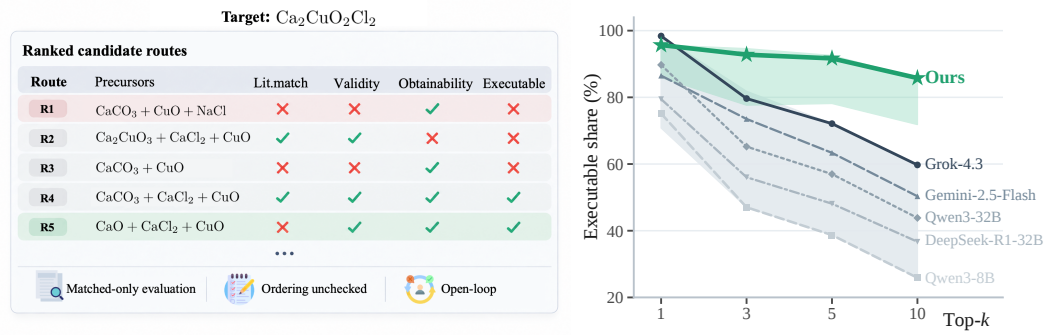
CAN AN INORGANIC SYNTHESIS ROUTE BE RUN? BENCHMARKING LABORATORY EXECUTABILITY AND IMPROVING IT FROM PRACTICE FEEDBACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Synthesis planning for inorganic materials is evaluated by whether a generated route matches a recorded literature recipe. That criterion rewards reproducing knowledge that has already been published, and it leaves three things undetermined: whether a route can actually be run, whether the candidates are ordered by how reliable they are, and whether a model can revise its output once a route fails in the laboratory. We introduce SynBench, an executability benchmark that scores a route by whether its chemistry balances and whether every precursor can be commercially obtained, verified one by one against a real reagent platform. On 185 target materials, we evaluate sixteen LLM configurations and find that executability collapses as more candidates are requested: the executable share of the returned routes falls by 37% to 66% between the first candidate and the tenth, and the first route that cannot be run appears among the first five. We then present ExecuSyn, which uses SynBench as a training environment and, by having an LLM judge simulate a chemist's feedback, closes the loop between the planner and the chemist so that it keeps refining its decisions during an experiment. Against the strongest evaluated model, ExecuSyn raises the executable share of the top ten predictions by 44% and moves the first unexecutable route 86% later, while the rubric score rises throughout training.



(a) Limitations of existing synthesis planning

(b) Overall leaderboard evaluation on SynBench

Figure 1: Overview of **(a)** the limitations of existing synthesis planning methods and **(b)** overall leaderboard evaluation on SynBench. **(a)** Candidates are scored only on whether they match a literature recipe (Lit. match), their order is not examined, and the set is returned in one shot. The remaining columns are what SynBench scores instead: chemical Validity, Obtainability, and Executable. **(b)** The four ExecuSyn backbones (green) sustain their executable share as more candidates are requested, where the 16 baselines (grey) do not.

1 INTRODUCTION

Computational screening and generative models, run against large open databases of computed materials properties (Jain et al., 2013; Kirklin et al., 2015; Curtarolo et al., 2012), now propose candidate materials far faster than laboratories can make them. Autonomous laboratories have begun to close part of this gap by executing literature-derived recipes end to end (Szymanski et al., 2023),

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.

but the recipe such a laboratory receives is exactly what it has to be able to run. The synthesis plan is what stands between a promising prediction and a real sample, and for many inorganic materials it is a solid-state route: a set of starting materials that react, on heating, to give the target composition. Synthesis planning is therefore posed as precursor recommendation. Given a target formula, a synthesis planning method returns the precursor combinations most likely to yield it.

Existing methods are trained on literature recipes and evaluated by top-*k* exact match against them (Noh et al., 2024; Prein et al., 2025a; Noh et al., 2026). This objective rewards the reproduction of synthesis knowledge that has already been reported, and it does not train or evaluate the ability to produce routes that can be executed in a laboratory. Yet the purpose of synthesis planning is to guide real synthesis experiments. A route that looks reasonable on paper is useful to a chemist only if it can actually be run. This gap between what is optimized and what is needed leaves three limitations. (i) Evaluation is granted only to candidates that match a literature recipe, so whether an unmatched candidate can be executed remains undetermined. (ii) The ordering of candidates is not examined, whereas a more reliable route should rank earlier, which matters greatly to a chemist consulting the predictions and carrying out experiments efficiently. (iii) Planning is open-loop: a fixed set of candidate routes is returned for a target. Real synthesis rarely succeeds on the first attempt, and a chemist runs a route and then adjusts the starting materials according to what failed. The loop between the planner and the chemist is open: the planner can neither receive that feedback nor refine its decisions from it. Together, these three limitations restrict how far synthesis planning can guide experiments (Figure 1).

To fill this gap, we build SynBench, an executability benchmark for inorganic synthesis planning, and use it to evaluate leading LLMs. SynBench makes executability a measurable and comparable target, moving evaluation from whether a literature recipe has been reproduced to whether a route can actually be executed. On this basis we further propose ExecuSyn, which produces candidate routes that are executable and can also be improved from feedback arising in experimental practice.

Contributions.

- **SynBench: an executability benchmark for inorganic synthesis planning.** SynBench evaluates whether the candidate routes a model returns can actually enter a laboratory, and whether those routes are ordered sensibly by reliability. It comprises 185 target materials and pairs offline and online precursor-obtainability retrieval, verifying the starting materials each route requires one by one against a real reagent platform.
- **We show that executability collapses in current LLMs as more candidates are requested.** Across sixteen LLM configurations, the executable share of the returned routes falls by 37% to 66% between the first candidate and the tenth (Exec@K). The ordering is no more dependable: averaged over targets, the first route that cannot be run arrives between the third and the fifth position (FirstFail@10), so unexecutable routes are already interleaved among the leading candidates.
- **ExecuSyn: executable synthesis planning with practice feedback.** We use SynBench as the environment in which a model is optimized directly for executability. Against the strongest evaluated model, the executable share of the top ten predictions rises by 44% (Exec@10), and the position at which the first unexecutable route appears moves 86% later (FirstFail@10). We further use an LLM judge to stand in for the chemist and return the feedback a real one would give, so that the planner keeps refining its decisions from that feedback rather than converging after one update to a fixed set of routes.

2 SynBench: From Literature Match to Laboratory Execution

2.1 Task Formulation

Synthesis planning for inorganic materials takes a target material and predicts a set of precursors that can be used to prepare it. We define a **route** R as a set of precursor formulas, $R = \{p_1, p_2, \ldots, p_m\}$, where each p_i is the formula of one starting material the route uses. A model takes a target formula t as input and returns K candidate routes, $f(t) = (R_1, R_2, \ldots, R_K)$. Let L_t denote the routes recorded for t in the literature. Existing methods are supervised on literature recipes and judged by whether a candidate route matches a recorded one, $\mathrm{Match}(R, L_t) = \mathbf{1}[R \in L_t]$. This criterion

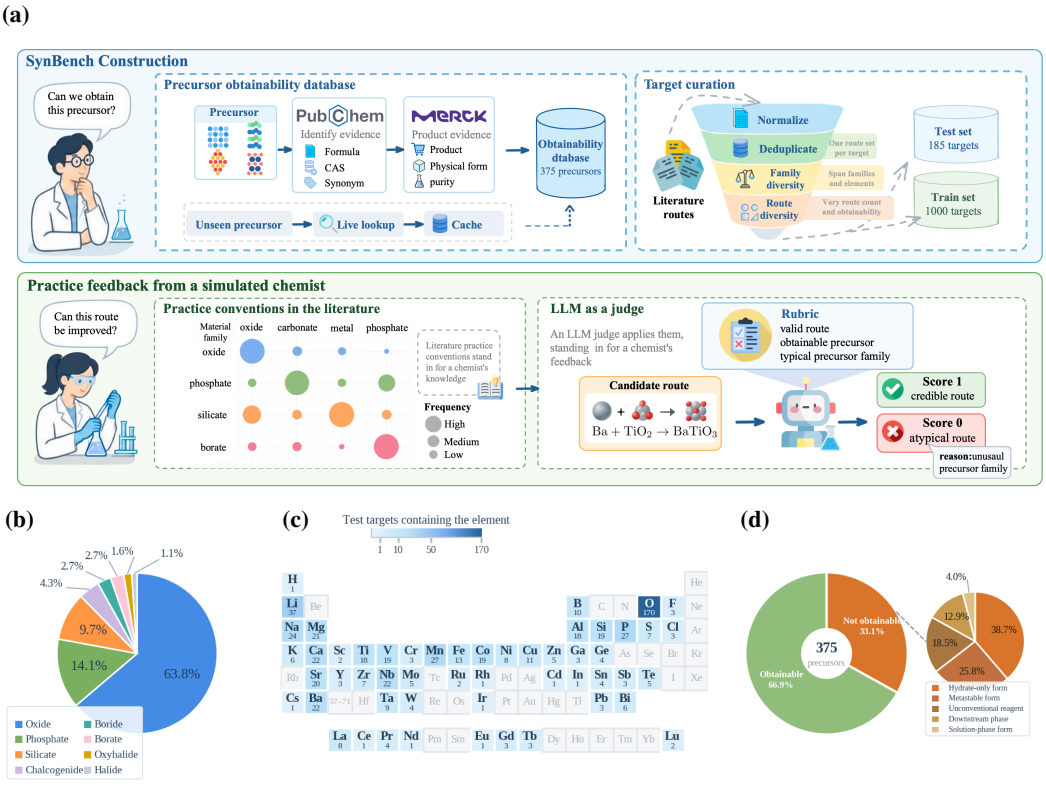


Figure 2: **Construction of SynBench and the composition of the resulting benchmark.** **(a)** The pipeline turning literature routes into the precursor-obtainability database and the curated target splits (top), and the literature-derived rubric applied by an LLM judge as practice feedback (bottom). **(b)** Material-family distribution of the 185-target test split. **(c)** Its elemental coverage. **(d)** Obtainability composition of the 375-precursor database.

measures how well a model reproduces synthesis knowledge that has already been reported, and does not reflect whether a route can actually be carried out in a laboratory. We therefore redefine the objective. A route is **lab-executable** if and only if it meets both of the following conditions.

Chemical validity. Every element in the target formula must have a source among the precursors, and the precursors must not introduce an element that the target does not contain. We write $v(R) \in \{0, 1\}$ for whether R is chemically valid; the implementation is given in Appendix C.1.

