filtering details are provided in Appendix A. The final temporal split contains 1,495 training examples from papers published before 2024 and 159 test examples from papers published in 2024 or later.

All experiments use Qwen3.5-27B as the base model. During training, we build unstructured and GoE memory variants from the training set. At test time, we run five independent generations per prompt and report aggregate scores. We compare four inference settings. Baseline uses only the task prompt. Prompting adds the scientific reasoning guidance in Appendix Figure 9 without learned memory. The inference, summarization, extraction, and memory update prompt templates are provided in Appendix E.

4.2. Evaluation Metrics

We evaluate final outputs and reasoning traces separately for both tasks using an LLM-as-a-judge pipeline with GPT-5.4 high, together with one programmatic metric. Judge prompts are provided in Appendix G. This separation prevents a plausible final answer from masking weak scientific reasoning, while ensuring that strong reasoning remains visible even when the final output is imperfect. All metrics use a common 0 to 5 scale.

For property prediction, the final answer is evaluated by *Value Alignment*, and the reasoning trace by *Mechanistic Validity*. Value Alignment compares the normalized prediction $\hat{y}$ with the ground-truth query property y_q, rewarding accurate estimates and penalizing overly broad predictions. The full definition is provided in Appendix B. Mechanistic Validity evaluates whether the reasoning identifies the dominant physical mechanism, captures its direction, and links synthesis differences to the target property through a plausible causal chain.

For property-conditioned synthesis planning, the generated procedure is evaluated by *Feasibility & Specificity* and *Observed Consistency*, and the reasoning trace by *Mechanistic Validity*. Feasibility & Specificity measures whether the procedure is experimentally coherent and specific enough to execute. Observed Consistency measures whether it plausibly reaches a chemically and physically similar outcome state as the observed procedure x_q^{proc}, without rewarding superficial overlap in wording or step order. Mechanistic Validity evaluates whether the reasoning links procedural choices to the target-relevant outcome through a physically plausible causal chain.

4.3. Results

4.3.1. Effect of Memory and Structure

Table 1 shows that learned memory improves performance over both the baseline LLM and the prompting variant across most metrics. Prompting alone improves over the baseline by encouraging condition-aware scientific reasoning, but its gains are limited because it provides fixed guidance without learned experience. In contrast, memory-based methods show larger improvements, indicating that insights distilled from training data provide reusable guidance for later generations.

Table 1 also shows that GoE memory improves value calibration, observed procedure consistency, and mechanistic validity compared with unstructured memory. This suggests

Table 1. Main evaluation results for property prediction and property-conditioned synthesis planning. Scores are averaged over five independent generations per test query and reported on a 0–5 scale.

Method	Property Prediction		Property-Conditioned Synthesis Planning		
	Value Alignment ↑	Mechanistic Validity ↑	Feasibility & Specificity ↑	Observed Consistency ↑	Mechanistic Validity ↑
Baseline	3.33	2.32	2.45	2.45	2.33
Prompting	3.38	2.37	2.40	2.66	2.54
Unstructured Memory	3.56	2.69	**2.58**	2.84	2.71
GoE memory (ours)	**3.71**	**2.95**	2.55	**3.08**	**2.98**

Table 2. Ablation of Graph-of-Experience (GoE) memory organization. All rows use the same prompting setup and GoE memory representation, differing only in the context unit used to build and update memory. Scores are averaged over five independent generations per test query and reported on a 0–5 scale.

Method	Property Prediction		Property-Conditioned Synthesis Planning		
	Value Alignment ↑	Mechanistic Validity ↑	Feasibility & Specificity ↑	Observed Consistency ↑	Mechanistic Validity ↑
GoE memory (all)	3.71	2.95	**2.55**	3.08	**2.98**
GoE memory (MF)*	3.72	**2.99**	2.53	3.07	2.99
GoE memory (MP)**	**3.74**	2.92	2.53	**3.12**	2.91

