BEFORE THE FURNACE: TRAINING LLM AGENTS FOR INORGANIC RETROSYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Inorganic retrosynthesis connects materials discovery with synthesis planning by identifying precursor sets for a target composition. Effective recommendations require models to acquire evidence beyond composition and use it to assess and revise precursor candidates. We introduce PreWeave, an LLM agent trained for inorganic precursor recommendation through tool interaction. We train its interaction policy through supervised fine-tuning on distilled trajectories followed by agentic reinforcement learning. The policy learns to select tools and use their feedback to guide subsequent precursor decisions. Agentic training improves precursor-set Pass@1 accuracy from 53.81% for the base agent to 73.02%. PreWeave achieves competitive performance with task-specific models and tool-augmented frontier LLMs. Further analyses show that training makes reference-matching recommendations more consistent across runs and shifts tool use toward candidate construction, with retrieval contributing most to accuracy among the available tools. We will release tools, interaction trajectories, and training and evaluation code to support further research on agent policy training for materials synthesis.

1 INTRODUCTION

Advancing inorganic materials discovery requires not only identifying promising target materials but also determining how to synthesize them. Precursor selection is a key decision in solid-state synthesis because the starting compounds influence the reaction pathways through which the target material forms (Aykol et al., 2021). In this work, we study inorganic retrosynthesis as the task of recommending precursor sets for a given target composition (He et al., 2023; Noh et al., 2024).

This composition-to-precursor task presents two connected challenges. First, a target composition specifies the desired elemental proportions but does not uniquely determine the starting compounds. Selecting among possible precursor sets therefore requires evidence beyond the composition itself, including related synthesis recipes and chemical knowledge (He et al., 2023; Noh et al., 2024). Second, evaluating a candidate set requires determining whether retrieved precursor choices transfer to the target and whether chemical checks call for revising the proposed combination. Addressing these questions involves connecting candidate generation with external feedback: proposing a set, obtaining evidence relevant to that proposal, and using the returned information to compare alternatives or revise the selection (Kim et al., 2026). These challenges motivate a multi-turn decision process that uses external feedback to guide precursor recommendation.

Existing methods address these challenges through learned prediction, recipe retrieval, and chemical reasoning. Traditional machine learning approaches learn precursor choices from synthesis records and use materials similarity, retrieved recipes, and domain knowledge to guide recommendations (He et al., 2023; Noh et al., 2024; Kim et al., 2020; 2024b). Recent approaches based on large language models (LLMs) support precursor generation and synthesis planning (Kim et al., 2024a; Prein et al., 2025; Noh et al., 2026), with post-training improving recipe-aligned recommendations (Pan et al.) and agentic systems incorporating tool interaction and deliberation (Zhu et al., 2026; Kim et al., 2026; Fei et al., 2026). However, task-specific training of agent policies for tool selection and feedback-informed precursor decisions remains insufficiently studied, and its benefits over tool access without policy training or answer-only supervision remain unclear. Therefore, training an agentic policy that connects tool interaction with precursor decisions conditioned on returned evidence remains underexplored for inorganic precursor recommendation.

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

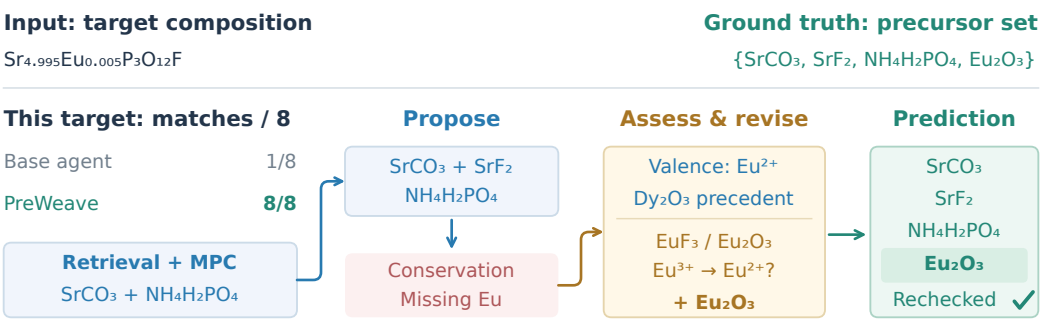


Figure 1: An illustrative example of PreWeave. Retrieval and conditional precursor suggestions guide candidate construction, while chemical checks and evidence from related syntheses inform revision. Matches/8 denotes the number of exact matches to the reference precursor set across eight independent runs for the displayed target.

To address this gap, we introduce PreWeave, an LLM agent trained to recommend inorganic precursors through tool interaction (Figure 1). The agent operates in a materials-specific environment, where tools provide synthesis precedents, conditional precursor suggestions, and compositional checks to inform candidate construction and assessment. We train tool selection and feedback-informed precursor decisions as a unified policy, linking evidence acquisition to how the agent evaluates and revises its candidates. Based on PreWeave, we investigate how trajectory supervision and online reinforcement learning shape the interaction policy, and empirically examine their effects on recommendation performance and tool-use behavior.

Concretely, we initialize the agent policy through supervised fine-tuning on distilled interaction trajectories and refine it through online agentic reinforcement learning. The agentic training of PreWeave raises single-run exact-match accuracy from 53.81% to 73.02%. Our trained 9B agent also achieves competitive performance with the evaluated tool-augmented frontier LLMs on a shared test subset. Across repeated runs, training primarily improves the consistency of reference-matching recommendations, though with limited change in aggregate coverage. SFT introduces a large leap in performance and change in tool-use patterns, while online RL provides additional accuracy gains. Tool ablations and budget analyses further show that retrieval tools has the largest individual effect on accuracy and that gains from additional tool calls diminish beyond a modest interaction budget.

Our contributions are threefold:

- We introduce PreWeave, an LLM agent for inorganic precursor recommendation, and train its tool-interaction policy through trajectory-based supervised fine-tuning and online reinforcement learning. The policy connects tool selection and returned evidence with subsequent precursor decisions.
- We characterize the effects of policy training, tool access, and interaction budgets on recommendation performance and agent behavior. Our results identify improved single-run accuracy and cross-run consistency as the primary training gains, alongside changes in tool-use patterns and diminishing returns from additional interactions.
- We will release reusable tools, distilled interaction trajectories, and training and evaluation code to support reproducible research on agent policy training for materials synthesis.

2 Related Work

2.1 Machine Learning for Inorganic Retrosynthesis

Inorganic retrosynthesis is commonly formulated as recommending precursor sets for a target composition. Literature-derived synthesis datasets (Kononova et al., 2019) have enabled models to learn this mapping from reported experimental recipes. Early neural approaches modeled synthesis choices using conditional generative models (Kim et al., 2020). Subsequent methods introduced

more specialized representations and inductive biases: He et al. (2023) learned materials similarity to transfer precursor choices from related syntheses, while elementwise template formulation (Kim et al., 2024b) structured precursor prediction around constituent elements. Retrieval-Retro (Noh et al., 2024) further incorporated retrieved reference syntheses and thermodynamic expertise into precursor prediction. These approaches establish complementary ways to exploit compositional information, synthesis precedents, and domain knowledge.

Large language models have expanded this line of work through precursor generation, synthetic training data, and structured synthesis planning. SynthGPT (Kim et al., 2024a) studied language models for inorganic synthesis prediction, including precursor selection. Prein et al. (2025) evaluated off-the-shelf language models and used generated recipes to augment training data for synthesis prediction. MSP-LLM (Noh et al., 2026) connected precursor prediction and synthesis operation prediction through an intermediate material-class representation. More recently, SynReason (Pan et al.) applied reinforcement learning with an experimental-alignment reward to improve ranked precursor recommendations. Its reward evaluates agreement between generated precursor sets and recorded experimental recipes. In contrast, our work trains an agentic tool-interaction policy for inorganic retrosynthesis through trajectory-based supervised fine-tuning and online reinforcement learning, enabling the model to use external feedback when forming precursor recommendations.

2.2 LLMs for Chemical and Materials Science

Large language models have been applied to synthesis planning, reaction prediction, and materials structure prediction. In synthesis planning, pretrained language models generate precursor recommendations and synthetic training recipes (Kim et al., 2024a; Prein et al., 2025), while reasoning-based post-training improves precursor ranking and multi-step molecular route generation (Pan et al.; Wang et al., 2026). Related progress in molecular reaction prediction includes RetroChimera (Maziarz et al., 2026), which ensembles a graph-based editing model with an autoregressive Transformer to combine complementary inductive biases. In material structure prediction, MOF-LLM (Pan et al., 2026) leverages spatially aware pretraining, supervised fine-tuning, and reinforcement learning to predict the three-dimensional configurations of metal-organic framework assemblies. These studies illustrate how chemical representations and task-specific training adapt language generation to reaction prediction, synthesis reasoning, and structural modeling.

Agentic LLMs extend these capabilities by interacting with scientific tools and using their outputs to guide subsequent actions. ChemCrow and Coscientist (M. Bran et al., 2024; Boiko et al., 2023) connect language models with chemistry tools and experimental resources, while RETRO-R1 and RetroAgent (Zhu et al., 2026; Liu et al., 2026) apply multi-turn reasoning and tool interaction to molecular retrosynthesis. For materials discovery, MatAgent (Takahara et al., 2025) couples crystal generation with property evaluation to guide iterative exploration, and MAPPS (Zhou et al., 2025) integrates workflow planning, executable tool-code generation, and scientist feedback. MatBrain (Shi et al., 2026) separates domain reasoning and tool execution into two specialized models, supporting structure generation, property prediction, and synthesis planning. Materealize (Kim et al., 2026) uses domain tools and multi-agent deliberation to develop synthesis recommendations, and A-Lab GPSS (Fei et al., 2026) connects agentic reasoning with experimental observations in solid-state materials discovery. Building on this line of work, PreWeave trains a single interaction policy for inorganic precursor recommendation, interleaving evidence acquisition, candidate selection, and compositional checks from a target composition to a final precursor set.

3 Method

Given a target inorganic composition, PreWeave learns an agent policy that reasons over precursor candidates, gathers relevant evidence, and revises its choices before producing a final recommendation. Each decision is conditioned on the preceding reasoning and observations, allowing the recommendation to develop through an evolving interaction history (Section 3.1). We train this policy in two stages. Supervised fine-tuning on distilled teacher trajectories provides demonstrations of intermediate decisions and their connection to final recommendations (Section 3.2). Online reinforcement learning then refines the policy using fresh trajectories generated by the agent itself, with rewards based on final-answer agreement and interaction-protocol compliance (Section 3.3).