Precursor obtainability. A chemist must be able to gather every starting material a route calls for, and a single unobtainable precursor makes the route impossible to run. We use the reagent platform Sigma-Aldrich to determine whether each precursor of a route can be obtained in the chemical form the route declares; the platform and the retrieval procedure are described in Appendix C.3. Let $o(R) = o_R/m_R$ be the fraction of obtainable precursors, where m_R is the number of precursors in the route and o_R the number of them that are obtainable.

A route is thus **lab-executable** if and only if it is chemically valid and all of its precursors are obtainable, $\mathrm{Exec}(R) = v(R) \cdot \mathbf{1}[o(R) = 1]$. Synthesis planning methods generally return a set of candidate routes, so we evaluate a model on the whole output $f(t)$ rather than on any single route within it. The corresponding metrics are given in Section 2.3.

2.2 Evidence-Grounded Benchmark Construction

SynBench is built on the text-mined dataset of inorganic synthesis recipes assembled by Kononova et al. (2019), which records the target material and the precursor combinations of each literature route. From it we build a precursor database, and a target database split into 185 materials held out for evaluation and 1,000 used for training, with no formula in both. Figure 2a gives an overview of the pipeline.

Precursor obtainability database. A route can only be carried out if its precursors can be purchased, so we build a database in which every precursor a model names is judged against commercial evidence. Formulas extracted from the literature routes are first checked against PubChem (Kim et al., 2025) for molecular formula and common synonyms, which catches parsing errors introduced during extraction; a PubChem record establishes that a substance is known, not that it can be bought (Appendix C.2). The verified precursors are then looked up on the reagent platform Sigma-Aldrich, falling back to the CAS Registry Number and to common synonyms when a formula returns nothing. A listing counts as evidence only if it meets the specification of a solid-state precursor, such as a solid powder form and a stated purity (Appendix C.3). Precursors absent from the offline database are resolved by an online lookup under the same criteria (Appendix C.4).

Target curation. Targets are normalized, deduplicated by formula, and then sampled to diversify both material family and route landscape (Appendix C.6). The test split holds 185 targets spanning eight material families (Figure 2b, Appendix C.7) and 53 elements across the main group, the transition series, and the lanthanides (Figure 2c), covering families whose chemistry differs markedly, such as borides and oxyhalides. Its targets carry between 1 and 14 recorded routes, and some of them carry routes that differ among themselves in obtainability, so that the split can test whether a method finds the executable route rather than merely a recorded one. The training split holds 1,000 targets, spanning 11 material families and 73 elements.

Data statistics. The precursor database holds 375 precursors, 66.9% of which are labelled obtainable (Figure 2d). None of the remainder is a substance that fails to exist; each runs into a particular obstacle to procurement (Appendix C.5). Most of these come from how recipes are written. A route names the species that reacts, so it writes an anhydrous nitrate such as $Fe(NO_3)_3$; the anhydrous compound is hygroscopic, and what suppliers stock is the hydrate. Others, such as Ca_2CuO_3, are downstream phases a chemist prepares in the lab rather than buys, so few commercial listings exist for them. Still others are reagents sold less widely, or available only in solution rather than as the solid a solid-state route needs. Ease of purchase therefore varies across what the literature records, and a route a chemist can act on is the more useful one to propose.

2.3 Evaluation Protocol

To assess whether the candidate routes a synthesis planning method returns can actually be carried out in a laboratory, we evaluate a model on its whole set of candidate routes and use two primary metrics: **Exec@K**, the fraction of those routes that are lab-executable, and **FirstFail@K**, the position at which the first non-executable route appears. All metrics are averaged over the targets in the test set.

Executable share (Exec@K). For a target t, let the fraction of the top-K candidate routes that are lab-executable be

$$\mathrm{Exec@}K(t) = \frac{1}{K}\sum_{i=1}^{K}\mathrm{Exec}(R_i), \tag{1}$$

with $\mathrm{Exec}(R_i)$ as defined in Section 2.1. Routes are deduplicated before scoring: where the same precursor combination occurs more than once, only its first occurrence is kept, and the rest count as failures and occupy the ranks they were given, so that a repeated route cannot raise the metric. When fewer than K routes are returned, the missing ranks count as failures as well.

Rank of first failure (FirstFail@K). This metric records the position of the first non-executable route,

$$\mathrm{FirstFail@}K(t) = \min\left(\{i \mid \mathrm{Exec}(R_i) = 0\} \cup \{K+1\}\right), \tag{2}$$

and is $K+1$ when all K routes are executable. A larger value means the model sustains executable routes further down the ranking. Exec@K measures how many routes in a candidate set can be run, while FirstFail@K measures how far down the set remains usable; together they describe how usable a candidate set is in practice.

Supporting metrics. We also report two supporting quantities. **AllExec@K** is the share of targets whose top-K routes are all executable, which indicates whether a candidate set can be trusted as a whole. **MeanObtain@K** is the mean precursor obtainability ratio over the top-K routes, which shows how far a route that does not satisfy obtainability outright falls short of it.

Table 1: **Executability of LLMs on SynBench, on the 185 test targets.** Higher is better throughout. Bold marks the best result within each access group and underline the second best; *(thinking)* denotes a model's long-reasoning mode.

	Exec@K				AllExec	MeanObtain	**FirstFail**
Model	@1	@3	@5	@10	@10	@10	@10
Proprietary models							
Claude-3.5-Sonnet	0.951	0.784	**0.721**	0.595	**0.124**	0.784	**4.89**
DeepSeek-V3	**0.984**	**0.818**	0.703	0.560	0.114	0.775	4.52
Gemini-2.5-Flash	0.865	0.735	0.634	0.503	0.065	0.710	4.15
Gemini-2.5-Flash-Lite	0.892	0.714	0.629	0.489	0.027	0.730	3.59
GPT-4.1-Nano	0.854	0.672	0.557	0.390	0.016	0.509	3.31
GPT-4o	**0.984**	0.778	0.672	0.521	0.103	0.731	4.42
GPT-4o-mini	0.838	0.625	0.497	0.367	0.005	0.596	2.89
Qwen-Flash	0.946	0.735	0.629	0.498	0.059	0.714	4.04
Qwen3-Max	0.951	0.667	0.588	0.434	0.027	0.641	3.43
Grok-4.3	**0.984**	0.796	**0.721**	**0.597**	0.119	**0.791**	4.66
Open-source models							
DeepSeek-R1-0528-Qwen3-8B	0.724	0.557	0.452	0.348	0.011	0.501	2.67
DeepSeek-R1-Distill-Qwen-32B	0.795	0.560	0.481	0.367	**0.049**	0.579	3.23
Qwen3-32B	**0.897**	**0.652**	**0.570**	**0.438**	**0.049**	**0.651**	**3.57**
Qwen3-32B (thinking)	0.849	0.625	0.530	0.424	0.027	0.643	3.24
Qwen3-8B	0.751	0.470	0.387	0.259	0.005	0.457	2.46
Qwen3-8B (thinking)	0.708	0.474	0.394	0.286	0.000	0.473	2.50

3 Evaluating LLMs as Synthesis Planners with SynBench

3.1 Experiments Setup

Baselines. We evaluate 14 models on SynBench, covering both closed- and open-source families. The closed-source models are Grok-4.3, Claude-3.5-Sonnet, DeepSeek-V3, GPT-4o, GPT-4o-mini, GPT-4.1-Nano, Gemini-2.5-Flash, Gemini-2.5-Flash-Lite, Qwen3-Max, and Qwen-Flash. The open-source models are Qwen3-32B, Qwen3-8B, DeepSeek-R1-Distill-Qwen-32B, and DeepSeek-R1-0528-Qwen3-8B. They span a wide range of scales, from lightweight to flagship, so that the evaluation reflects how executability varies with model scale. We additionally report a long-reasoning configuration for Qwen3-32B and Qwen3-8B, to examine how the reasoning mode affects executability. Each model's provenance, and the role it plays as a point of comparison, is recorded in Appendix E.2.

Evaluation protocol. Every model receives the same prompt and returns candidate routes without explanations, ordered by its own estimate of viability (Appendix I). A single scoring implementation processes every output under matched decoding settings, so results are comparable across models; full protocol details are given in Appendix E.3.

3.2 Results

Table 1 reports the executability of all evaluated models on the 185 test targets. Three observations follow.

Executability falls sharply as more candidates are requested. At the first candidate, Exec@1 averages 0.873 across models, with five models above 0.95 and a spread of only about 0.28 between the strongest and the weakest. Returning a single plausible route is not difficult for current models. As more candidates are taken into account, however, performance diverges quickly: for Grok-4.3, the strongest closed-source model, Exec@K drops from 0.984 to 0.597, a decline of 39.3%, and for Qwen3-32B, the strongest open-source model, from 0.897 to 0.438, a decline of 51.1%. Models can still produce executable routes near the top of the ranking, but further down they increasingly rely on candidates that cannot in fact be carried out.

Non-executable routes appear early. Across all models the first failure falls between ranks 2.46 and 4.89, so a chemist working through the output in order encounters a route that cannot be run within the first five entries on average. For a model whose purpose is to guide experiments, this means that only the very front of its output is trustworthy.

Where the failures come from, and the effect of scale. Models generally do select obtainable precursors (MeanObtain@10 averages 0.643) but rarely keep every route in a set obtainable (AllExec@10 averages 0.050): a single unobtainable precursor is enough to fail a route, and current models cannot keep such precursors out of their candidates. Scale does not close this gap, nor does long reasoning help consistently (Appendix G.3).

4 ExecuSyn: Executable Planning with Practice Feedback

This section introduces ExecuSyn. Using SynBench as the training environment, we train a model to return executable candidate routes, addressing the limitations of existing LLMs revealed above (Section 4.1). To address the remaining one-shot limitation, we add a practice-feedback loop: an LLM judge scores each route against a rubric, and the model is refined by on-policy distillation to keep improving its routes as such feedback arrives in experimental practice (Section 4.2).

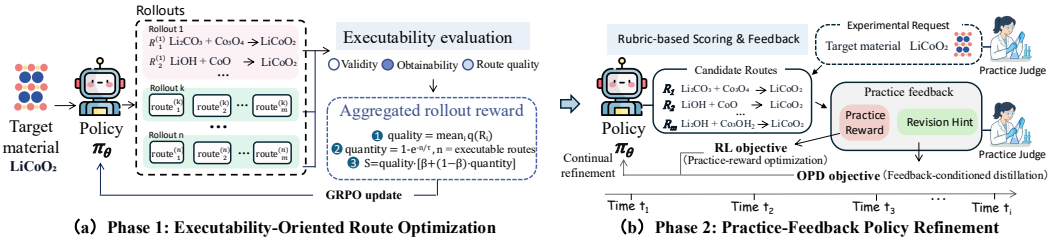


Figure 3: **Overview of ExecuSyn.** Phase 1 optimizes a policy for executability (Section 4.1); phase 2 refines it from practice feedback (Section 4.2).