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

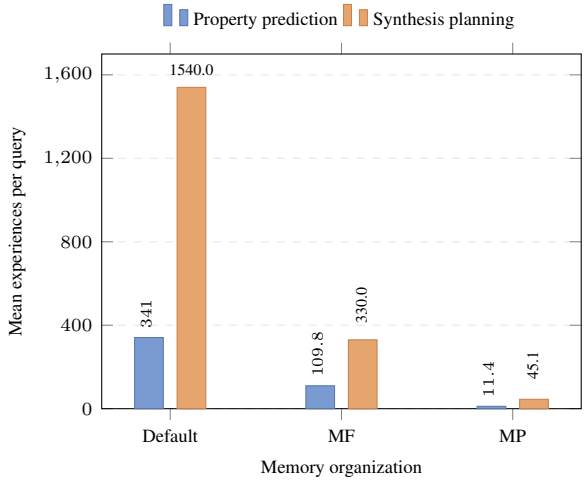

Figure 3. Active memory size under each GoE memory organization. Bars report the mean number of experiences inserted into each query for property prediction and synthesis planning. MF and MP grouping reduce active context by updating and retrieving experiences within scientifically meaningful partitions.

that memory structure is especially useful when reasoning requires condition-specific mechanisms, while unstructured memory remains competitive when procedural detail is the dominant signal.

Figure 4 illustrates this effect on a Fe_3O_4/PBN particle-size prediction query. In Figure 4(b), the baseline over-anchors to the ball-milling condition and predicts substantial particle growth through aggregation. Unstructured memory introduces a relevant prior about aggregation under mechanical processing and moves the prediction closer to the ground truth. However, it still over-applies this prior because it does not encode when precursor morphology should remain the dominant constraint.

Figure 4(a) shows why GoE improves over unstructured memory. Unstructured memory stores a broad rule about milling-induced aggregation. GoE instead represents the experience conditionally, separating precursor context, post-processing conditions, intermediate effects, structural constraints, and final property implication with state tags. As shown in Figure 4(b), this helps the model recognize that the Fe_3O_4 nanoparticles are pre-synthesized and are not exposed to a high-temperature coarsening step after introduction. GoE therefore predicts a value governed by precursor size rather than by a generic milling-agglomeration heuristic. An additional qualitative example is provided in Appendix F.

4.3.2. Effect of Disentangled Memory Updates

We evaluate whether disentangled memory updates reduce interference while preserving useful experience. Figure 3 shows that scientific-context grouping greatly reduces the active memory exposed to each query. Appendix Table 3 further shows that grouped memories are not just filtered at inference time. They are built within separate context units. In synthesis planning, this also reduces the total number of stored experiences, indicating that disentanglement makes memory construction more selective.

Table 2 shows that this reduction does not materially degrade performance. Grouped memories remain close to the global

(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 memory. *"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 is 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 inter-particle 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 memory	**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 a Fe_3O_4/PBN particle-size prediction query. The baseline overgeneralizes from the ball-milling condition and predicts substantial particle growth. Unstructured memory introduces a relevant prior about aggregation under mechanical processing, but still over-applies it. GoE stores the experience in a conditional form with explicit state tags, helping the model preserve the query-specific constraint that Fe_3O_4 nanoparticles are pre-synthesized and not exposed to a high-temperature coarsening step after introduction.

GoE memory, and MP is often competitive or stronger on answer-level metrics. These results suggest that useful materials experience is strongly context dependent. Weakly related memories can often be removed from the active context without losing the main memory benefit. The remaining differences reflect a trade-off between breadth and specificity. Broader groups retain more analogical evidence for mechanistic reasoning, while local groups better suppress irrelevant or conflicting experience. Overall, SciMem benefits not from exposing the largest memory, but from updating and retrieving experiences within scientifically meaningful contexts.

5. Conclusion

We presented *SciMem*, a memory-based framework for scientific reasoning in LLMs for materials design. SciMem combines the Graph of Experience with disentangled memory updates to capture condition-dependent relationships and organize experience within scientifically meaningful contexts. Our results show that learned memory provides larger gains than prompting alone, and that structured memory improves reasoning by preserving condition-specific effects, reducing overgeneralization, and enabling selective use of relevant experience. These findings suggest that effective materials reasoning depends not only on model reasoning capability, but also on how experience is represented, updated, and retrieved. Future work will extend SciMem to more complex multi-step design settings and memory organizations that better balance coverage and relevance.