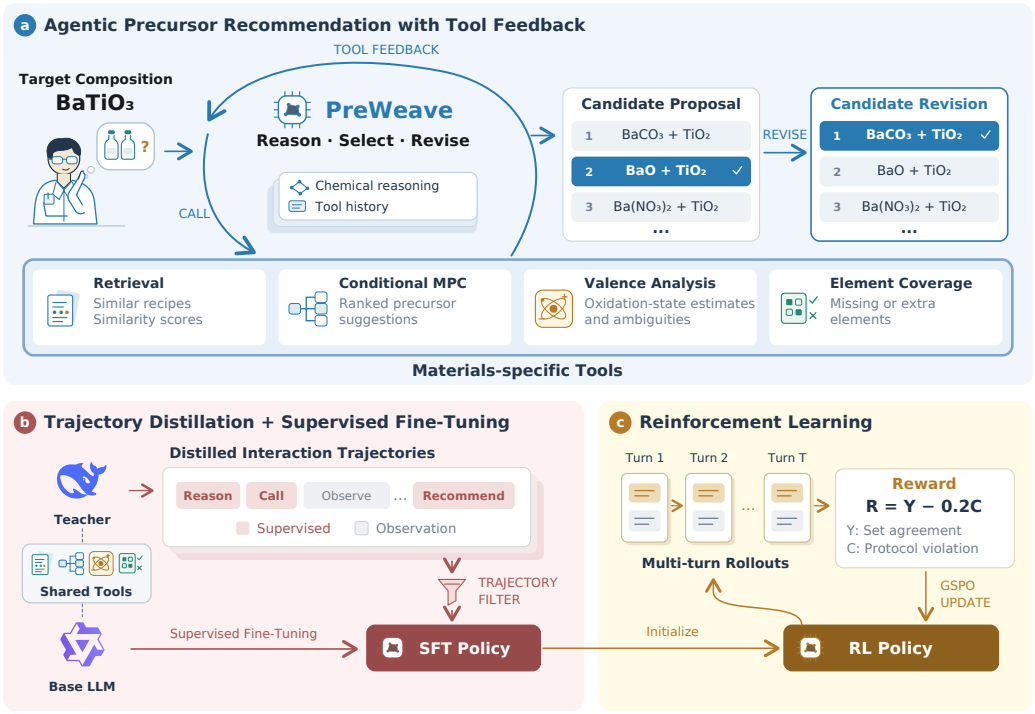


Figure 2: Overview of PreWeave. (a) Given a target composition, PreWeave agent reasons over precursor candidates, gathers evidence, and revises its choices through interaction before producing a final recommendation. The example illustrates a change in the selected candidate following tool feedback. (b) Teacher-generated trajectories are filtered and used for supervised fine-tuning, with reasoning, tool calls, and final recommendations as prediction targets and observations as context. (c) The SFT policy initializes online reinforcement learning, which optimizes the agent on online trajectories, using rewards based on final-answer agreement and protocol compliance.

3.1 Agent Interaction and Tool Environment

Agent–environment interaction. Given a target composition x, the agent reasons over precursor candidates and decides whether to gather further evidence or produce a final recommendation. Let h_t denote the interaction history before model turn t, including task instructions, previous model outputs, and tool observations. The policy generates reasoning r_t and an action a_t, consisting of one or more tool calls or a final recommendation. For a tool action, the environment returns an observation o_t, giving

$$(r_t, a_t) \sim \pi_\theta(\cdot \mid x, h_t), \qquad h_{t+1} = h_t \parallel (r_t, a_t, o_t). \tag{1}$$

Here, $\parallel$ denotes appending to the history, and o_t collects the observations when multiple calls are issued in one turn. This history conditions subsequent decisions, allowing the agent to reconsider candidates as evidence accumulates. Interaction proceeds within a fixed budget and yields a trajectory $\tau = (x, \{(r_t, a_t, o_t)\}_{t=0}^{T-1}, r_T, y)$, where y is the final precursor set.

Feedback from materials-specific tools. The tool environment reflects the intuition of a materials scientist approaching retrosynthesis: consulting synthesis precedents, considering which precursors could complement an existing choice, and checking the resulting candidates against basic chemical knowledge. We therefore provide tools for both candidate construction and consistency checks (Table 1). Retrieval and conditional precursor completion build on the models of He et al. (2023), while valence analysis and element-coverage checks use `pymatgen` with task-specific rules (Ong et al., 2013). Implementation details and tool settings are given in Appendix A.1.

Table 1: Tools available to the agent. Target compositions are supplied by the environment; query options and implementation details are given in Appendix A.1.

Tool	Query	Returned evidence
Retrieval	Target composition	Similar recipes and similarity scores
Conditional MPC	Target and partial precursor set	Ranked precursor suggestions
Valence analysis	Candidate precursors	Oxidation-state estimates and ambiguities
Element coverage	Candidate precursor set	Missing or extra elements

3.2 Supervised Fine-Tuning on Distilled Trajectories

We initialize the agent policy through supervised fine-tuning on teacher-generated interaction trajectories collected by rejection sampling. For each training target, the teacher generates a trajectory containing reasoning, tool calls, observations, and a final recommendation. Unsuccessful targets undergo bounded resampling according to the failure type. Teacher settings and the resampling procedure are detailed in Appendices A.2 and A.3.

Prior work on reasoning SFT highlights the value of carefully curated demonstrations (Ye et al., 2025). We apply three eligibility criteria:

- **Answer validity and agreement.** Valid precursor sets match a reference recipe and pass the required element-coverage checks.
- **Interaction integrity.** Complete trajectories contain valid tool interactions and no execution failures.
- **Training compatibility.** Untruncated conversations fit the student context limit, with no target overlap across training and evaluation splits.

For each target, we retain at most one trajectory that satisfies these criteria to form $\mathcal{D}_{\mathrm{SFT}}$.

Let $z^{\tau}_{1:L_\tau}$ denote the token sequence obtained from trajectory τ, and let m^{τ}_i indicate whether token i belongs to a supervised model output. We optimize the masked autoregressive objective

$$\mathcal{L}_{\mathrm{SFT}}(\theta) \propto -\sum_{\tau \in \mathcal{D}_{\mathrm{SFT}}} \sum_{i=1}^{L_\tau} m^{\tau}_i \log \pi_\theta(z^{\tau}_i \mid z^{\tau}_{<i}) \,. \tag{2}$$

The mask covers intermediate reasoning, tool names and arguments, and final recommendations across all model turns. Task instructions and tool observations are excluded from the prediction loss but remain in the conditioning context. This objective therefore supervises both tool-action generation and subsequent responses conditioned on returned evidence.

3.3 Online Reinforcement Learning for Policy Optimization

SFT initializes the policy from teacher demonstrations, but the student's own candidate choices lead to different tool observations and subsequent decisions. We therefore collect fresh interactions from the current policy and optimize them using outcome-based feedback.

Reward design. For a target x and a trajectory τ ending in precursor set y, we define

$$R(x, \tau) = Y(x, y) - \lambda_{\mathrm{p}} C(\tau), \qquad \lambda_{\mathrm{p}} = 0.2, \tag{3}$$

where $Y \in \{0, 1\}$ indicates composition-normalized, order-invariant agreement with a reference set, and $C \in \{0, 1\}$ indicates any interaction-protocol violation.

Policy optimization. For each sampled training target x, the current policy generates a group of K fresh interaction trajectories $\{\tau_i\}_{i=1}^{K}$. We compare their outcomes using group-centered rewards:

$$A_i = R(x, \tau_i) - \frac{1}{K} \sum_{j=1}^{K} R(x, \tau_j). \tag{4}$$

We use Group Sequence Policy Optimization (GSPO; Zheng et al., 2025) over multi-turn trajectories, assigning a shared advantage to reasoning, tool calls, and final answer within each trajectory. Tool observations remain in the conditioning context but are excluded from the optimization targets. A KL regularizer penalizes divergence from the fixed SFT policy. Each fresh batch is used for one optimizer step, after which the updated policy collects new trajectories. Appendix A.4 provides the trajectory likelihood ratio, full optimization objective, and implementation details.

4 Main Results

In this section, we evaluate PreWeave against task-specific precursor predictors and frontier LLMs on inorganic precursor recommendation. We assess both single-prediction accuracy and coverage across multiple candidates.

4.1 Experimental Setup

Data and scoring. We use the established inorganic precursor recommendation benchmark of He et al. (2023), which follows a chronological split with 24,034 training, 1,910 validation, and 2,654 test targets. A prediction is considered correct if the predicted precursor set exactly matches at least one reference set for the target. We compare precursor sets by normalized chemical composition, ignoring formula notation and precursor order. Appendix B provides further details.

Evaluation protocols. Prior precursor predictors produce ranked candidate lists. In contrast, PreWeave returns one final precursor set per independent run and does not assign scores for ranking candidates across runs. We retain ranked evaluation for prior methods and use empirical Pass@k to assess the reliability and coverage of sampled predictions.

- **Ranked evaluation.** For ranked candidate lists, we present Hit@k results: a target is counted as correct if any of the first k candidates matches a reference precursor set.
- **Sampled evaluation.** We report empirical Pass@k, the fraction of targets with a reference-matching prediction among the first k independent runs. Pass@1 measures single-run accuracy. Larger k measures coverage across repeated attempts.

Both protocols use $k \in \{1, 2, 4, 8\}$, subject to the available number of runs. Run ordering, decoding settings, and interaction budgets are detailed in Appendix B.

Baselines. He et al. (2023) recommend precursors by transferring synthesis information from materials with similar learned composition representations. CrabNet (Wang et al., 2021) uses an attention-based composition encoder, adapted here to predict precursor labels. Retrieval-Retro (Noh et al., 2024) combines retrieved synthesis precedents with thermodynamic expert knowledge for precursor prediction. MSP-LLM (Noh et al., 2026) generates precursors through an intermediate material-class representation with LLM beam search. To assess tool access and policy training, we also evaluate the untrained Qwen3.5-9B (Qwen Team, 2026a) backbone through direct generation and as a tool-enabled agent, denoted PreWeave (Base Agent), using the full tool environment without task-specific fine-tuning. We additionally evaluate Qwen3.5-9B (Answer-only SFT), fine-tuned to predict precursor sets without reasoning traces or tools. To limit API evaluation costs, we compare with tool-augmented frontier LLMs on a shared subset of 512 targets randomly sampled from the test set. Frontier LLMs use a smolagents (Roucher et al., 2025) ToolCallingAgent with the same tools and system prompt as PreWeave. Baseline implementations are detailed in Appendix B.2.