4.1 Executability-Oriented Route Optimization

Executability reward. Figure 3 gives an overview of the method. For each route R_i in a rollout, its quality is $q(R_i) = v(R_i) \cdot o(R_i) \cdot \rho(R_i) \cdot \mu(R_i)$, where $v(R_i)$ and $o(R_i)$ are the chemical validity and obtainable fraction defined in Section 2.1, $\rho(R_i) \in [0, 1]$ penalizes repeated precursors within the route, and $\mu(R_i) \in [0.3, 1]$ penalizes sourcing a single element from multiple precursors for no reason, which we judge to be the case only when the precursors in question supply exactly the same set of elements. A rollout returns m routes, of which $\mathcal{U}$ indexes the distinct ones, and its quality term divides by every route emitted while summing only over the distinct ones, $\text{quality} = \frac{1}{m} \sum_{i \in \mathcal{U}} q(R_i)$. Dividing by m rather than by the number of distinct routes means a repeated route occupies a slot exactly as a failed one does, so copying an already successful route earns nothing.

Quality alone rewards a rollout for containing one good route, not for containing many. We therefore add a quantity term that rewards a rollout for returning as many fully executable routes as possible. Writing n for the number of routes in a rollout that are chemically valid and fully obtainable, the quantity term is $\text{quantity} = 1 - e^{-n/\tau}$ with $\tau > 0$. This term saturates in n: past saturation, returning still more routes brings little further gain, so the model has no incentive to pad its output with additional routes. The rollout reward is the product of the two,

$$S = \text{quality} \cdot \big[\beta + (1 - \beta) \cdot \text{quantity}\big], \qquad \beta \in [0, 1], \tag{3}$$

where β lower-bounds the quantity factor so that early rollouts with a small n still receive a usable gradient.

Training setup. We optimize a policy π_θ with GRPO (Shao et al., 2024) to maximize the expected rollout reward, $\max_\theta \mathbb{E}_{t \sim \mathcal{D},\, \{R_i\}_{i=1}^m \sim \pi_\theta(\cdot|t)}[S]$, sampling a group of rollouts per target and estimating the advantage from their relative scores within the group. The prompt and the obtainability judgement match evaluation, and the training set $\mathcal{D}$ is the 1,000 training targets of Section 2.2, disjoint from

the test targets. We train four backbones separately: Qwen3-8B, Qwen3-4B-Instruct, Llama-3.2-3B-Instruct (Grattafiori et al., 2024), and Qwen3-1.7B (Yang et al., 2025). Full hyperparameters and training configuration are given in Appendix F.3.

4.2 Practice-Feedback Route Refinement

Literature-derived rubrics and feedback. We model synthesis planning as a process that is continually revised during an experiment: the planner proposes a route, a chemist would normally report back on what failed, and the planner would revise its next attempt accordingly. We simulate the chemist's side of this exchange with an LLM judge (Lee et al., 2024), so the model can be trained on it directly. We turn the practice knowledge of an experienced experimentalist into a rubric: for each target family, we count from literature routes which precursor families it conventionally draws its starting materials from, giving a conventional set $T(f)$ (Appendix F.2). The judge applies this rubric (Appendix I) to score a candidate route and to produce feedback (Figure 2a, bottom).

A route is judged in two stages (Appendix F.2). The first is a gate $a(R) \in \{0, 1\}$ that reuses the chemical-validity and obtainability criteria of Section 2.1 to reject a route, and return a reason for it. Only routes that clear the gate reach the second stage, an LLM judge that applies the rubric rather than its own chemistry intuition and likewise returns a reason for the routes it rejects. The two stages give the practice-feedback signal $\phi(R) = a(R) \cdot j(R)$, and $h(R)$ denotes the reason returned by whichever stage rejects a route.

Feedback-conditioned updating. To let the model keep improving from feedback, we continue optimizing the policy trained with GRPO in Section 4.1. The objective has two terms: a GRPO loss driven by practice feedback, and a distillation loss that turns a rejection reason into a signal over the student's own candidate tokens.

The reward behind the GRPO loss reuses the structure of Section 4.1, with the criterion switched from executability to the practice-feedback signal $\phi(R)$ defined above. Writing $n_\phi = \sum_{i=1}^{m} \phi(R_i)$ for the number of routes in a rollout that follow practice, the reward is $S_\phi = (n_\phi/m)\big[\beta + (1 - \beta)(1 - e^{-n_\phi/\tau})\big]$.

The second term turns a rejection reason into a gradient, following the on-policy distillation approach of training a student against a teacher's feedback (Revision Hint) on the student's own samples (Agarwal et al., 2024). For a route R that the gate or the judge rules out, $\phi(R) = 0$, we append its reason $h(R)$ to the original prompt, giving the feedback-conditioned context $s \oplus h(R)$. The teacher π_T is the current policy evaluated under that context, without backpropagation. At each response position i the update is a clipped surrogate, in the style of PPO (Schulman et al., 2017), over the student's own top-k candidates $S_i = \text{top-}k\{\pi_{old}(\cdot \mid s, y_{<i})\}$, rather than over the sampled token alone. Writing $w_v = \text{softmax}_{v \in S_i}(l_{old}(v))$, $\Delta_v = \text{clip}(l_T(v) - l_{old}(v), -C, +C)$, $A_v = \Delta_v \cdot w_v$, and $\rho_v = \exp(l_{cur}(v) - l_{old}(v))$, where $l_*(v)$ denotes $\log \pi_*(v \mid \cdot)$ and A_v is computed without gradient, the loss is

$$L^{\text{OPD}} = \frac{1}{T} \sum_{i=1}^{T} \sum_{v \in S_i} \max \Big[- A_v \rho_v,\ -A_v \,\text{clip}(\rho_v,\, 1 - \epsilon_{\text{lo}},\, 1 + \epsilon_{\text{hi}}) \Big], \tag{4}$$

and the final objective combines it with the GRPO loss L^{GRPO} driven by S_ϕ,

$$L = w_{\text{RL}} \cdot L^{\text{GRPO}} + w_{\text{OPD}} \cdot L^{\text{OPD}}. \tag{5}$$

Three details make the term stable. The softmax weight w_v concentrates the update on candidates the student was already likely to emit. The clip C is necessary rather than cosmetic: without it the raw teacher-student gap takes highly extreme values and the update diverges, which we observed directly before adding it. And a rollout that clears both stages carries no reason, so the augmented context equals the original one and $\Delta_v \equiv 0$. At $k = 1$ the candidate set is the sampled token, its weight is 1, and the expression reduces to the sampled-token form. In practice, feedback accumulates one interaction at a time: the model is updated only once enough interactions with the chemist have accumulated into a mini-batch, rather than waiting on a large batch collected all at once. Full experimental settings are given in Appendix F.3.

5 EXECUSYN PERFORMANCE AND ANALYSIS

5.1 ROUTE EXECUTABILITY

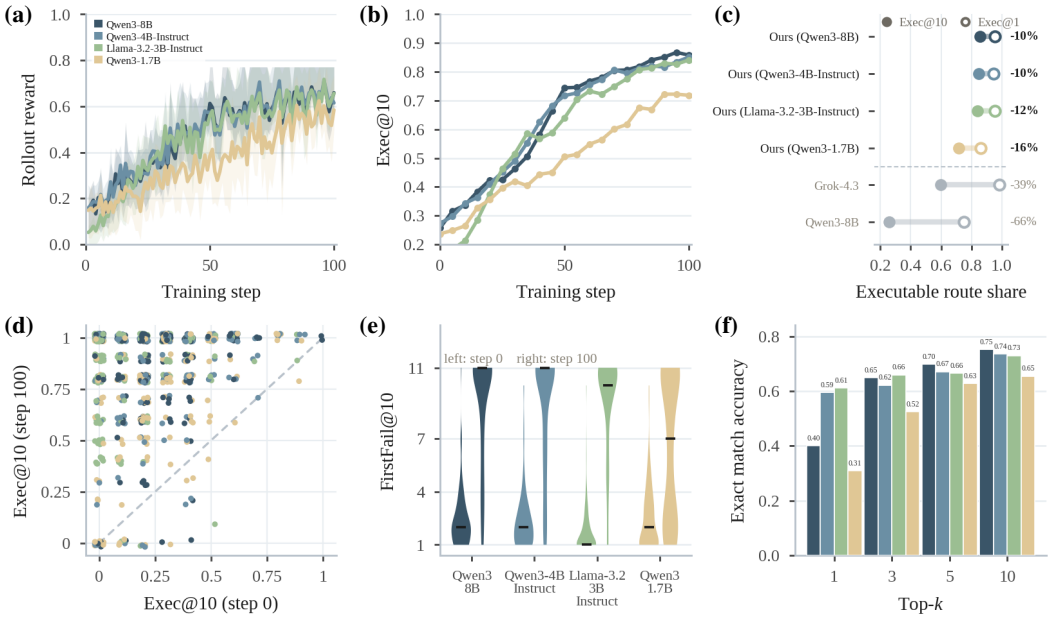


Figure 4: **Final performance of ExecuSyn.** (a) Rollout reward over training. (b) Test-set Exec@10. (c) Retention from Exec@1 to Exec@10. (d) Per-target Exec@10 before versus after training. (e) FirstFail@10 distribution. (f) Top-k exact match against literature recipes.

Executability no longer falls as more candidates are requested. On the 185 test targets, ExecuSyn reaches Exec@10 of 0.858, a relative gain of 44% over the strongest baseline (Table 2). The gain is not a late spike but the end state of training: rollout reward rises throughout, and test-set Exec@10 rises with it and then holds (Figure 4a,b). The improvement is broad-based rather than concentrated in a few targets, with most targets improving over their pre-training Exec@10 (Figure 4d). Figure 4c shows that this level is held as more candidates are requested: Exec@K falls by 10% to 16% from the first route to the tenth for the four ExecuSyn variants, against 39% for the strongest baseline. The model therefore returns executable routes reliably, and does not admit non-executable ones at a growing rate as more candidates are requested.

Model	Exec@10	FirstFail@10
Ours (Qwen3-8B)	**0.858** (+43.7%)	8.77 (+79.3%)
Ours (Qwen3-4B-Instruct)	0.851 (+42.5%)	8.57 (+75.3%)
Ours (Llama-3.2-3B-Instruct)	0.841 (+40.9%)	**9.10** (+86.1%)
Ours (Qwen3-1.7B)	0.718 (+20.3%)	6.60 (+35.0%)

Table 2: **Executability of the four ExecuSyn variants on the 185 test targets.** Gains are relative to the strongest baseline (Table 1).