References

Ahlawat, D., Mishra, V., Singh, S., Zaki, M., Bihani, V., Grover, H. S., Mishra, B., Miret, S., Mausam, and Krishnan, N. M. A. A family of large language models for materials research with insights into model adaptability in continued pretraining. *Nature Machine Intelligence*, 8: 435–448, 2026. doi: 10.1038/s42256-026-01199-8.

Alampara, N., Aneesh, A., Ríos-García, M., Mirza, A., Schilling-Wilhelmi, M., Aghajani, A. A., Sun, M., Prastalo, G., and Jablonka, K. M. General-purpose models for the chemical sciences: LLMs and beyond. *Chemical Reviews*, 126(4):2484–2549, 2026. doi: 10.1021/acs.chemrev.5c00583.

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Van Den Driessche, G. B., Lespiau, J.-B., Damoc, B., Clark, A., et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Bran, A. M., Cox, S., Schilter, O., Baldassari, C., White, A. D., and Schwaller, P. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535, 2024. doi: 10.1038/s42256-024-00832-8. Published version of ChemCrow arXiv:2304.05376.

Cai, Y., Cai, S., Shi, Y., Xu, Z., Chen, L., Qin, Y., Tan, X., Li, G., Li, Z., Lin, H., Mao, Y., Li, K., and Sun, X. Training-free group relative policy optimization. *arXiv preprint arXiv:2510.08191*, 2025. doi: 10.48550/arXiv.2510.08191. URL https://arxiv.org/abs/2510.08191.

Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Cheng, M., Fu, C.-L., Okabe, R., Chotrattanapituk, A., Boonkird, A., Hung, N. T., and Li, M. Artificial intelligence-driven approaches for materials design and discovery. *Nature Materials*, 25(2):174–190, 2026. doi: 10.1038/s41563-025-02403-7.

Chervonyi, Y., Trinh, T. H., Olšák, M., Yang, X., Nguyen, H. H., Menegali, M., Jung, J., Kim, J., Verma, V., Le, Q. V., and Luong, T. Gold-medalist performance in solving olympiad geometry with alphageometry2. *Journal of Machine Learning Research*, 26(241):1–39, 2025. URL http://jmlr.org/papers/v26/25-1654.html.

Choi, S. E., Jang, M., Yoon, S., Yoo, S., Ahn, J., Kim, M., Kim, H.-G., Jung, Y., Park, S., Kim, Y.-S., et al. LLM-driven synthesis planning for quantum dot materials development. *Journal of Chemical Information and Modeling*, 65(6):2748–2758, 2025. doi: 10.1021/acs.jcim.4c01529.

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799. PMLR, 2023.

Gülmez, B. Code generation with large language models: A survey from neural program synthesis to autonomous software development. *Applied Intelligence*, 56(200), 2026. doi: 10.1007/s10489-026-07230-0.

Hubert, T., Mehta, R., Sartran, L., Vieillard, N., Andreev, A., Crepy, C., Oliveira, A. S. P., Bachem, O., Ono, Y., Hussenot, L., et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613, 2026. doi: 10.1038/s41586-025-09833-y. Published online 2025; version of record in 2026.

Jain, A., Ong, S. P., Hautier, G., Chen, W., Richards, W. D., Dacek, S., Cholia, S., Gunter, D., Skinner, D., Ceder, G., and Persson, K. A. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013. doi: 10.1063/1.4812323.

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

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Karpovich, C., Pan, E., and Olivetti, E. A. Deep reinforcement learning for inverse inorganic materials design. *npj Computational Materials*, 10(1), 2024. doi: 10.1038/s41524-024-01474-5.