4.2 Precursor Recommendation Performance

In Panel A of Table 2, MSP-LLM achieves the highest Hit@k at all four budgets. Among the other predictors, He et al. (2023) achieve substantially greater coverage as the shortlist grows, while CrabNet and Retrieval-Retro perform better at Hit@1 but gain less from additional candidates. Candidate search and ranking can yield higher shortlist coverage than independent sampling, which may repeatedly produce the same answer; these scores therefore reflect different candidate-generation procedures. Additionally, these baselines return precursor candidates without explicit reasoning and tool-verification traces, limiting interpretability and the supporting evidence available to users.

Table 2: **PreWeave improves single-run reliability under a common 9B backbone and remains competitive with tool-augmented frontier LLMs.** Panel A reports ranked Hit@k, while Panels B and C report empirical Pass@k. Panel B compares the backbone, answer-only fine-tuning, tool access, and agentic training; Panel C places the final policy against frontier systems on a shared 512-target subset. API costs limit Gemini, GPT, and Opus to Pass@2. Gray shading distinguishes ranked evaluation; bold and underlined values denote the best and second-best results within each panel.

Method	$k=1$	$k=2$	$k=4$	$k=8$
A. Ranked precursor prediction: Hit@k ($N = 2{,}654$)				
CrabNet (Wang et al., 2021)	54.94	57.01	57.91	59.57
He et al. (2023)	36.32	<u>73.74</u>	<u>81.16</u>	<u>84.74</u>
Retrieval-Retro (Noh et al., 2024)	<u>57.23</u>	60.78	62.06	63.90
MSP-LLM (Qwen3.5-9B) (Noh et al., 2026)	**73.89**	**78.03**	**82.03**	**85.34**
B. Independent sampling: Pass@k ($N = 2{,}654$)				
Qwen3.5-9B (Thinking) (Qwen Team, 2026a)	34.82	51.54	65.11	72.46
Qwen3.5-9B (Answer-only SFT)	68.88	72.08	74.34	76.45
PreWeave (Base Agent)	53.81	66.69	74.04	78.22
PreWeave (SFT)	<u>71.89</u>	<u>74.94</u>	**76.90**	<u>78.26</u>
PreWeave (SFT + RL)	**73.02**	**75.21**	<u>76.83</u>	**78.33**
C. Tool-augmented frontier LLMs: Pass@k ($N = 512$)				
Qwen 3.8 Max (Qwen Team, 2026b)	73.24	<u>75.78</u>	<u>77.73</u>	78.71
GLM 5.3 (Z.ai, 2026)	71.09	74.41	76.17	77.15
Kimi K3 (Kimi Team et al., 2026)	72.46	75.39	76.37	77.93
DeepSeek V4 Flash (DeepSeek-AI, 2026b)	68.55	74.22	**78.13**	<u>79.49</u>
Gemini 3.5 Flash (Google DeepMind, 2026)	<u>73.83</u>	75.00	—	—
GPT 5.6 Sol (OpenAI, 2026)	73.44	75.59	—	—
Opus 5 (Anthropic, 2026)	**74.22**	75.39	—	—
PreWeave (SFT + RL)	73.44	**75.98**	**78.13**	**80.08**

Policy training improves the agent's single-run accuracy. In Panel B, under the shared agent evaluation, SFT raises Pass@1 from 53.81% to 71.89%, with RL further reaching 73.02%. The training-stage results suggest that policy learning primarily increases the likelihood of obtaining a reference-matching answer in one run, rather than substantially expanding coverage across repeated attempts. Compared with the established precursor-prediction baselines in Panel A, PreWeave additionally provides an inspectable tool-interaction trace. We also implement independent sampling for MSP-LLM, the strongest ranked baseline in Panel A; PreWeave (SFT + RL) retains higher Pass@1 under both validation-selected sampling configurations (Appendix B.2).

PreWeave achieves competitive accuracy with tool-augmented frontier LLMs. On the shared 512-target subset (Panel C), the RL-trained agent attains the highest reported Pass@2 and Pass@8, and ties DeepSeek at Pass@4. Its Pass@1 remains close to the strongest frontier models. These results show that task-specific agent training enables a 9B model to achieve competitive single-run accuracy and multi-run coverage.

5 Analysis

We analyze how PreWeave acquires evidence, allocates interactions, and learns to turn tool feedback into precursor recommendations. We examine four complementary aspects of the agent: the contribution of each tool (Section 5.1), the interaction budget needed for effective recommendation (Section 5.2), changes in reliability and tool use through training (Section 5.3), and the use of feedback in revision and candidate assessment (Section 5.4). Together, these analyses characterize the agent's recommendation reliability and inspectable reasoning process. Appendix C.5 further examines online RL dynamics, including optimization gains, and evolving generation behavior.

Table 3: Tool ablation results on test set.

Configuration	P@1	P@2	P@4
Full (four tools)	71.89	74.94	**76.90**
w/o Retrieval	70.08	73.70	75.77
w/o MPC	71.82	74.68	76.53
w/o Valence	**72.04**	**75.24**	76.83
w/o Element coverage	71.97	74.94	76.60
No tools	68.31	72.65	75.66

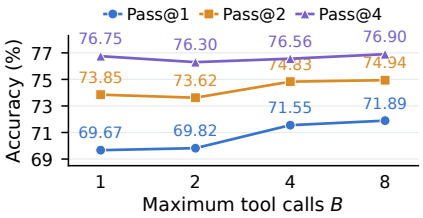


Figure 3: Pass@1/2/4 versus call budget.

5.1 Tool-Access Ablation

We examine how synthesis precedents, precursor suggestions, and compositional checks contribute to precursor recommendation. Using the SFT checkpoint, we compare the full tool set with four single-tool ablations and a no-tool variant (Table 3). Each ablation removes the corresponding tools and instructions at inference time without retraining. Pass@1/2/4 captures their effects on single-run accuracy and coverage across repeated attempts.

Result. Tool access primarily improves single-run reliability: the full agent exceeds the no-tool variant by 3.58 percentage points at Pass@1, but only 1.24 points at Pass@4. Retrieval contributes the largest individual gain, suggesting that synthesis precedents help identify a reference-matching set earlier. Removing Valence analysis or Element coverage has little effect on exact-match accuracy and slightly improves Pass@1. Valence-analysis and element-coverage outputs make oxidation-state assumptions and elemental accounting explicit in the interaction trace.

5.2 Tool-Call Budget Analysis

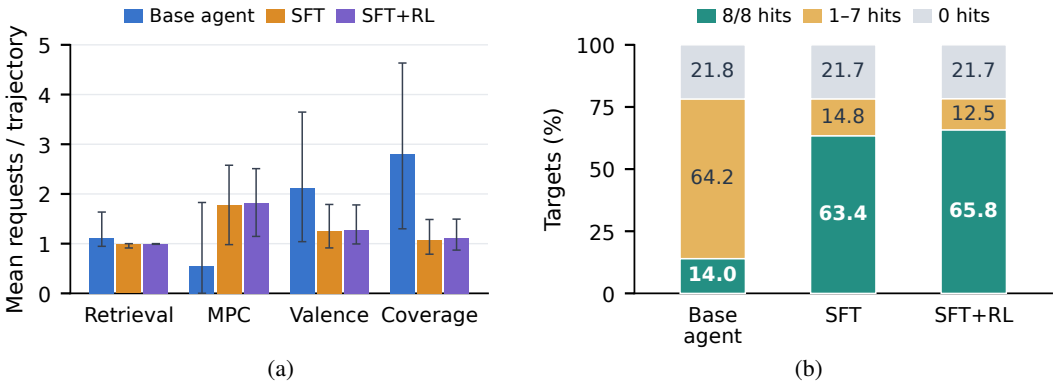
We evaluate how tool-call budgets ($B \in \{1, 2, 4, 8\}$) affect SFT recommendation accuracy (Figure 3). All settings use the full tool set and the same prompts and decoding configuration, evaluated on 2,654 test targets with four seeds. Once the call budget or generation-length limit is reached, the agent is prompted to give its final recommendation.

Result. Larger tool budgets improve single-run accuracy more than coverage across repeated attempts. Pass@1 increases with the call allowance, whereas Pass@4 varies by only 0.60 percentage points and shows no monotonic trend. This suggests that additional interactions primarily help the agent reach a reference-matching recommendation within a single attempt, without substantially expanding aggregate coverage across attempts. Gains diminish beyond four calls: increasing the allowance to eight improves Pass@1 by only 0.34 percentage points. Together, these results indicate that much of the benefit of tool interaction is realized within a modest call budget.

5.3 Training Effects on Reliability and Tool Use

We examine how agentic training changes tool use and recommendation consistency across repeated runs. Figure 4 compares the base agent, SFT, and SFT+RL on the same 2,654 test targets, using eight runs per stage with shared decoding settings. We measure tool-request counts and the number of reference-matching runs per target; Appendix C.3 provides additional statistics.

Post-training makes reference-matching recommendations more consistent across runs. The proportion of targets matched in all eight runs rises from **13.98%** for the base agent to **63.41%** after SFT and **65.79%** after RL, whereas coverage by at least one run remains nearly unchanged. The main gain thus lies in reliably selecting a reference-matching set rather than substantially expanding observed coverage. Alongside this change, SFT increases MPC use and reduces valence-analysis and element-coverage checks, lowering mean requests from 6.59 to 5.08. This redistribution is consistent with the small exact-match effects of removing either checking tool and the diminishing returns from larger call budgets. RL largely preserves the SFT interaction pattern while making retrieval nearly universal, increasing its use from 98.39% to 99.81% of trajectories. The largest changes in both outcome consistency and request allocation therefore occur during SFT, with smaller refinements at the evaluated RL checkpoint.



(a) (b)

Figure 4: Training effects on 2,654 test targets with eight runs per stage. (a) Mean tool requests per trajectory, including rejected requests; whiskers span the mean per-target minimum and maximum across runs. (b) Target fractions with eight, one to seven, or zero reference-matching runs.