Non-executable routes no longer appear early. Table 2 gives FirstFail@10, the rank of the first non-executable route: our strongest variant puts it at 9.10 against 4.89 for the strongest baseline, a relative gain of 86%. Figure 4e shows why: before training, the first non-executable route falls among the first few candidates for every model. After training it falls at or near the tenth for three of the four variants. Where existing models leave only the very front of their output trustworthy, ExecuSyn extends that trust across nearly the whole top ten.

Optimizing for executability does not cost literature knowledge. A natural question is whether training against executability gives up the synthesis knowledge recorded in the literature. Figure 4f shows that it does not: on 65% to 75% of targets, at least one of the top-ten routes ExecuSyn returns agrees with a literature recipe for that target. The model selects purchasable and executable routes on top of literature knowledge rather than meeting the executability criterion with combinations unrelated to it.

5.2 PRACTICE-FEEDBACK ADAPTATION

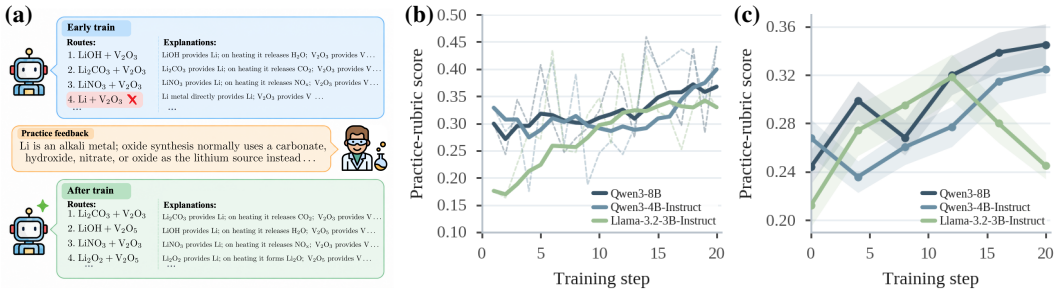


Figure 5: **Practice-feedback adaptation.** (a) One target's routes before and after training, with the practice feedback between them. (b) Practice-rubric score on the training targets. (c) Practice-rubric score on held-out targets.

The planner keeps refining its decisions from feedback. Figure 5 tracks the practice-rubric score over 20 training steps, with the measurement conventions of Appendix G.2. On the training targets, the score dips briefly at the start of training and then rises steadily through step 20 for the three backbones in Figure 5 (Figure 5b). On the held-out targets it rises by a comparable margin (Figure 5c), improving on targets the model never trained on. The one exception is Llama-3.2-3B-Instruct, whose held-out score peaks at step 12 and then falls back over the last two evaluations, though it still ends above where it began. The loop therefore keeps moving after one update: the planner refines its decisions from the judge's feedback, including on targets it never trained on, rather than memorizing the ones it happened to see.

Ablation. Table 3 switches one part of the refinement objective at a time, holding the backbone and the number of training steps fixed (Appendix F.3). Removing either term costs every backbone: k=4 exceeds GRPO-only by 0.0085 to 0.0757 and OPD-only by 0.0187 to 0.0578, so the two signals are complementary. Width beyond $k = 4$ does not help: it is the best setting on three of the four backbones. On Qwen3-8B the sampled-token form already suffices, but both widths clear every single-term ablation by at least 0.0234, so on that backbone the choice of width matters less than having both terms.

Model	k=1	**k=4**	k=8	k=16	GRPO only	OPD k4	OPD samp
Qwen3-8B	**0.3453**	0.3183	0.2771	0.3261	0.2919	0.2949	0.2927
Qwen3-4B	0.3248	**0.3498**	0.2592	0.2951	0.2741	0.2920	0.2584
Llama-3.2-3B	0.2450	**0.2735**	0.2497	0.2508	0.2650	0.2548	0.0896
Qwen3-1.7B	—	**0.2621**	0.2077	0.2581	0.2198	0.2067	0.2259

Table 3: **Ablation of the refinement objective (20 steps).** k=candidates in loss; GRPO-only drops OPD; OPD-only drops practice reward. Values are practice-rubric score on held-out targets.

Case Study. Figure 5a traces $LiVO_2$, a held-out target, through this loop. Early in training the model sources lithium from the bare metal; the feedback rules it out as an alkali-metal source and names the carbonate, hydroxide, nitrate, and oxide families oxide synthesis draws on instead. Within twenty steps the metal route is gone. $LiVO_2$ never appears among the training targets, so the shift shows the model learning what the feedback requires, not recalling a memorized recipe.

6 CONCLUSION

We presented SynBench, a benchmark that measures whether the routes a synthesis planner returns can actually be executed in a laboratory, scoring each route by whether its chemistry balances and whether its precursors can be commercially obtained. Across sixteen LLM configurations, executability collapses as more candidates are requested: the executable share falls by 37% to 66% from the first to the tenth candidate, and the first unexecutable route arrives among the leading candidates. ExecuSyn removes both failures, raising the executable share of the top ten predictions by 44% and moving the first unexecutable route 86% later, and it keeps improving from practice feedback rather than converging after one update.

AI USE STATEMENT

Generative AI tools assisted with manuscript language editing and result analysis. The authors verified all outputs and accept full responsibility for the final manuscript.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024.

Anthropic. Claude 3.5 sonnet model card addendum. `https://www.anthropic.com/news/claude-3-5-sonnet`, 2024.

Evan R. Antoniuk, Gowoon Cheon, George Wang, Daniel Bernstein, William Cai, and Evan J. Reed. Predicting the synthesizability of crystalline inorganic materials from the data of known material compositions. *npj Computational Materials*, 9:155, 2023. doi: 10.1038/s41524-023-01114-4.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Binghong Chen, Chengtao Li, Hanjun Dai, and Le Song. Retro*: Learning retrosynthetic planning with neural guided a* search. In *International Conference on Machine Learning*, 2020.

Shuan Chen and Yousung Jung. Deep retrosynthetic reaction prediction using local reactivity and global attention. *JACS Au*, 1(10):1612–1620, 2021. doi: 10.1021/jacsau.1c00246.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

E. J. Corey and W. Todd Wipke. Computer-assisted design of complex organic syntheses. *Science*, 166(3902):178–192, 1969. doi: 10.1126/science.166.3902.178.

Stefano Curtarolo, Wahyu Setyawan, Gus L. W. Hart, Michal Jahnátek, Roman V. Chepulskii, Richard H. Taylor, Shidong Wang, Junkai Xue, Kesong Yang, Ohad Levy, Michael J. Mehl, Harold T. Stokes, Denis O. Demchenko, and Dane Morgan. AFLOW: An automatic framework for high-throughput materials discovery. *Computational Materials Science*, 58:218–226, 2012. doi: 10.1016/j.commatsci.2012.02.005.

Hanjun Dai, Chengtao Li, Connor W. Coley, Bo Dai, and Le Song. Retrosynthesis prediction with conditional graph logic network. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1:8, 2009. doi: 10.1186/1758-2946-1-8.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. KTO: Model alignment as prospect theoretic optimization. In *International Conference on Machine Learning*, 2024.

Samuel Genheden, Amol Thakkar, Veronika Chadimová, Jean-Louis Reymond, Ola Engkvist, and Esben Bjerrum. Aizynthfinder: a fast, robust and flexible open-source software for retrosynthetic planning. *Journal of Cheminformatics*, 12:70, 2020. doi: 10.1186/s13321-020-00472-1.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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. doi: 10.1126/sciadv.adg8180.

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. Chemformer: A pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3 (1):015022, 2022. doi: 10.1088/2632-2153/ac3ffb.

Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin A. Persson. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013. doi: 10.1063/1.4812323.

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, 2024. doi: 10.1039/D3SC03538G.

Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A. Shoemaker, Paul A. Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E. Bolton. PubChem 2025 update. *Nucleic Acids Research*, 53(D1):D1516–D1525, 2025. doi: 10.1093/nar/gkae1059.

Scott Kirklin, James E. Saal, Bryce Meredig, Alex Thompson, Jeff W. Doak, Muratahan Aykol, Stephan Rühl, and Chris Wolverton. The open quantum materials database (OQMD): assessing the accuracy of DFT formation energies. *npj Computational Materials*, 1:15010, 2015. doi: 10.1038/npjcompumats.2015.10.

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*, 6:203, 2019. doi: 10.1038/s41597-019-0224-1.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In *International Conference on Machine Learning*, 2024.

Sanghoon Lee, Kevin Cruse, Viktoriia Baibakova, Gerbrand Ceder, and Anubhav Jain. Text-mined dataset of solid-state syntheses with impurity phases using a large language model. *Scientific Data*, 2025. doi: 10.1038/s41597-025-06222-y.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

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

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

OpenAI. Introducing GPT-4.1 in the API. `https://openai.com/index/gpt-4-1/`, 2025.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Elton Pan, Thorben Prein, Juno Nam, Xiaochen Du, Soojung Yang, Pengfei Cai, Jennifer L. M. Rupp, Rafael Gómez-Bombarelli, and Elsa Olivetti. Synreason: Enhancing synthesis reasoning via reinforcement learning experimental feedback, 2026. URL `https://openreview.net/forum?id=1kLMpYT9AW`. OpenReview preprint.

Zongrui Pei, Junqi Yin, and Jiaxin Zhang. Language models for materials discovery and sustainability: Progress, challenges, and opportunities. *Progress in Materials Science*, 154:101495, 2025. doi: 10.1016/j.pmatsci.2025.101495.

Thorben Prein, Elton Pan, Sami Haddouti, Marco Lorenz, Janik Jehkul, Tymoteusz Wilk, Cansu Moran, Menelaos Panagiotis Fotiadis, Artur P. Toshev, Elsa Olivetti, and Jennifer L. M. Rupp. Retro-rank-in: A ranking-based approach for inorganic materials synthesis planning. *arXiv preprint arXiv:2502.04289*, 2025a.

Thorben Prein, Elton Pan, Janik Jehkul, Steffen Weinmann, Elsa Olivetti, and Jennifer L. M. Rupp. Language models enable data-augmented synthesis planning for inorganic materials. *ACS Applied Materials & Interfaces*, 2025b. doi: 10.1021/acsami.5c11229.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Philippe Schwaller, Teodoro Laino, Theophile Gaudin, Peter Bolgar, Christopher A. Hunter, Costas Bekas, and Alpha A. Lee. Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction. *ACS Central Science*, 5(9):1572–1583, 2019. doi: 10.1021/acscentsci.9b00576.