Kim, H., Jeon, T., Choi, S., Hong, J. H., Jeon, D. W., Cho, S. B., Baek, G.-Y., Kwak, G.-W., Lee, D.-H., Choi, S.-J., Bae, J., Lee, C., Kim, Y., Park, J., and Cho, H. 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*, pp. 1302–1312, 2025. doi: 10.1145/3746252.3761359. Also available as arXiv:2502.16457.

Kim, K., Kang, M., Kim, T., Yang, Y., Ren, M., and Hwang, S. J. Memory transfer learning: How memories are transferred across domains in coding agents. *arXiv preprint arXiv:2604.14004*, 2026a. doi: 10.48550/arXiv.2604.14004. URL https://arxiv.org/abs/2604.14004.

Kim, S., Choi, J., Jang, K., Park, J., Bernales, V., Aspuru-Guzik, A., and Jung, Y. Materealize: a multi-agent deliberation system for end-to-end material design and synthesis. *arXiv preprint arXiv:2601.15743*, 2026b.

Kononova, O., Huo, H., He, T., Rong, Z., Botari, T., Sun, W., Tshitoyan, V., and Ceder, G. Text-mined dataset of inorganic materials synthesis recipes. *Scientific Data*, 6: 203, 2019. doi: 10.1038/s41597-019-0224-1.

Kuniyoshi, F., Makino, K., Ozawa, J., and Miwa, M. Annotating and extracting synthesis process of all-solid-state batteries from scientific literature. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 1941–1950, 2020.

Lee, H., Yoon, S., Park, J., Chae, S. I., Park, S., Ahn, J., Jung, Y., Chung, Y., Chang, H., Park, S., Kang, M., Kim, J., Kim, H.-G., and Jeong, M. Aligning reasoning LLMs for materials discovery with physics-aware rejection sampling. *arXiv preprint arXiv:2509.00768*, 2025a. doi: 10.48550/arXiv.2509.00768. URL https://arxiv.org/abs/2509.00768.

Lee, S., Cruse, K., Baibakova, V., Ceder, G., and Jain, A. Text-mined dataset of solid-state syntheses with impurity phases using large language model. *Scientific Data*, 12 (1), 2025b. doi: 10.1038/s41597-025-06222-y.

Lei, G., Docherty, R., and Cooper, S. J. Materials science in the era of large language models: A perspective. *Digital Discovery*, 3(7):1257–1272, 2024. doi: 10.1039/D4DD00074A.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Li, X., Li, P., Qian, L., Liu, M., Wang, D., Liu, J., Kang, B., Ma, X., Wang, X., Guo, D., Kong, T., Zhang, H., et al. What matters in building vision–language–action models for generalist robots. *Nature Machine Intelligence*, 8: 158–172, 2026. doi: 10.1038/s42256-025-01168-7.

Liang, X., Tao, M., Xia, Y., Wang, J., Li, K., Wang, Y., He, Y., Yang, J., Shi, T., Wang, Y., Zhang, M., and Wang, X. SAGE: Self-evolving agents with reflective and memory-augmented abilities. *Neurocomputing*, 647:130470, 2025. doi: 10.1016/j.neucom.2025.130470.

Lu, C., Lu, C., Lange, R. T., Yamada, Y., Hu, S., Foerster, J., Ha, D., and Clune, J. Towards end-to-end automation of ai research. *Nature*, 651:914–919, 2026. doi: 10.1038/s41586-026-10265-5.

Lu, W., Luu, R. K., and Buehler, M. J. Fine-tuning large language models for domain adaptation: exploration of training strategies, scaling, model merging and synergistic capabilities. *npj Computational Materials*, 11(84), 2025. doi: 10.1038/s41524-025-01564-y.

Luo, J., Luo, X., Chen, X., Xiao, Z., Ju, W., and Zhang, M. Semi-supervised fine-tuning for large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2795–2808, 2025a. doi: 10.18653/v1/2025.findings-naacl.151.