5.4 Case Studies

The selected trajectories illustrate how the agent uses tool feedback to revise and assess precursor candidates. For the Zr–Ni–Pb–Sn–Bi target, an unresolved-oxygen warning prompts the agent to revisit its initial oxide interpretation and switch to elemental precursors (Appendix C.4.1). For the Eu-doped fluorophosphate, the agent responds to a missing-element check by adding the omitted Eu source and rechecking the complete set, showing how explicit accounting can correct an incomplete recommendation (Appendix C.4.2). For the Ce-doped nitride, the agent combines valence-analysis and element-coverage observations with stoichiometric reasoning to retain an independently proposed CeN candidate beyond the returned MPC suggestions (Appendix C.4.3). All three final sets match their references. These cases illustrate how the agent integrates tool feedback with chemical reasoning to revise or retain precursor choices.

6 Discussion

Broader Impact. The training approach studied here may inform materials agents beyond precursor recommendation, particularly for tasks that combine literature evidence, scientific models, and candidate assessment. Extending the tool environment would allow domain researchers to contribute additional sources of materials knowledge and examine how agents use them. This offers a route to incorporating specialized scientific expertise into computational materials workflows.

Limitations and Future Work. Our tool environment covers synthesis precedents, conditional precursor suggestions, and compositional checks, but provides limited evidence on reaction thermodynamics and kinetics. Integrating additional scientific models and experimental feedback would broaden the basis for candidate assessment. Evaluation by agreement with reported precursor sets leaves the experimental feasibility of alternative recommendations unresolved. Extending the agent to synthesis operation and temperature prediction would support more complete synthesis plans.

Conclusion. We introduced PreWeave, an LLM agent whose tool-interaction policy connects evidence acquisition with inorganic precursor decisions. Supervised fine-tuning on distilled trajectories and online reinforcement learning improve single-run accuracy, yielding performance competitive with the evaluated task-specific models and tool-augmented frontier LLMs. Training also makes reference-matching recommendations more consistent across runs and shifts tool use toward candidate construction, with inspectable traces of candidate assessment and revision. These findings support agent policy training as a practical approach to improving precursor recommendation and characterize how training changes tool use. We will release tools, trajectories, and code to enable further research on materials agents that turn scientific evidence into synthesis decisions.

AI USE STATEMENT

We used generative AI tools to assist with manuscript drafting and polishing, literature retrieval and discovery, code implementation and debugging, figure preparation, and experimental result analysis. We also used LLMs to generate the synthetic interaction trajectories used for training, as described in Section 3. The authors determined the core research direction and retain responsibility for all methodological decisions, verification of AI-assisted content and code, and the accuracy and integrity of the final manuscript and reported results.

ETHICS STATEMENT

This work explores LLM agents for inorganic precursor recommendation using literature-derived synthesis data. Such recommendations may involve hazardous substances or be applied in inappropriate contexts. Our evaluation is computational and measures agreement with recorded precursor sets; it does not establish experimental feasibility or safety. The framework is intended to support materials research, and any experimental use requires independent assessment of chemical compatibility, toxicity, handling requirements, and applicable laboratory safety procedures by qualified researchers.

REPRODUCIBILITY STATEMENT

Section 3 describes the tool environment, agent interaction, and training objectives. Appendix A details tool implementations, trajectory generation and filtering, and SFT and online RL configurations. Appendix B documents scoring, run aggregation, baseline implementations, and inference settings. Appendix C provides additional analyses and representative interaction traces. An initial anonymized piece of code is included in the supplementary material. We will publicly release the code, data, and interaction trajectories, along with configuration files and instructions for data preparation, training, and evaluation, to support reproduction and further research.

REFERENCES

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. POLARIS: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL https://hkunlp.github.io/blog/2025/Polaris/.

Anthropic. System card: Claude Opus 5, July 2026. URL https://www-cdn.anthropic.com/b514064af1408018e64b1ad24e7d5e75850b4ffd/Claude%20Opus%205%20System%20Card.pdf.

Muratahan Aykol, Joseph H Montoya, and Jens Hummelshøj. Rational solid-state synthesis routes for inorganic materials. *Journal of the American Chemical Society*, 143(24):9244–9259, 2021.

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026a. URL https://arxiv.org/abs/2606.19348.

DeepSeek-AI. DeepSeek-V4-Flash-0731, 2026b. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash-0731. Model card.

Yuxing Fei, Bernardus Rendy, Xiaochen Yang, Junhee Woo, Xu Huang, Chang Li, Shilong Wang, David Milsted, Yan Zeng, and Gerbrand Ceder. Agentic llm reasoning in a self-driving laboratory for air-sensitive lithium halide spinel conductors. *arXiv preprint arXiv:2604.11957*, 2026.

Google DeepMind. Gemini 3.5 Flash, May 2026. URL https://deepmind.google/models/model-cards/gemini-3-5-flash/. Model card.

Tanjin He, Haoyan Huo, Christopher J Bartel, Zheren Wang, Kevin Cruse, and Gerbrand Ceder. Precursor recommendation for inorganic synthesis by machine learning materials similarity from scientific literature. *Science advances*, 9(23):eadg8180, 2023.

Edward Kim, Zach Jensen, Alexander van Grootel, Kevin Huang, Matthew Staib, Sheshera Mysore, Haw-Shiuan Chang, Emma Strubell, Andrew McCallum, Stefanie Jegelka, et al. Inorganic materials synthesis planning with literature-trained neural networks. *Journal of chemical information and modeling*, 60(3):1194–1201, 2020.

Seongmin Kim, Yousung Jung, and Joshua Schrier. Large language models for inorganic synthesis predictions. *Journal of the American Chemical Society*, 146(29):19654–19659, 2024a.

Seongmin Kim, Juhwan Noh, Geun Ho Gu, Shuan Chen, and Yousung Jung. Predicting synthesis recipes of inorganic crystal materials using elementwise template formulation. *Chemical Science*, 15(3):1039–1045, 2024b.

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

Kimi Team et al. Kimi K3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Olga Kononova, Haoyan Huo, Tanjin He, Ziqin Rong, Tiago Botari, Wenhao Sun, Vahe Tshitoyan, and Gerbrand Ceder. Text-mined dataset of inorganic materials synthesis recipes. *Scientific data*, 6(1):203, 2019.

Wei Liu, Jiangtao Feng, Hongli Yu, Yuxuan Song, Yuqiang Li, Shufei Zhang, Lei Bai, Wei-Ying Ma, and Hao Zhou. Retro-r1: Llm-based agentic retrosynthesis. *Advances in Neural Information Processing Systems*, 38:70709–70737, 2026.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature machine intelligence*, 6(5):525–535, 2024.

Krzysztof Maziarz, Guoqing Liu, Felix Pultar, John Gardner, Tobias Gensch, Jean Helie, Hubert Misztela, Austin Tripp, Junren Li, Aleksei Kornev, et al. Chemist-aligned retrosynthesis by ensembling diverse inductive bias models. *Nature*, pp. 1–3, 2026.

Heewoong Noh, Namkyeong Lee, Gyoung S Na, and Chanyoung Park. Retrieval-retro: retrieval-based inorganic retrosynthesis with expert knowledge. *Advances in Neural Information Processing Systems*, 37:25375–25400, 2024.

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

Shyue Ping Ong, William Davidson Richards, Anubhav Jain, Geoffroy Hautier, Michael Kocher, Shreyas Cholia, Dan Gunter, Vincent L Chevrier, Kristin A Persson, and Gerbrand Ceder. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68:314–319, 2013.

OpenAI. GPT-5.6 system card, July 2026. URL https://deploymentsafety.openai.com/gpt-5-6.

Elton Pan, Thorben Prein, Juno Nam, Xiaochen Du, Soojung Yang, Pengfei Cai1 Jennifer LM Rupp, Rafael Gomez-Bombarelli, and Elsa Olivetti. Synreason: Enhancing synthesis reason-ing via reinforcement learning experimental feedback.

Mianzhi Pan, Jianfei Li, Peishuo Liu, Botian Wang, Yawen Ouyang, Yiming Rong, Wei-Ying Ma, Hao Zhou, and Jianbing Zhang. Enhancing spatial reasoning in large language models for metal-organic frameworks structure prediction. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 11750–11761, 2026.

Thorben Prein, Elton Pan, Janik Jehkul, Steffen Weinmann, Elsa Olivetti, and Jennifer LM Rupp. Language models enable data-augmented synthesis planning for inorganic materials. *ACS Applied Materials & Interfaces*, 17(51):69221, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026b. URL https://qwen.ai/blog?id=qwen3.8.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 'smolagents': a smol library to build great agentic systems. https://github.com/huggingface/smolagents, 2025.

Tongyu Shi, Yutang Li, Zhanyuan Li, Qian Liu, Jie Zhou, Wenhe Xu, Yang Li, Dawei Dai, Rui He, Wenhua Zhou, et al. A collaborative agent with two lightweight synergistic models for autonomous crystal materials research. *Nature Machine Intelligence*, pp. 1–16, 2026.

Izumi Takahara, Teruyasu Mizoguchi, and Bang Liu. Accelerated inorganic materials design with generative ai agents. *Cell Reports Physical Science*, 6(12), 2025.

Anthony Yu-Tung Wang, Steven K Kauwe, Ryan J Murdock, and Taylor D Sparks. Compositionally restricted attention-based network for materials property predictions. *Npj Computational Materials*, 7(1):77, 2021.

Yifei Wang, Qizhi Pei, Jiangtao Feng, Yuntian Shi, Yi Duan, Lihao Wang, Lei Bai, Lijun Wu, Wei-Ying Ma, and Hao Zhou. R3: End-to-end reasoning-based planning for multi-step retrosynthesis via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 37618–37632, 2026.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

Z.ai. GLM-5.3: Frontier coding with emergent cyber capabilities, August 2026. URL https://z.ai/blog/glm-5.3.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Gspo: Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Lianhao Zhou, Hongyi Ling, Keqiang Yan, Kaiji Zhao, Xiaoning Qian, Raymundo Arróyave, Xiaofeng Qian, and Shuiwang Ji. Toward greater autonomy in materials discovery agents: Unifying planning, physics, and scientists. *arXiv preprint arXiv:2506.05616*, 2025.

Yanqiao Zhu, Jingru Gan, Xiaoqi Sun, Fang Sun, Yidan Shi, Md Mofijul Islam, Chao Shang, Wenhao Gao, Connor W Coley, Yizhou Sun, et al. Retroagent: Harnessing llms to search over structured memory for agentic retrosynthesis planning. *arXiv preprint arXiv:2607.14512*, 2026.