Marwin H. S. Segler, Mike Preuss, and Mark P. Waller. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, 555:604–610, 2018. doi: 10.1038/nature25978.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

Zhilong Song, Shuaihua Lu, Minggang Ju, Qionghua Zhou, and Jinlan Wang. Accurate prediction of synthesizability and precursors of 3D crystal structures via large language models. *Nature Communications*, 16:6530, 2025. doi: 10.1038/s41467-025-61778-y.

Nathan J. Szymanski, Bernardus Rendy, Yuxing Fei, Rishi E. Kumar, Tanjin He, David Milsted, Matthew J. McDermott, Max Gallant, Ekin Dogus Cubuk, Amil Merchant, Haegyeom Kim, Anubhav Jain, Christopher J. Bartel, Kristin Persson, Yan Zeng, and Gerbrand Ceder. An autonomous laboratory for the accelerated synthesis of inorganic materials. *Nature*, 624:86–91, 2023. doi: 10.1038/s41586-023-06734-w.

Austin Tripp, Krzysztof Maziarz, Sarah Lewis, Marwin Segler, and José Miguel Hernández-Lobato. Retro-fallback: retrosynthetic planning in an uncertain world. In *International Conference on Learning Representations*, 2024.

Zhengkai Tu, Sourabh J. Choure, Mun Hong Fong, Jihye Roh, Itai Levin, Kevin Yu, Joonyoung F. Joung, Nathan Morgan, Shih-Cheng Li, Xiaoqi Sun, Huiqian Lin, Mark Murnin, Jordan P. Liles, Thomas J. Struble, Michael E. Fortunato, Mengjie Liu, William H. Green, Klavs F. Jensen, and Connor W. Coley. ASKCOS: Open-source, data-driven synthesis planning. *Accounts of Chemical Research*, 58(11):1764–1775, 2025. doi: 10.1021/acs.accounts.5c00155.

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

Zheren Wang, Olga Kononova, Kevin Cruse, Tanjin He, Haoyan Huo, Yuxing Fei, Yan Zeng, Yingzhi Sun, Zijian Cai, Wenhao Sun, and Gerbrand Ceder. Dataset of solution-based inorganic materials synthesis procedures extracted from the scientific literature. *Scientific Data*, 9:231, 2022b. doi: 10.1038/s41597-022-01317-2.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

xAI. Grok 4.3. `https://docs.x.ai/developers/models/grok-4.3`, 2026.

Tian Xie and Jeffrey C. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical Review Letters*, 120:145301, 2018. doi: 10.1103/PhysRevLett.120.145301.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 36, 2023.

Appendix

A Related Work . . . 15
A.1 Synthesis Planning for Inorganic Materials . . . 15
A.2 Purchasability in Organic Retrosynthesis . . . 15
A.3 Learning from Feedback . . . 16
B Limitations and Broader Impacts . . . 16
B.1 Scope of the Executability Claim . . . 16
B.2 Evidence Base and Portability . . . 16
B.3 Broader Impacts . . . 16
C Benchmark Construction Details . . . 17
C.1 Chemical Validity . . . 17
C.2 Chemical Identity Verification . . . 17
C.3 Commercial Evidence Retrieval . . . 17
C.4 Online Extension and Caching . . . 18
C.5 Why a Precursor Is Not Obtainable . . . 19
C.6 Target Curation . . . 19
C.7 Compound Family Classification . . . 19
D Benchmark Statistics . . . 21
D.1 Obtainability by Precursor Class . . . 21
D.2 Literature Routes per Target . . . 21
D.3 Route Composition . . . 21
D.4 Precursor Reuse Across Targets . . . 21
D.5 Elemental Coverage . . . 22
E Evaluation Protocol . . . 22
E.1 Scoring Conventions . . . 22
E.2 Models Evaluated . . . 23
E.3 Generation and Scoring . . . 24
F ExecuSyn Training Details . . . 25
F.1 Executability Reward . . . 25
F.2 Practice Rubric: Family Conventions . . . 26
F.3 Hyperparameters . . . 27
G Extended Results . . . 28
G.1 Retention Across Rank Depth . . . 28
G.2 Hint Dependence Over Training . . . 28
G.3 Failure-Mode Decomposition . . . 28
G.4 Performance by Target Stratum . . . 29
G.5 Agreement with Literature Routes . . . 29
G.6 Optimization Beyond the Reported Backbones . . . 30
H Case Studies . . . 30
H.1 Chemistry That Balances and Shopping That Does Not . . . 30
I Prompt Templates . . . 31

A Related Work

Table A.1 groups representative lines of work by what they take as input, what they return, and what that output is evaluated against. The grouping is by evaluation target rather than by model class, because methods that are similar in architecture can differ entirely in what they are held accountable for.

Table A.1: Representative lines of work by input, output, and what the output is evaluated against.

Line of work	Input	Output	Evaluated against
Inorganic synthesis planning	Target formula	Precursor set, optionally with synthesis operations	Agreement with literature precursors; thermodynamic feasibility
Organic retrosynthesis	Target molecule	Route terminating in catalogue compounds	Whether every leaf compound appears in a supplier catalogue
Feedback-driven self-improvement	Model output and a feedback signal	Revised output	Improvement over the unrevised output

A.1 Synthesis Planning for Inorganic Materials

Text-mined recipe collections are the common starting point for this line of work. Kononova et al. (2019) assembled solid-state synthesis recipes from the literature, Wang et al. (2022b) did the same for solution-based procedures, and Lee et al. (2025) used a language model to extract syntheses together with the impurity phases they report. These collections fixed the input representation the field works with, and with it the standard later methods are judged by: agreement with what the literature records.

Methods built on them approach the task from several directions. He et al. (2023) recommend precursors by learning materials similarity over the same literature-mined recipes, and Kim et al. (2024) predict a synthesis recipe from an element-wise template formulation of the target composition. Retrieval-Retro and Retro-Rank-In cast inorganic retrosynthesis as retrieval and ranking over known precursors (Noh et al., 2024; Prein et al., 2025a). MSP-LLM unifies precursor prediction and operation prediction in one language-model framework (Noh et al., 2026). Prein et al. (2025b) show that models without task-specific fine-tuning already recall synthesis conditions reported in the literature, and Song et al. (2025) predict synthesizability and precursors directly from crystal structure, building on graph representations of the crystal originally developed for property prediction rather than synthesis (Xie & Grossman, 2018); a composition-only counterpart reaches the same synthesizability judgement without requiring a structure at all (Antoniuk et al., 2023). Language models more broadly are being applied across materials science well beyond synthesis planning, from property prediction to autonomous experimentation (Miret & Krishnan, 2025; Pei et al., 2025).

Across this line of work the output is judged against what the literature records or what thermodynamics permits. Both ask whether a route is chemically sound. Neither asks whether the starting materials it names can be obtained.

A.2 Purchasability in Organic Retrosynthesis

Organic retrosynthesis has long treated purchasable starting materials as the terminating condition of a route: a search is complete only when every leaf of the retrosynthetic tree falls within a catalogue of building blocks. Computer-assisted systems adopted this criterion from the outset (Corey & Wipke, 1969), and the neural search that replaced hand-coded transform rules (Segler et al., 2018) kept it: a route terminates only when it reaches a catalogue entry. The single-step models each backward step draws on take several forms – graph logic networks that score a template against a product (Dai et al., 2019), local reactivity models that predict an edit rather than retrieve a whole template (Chen & Jung, 2021), and pre-trained translation models fine-tuned for forward reaction and retrosynthesis prediction alike (Schwaller et al., 2019; Irwin et al., 2022) – and a learned cost function can then guide the search over them, as in the neural A* search of Retro* (Chen et al., 2020). Current search-based planners inherit the same termination rule, with the catalogue drawn from supplier inventories (Genheden et al., 2020; Tu et al., 2025). A route that has not yet reached a catalogue can still be scored on how close it is, and synthetic accessibility scores estimate exactly that from a molecule's fragments and complexity alone, without reference to any particular route (Ertl & Schuffenhauer, 2009). Later work refines what availability itself means, separating stocked compounds from make-on-demand ones and

folding price and lead time into how routes are ranked (Tripp et al., 2024). Purchasability there is not a filter applied after planning. It is part of what a completed route is.
Inorganic synthesis planning has no counterpart. Routes are returned as precursor sets without reference to whether those precursors can be obtained in the form the route declares.

A.3 Learning from Feedback

Improving a model's output from a signal it receives after producing that output has an established line of work. Preference signals, first learned from human comparisons over trajectory pairs (Christiano et al., 2017), now drive RLHF and RLAIF (Ouyang et al., 2022; Bai et al., 2022; Lee et al., 2024) as well as methods that fold the preference model directly into the policy update (Rafailov et al., 2023), or that replace paired comparisons with a binary desirable/undesirable signal altogether (Ethayarajh et al., 2024); a model can also be fine-tuned on the reasoning that led it to a correct answer, bootstrapping its own supervision rather than receiving an external signal (Zelikman et al., 2022); Self-Refine and Reflexion have a model revise its own output from feedback it generates or observes (Madaan et al., 2023; Shinn et al., 2023); LLM judges substitute model scoring for human scoring and agree closely with human raters when the scoring rubric is explicit (Zheng et al., 2023). In synthesis, SynReason post-trains with reinforcement learning on experimental feedback (Pan et al., 2026).
A related line improves an output at test time rather than through training: sampling several reasoning paths and taking their majority answer (Wang et al., 2022a), searching over intermediate thoughts with explicit lookahead (Yao et al., 2023a), and interleaving reasoning with actions that return new information from an environment or an external tool (Yao et al., 2023b; Schick et al., 2023) all revise what a model produces without touching its weights. These operate at inference time and read no rubric; the practice-feedback signal we build in Section 4.2 instead trains the weights themselves, so the resulting judgement survives to the next target rather than being recomputed on every prompt.
What a rubric encodes decides what the feedback can teach, a distinction mathematical reasoning has already found supervising each step rather than only the final answer sharpens (Lightman et al., 2023). A judge asked for its own chemical opinion returns an opinion. A judge given a rubric drawn from what the literature does returns evidence. Section 4.2 builds its rubric the second way.

B Limitations and Broader Impacts

B.1 Scope of the Executability Claim

Section 2.1 defines a route as lab-executable when it is chemically valid and every precursor it names can be obtained in the declared form. Both conditions are decided from evidence: element balance from the formula, obtainability from a commercial listing. The definition is confined to what can be checked this way. It does not assert that a route will yield the target under a given temperature profile, nor that the product will be phase-pure. Those are questions for the bench, and a route reaches them only after it satisfies both conditions here.
SynBench covers solid-state inorganic synthesis, a scope that follows from what makes obtainability decidable: solid-state precursors are commodity powders with a stated form and purity, so a listing either matches the declared form or it does not. Solution-phase and organic syntheses specify starting materials by concentration, solvent, and formulation, which calls for different product criteria. The construction procedure carries over; the criteria would be replaced.