Luo, Z., Yang, Z., Xu, Z., Yang, W., and Du, X. LLM4SR: A survey on large language models for scientific research. *arXiv preprint arXiv:2501.04306*, 2025b. doi: 10.48550/arXiv.2501.04306. URL https://arxiv.org/abs/2501.04306.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Miret, S. and Krishnan, N. M. A. Enabling large language models for real-world materials discovery. *Nature Machine Intelligence*, 7:991–998, 2025. doi: 10.1038/s42256-025-01058-y.

Mysore, S., Kim, E., Strubell, E., Liu, A., Chang, H.-S., Kompella, S., Huang, K., McCallum, A., and Olivetti, E. Automatically extracting action graphs from materials science synthesis procedures. *arXiv preprint arXiv:1711.06872*, 2017.

Niyongabo Rubungo, A., Li, K., Hattrick-Simpers, J., and Bousso Dieng, A. LLM4Mat-Bench: Benchmarking large language models for materials property prediction. *Machine Learning: Science and Technology*, 6(2):020501, 2025. doi: 10.1088/2632-2153/add3bb.

Noh, H., Na, G. S., Lee, N., and Park, C. Msp-llm: A unified large language model framework for complete material synthesis planning. *arXiv preprint*

arXiv:2602.07543, 2026. URL https://arxiv.org/abs/2602.07543.

Nwabara, U., Yang, K., Talekar, A., Bernales, V., González, J., Miller, S., and Wu, J. High throughput computational and experimental methods for accelerated electrochemical materials discovery. *Journal of Materials Chemistry A*, 13(32):26041–26066, 2025. doi: 10.1039/d5ta00331h.

Pan, E., Kwon, S., Liu, S., Xie, M., Hoffman, A. J., Duan, Y., Prein, T., Sheriff, K., Roman-Leshkov, Y., Moliner, M., Gomez-Bombarelli, R., and Olivetti, E. A. DiffSyn: A generative diffusion approach to materials synthesis planning. *Nature Computational Science*, 6:404–416, 2026. doi: 10.1038/s43588-025-00949-9.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Peng, Z., Stingelin, N., Ade, H., and Michels, J. J. A materials physics perspective on structure–processing–function relations in blends of organic semiconductors. *Nature Reviews Materials*, 8:439–455, 2023. doi: 10.1038/s41578-023-00541-5.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Sood, A., Poletayev, A. D., Cogswell, D. A., Csernica, P. M., Mefford, J. T., Fraggedakis, D., Toney, M. F., Lindenberg, A. M., Bazant, M. Z., and Chueh, W. C. Electrochemical ion insertion from the atomic to the device scale. *Nature Reviews Materials*, 6:847–867, 2021. doi: 10.1038/s41578-021-00314-y.

Spotte-Smith, E. W. C., Cohen, O. A., Blau, S. M., Munro, J. M., Yang, R., Guha, R. D., Patel, H. D., Vijay, S., Huck, P., Kingsbury, R., et al. A database of molecular properties integrated in the materials project. *Digital Discovery*, 2(6):1862–1882, 2023. doi: 10.1039/D3DD00153A.

Sun, W. and David, N. A critical reflection on attempts to machine-learn materials synthesis insights from text-mined literature recipes. *Faraday Discussions*, 256:614–638, 2025. doi: 10.1039/d4fd00112e.

Trinh, T. H., Wu, Y., Le, Q. V., He, H., and Luong, T. Solving olympiad geometry without human demonstrations. *Nature*, 625:476–482, 2024. doi: 10.1038/s41586-023-06747-5.

Tsuruta, H. and Kumagai, M. MatPROV: A provenance graph dataset of material synthesis extracted from scientific literature. *arXiv preprint arXiv:2509.01042*, 2025. doi: 10.48550/arXiv.2509.01042. URL https://arxiv.org/abs/2509.01042.

Verma, A. K., Zhang, Z., Seo, J., Kuo, R., Jiang, R., Strubell, E., and Rollett, A. D. Text mining for process–structure–properties relationships in metals. *Integrating Materials and Manufacturing Innovation*, 2025. doi: 10.1007/s40192-025-00420-7. Published-title metadata matched to DOI; earlier draft title: Structured Extraction of Process Structure Properties Relationships in Materials Science.