A Method Implementation Details

A.1 Tool Implementations and Information Access

Retrieval returns reference compositions, precursor sets, and similarity scores; conditional MPC returns ranked candidates, model scores, and element annotations. Similarity and model scores are not calibrated synthesis-success probabilities. Valence analysis reports oxidation-state estimates, assignment status, and ambiguity, without establishing redox feasibility. Element coverage reports missing or extra elements and the accounting status under declared external sources and byproducts. We use Element coverage for the tool implemented as `check_element_conservation`; verbatim trajectory excerpts retain the original wording.

The retrieval tool and conditional MPC tool use the released encoder and precursor model of He et al. (2023). The retrieval collection contains 26,781 recipes from the training split. Compositions are

represented by normalized 32-dimensional embeddings and ranked by cosine similarity, excluding exact target-composition matches. The agent can request 1–20 neighbors (default 5), with retrieval limited to one call per trajectory. Conditional MPC accepts a nonempty partial precursor set and returns 1–5 suggestions (default 3) from the released 417-class vocabulary. If all supplied precursors are out of vocabulary, it returns a no-query observation. This vocabulary limits the completion tool, not the precursors allowed in the final recommendation; an out-of-vocabulary response does not establish chemical invalidity. The released model weights are used without retraining; restricting the retrieval collection to training recipes does not constitute an audit of the pretrained model's information access.

The valence tool builds on `pymatgen` composition parsing and oxidation-state assignments (Ong et al., 2013), with hydrate handling and a fallback based on common oxidation states and fractional assignments. The element-coverage tool applies custom rules to parsed compositions. Its default accounting permits C, H, and N as potential byproduct elements. Optional declarations cover external H, N, or O sources and removal of O, F, Cl, Br, I, S, Se, or P; these declarations are checked against the wrapper's rules. An accepted element-coverage check denotes element presence under this accounting, rather than a balanced reaction. Both tools operate on formulas without crystal structures, energies, or reaction conditions.

A.2 Teacher Generation and SFT Configuration

We generate demonstrations with DeepSeek-v4-Flash (DeepSeek-AI, 2026b;a) in thinking mode with high reasoning effort. The API requests omit temperature and top-p in this mode. The main generation budget permits 6,144 output tokens per model reply, with at most ten replies and eight tool calls over the full trajectory. Initial recovery configurations increase the per-reply limit to 12,288 tokens; later length-recovery attempts use up to 16,384. Length recovery retains the eight-call and ten-reply limits, while budget recovery permits ten calls and twelve replies. These are generation limits; the student context limit applies separately to the complete encoded conversation. Reference precursor sets are used for screening and are not supplied in the teacher's task context.

We train Qwen3.5-9B Base (Qwen Team, 2026a) with full language-model fine-tuning, keeping the vision encoder and projector frozen. The conversation template preserves reasoning and tool calls. All assistant reasoning, tool-call arguments, and final answers receive supervision; instructions, tool observations, and runtime reminders provide context only. The student context limit is 32,768 tokens, measured over the complete conversation using the student tokenizer and template. Overlength examples are excluded before training; retained conversations are neither truncated nor packed. Table 4 summarizes the training configuration. Online RL initializes from the first-epoch checkpoint (update 1,313), which is also the SFT reference in the main evaluation.

Table 4: SFT configuration. The run spans five epochs; the reported SFT model and RL initialization use the first-epoch checkpoint.

Hyperparameter	Value
Backbone / trainable parameters	Qwen3.5-9B Base / full language model
Training examples / context limit	21,008 / 32,768 tokens
Global / per-device batch size	16 / 1 (gradient accumulation: 1)
Learning rate / schedule	10^{-5} / cosine decay
Warmup / weight decay	276 updates / 0.01
Gradient clipping	1.0
Training epochs / updates per epoch	5 / 1,313
Reported checkpoint	Epoch 1, update 1,313
Model / data seed	42 / 42
Precision	BF16; FP32 master parameters and optimizer states
Hardware / parallelism	16 A100 80GB GPUs on two nodes / hybrid sharding
Sequence handling	No packing or truncation; gradient checkpointing enabled

A.3 Trajectory Filtering and Dataset Construction

We first generate one trajectory for each training target. Targets without an accepted trajectory undergo a bounded sequence of fresh attempts selected by failure type: length failures receive a larger per-reply output budget, budget failures or missing final element-coverage checks receive an expanded interaction budget, and protocol-valid answers that fail reference matching receive an independent resample under the main budget. The initial procedure allows one recovery stage of each applicable type and one resample for a reference mismatch, stopping once a protocol-valid, reference-matching trajectory is obtained. A subsequent recovery pass generates one additional trajectory for each of 176 unresolved length-related cases using the budgets in Appendix A.2. Each attempt starts a new interaction, and reference answers are used only for screening. The initial procedure generated 28,675 complete trajectories across 24,034 targets, with one to four trajectories per target; the additional recovery increased the maximum to five. Multiple model replies within a trajectory and HTTP request retries are not independent trajectory samples. We select at most one accepted trajectory per target before applying the student-context length check.

Table 5 expands the three eligibility criteria in Section 3.2. Screening uses the completed interaction record and reference precursor sets. Intermediate negative observations may remain in an accepted trajectory; the checks do not label each reasoning step or tool choice as correct.

Table 5: Eligibility checks for teacher demonstrations.

Criterion	Required checks
Answer validity and agreement	Parseable final precursor set; composition-normalized, order-invariant match to at least one reference; a matching accepted element-coverage check and offline recheck, including supported external-source or byproduct declarations where applicable.
Interaction integrity	Recorded final answer at termination; no length-truncated replies, model-request failures, or tool-backend failures; valid tool names and parseable arguments; paired calls and observations; compliance with tool availability and call budgets.
Training compatibility	Complete conversation within the student context limit under its tokenizer and template; model outputs preserved for supervision; no composition-normalized target overlap with validation or test. Overlength conversations are excluded without truncation.

Generation and recovery yield 21,010 candidate demonstrations for the 24,034 training targets, leaving 3,024 without an accepted demonstration. The candidate pool comprises 20,976 trajectories from the initial generation and bounded-resampling procedure and 34 from the subsequent recovery pass. Within the recorded generation procedure, 20,171 targets yielded a qualified trajectory in the initial main stage and 805 through subsequent resampling or recovery. The initial-stage count refers to this recorded procedure rather than the first-ever sample for each target; some targets had appeared in earlier pilot or smoke runs. Imported attempts retain their original stage and attempt numbering. Applying the student tokenizer and conversation template excludes two overlength conversations, producing 21,008 SFT examples. The longest retained sequence contains 31,300 tokens within the 32,768-token limit. Composition-normalized checks find no overlap between the retained training targets and the validation or test targets.

A.4 Online Rollout and Policy Optimization

Fresh rollouts and target sampling. Each batch contains 64 training targets with $K = 8$ fresh trajectories per target (512 trajectories). Historical SFT rollouts are used only to construct the target-sampling schedule; their trajectories are not reused for policy updates. The original sampling pool contains 24,015 literal target groups from the 24,034 training records. A mixed-outcome subset contains 3,447 targets for which the historical eight-sample group has between one and seven correct answers. Over each five-batch cycle, 80% of target slots are drawn from this subset and the remainder from the original pool, excluding targets already selected in that batch. The pools overlap, so 80% describes the sampling allocation rather than the exact fraction of mixed-outcome targets in a batch. The schedule uses seed 20260922. Fresh groups are retained regardless of correctness; equal-reward groups have zero task advantage but still participate in KL regularization.

Reward and execution failures. The protocol indicator in Equation 3 covers invalid tool calls, budget violations, and invalid final-answer formats. Reaching the interaction budget triggers finalization and does not itself constitute a protocol violation. Budget violations refer to requests that exceed the permitted call limits. Such violations and invalid final-answer formats each set the same binary indicator C to one; multiple violations incur only one penalty. Valid out-of-vocabulary queries and empty suggestions are observations rather than violations. Infrastructure failures or unclassified errors interrupt batch processing and prevent an update on that batch. Centering without standard-deviation normalization preserves the smaller reward differences induced by protocol penalties.

Trajectory masks and likelihoods. Let π_{old} denote the policy snapshot used to collect a fresh group $\mathcal{G}_x = \{\tau_i\}_{i=1}^{K}$. For generated-token indices $\mathcal{I}_i$ across all turns of trajectory i, the length-normalized likelihood ratio is

$$\rho_i(\theta) = \exp\left[\frac{1}{|\mathcal{I}_i|}\sum_{\ell\in\mathcal{I}_i}\log\frac{\pi_\theta(z_\ell^{\tau_i} \mid z_{<\ell}^{\tau_i})}{\pi_{\text{old}}(z_\ell^{\tau_i} \mid z_{<\ell}^{\tau_i})}\right]. \tag{5}$$

Each model turn is encoded with its recorded runtime prefix. The action mask includes generated reasoning, tool calls, and final answers, while excluding instructions, observations, runtime reminders, and end-of-sequence tokens inserted by the runtime. For turn t of trajectory i, let L_{it} be its number of participating tokens and $L_i = \sum_t L_{it}$. Masked log-probability differences are summed across turns and divided by L_i to obtain Equation 5. Separately encoded turns share the trajectory advantage and ratio, with contributions weighted by L_{it}/L_i.

The denominator is recomputed under the pre-update policy and detached for loss computation. It is refreshed for each new batch, whereas the KL reference remains the initial SFT model. The ratio uses raw model probabilities without decoding transformations such as temperature or sampling truncation; it is therefore a model-likelihood surrogate rather than an exact correction for the decoding distribution.

Optimization objective. Using the centered advantages in Equation 4, the implemented GSPO objective is

$$\begin{aligned}\mathcal{J}(\theta) = \mathbb{E}_{\mathcal{G}_x\sim\mathcal{D}_{\text{old}}}\left[\frac{1}{K}\sum_{i=1}^{K}\min(\rho_i A_i, \text{clip}(\rho_i, 1-\epsilon_-, 1+\epsilon_+)A_i)\right] \\ - \beta\mathcal{L}_{\text{KL}}(\theta;\pi_{\text{ref}}),\end{aligned} \tag{6}$$

where $\mathcal{D}_{\text{old}}$ is the distribution of fresh trajectory groups and $\pi_{\text{ref}} = \pi_{\text{SFT}}$ is fixed throughout training. Each complete trajectory contributes one shared advantage and ratio. The clipping bounds, KL coefficient, and optimizer settings are listed in Table 6.

Optimization settings. We take one optimizer step per fresh batch, synchronize actor weights with the sampler, and then collect the next batch. With consistent pre-update scoring, the ratio equals one at gradient evaluation, so clipping normally does not constrain this single step; the detached denominator still permits a policy gradient through the numerator. Table 6 gives the optimizer and rollout configuration for the run underlying the reported step-200 checkpoint. The main evaluation uses the step-200 checkpoint; Appendix C.5 discusses its selection from the longer training run.