B.2 Evidence Base and Portability

Obtainability is decided against Sigma-Aldrich, a supplier catalogue rather than a compound database. The distinction is the one the benchmark rests on: a compound database records that a substance is known, a supplier catalogue records that it can be bought. Listings state form and purity per product, the granularity the product criteria require.
Every label traces to a specific listing and can be rechecked against it. Because catalogues change, we release a timestamped snapshot of the database alongside the online lookup, which applies the same criteria to the live catalogue. The criteria are stated independently of the source, so another catalogue or a laboratory's own stock list can be substituted by replacing the lookup alone.

B.3 Broader Impacts

Synthesis planning that accounts for procurement reduces the cost of trial and error. A route naming an unobtainable precursor consumes a chemist's time before any material is made, and that cost falls

hardest on laboratories with small budgets and limited reagent inventories. Both the benchmark and the obtainability database are released, so they can be used to screen routes from any planner.
An obtainability label states only that a precursor can be purchased in the declared form. It carries no judgement about handling safety, toxicity, or regulatory status, and routes produced or scored with SynBench remain subject to review under local laboratory safety procedures.

C Benchmark Construction Details

This section gives the procedures Section 2.2 summarizes, in enough detail that any single label can be rechecked against the same evidence. Two questions are kept apart throughout: what a formula denotes, and whether that substance can be bought in the form a route declares.

C.1 Chemical Validity

Balance is checked over the core elements of a formula, meaning all elements except carbon, nitrogen, hydrogen, and oxygen. Those four leave a solid-state reaction as CO_2, N_2, H_2O, or O_2 and enter it from the atmosphere, so their presence on either side carries no information about whether a route balances. A route passes when the core elements of the target and those of the precursors coincide exactly: an element in the target with no source among the precursors fails the route, and so does an element the precursors introduce that the target does not contain, since it would be left behind in the product.
A separate check rejects any route citing a gas-phase species as a starting material. Solid-state synthesis weighs out solids, so this is not a borderline case to be scored down.
The check returns the offending element sets, not the verdict alone. Appendix F.2 returns them to the model as the stated reason a route was rejected, and Appendix G.3 reports which elements go unsupplied most often.

Algorithm C.1 Chemical validity of a route

Require: target formula t, route $R = \{p_1, \ldots, p_m\}$
Ensure: $v(R) \in \{0, 1\}$, with the offending element sets
for each $p \in R$ **do**
  if $\mathrm{phase}(p) = \text{gas}$ **then**
    return 0, "gas-phase precursor"
  end if
end for
$E_t \leftarrow \mathrm{elements}(t) \setminus \{\mathrm{C}, \mathrm{N}, \mathrm{H}, \mathrm{O}\}$
$E_R \leftarrow \bigcup_{p \in R} (\mathrm{elements}(p) \setminus \{\mathrm{C}, \mathrm{N}, \mathrm{H}, \mathrm{O}\})$
$missing \leftarrow E_t \setminus E_R$ ▷ target element with no source
$extra \leftarrow E_R \setminus E_t$ ▷ element left behind in the product
return $(missing = \emptyset \text{ and } extra = \emptyset)$, $(missing, extra)$

C.2 Chemical Identity Verification

Extraction from text introduces errors that no downstream check would catch: a dropped subscript, a misread symbol, a formula truncated at a line break. PubChem, the open chemical database maintained by the US National Center for Biotechnology Information, is queried for each extracted formula, and the molecular formula it returns is compared with the formula as extracted. A disagreement marks an extraction error, and the formula is corrected or set aside before the next stage.
What this step does not decide. A PubChem record establishes that a substance is known and described. It says nothing about whether any supplier sells it, in what form, or at what purity. Treating a database record as evidence of availability is what lets a route be feasible on paper and impossible at the bench. The two questions stay separate: Table C.1 states which source settles which.

C.3 Commercial Evidence Retrieval

Sigma-Aldrich is the research-chemicals catalogue of Merck KGaA, among those most widely stocked by synthesis laboratories, and each entry states the physical form and purity of the product sold. A catalogue records what can be bought, which is the question a chemist planning a synthesis faces.
Product criteria. A listing counts as evidence only if it meets all three criteria in Table C.2. A listing failing any of them is not accepted, however closely the substance matches.

Table C.1: The two sources and what each decides. Neither substitutes for the other.

Source	Question it answers	Fields used	Role in the decision
PubChem	What substance does this formula denote?	Molecular formula, synonyms, CAS number	Identity check; fallback keys
Sigma-Aldrich	Can that substance be bought in the declared form?	Product listing, physical form, stated purity	The obtainability label

Related forms do not substitute. Retrieval often returns a listing close to the declared precursor without being it: the nonahydrate where the route declares the anhydrous salt, one oxidation state where it declares another. In every such case the declared formula stays labelled not obtainable. A chemist ordering from that listing would receive a different starting material, so evidence for a related form is not carried over to the form the route names. Where the related form is itself a plausible precursor it is filed as its own entry, available to any route declaring it.
This rule is what makes the labels conservative, and it is the single largest source of not-obtainable labels, as Appendix C.5 reports.

Table C.2: Criteria a listing must meet to count as evidence of obtainability.

Criterion	Requirement	Why it is required
Chemical form	The exact form the route declares, not a hydrate, salt, or oxidation state of it	A different form has a different mass and behaves differently on heating
Physical state	Sold as a solid, typically a powder	Solid-state synthesis weighs out solids
Purity	A purity is stated on the listing	An unstated purity gives no basis for planning stoichiometry

C.4 Online Extension and Caching

A precursor absent from the offline database is not thereby unobtainable, so an online lookup extends the database on demand. It applies the criteria of Appendix C.3 unchanged; what differs is only when the lookup runs. Each resolved precursor is written back under the same schema, so a formula is retrieved once and every later route naming it reads the stored label. Algorithm C.2 states the order of these steps.
Training and evaluation pin ALLOWLIVE to false, so every label is read from the frozen snapshot rather than triggered live at scoring time. This controls only *when* the lookup of Algorithm C.2 is allowed to run, not whether it works: the same retrieval chain is what extends the database in the first place. Two constraints motivate pinning it off here. Training and evaluation both iterate in tight batches where a network round-trip would stall the loop, and a label that depended on when in the loop a query happened to fire would not be reproducible across reruns. Under this setting a precursor in neither the database nor the cache is scored as not obtainable, the conservative direction.

Algorithm C.2 Obtainability lookup. The retrieval chain on lines 4 to 6 is the same one used offline; ALLOWLIVE gates only whether it may run at this point.

Require: precursor formula p, flag ALLOWLIVE
Ensure: label $\in \{\text{obtainable}, \text{not obtainable}\}$
1: **if** $p \in$ database or $p \in$ cache **then**
2: **return** stored label
3: **end if**
4: **if not** ALLOWLIVE **then**
5: **return** not obtainable ▷ keeps the label reproducible
6: **end if**
7: $\ell \leftarrow \text{search}(p)$ ▷ exact declared form
8: **if** $\ell = \varnothing$ **then** $\ell \leftarrow \text{search}(\text{CAS}(p))$
9: **if** $\ell = \varnothing$ **then** $\ell \leftarrow \text{search}(\text{synonyms}(p))$ ▷ keys from Appendix C.2
10: label $\leftarrow$ obtainable **if** ℓ meets Table C.2 **else** not obtainable
11: write (p, label, ℓ) to cache
12: **return** label

C.5 Why a Precursor Is Not Obtainable

Each of the 124 precursors in this group is a real and known compound. Table C.3 gives the five kinds, their definitions, and their counts, which sum to the 124.
The largest kind follows directly from the rule of Appendix C.3: where a route declares an anhydrous salt and the catalogue carries the nonahydrate, the two are different starting materials. Read together, the five kinds show that a precursor is rarely unavailable because it is exotic. Far more often, the exact form a route names is not the form that is sold.

Table C.3: The five kinds of not-obtainable precursor. The counts are over all 124 precursors so labelled and sum to that total.

Kind	n	What it means
Hydrate-only form	48	The anhydrous form is declared; the catalogue carries only a hydrate
Metastable form	32	The declared form is not thermodynamically stable and is not sold
Unconventional reagent	23	Not used as a starting material in ordinary practice
Downstream phase	16	A synthesis target or intermediate, not a reagent
Solution-phase form	5	Supplied only as a solution, not as a solid
Total	124	

C.6 Target Curation

The target database is built in four stages.
Normalize. Formulas still carrying a substitution placeholder, such as *M* or *Ln* standing for an unspecified element, are unresolved rather than wrong, and are set aside here.
Deduplicate. Targets are collapsed by normalized formula, so a material reported by several papers contributes one target carrying all of its recorded routes. Collapsing by formula rather than by string is what makes the two splits disjoint.
Diversify by family. Targets are sampled across material families so the split is not dominated by the families most frequent in the literature.
Diversify by route. Within the families, targets are chosen to vary in how their routes are composed. Of the 185 test targets, 54 carry routes that differ among themselves in executability. Those targets are what makes route quality testable: without them, a method returning any recorded route would score the same as one returning the executable recorded route.

C.7 Compound Family Classification

Both the curation above and the practice rubric of Appendix F.2 ask what kind of compound a formula is. The same taxonomy answers that question for a target and for a precursor, because both are described in the same chemical terms.
The classes are the standard ones. The families are those solid-state chemistry already uses to organize itself, as in the section tags of the subject literature: oxides, halides, chalcogenides, nitrides, phosphides, borides, carbides, silicides, hydrides, and intermetallics. One split is ours, and for a stated reason: the oxygen-bearing compounds are divided by the further non-metal they carry, since that is what decides which additional precursor a route needs. A structural taxonomy has no cause to separate a silicate from a borate; a synthesis planner does, because one route buys SiO_2 and the other H_3BO_3. The one non-mineralogical name, *carboxylate*, is the standard coordination-chemistry term for oxalates and acetates.
Assignment is a fixed chain of tests. A formula is tried against the tests in the order of Table C.4 and takes the first class that matches, so it is assigned exactly one class and the same input always gives the same answer. Two points in the order carry chemistry rather than convenience. Oxygen is tested before anything else, so that $Ba(NO_3)_2$ is read as the nitrate a chemist would call it rather than as a nitride on the strength of its nitrogen; and among oxygen-free compounds a chalcogen is tested before phosphorus, so that thiophosphates such as Li_3PS_4 join the chalcogenides, where solid-state chemistry places them, rather than the phosphides.
Coverage. The taxonomy was fixed against the whole source corpus, not against the benchmark alone: every one of its 8,788 distinct target formulas and 1,987 distinct precursor formulas receives a class, as do all 1,185 benchmark targets and all 375 precursors of the released database. Gas-phase elements and hydrocarbons are deliberately left outside it. Whether a route may cite O_2 or a pyrolytic

Table C.4: The compound family taxonomy, in the order the tests are applied. A formula takes the first class it matches. Counts are distinct formulas in the source corpus, as targets (T) and as precursors (P); a class may serve as either. Classes with no corpus targets are precursor-only in this literature, and the three elemental classes are starting materials that are never themselves a target here.