Wang, H., Li, K., Ramsay, S., Fehlis, Y., Kim, E., and Hattrick-Simpers, J. Evaluating the performance and robustness of llms in materials science q&a and property predictions. *Digital Discovery*, 4:1612–1624, 2025. doi: 10.1039/D5DD00090D.

Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., and Lim, E.-P. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 2609–2634, 2023.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

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

Yan, S., Yang, X., Huang, Z., Nie, E., Ding, Z., Li, Z., Ma, X., Schütze, H., Tresp, V., and Ma, Y. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025. doi: 10.48550/arXiv.2508.19828. URL https://arxiv.org/abs/2508.19828.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

A. 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 property-conditioned 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 5, 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 $\tilde{I}_{\hat{y}}$ and $\tilde{I}_q$. We then compare both the interval center and the interval width. Let $c(\cdot)$ and $w(\cdot)$ denote interval center and width, respectively. To stabilize evaluation, we penalize not only center mismatch but also predictions that are much broader than the ground-truth interval:

$$s_{\text{range}} = 5 \exp\left(- \left(\frac{|c(\tilde{I}_{\hat{y}}) - c(\tilde{I}_q)|}{\tau_c} \right)^2 - \left(\frac{\max\{0,\, w(\tilde{I}_{\hat{y}}) - w(\tilde{I}_q)\}}{\tau_w} \right)^2 \right).$$

The first term rewards a correctly located interval center. The second term penalizes overly broad predictions, 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. Disentangled Memory

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 5. 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.

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%

The MF labels are defined by broad mechanism regimes, while MP labels combine normalized material and property labels. Figures 5–8 list the full textual descriptions used for these labels.

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 5. 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 6. 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 7. 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 8. Full material-name descriptions used for the material dimension of MP memory organization, part 2 of 2.

D. Prompt Templates

E. Prompt Templates

This section lists the prompt templates used to instantiate the experimental pipeline described in Sections 3 and 4. Figure 9 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 baseline recipe, material and property descriptors, and explicit scientific reasoning constraints. Figure 10 shows the property-prediction prompt, which asks the model to infer a structured target-property object from a full synthesis record. Figure 11 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 12 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 13 shows the inference prompt that exposes learned experiences at test time. Figure 14 summarizes model trajectories against the ground-truth property, Figure 15 extracts scientific experience edits from those summaries, and Figure 16 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 17 shows the GoE inference prompt. Figures 18 and 19 define the GoE trajectory-summary prompt, including node types, edge rules, and dominant path extraction. Figures 20 and 21 define the experience-extraction prompt that turns GoE summaries into add, modify, merge, or delete operations, and Figure 22 shows the final update prompt.

For property-conditioned 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 23 shows the inference prompt for proposing a recipe under memory guidance. Figure 24 summarizes inverse-design rollouts with respect to the desired property and observed recipe, Figure 25 extracts planning experiences, and Figure 26 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 27 shows the inference prompt. Figures 28 and 29 define the GoE summary prompt for inverse design. Figures 30 and 31 define the corresponding experience-extraction prompt, and Figure 32 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 9. 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 baseline 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 baseline 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 baseline and query records; here we show the beginning and end of each record and omit the middle with `...`.

> **Baseline.** 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.

Scientific reasoning requirements. All reasoning must be grounded in real materials-science principles such as thermodynamics, kinetics, defect chemistry, crystallography, surface/interface phenomena, transport processes, and precursor chemistry. The model must justify each synthesis modification by referencing physical or chemical mechanisms, explain why the modification changes the property, use real synthesis parameters, and give mechanistically correct reasoning. The model must not introduce symbolic variables, perform algebraic or abstract optimization, skip scientific explanation, or rely on superficial keyword matching without causal, physics-based justification.

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

Property-Conditioned 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 baseline 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 and often conditioned on pH, temperature, and isotherm regime.