Stage-specific interaction settings. The online prompt clarifies that the MPC vocabulary limits tool queries rather than valid final precursor sets: an out-of-vocabulary result does not establish chemical invalidity, and partial queries are interpreted only under their reported conditions. This clarification is also applied in matched evaluation; the teacher demonstrations are not regenerated. Training uses fresh interaction episodes without conditional length recovery. Evaluation stopping and recovery rules are described in Appendix B.

Table 6: Online RL configuration for the reported model, trained for 200 updates.

Hyperparameter	Value
Initialization / fixed KL reference	SFT epoch 1 (update 1,313)
Targets / trajectories per target	64 / 8 (512 trajectories per update)
Target sampling	80% mixed-outcome pool, 20% original pool
Sampling-schedule seed	20260922
Reward / advantage	$Y - 0.2C$ / group centering, no standard-deviation scaling
Optimizer / steps per rollout batch	Adam (Kingma & Ba, 2014) with CPU offload / 1
Learning rate / schedule / warmup	10^{-6} / constant / 10 updates
Adam $(\beta_1, \beta_2, \epsilon)$	$(0.9, 0.98, 10^{-8})$
Weight decay / gradient clipping	0.05 / 1.0
Ratio clipping interval	$[0.80, 1.28]$
KL estimator / coefficient	`low_var_kl` / 0.01
Entropy regularization coefficient	0
Training updates	200
Temperature / top-p / top-k	1.0 / 0.95 / 20
Output limit / final-answer reserve	5,120 / 4,096 tokens
Soft / hard context limit	16,384 / 32,768 tokens
Tool calls / tool rounds per trajectory	8 / 10
Dynamic-batch token budget	6,000 tokens per GPU
Tensor / pipeline / context parallelism	2 / 1 / 1

A.5 Computational Resources and Cost

Supervised fine-tuning uses 16 NVIDIA A100 80GB GPUs and 21,008 tool-interaction demonstrations. Online reinforcement learning initially uses 64 NVIDIA H200 GPUs and subsequently scales to 128 GPUs. The reported checkpoint is trained for 200 updates, with 64 targets and eight fresh trajectories per target in each update, corresponding to 102,400 training rollouts.

We estimate the API expenditure at approximately US$500 for teacher-trajectory generation and US$1,000 for frontier-LLM evaluation. These approximate costs exclude local GPU computation. To accommodate evaluation costs, frontier models are evaluated on a shared, randomly sampled subset of 512 test targets, with fewer independent runs for the most expensive models.

B Experimental Details

B.1 Scoring and Run Aggregation

Composition matching. The shared agent scorer parses each precursor formula with `pymatgen.Composition`, reduces its composition, and constructs a canonical key from sorted element–amount pairs, rounding amounts to eight decimal places. It sorts the precursor keys before comparing the prediction with each reference set. Formula notation and precursor order therefore do not affect the score. The implementation preserves the number of entries rather than silently deduplicating repeated precursors. A match to any reference set counts as success; this composition-level criterion does not distinguish crystal structures.

Scoring. For target i, let $y_{ij} \in \{0, 1\}$ indicate whether independent run j exactly matches any reference precursor set. Runs follow a fixed, prespecified order. We compute empirical Pass@k as

$$\text{Pass@}\,k = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[\max_{1 \le j \le k} y_{ij} = 1\right], \qquad k \le n. \tag{7}$$

We report empirical Pass@k using the first k independent runs in a prespecified run order. Completed rows in Panel B use $n = 8$. In Panel C, Qwen, GLM, Kimi, DeepSeek, and the RL subset reference use $n = 8$, while Gemini, GPT, and Opus use $n = 2$. Unparseable or incomplete answers remain in the denominator. Pass@1 is the accuracy of the first run, while Pass@8 measures union

coverage rather than the accuracy of an answer-selection rule. Ranked Top-k instead checks whether a reference set appears among the first k ranked candidates. Full-test scores retain out-of-vocabulary targets. For CrabNet and Retrieval-Retro, the 798-label precursor vocabulary yields a precursor-IV subset of 2,558 targets; the remaining 96 targets have no fully representable reference set. Table 2 reports their full-test scores, counting these 96 targets as misses.

B.2 Baseline Implementations and Inference Settings

Ranked precursor predictors. Panel A retains the ranked-output evaluation of He et al., CrabNet, Retrieval-Retro, and the Qwen3.5-9B adaptation of MSP-LLM. Their ordered candidate lists are evaluated separately from independent samples in Panels B and C. CrabNet and Retrieval-Retro use the full-test denominator described above, including targets outside their precursor vocabulary.

Policy checkpoints and run aggregation. Tables 4 and 6 specify the training configurations. The RL row uses the step-200 checkpoint evaluated with eight runs, each retaining 32 logical shards and their target order. We aggregate first-pass predictions only, using the first k runs rather than the combinatorial estimator averaged over all subsets. The resulting correct-target counts at $k = 1, 2, 4, 8$ are 1,938, 1,996, 2,039, and 2,079 out of 2,654.

MSP-LLM. The ranked Qwen3.5-9B adaptation of MSP-LLM uses beam width 10; candidates retain their original order, with unparseable candidates omitted as in the original evaluator and no precursor-set deduplication. For the supplementary sampled evaluation, eight independent samples per target use seeds 42–49. MSP-LLM sampling uses temperature 1.2, top-p 1.0, and no top-k truncation, selected by validation Pass@4. Base and SFT agents share a Qwen3.5-9B backbone and evaluation protocol, with the SFT checkpoint selected on validation data.

Direct-generation backbone. The untrained direct-generation baseline, labeled Qwen3.5-9B (Thinking) in Table 2, uses Qwen3.5-9B (Qwen Team, 2026a) with a prompt requesting a material group followed by a precursor set. Reasoning is enabled, but each target receives one generation with no tools or recovery, capped at 5,120 output tokens. Sampling uses temperature 1.0, top-p 0.95, top-k 20, and seeds 42–49. The parser considers only text after the closing thinking tag and extracts the last valid nested list containing a material group and a nonempty precursor list. Only the precursor list is scored; material-group classification is not part of exact-match accuracy. Missing or malformed final answers count as failures. The per-target sampling seed is $(s + 1009i) \bmod 2^{32}$ for run seed s and zero-based target index i. The multi-turn base agent has a different generation-budget structure, so their comparison is not compute-matched.

Answer-only SFT. This baseline fine-tunes Qwen3.5-9B Base on all 24,034 training records using final precursor lists as supervision, without reasoning traces or tool interactions. Each record uses its most frequent reference route, breaking ties by source order; evaluation accepts any reference route. Full-parameter training runs for five epochs, and the epoch-five checkpoint (update 7,515) is selected by greedy exact-match accuracy on the 1,910 validation targets. Sampling parameters are selected by validation Pass@4 among three configurations: $(T, p, k) = (0.4, 0.95, 20)$, $(1.0, 0.95, 20)$, and $(1.2, 1.0, \text{all})$. The selected configuration uses temperature 1.2, top-p 1.0, and no top-k truncation. Test evaluation disables thinking and tools, allows 256 output tokens, and uses eight runs with seeds 42–49. All 21,232 outputs are parseable and none are truncated. This baseline compares answer supervision with trajectory supervision; the training data and generation budgets differ from those of the agent.

Tool-augmented frontier models. Panel C evaluates Qwen 3.8 Max, GLM 5.3, Kimi K3, DeepSeek V4 Flash, Gemini 3.5 Flash, GPT 5.6 Sol, and Opus 5 on the same fixed subset of 512 randomly sampled test targets, chosen to limit API evaluation costs. It uses a smolagents (Roucher et al., 2025) ToolCallingAgent with the same system prompt and four tools as PreWeave: retrieval, conditional precursor completion, valence analysis, and element-coverage checks. The API harness allows eight chemistry-tool calls, ten agent steps, and an initial output limit of 32,768 tokens per response. Qwen, GLM, Kimi, and DeepSeek each have eight complete runs (labels 42–49), yielding 4,096 samples per model. Their main-table results use original first-pass trajectories and exclude

subsequent length recovery. Kimi run labels distinguish independent requests rather than provider-enforced seeds. Gemini, GPT, and Opus each have two complete runs, yielding 1,024 samples per model and Pass@1/2 results. The same-subset RL reference uses the step-200 checkpoint and is reaggregated from the eight full-test runs, restricted to the same 512 targets by their original indices. First-pass prefix counts are 376, 389, 400, and 410 for $k = 1, 2, 4, 8$. The RL run seeds follow $42 + 1000003j$ for $j = 0, \ldots, 7$; aggregation preserves this fixed order and uses outputs from its native chemistry-v2 harness. All main comparisons exclude length recovery. The system prompt and available tools are shared; the API and local-agent generation budgets are specified separately above and below. API and answer-format failures remain in the denominator.

MSP-LLM decoding sensitivity. Table 7 supplements the ranked results in Panel A with independent sampling from the same MSP-LLM checkpoint. Selecting decoding by validation Pass@1 gives temperature 0.4, top-p 0.95, and top-k 20; selecting by validation Pass@4 gives temperature 1.2, top-p 1.0, and no top-k truncation. Each sampled row uses a single configuration for all four budgets. The latter attains 76.98% Pass@4 and 78.75% Pass@8, slightly exceeding the corresponding PreWeave scores, whereas PreWeave (SFT + RL) achieves higher Pass@1 and Pass@2. Beam-search Hit@k and sampled Pass@k assess different candidate-generation procedures.

Table 7: MSP-LLM decoding comparison on 2,654 test targets. Beam search reports Hit@k; independent sampling reports empirical Pass@k. Sampling settings are selected on validation data.

Decoding / selection criterion	$k = 1$	$k = 2$	$k = 4$	$k = 8$
Beam search (width 10)	73.89	78.03	82.03	85.34
Sampling / validation Pass@1	72.38	74.04	74.64	75.32
Sampling / validation Pass@4	70.72	74.76	76.98	78.75

Local tool-enabled agents. For the base, SFT, and RL agents, we keep decoding fixed at temperature 1.0, top-p 0.95, and top-k 20 to compare training stages under a common sampling configuration. We do not conduct a separate decoding sweep for each stage because evaluating multi-turn reasoning and tool interaction is substantially more expensive than answer-only generation.