Class	Membership rule	Example target	Example precursor	T	P
Oxygen-bearing families, tested first and in this order					
Carbonate	CO_3 group	$Ca_3Na_2(CO_3)_4$	$CaCO_3$	11	95
Nitrate	NO_3 group	—	$Ba(NO_3)_2$	0	238
Sulfate	SO_4 group	$FeSO_4$	Li_2SO_4	40	36
Phosphate	PO_4 or HPO group	$LiFePO_4$	$NH_4H_2PO_4$	592	109
Carboxylate	C_2O_4 or CH_3COO group, or any other carbon with oxygen	$Li_4Ti_5O_{12}C$	$Mn(CH_3COO)_2$	8	264
Hydroxide	Terminal OH with a metal	$NiMn(OH)_4$	$Al(OH)_3$	2	73
Borate	Oxygen with boron	$BaCu(B_2O_5)$	H_3BO_3	307	20
Silicate	Oxygen with silicon	Ca_2SiO_4	SiO_2	528	32
Oxyhalide	Oxygen with a halogen	$Ca_2CuO_2Cl_2$	BiOCl	126	28
Oxide	Oxygen with none of the above	$BaTiO_3$	TiO_2	6,234	603
Oxygen-free families					
Chalcogenide	S, Se or Te	Bi_2Te_3	Li_2S	307	110
Halide	F, Cl, Br or I	$BaMgF_4$	LiF	92	112
Nitride	Nitrogen	Ti_2AlN	Si_3N_4	34	48
Phosphide	Phosphorus	FeP	Fe_2P	21	14
Boride	Boron	MoAlB	TiB_2	13	14
Carbide	Carbon with a metal or metalloid	Ti_3AlC_2	SiC	44	16
Silicide	Silicon	$MoSi_2$	Mg_2Si	57	10
Hydride	Hydrogen	$NaMgH_3$	MgH_2	17	18
Intermetallic	Metals only	NiAl	FeAs	355	69
Elements in uncombined form, tested in this order, the last taking whatever the two lists do not name					
Alkali / alkaline-earth metal	Li, Na, K, Rb, Cs, Be, Mg, Ca, Sr, Ba	—	Na	0	11
Non-metal powder	B, C, P, S, Si, Te	—	S	0	7
Other elemental precursor	Every element neither list names	—	Cu	0	60

carbon source as a starting material is a feasibility question, settled by the gate of Appendix C.1 before the taxonomy is consulted, not a question about what kind of compound something is.

The three elemental classes are separated by how a chemist handles them rather than by position in the periodic table, because that is the distinction a synthesis route turns on. Sodium ignites in moist air and is weighed in a glovebox; copper or iron powder sits on an open bench; boron or silicon powder is the ordinary starting material for a boride or a silicide. Treating all three as one class would leave the rubric unable to tell an everyday metal powder from something nobody handles in air. The two lists are fixed enumerations, not periodic-table groups, and the third class takes every element they do not name, so an element is assigned by the same first-match rule as a compound. That third class is mostly the ordinary metal powders, Cu, Fe, Ti and the like, which is the role it plays in a route; it is named for that role rather than for a periodic-table group because it also carries the handful of metalloids a route may call for in elemental form. The rubric prompt of Appendix I names the same class after the transition metals, its largest membership, and states there that it also covers the rare earths and the post-transition metals. The two names denote one class: the prompt's wording is addressed to the judge, and the rule in Table C.4 is the definition.

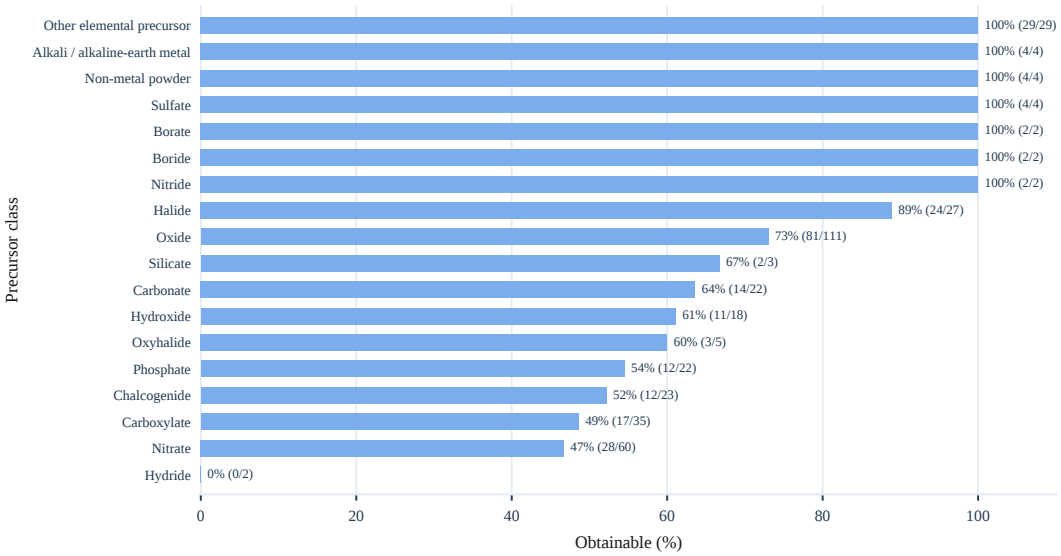


Figure D.1: Obtainability by precursor class, over the 375 precursors of the released database. Bars give the share of each class that is obtainable, with the count and class size beside them. Classes are ordered by rate.

D Benchmark Statistics

D.1 Obtainability by Precursor Class

Figure D.1 resolves the overall obtainability rate of Section 2.2 by precursor class, using the classification of Appendix C.7. The spread is wide: elemental metals and the simple anion classes are close to fully purchasable, while nitrates, carboxylates, chalcogenides, and phosphates sit at or below half. Each bar carries its own count, so a rate resting on a handful of entries can be read as such.

D.2 Literature Routes per Target

The 185 test targets carry 452 recorded routes between them, a mean of 2.4 each. Figure D.2 gives the distribution. Most targets carry more than one route, which is what allows route quality to be compared within a target rather than only across targets: where a target has a single recorded route, returning it is the only thing a method can be asked to do.

Figure D.2b splits the same bars by what each target's own routes do. Of the 185 targets, 54 carry routes that differ among themselves in obtainability, and those are the targets on which the ordering of candidates can be scored at all: elsewhere any recorded route is as executable as any other, so a method that ranks well and one that ranks arbitrarily receive the same score. The share rises with route count, from none of the single-route targets to 15 of the 25 carrying four or more, which is the second reason the split was built to favour well-documented targets. Only two targets have no obtainable recorded route at all.

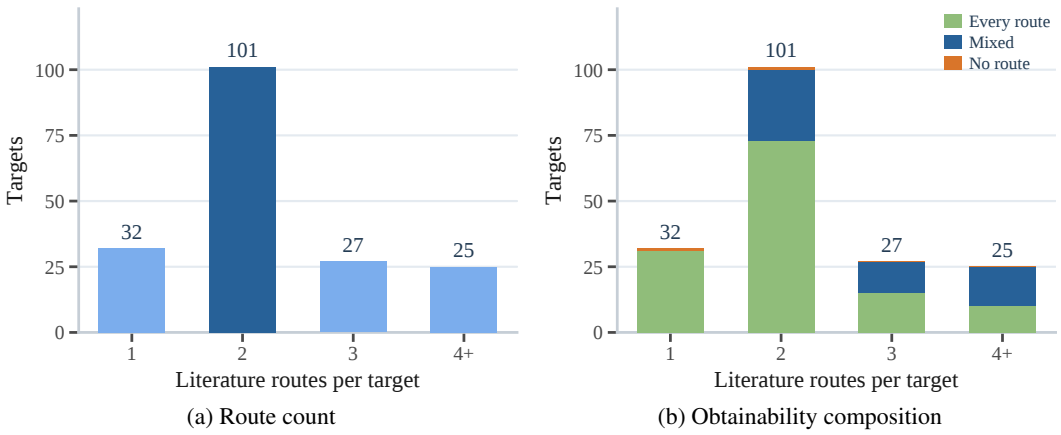
D.3 Route Composition

The 452 test routes draw on 159 distinct precursors across 1,122 precursor slots, so a precursor is used seven times on average. Routes are short: 264 use two precursors, 155 use three, and 32 use four, with one route using a single precursor. This is what a solid-state route looks like, and it is why obtainability is decided precursor by precursor rather than estimated over a route as a whole. A route of two or three starting materials has no redundancy; one unobtainable precursor removes it.

D.4 Precursor Reuse Across Targets

Figure D.3 ranks the precursors the test routes draw on most widely. Li_2CO_3 is called for by 32 of the 185 targets, and the reagents behind it are the staples of a solid-state bench: alkali and alkaline-earth carbonates, binary oxides, ammonium phosphates. Behind them lies a long tail, with 61 of the 159 precursors serving a single target each.

The shape matters for what obtainability measures. A benchmark whose precursors were all common would make purchasability trivial, and one with no common precursors at all would not resemble a



(a) Route count (b) Obtainability composition

Figure D.2: Recorded literature routes per test target, over the same four buckets. **(a)** How many targets carry each route count; targets with four or more are pooled, and the largest carries 14. **(b)** The same bars split by whether every recorded route of a target has all its precursors obtainable, only some do, or none do.