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

> **Baseline.** Recipe: materials include $FeSO_4{\cdot}7H_2O$, $FeCl_3{\cdot}6H_2O$, tea waste, distilled water, ammonium hydroxide, and N_2. Synthesis 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. `...` Class: `bulk_phase_microstructure`, `iron_oxide`, `adsorption_capacity`. Property: 75.76 mg/g, maximum adsorption capacity for Cr(VI).
>
> **Query.** Recipe: 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.

Scientific reasoning requirements for inverse design. All design choices must be grounded in real materials-science principles, including thermodynamics, kinetics, defect chemistry, crystallography, surface/interface phenomena, transport processes, and precursor chemistry. The model must design toward the requested target regime rather than a generic "better" direction; justify each proposed synthesis choice by identifying the changed control variable, affected mechanism, and expected property shift; use actionable synthesis parameters; remain experimentally coherent under the provided constraints; and identify likely failure modes, overshoot risks, or uncertainty sources. The model must not introduce symbolic mathematical variables, replace the provided material system, introduce unsupported materials or equipment, recommend vague actions without concrete conditions, or assume that higher crystallinity, smaller particles, higher surface area, or stronger porosity are always beneficial.

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 nickel nitrate and cobalt nitrate with urea in the provided
deionized-water solvent until the solution is clear.
n2. Transfer the solution into the provided Teflon-lined autoclave and
maintain hydrothermal growth at a moderate temperature for several hours to
form a uniform precursor coating on the provided substrate.
n3. Remove the sample, rinse thoroughly, and dry under the provided oven
conditions.
n4. Apply a short moderate-temperature post-anneal using the provided
furnace equipment to improve connectivity while preserving accessible
nanosheet morphology.
n5. Cool to room temperature and collect the final electrode for downstream
evaluation."
}
```


Figure 11. Property-conditioned synthesis-planning prompt. The displayed prompt preserves the task context, scientific conditioning labels, inverse-design requirements, task definition, and output schema. Long baseline 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 12. 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 13. 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 14. 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 12, 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 15. 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 16. 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 17. 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 18. 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 19. 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 20. 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 21. 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 22. 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 23. 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 24. 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 25. 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 26. 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 27. 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 28. 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 29. 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 30. 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 31. 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 32. 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.

F. Additional Qualitative Example for Experience Memory

Figure 33 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 4.3 that memory improves calibration by moving the model from generic process analogy toward the mechanism controlling the queried property.

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 property-conditioned 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 34 summarizes the judge payload and metric schema, while Figures 35–37 provide the task-specific judge prompts and scoring anchors.

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 baseline. Complex biological extraction processes often suffer handling losses and inefficient cell disruption."*
GoE memory	**47%**	*"Unlike the chemical driving force in the baseline, 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 33. 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.

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	*Input Grounding*	Judge-scored reasoning metric. Uses only `MODEL REASONING TRACE`; measures whether the reasoning stays anchored to baseline/query facts rather than unsupported memory, benchmarks, or invented details.
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	*Input Grounding* / *Mechanistic Validity*	Judge-scored reasoning metrics. Use only `MODEL REASONING TRACE`; evaluate record grounding and chemically specific causal reasoning from processing choices to outcome-relevant structure or mechanism.

Structured responses. `recipe_to_prop` returns reasoning subscores, flags, a reasoning rationale, and an overall verdict; the programmatic evaluator appends *Value Alignment*. `prop_to_recipe` returns output subscores, reasoning subscores, 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 34. 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. `input_grounding`
2. `mechanistic_validity`

Use only `MODEL REASONING TRACE` when scoring these reasoning metrics. 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 `input_grounding`, penalize unsupported references to memory items, literature benchmarks, typical-value ranges, or precedents that are not explicitly evidenced in the provided record.

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 35. 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 metrics:
3. *Input Grounding*
4. *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 reasoning metrics, 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 36. 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 *Input Grounding*, penalize invented conditions, unsupported assumptions, or reasoning that ignores the provided record.

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, weakly grounded, 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 37. System prompt for judging synthesis-planning outputs, part 2. This part defines Observed Consistency, reasoning metrics, flag handling, and Mechanistic Validity anchors.