Base and SFT agents use a per-generation limit of 5,120 tokens, a soft context threshold of 16,384, and a hard context limit of 32,768. Each first-pass run allows at most eight tool requests and ten tool rounds, with 4,096 tokens reserved for finalization. Upon exhausting the call budget or reaching a generation-length limit, the runtime switches to answer-only mode, removes the tool schemas, and requests a final precursor list; a format retry may follow if necessary. This finalization is part of the initial agent trajectory and is distinct from the fresh-trajectory recovery evaluated below. Maintable scores use first-pass outputs without that conditional recovery. API and answer-format failures remain in the denominator. Provider-default sampling does not guarantee seed reproducibility.

C More Analysis

C.1 Conditional Length Recovery

Setting. Conditional length recovery tests whether a larger generation allowance mitigates truncation of long agent trajectories. Selection is label-blind: a fresh trajectory is run if an ordinary generation reaches its token cap, or if the soft context reserve triggers finalization without a simultaneous tool-budget trigger. The second trajectory increases the per-generation limit from 5,120 to 6,144 tokens and the soft context threshold from 16,384 to 18,432, retaining the 32,768-token hard context limit and the same per-trajectory tool budget. Its answer unconditionally replaces the initial answer, including when replacement changes a correct answer to an incorrect one. Recovery is therefore neither an oracle selection nor a union of two answers. For recovered targets, total compute includes both trajectories; the standard eight-call configuration can consume up to sixteen tool requests across them. Each outer seed still contributes exactly one final answer to empirical Pass@k.

Results. Table 8 shows that recovery raises SFT Pass@1 by 0.87 percentage points on the full test set and 0.39 points on the 512-target subset. The gain is smaller for the base agent (0.04 percentage

Table 8: Representative results with and without conditional length recovery (%). Empirical Pass@k uses fixed seed prefixes 42–49. All rows use the full tool environment. Recovery replaces selected answers and need not improve cumulative coverage.

Model / test set	Recovery	P@1	P@2	P@4	P@8
Base agent / 2,654	None	53.81	66.69	74.04	78.22
Base agent / 2,654	Enabled	53.84	66.65	74.08	78.26
PreWeave SFT / 2,654	None	71.89	74.94	76.90	78.26
PreWeave SFT / 2,654	Enabled	72.76	74.91	76.79	78.26
PreWeave SFT / 512	None	74.22	76.56	77.73	78.52
PreWeave SFT / 512	Enabled	74.61	76.17	77.34	77.93

points). Higher-k coverage does not consistently improve: full-test SFT Pass@8 is unchanged, and subset Pass@8 decreases by 0.59 points. These fixed-prefix results characterize the complete recovery procedure, including its answer replacements. The main paper uses first-pass outputs throughout to keep recovery separate from comparisons of tool access, call budgets, and control strategies.

C.2 Model-Selected versus Fixed Tool Sequences

Motivation. Access to the same tools does not determine when or in what order they are used. Table 9 diagnoses the behavior of the trained policy under two runtimes that attempt to enforce fixed tool sequences. These runtimes also change batching and correction rules, so this comparison does not isolate tool ordering.

Table 9: Control-protocol diagnostics at $B = 4$ (2,654 targets; one seed). Complete denotes completion of the assigned four-tool sequence.

Control protocol	P@1	Complete (%)
Model-selected	71.55	—
Assigned R→M→V→C	71.40	30.0
Assigned R→V→M→C	70.84	47.9

Setting. We compare model-selected interaction with the two assigned orders in Table 9, where R/M/V/C denote Retrieval/MPC/Valence analysis/Element coverage. All protocols use checkpoint-1313, all 2,654 targets, $B = 4$, one seed, and matched decoding. The constrained runtime exposes the next prescribed tool, accepts at most one call per generation, and requests retries after sequence violations. Generation limits may force a final answer before all four tools are executed. We report full-test Pass@1, retaining unsuccessful and incomplete trajectories in the denominator, and sequence completion rates.

Result. Model-selected interaction achieves 71.55% Pass@1, compared with 71.40% and 70.84% under the constrained runtimes. Only 797/2,654 (30.0%) and 1,270/2,654 (47.9%) trajectories complete their assigned sequence. Accuracy therefore characterizes the complete runtime, including correction attempts and forced finalization, rather than successful execution of a fixed order on every target. These results do not establish an isolated benefit of adaptive ordering.

C.3 Additional Analysis of Tool Interaction

Data and aggregation. We analyze eight runs on the same 2,654 test targets for each of the base agent, SFT checkpoint-1313, and RL step-200 policies, giving 21,232 trajectories per stage. We use the original first-pass records, exclude recovery trajectories and duplicate intermediate files, and verify target alignment across stages. Request counts include rejected requests. Reported behavior frequencies use all trajectories as the denominator, and patterns may overlap. For request counts in Figure 4, we take each target's minimum and maximum over its eight runs, then average each extreme across targets to obtain the whisker endpoints. Bar heights average all target–run counts.

These whiskers summarize within-target variation across runs, not confidence intervals. For behavior frequencies in Figure 5, whiskers retain the minimum and maximum of the eight run-level percentages.

Outcome consistency across runs. For each target, we count reference-matching answers among eight runs. The numbers matched zero, one to seven, and eight times are respectively 578/1,705/371 for the base agent, 577/394/1,683 after SFT, and 575/333/1,746 after RL (Figure 4b). Eight-of-eight matching describes observed consistency and differs from Pass@8, which requires at least one match. Of the targets matched in only some base-agent runs, 1,306 are matched in every SFT run. Similar aggregate coverage does not imply identical covered targets: SFT gains 98 and loses 97 relative to the base agent; RL gains 52 and loses 50 relative to SFT. Comparisons align target identities; finite sampling contributes to observed changes in per-target hit counts.

Behavior definitions. *Batched requests* means that a model turn contains at least two recorded tool requests, without implying parallel backend execution. *Multiple MPC contexts* means at least two distinct partial precursor sets are submitted to MPC; these may be different subsets of the same route. *Multiple checked sets* means at least two distinct sets are submitted to the element-coverage tool. A *cross-round set switch* occurs when a later model turn submits a set different from one checked earlier. A *same-set recheck* occurs when a later turn checks a previously checked set; external-source or removal declarations may differ. Set comparisons use composition normalization and ignore precursor order while retaining duplicate entries. Unparseable sets are excluded from set comparisons, but their trajectories remain in the denominator. These indicators describe observable requests rather than assigning correctness or attributing changes causally to feedback.

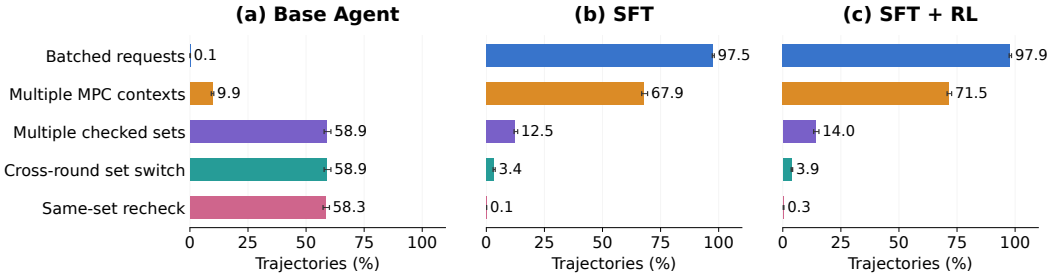


Figure 5: Interaction patterns across training stages. Bars show percentages of trajectories exhibiting each pattern. Checked sets are inputs to the element-coverage tool; cross-round switches and rechecks require a later model turn. Patterns may overlap. Whiskers span the minimum and maximum of the eight run-level percentages.

Behavior patterns. The agentic training of PreWeave substantially changes how the agent organizes tool interaction (Figure 5). The base agent frequently checks different precursor sets across turns and revisits previously checked sets. After SFT, multi-request turns become prevalent, while the fraction querying MPC with multiple partial sets rises from 9.9% to 67.9%. Cross-round changes between element-coverage check inputs become much less common. RL largely preserves this organization, with multiple-input MPC querying increasing to 71.5%. Multiple sets checked within one turn represent concurrent requests rather than revision in response to that turn's feedback.

Retrieval consistency. Retrieval appears in 20,891/21,232 SFT trajectories (98.39%) and 21,192/21,232 RL trajectories (99.81%). Its per-run frequency ranges from 98.15% to 98.68% for SFT and from 99.70% to 99.89% for RL. Across targets, 2,423/2,654 use retrieval in all eight SFT runs, compared with 2,618/2,654 under RL. The change reflects more consistent retrieval use, without establishing that it causes the accuracy difference.

Transition aggregation. For consecutive tool-bearing turns containing tool-type sets A and B, we assign weight $1/(|A||B|)$ to each pair in $A \times B$. From the last tool-bearing turn, each type receives terminal weight $1/|A|$ to End. We aggregate weights across trajectories and normalize each

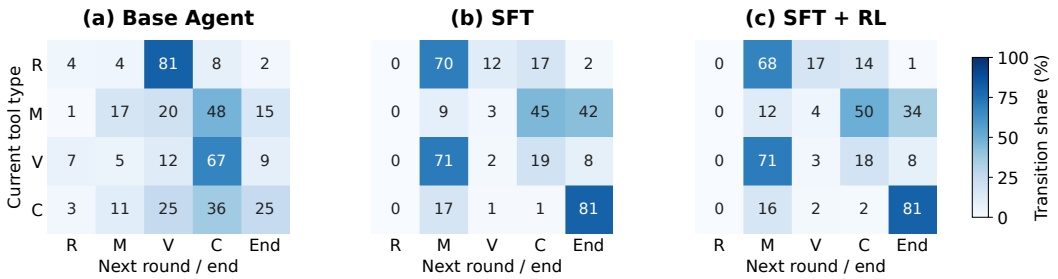


Figure 6: Transitions between tool-bearing model turns. R, M, V, and C denote Retrieval, MPC, Valence analysis, and Element coverage. Transition weights are split across source–destination tool-type pairs and normalized by source row. End denotes no subsequent tool request; same-turn calls are treated as an unordered batch.

source row. Intervening tool-free turns are omitted, and End does not imply a correct or successfully formatted answer.

Transition patterns. Figure 6 reveals how evidence-gathering and checking operations are arranged. For the base agent, retrieval is commonly followed by valence analysis and then element-coverage checking, with further checking often continuing afterward. In the SFT and RL policies, transitions from retrieval or valence analysis concentrate more strongly on MPC, and element-coverage checks are more closely associated with the end of tool interaction. This agrees with the frequent recorded sequence of joint retrieval and valence requests, followed by MPC and element-coverage checking. The similarity between SFT and RL suggests that the reported RL checkpoint preserves much of the interaction structure established during SFT.

C.4 Additional Case Studies

We present three selected trajectories from the SFT+RL policy after 200 updates. Reasoning boxes reproduce verbatim excerpts; bracketed ellipses mark omitted text and intermediate exchanges. Round labels are editorial guides, and tool boxes separately summarize actual returned observations. These examples illustrate individual decision processes rather than their prevalence. Reference matching concerns precursor-set agreement, not experimental synthesis validation.

C.4.1 Revisiting an Incorrect Route Assumption

Target. $ZrNiPb_{0.975}Sn_{0.005}Bi_{0.02}$

This trajectory expands the example discussed in Section 5.4: an element-coverage warning prompts the agent to reconsider its initial oxide interpretation.

Selected reasoning trace (verbatim excerpts)

Round 0 — Initial assumption

```
The target is an oxide, oxygen-containing.  Precursor set likely
oxides/carbonates:  ZrO2, NiO, PbO, SnO2, Bi2O3.
[...]
```

Round 1 — Response to element-coverage feedback

```
Wait target elements are ["Bi","Ni","Pb","Sn","Zr"].  O is not in
target elements!
[...]
```

Round 1 — Route revision

```
The formula is a composition, not an oxide.  So we need elemental
precursors.
[...]
```

Round 2 — Recheck
```
Conservation accepted for elemental set.
[...]
```
Round 3 — Assess MPC suggestions
```
MPC suggests Bi2(CO3)3, Bi(NO3)3, Bi, Bi2O3.  For oxygen-free target,
elemental Bi is preferred.  The oxide/carbonate/nitrate options add
O/C/N.
```

Selected tool observations (condensed)

Round	Tool	Returned evidence
0	Element coverage	Oxide set: unresolved extra element O; check not accepted.
1	Element coverage	Zr, Ni, Pb, Sn, Bi: no missing or extra elements; accepted.
2	MPC	Conditioned on Zr, Ni, Pb, Sn: Bi2(CO3)3, Bi(NO3)3, Bi, Bi2O3.

Final answer
```
["Zr", "Ni", "Pb", "Sn", "Bi"]
```

Interpretation. The feedback exposes a mistaken reading of the target and is followed by an explicit change from oxides to elements. MPC is queried after this revision and supports assessment of the Bi source; it does not initiate the route change.

C.4.2 Recovering an omitted dopant source

Target. $Sr_{4.995}Eu_{0.005}P_3O_{12}F$

The initial set covers the host composition but omits Eu. The subsequent exchange makes the missing dopant source explicit and records its addition.

Selected reasoning trace (verbatim excerpts)

Round 1 — Use retrieved precedent
```
The retrieval shows Sr/P/O with SrCO3 + PH6NO4 (NH4H2PO4) precedent.
Good.
[...]
```
Round 2 — Choose a host-level set
```
MPC confirms PH6NO4 = NH4H2PO4 as top P source.  Alternatives:  P2O5.
[...]
```
Round 2 — Submit element-coverage check
```
Now element conservation check.  Precursors:  SrCO3, SrF2, NH4H2PO4.
[...]
```
Round 3 — Respond and revise
```
The Eu is missing.  Need a Eu source:  Eu2O3 or EuF3 or Eu2O3 with a
small amount.
[...]
```
Round 3 — Request another check
```
Now the check needs to be redone with Eu2O3.  Let me run the new
conservation check with the full set ["SrCO3", "SrF2", "NH4H2PO4",
"Eu2O3"].
[...]
```
Round 4 — Finalize
```
Conservation accepted.  Route:  SrCO3 + SrF2 + NH4H2PO4 + Eu2O3.
```

Selected tool observations (condensed)

Round	Tool	Returned evidence
1	MPC	PH6NO4 ranked first; remaining missing target element: Eu.
2	Element coverage	SrCO3, SrF2, NH4H2PO4: missing Eu; check not accepted.
3	Element coverage	After adding Eu2O3: no missing target elements; accepted.

Final answer

```
["SrCO3", "SrF2", "NH4H2PO4", "Eu2O3"]
```

Interpretation. MPC already flags the missing Eu source, but the agent acts on it after the element-coverage check. The example shows a local correction enabled by explicit element accounting, complementing the small aggregate effect of removing Element coverage in the ablation.

C.4.3 Comparing alternative Ce sources

Target. $Ca_{0.99}Ce_{0.01}Al_{1.01}Si_{0.99}N_3$

The agent proposes CeN before querying MPC, then compares it with Ce and CeO2 using returned observations and its own compositional calculation.

Selected reasoning trace (verbatim excerpts)

Round 1 — Compare two routes

```
Candidate route A: Ca3N2, AlN, Si3N4, CeO2.  Candidate route B: Ca3N2,
AlN, Si3N4, CeN.
[...]
```

Round 2 — Interpret valence output

```
Valence:  CeO2 has Ce4+ while target Ce3+.  That suggests redox
needed.  CeN has Ce3+, matches.
[...]
```

Round 3 — Consider the candidate list

```
MPC suggests Ce metal (rank 1) and CeO2 (rank 2).  CeN not in top 5?
Interesting.  But CeN is plausible.
[...]
```

Round 4 — Read element-coverage results

```
CeN route is accepted, CeO2 route is not accepted due to O extra.
[...]
```

Round 4 — Check the composition

```
0.33 Ca3N2 + 1.01 AlN + 0.33 Si3N4 + 0.01 CeN ->
Ca0.99Ce0.01Al1.01Si0.99N3.
```

Selected tool observations (condensed)

Round	Tool	Returned evidence
1	Valence	Under charge assumptions: target Ce(III), CeO2 Ce(IV), CeN Ce(III).
2	MPC	Ce ranked first, CeO2 second; CeN absent from the five suggestions.
3	Element coverage	CeO2-containing set: unresolved O. CeN-containing set: accepted.

Final answer

```
["Ca3N2", "AlN", "Si3N4", "CeN"]
```

Interpretation. The agent retains an independently proposed option rather than adopting MPC ranking alone. The stated equation balances elemental amounts. Valence estimates are conditional, and the unresolved oxygen fate does not establish that a CeO2 route is infeasible; neither check verifies synthesis feasibility.

C.5 Online RL Dynamics and Further Directions

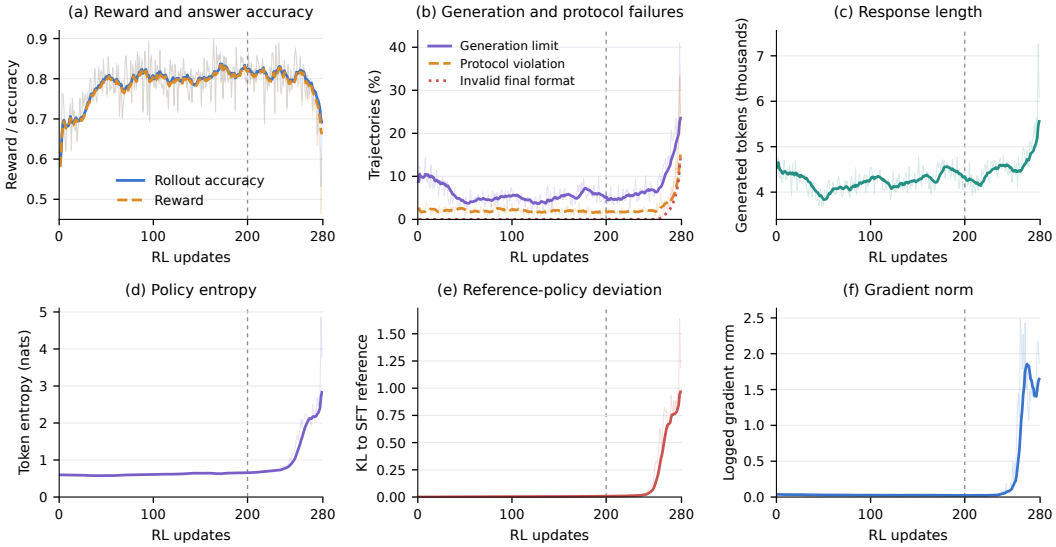


Figure 7: First-trial online RL dynamics: (a) rollout reward and accuracy; (b) overlapping protocol and generation failures; (c) generated tokens per trajectory; (d) action-token entropy; (e) clipped KL estimate relative to SFT; (f) logged gradient norm. Training curves cover updates 1–279: faint lines show raw values and foreground lines show trailing means over up to ten updates. Dashed vertical lines mark 200 updates. All metrics are measured during training updates.

Answer-accuracy gains during RL. Figure 7 summarizes the first online RL trial, which generates fresh groups of eight trajectories per target and applies trajectory-level policy optimization with KL regularization toward the initial SFT policy. Mean rollout reward increases from 0.686 over updates 1–20 to 0.812 over updates 151–200, accompanied by an increase in reference-matching accuracy from 69.02% to 81.54%. Protocol violations change only modestly, from 2.10% to 1.79%, and mean response length remains approximately 4.4k tokens. The reward gain primarily reflects more accurate recommendations, with little contribution from reduced penalties or longer responses; these statistics concern changing training batches.

Reward saturation and policy drift. By approximately 200 updates, the main training-reward gains have largely saturated: mean reward remains near 0.812 over the following 40 updates. Meanwhile, entropy and divergence from the SFT reference continue to increase and subsequently accelerate. Between updates 200 and 260, logged entropy rises from 0.644 to 2.085, reference KL from 0.0079 to 0.5726, and gradient norm from 0.024 to 2.212. Continued optimization thus produces increasing policy drift with little additional reward benefit. We selected the step-200 checkpoint based on the training-reward plateau and the subsequent increase in policy drift, retaining a comparatively stable checkpoint before the pronounced late-stage instability.

Generation stability and future directions. In the final nine recorded updates, mean response length reaches 5.6k tokens, generation-limit triggers affect 24.09% of trajectories, and invalid final formats reach 15.17%. Tool-parameter validation errors do not increase, suggesting that generation and finalization become the more prominent operational bottlenecks. Building on POLARIS (An et al., 2025), future work could recalibrate training-task difficulty as the policy evolves and adjust rollout temperature across stages using both diversity and accuracy measurements. These directions could help sustain informative training signals and controlled exploration; their effects on precursor recommendation and training stability remain to be evaluated.