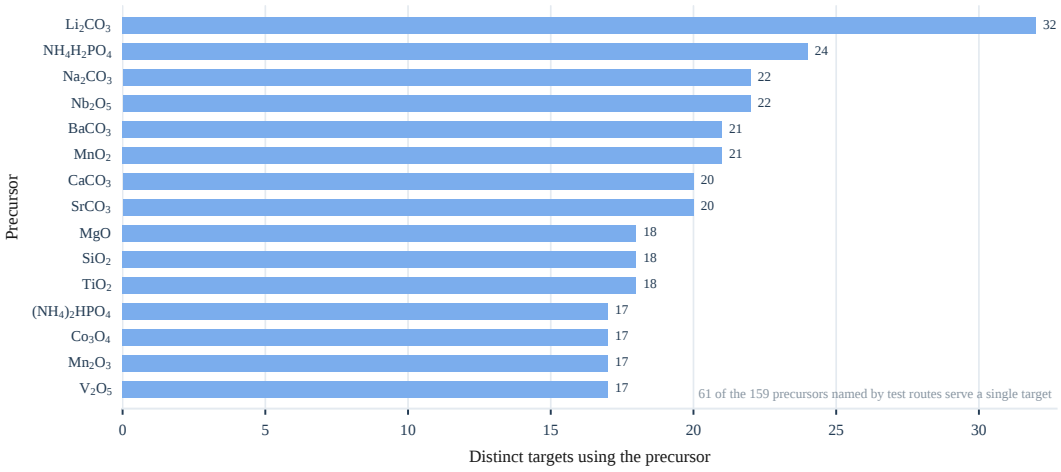


Figure D.3: The fifteen most widely reused precursors in the test split, by the number of distinct targets whose routes call for them.

real bench. The split has both, so the obtainability constraint bites on the specialised tail while the routine chemistry stays routine.

D.5 Elemental Coverage

Table D.1 gives the number of test targets containing each of the 53 elements the split reaches. Oxygen appears in 170 of the 185 targets, lithium in 37, and manganese and phosphorus in 27 each. The distribution has a long tail: 36 of the 53 elements appear in ten targets or fewer. Coverage therefore extends well past the elements that dominate the literature, into parts of the periodic table that a split built from frequency alone would not reach.

E Evaluation Protocol

E.1 Scoring Conventions

Section 2.3 defines the four metrics. This section records the conventions under which they are computed, since each of them decides how a degenerate candidate set is treated, and a set that is short or repetitive should not score as though it were neither.

Deduplication. Two routes are the same when their scored precursor sets are equal after formula normalization, so a route repeated in a different order or spelling is recognised as the repeat it is. Only the first occurrence is scored. Later occurrences count as failures and hold the ranks they were

Table D.1: Test targets containing each element, over the 185 targets of the test split. Read down each column in turn.

El.	n	El.	n	El.	n	El.	n
O	170	Ti	18	Ge	4	Ru	2
Li	37	Fe	13	Pr	4	Sc	2
Mn	27	Cu	11	Sn	4	Cd	1
P	27	B	10	W	4	Ce	1
Na	24	Ta	9	Cl	3	Cs	1
Ba	22	La	8	Cr	3	Eu	1
Ca	22	Ni	8	F	3	H	1
Nb	22	S	7	Ga	3	In	1
Mg	21	Zr	7	Gd	3	Ir	1
Sr	20	Bi	6	Pb	3	Nd	1
Co	19	K	6	Sb	3	Rh	1
Si	19	Mo	5	Tb	3		
V	19	Te	5	Y	3		
Al	18	Zn	5	Lu	2		

Table E.1: Configurations evaluated on SynBench. The four ExecuSyn rows are the backbones of Section 4, trained on the training split and evaluated on the same held-out targets as every other row.

Model	Access	Params	Model	Access	Params
Grok-4.3	API	–	Qwen3-32B	Open	32B
Claude-3.5-Sonnet	API	–	Qwen3-32B (think)	Open	32B
DeepSeek-V3	API	–	Qwen3-8B	Open	8B
GPT-4o	API	–	Qwen3-8B (think)	Open	8B
GPT-4o-mini	API	–	DeepSeek-R1-Distill-32B	Open	32B
GPT-4.1-Nano	API	–	DeepSeek-R1-0528-Qwen3-8B	Open	8B
Gemini-2.5-Flash	API	–			
Gemini-2.5-Flash-Lite	API	–	ExecuSyn (Qwen3-8B)	Ours	8B
Qwen3-Max	API	–	ExecuSyn (Qwen3-4B-Instruct)	Ours	4B
Qwen-Flash	API	–	ExecuSyn (Llama-3.2-3B-Instruct)	Ours	3B
			ExecuSyn (Qwen3-1.7B)	Ours	1.7B

given, so padding a candidate set with copies of one executable route lowers Exec@K rather than protecting it.

Short candidate sets. A model returning fewer than K routes is scored over K all the same, with the absent ranks counting as failures. Returning one executable route and stopping therefore gives Exec@$10 = 0.1$, not 1.0.

FirstFail at the boundary. FirstFail@K takes the value $K + 1$ when every one of the K routes is executable, so the metric is defined on all candidate sets and is bounded above by $K + 1$ rather than being unbounded or undefined at the top.

Obtainability at scoring time. Every precursor is resolved against the released database alone, with ALLOWLIVE pinned to false as described in Appendix C.4. Scores therefore depend only on the released snapshot, and rerunning the evaluation later returns the same numbers.

E.2 MODELS EVALUATED

Table E.1 lists the evaluated configurations, and the entries below record what each of them is. Fourteen distinct models are evaluated, two of which, Qwen3-32B and Qwen3-8B, are additionally run in their long-reasoning mode, giving sixteen configurations in total. Parameter counts are given where a model publishes one. The panel is deliberately heterogeneous: it spans both access families and a wide range of scales, and it includes models trained for long chain-of-thought reasoning (Wei et al., 2022) alongside models that are not, so a difference between two rows can be read against the kind of planner it came from.

Closed-source models. These are served through their providers' APIs and are taken as they are published, so each one stands for the capability its vendor ships at that tier.

- **Grok-4.3** (xAI, 2026) is the flagship of the Grok line, trained with large-scale reinforcement learning on reasoning-intensive tasks. It is the strongest baseline on several of the metrics reported here, so it sets the ceiling that ExecuSyn is measured against.
- **Claude-3.5-Sonnet** (Anthropic, 2024) is a general-purpose assistant model tuned for long-context reasoning and instruction following. It produces the best-ordered candidate sets of the closed-source group, staying executable furthest down the ranking (FirstFail@10 of 4.89, the best of any existing method), which makes it the right comparator for the question of whether fluent, well-structured candidate sets are also executable ones.
- **DeepSeek-V3** (DeepSeek-AI et al., 2024) is a large mixture-of-experts model with roughly 671B total parameters, of which about 37B are active per token. Its scale makes it the natural test of whether parameter count alone buys executability, and it reaches the highest three-candidate executability of any model evaluated here.
- **GPT-4o** and **GPT-4o-mini** (Hurst et al., 2024) (OpenAI) are the same model at two serving tiers, and **GPT-4.1-Nano** (OpenAI, 2025) is the lightweight member of a later OpenAI family. Taken together they span a wide spread of cost and capability within one vendor, which is what makes them useful for reading executability against the compute a deployment is willing to spend.
- **Gemini-2.5-Flash** and **Gemini-2.5-Flash-Lite** (Comanici et al., 2025) are the low-latency tiers of the Gemini 2.5 line, which is built for high-throughput serving with a thinking budget. They test whether a cost-optimized configuration retains the chemical judgement of a full-scale model.
- **Qwen3-Max** and **Qwen-Flash** (Yang et al., 2025) (Alibaba Qwen) are the largest and the fastest serving tiers of the Qwen line respectively. The Qwen family supplies both the frontier API tier evaluated here and, in its open weights, three of the four ExecuSyn backbones, so these two anchor the top and the bottom of the lineage our own models are built on.

Open-source models. These are run at their published weights and checkpoints, so every number below is reproducible without an API.

- **Qwen3-32B** and **Qwen3-8B** (Yang et al., 2025) (Qwen3 dense series) are the 32B and 8B dense members of the Qwen3 family, each available in an instruction-tuned release that supports a switchable long-reasoning mode. They are the open-weight models closest in scale and lineage to our own setting, and Qwen3-8B is also the backbone of our strongest ExecuSyn configuration, so the gap between it and that configuration isolates what training for executability contributes at a fixed backbone.
- **DeepSeek-R1-Distill-Qwen-32B** (DeepSeek-AI et al., 2025) (Qwen2.5-32B (Yang et al., 2024)) is a distillation of the DeepSeek-R1 teacher into a Qwen2.5-32B student. It carries the teacher's long chain-of-thought behaviour at a compact size, and it is the strongest reasoning-distilled model in the open-weight group.
- **DeepSeek-R1-0528-Qwen3-8B** (DeepSeek-AI et al., 2025) (Qwen3-8B) is the later DeepSeek-R1 release distilled into Qwen3-8B. It shares a backbone with the 8B configurations above while differing in what it was trained on, which makes the pair a controlled comparison of reasoning distillation against instruction tuning at equal scale.

The two long-reasoning rows in Table E.1 put Qwen3-32B and Qwen3-8B in their thinking mode and change nothing else, so their difference from the corresponding base rows is attributable to the reasoning mode rather than to the model.

E.3 Generation and Scoring

Every configuration receives the same prompt, given in Appendix I, which asks for 10 to 15 candidate routes ordered by the model's own estimate of viability. The scorer reads up to $M = 20$, so a model that returns more than asked is scored on what it returned rather than truncated at the request. Metrics are reported at $K \in \{1, 3, 5, 10\}$. Decoding is held fixed across configurations; the long-reasoning rows differ from their base models only in that mode being enabled.

One scoring implementation processes every output. It parses routes from the raw generation, normalizes each precursor formula, resolves obtainability, and aggregates all four metrics. No model receives a parser of its own, and no metric is recomputed by a second path, so a difference between two rows of Table E.1 is a difference in what the models returned.

F ExecuSyn Training Details

Section 4 states each phase as its own objective. Algorithm F.1 puts the two in one place, since what a reader needs in order to reproduce the method is not either objective alone but the order the pieces run in: where refinement picks up from optimization, which routes reach the judge and which are settled before it, and where the reason a route was rejected re-enters as a training signal rather than as a message.

Algorithm F.1 ExecuSyn. Phase 1 optimizes for executability; phase 2 continues from its checkpoint and optimizes for practice, adding a distillation term carried by the rejection reason.

Require: target splits $\mathcal{T}_{\text{train}}$, base policy π_0, family conventions $T(\cdot)$ (Table F.1)
Ensure: refined policy π

Phase 1: executability-oriented route optimization

$\pi \leftarrow \pi_0$
for each GRPO step **do**
  sample a rollout of m routes per target from π
  for each route R_i **do**
    $v_i \leftarrow$ element balance ▷ Algorithm C.1
    $o_i \leftarrow$ obtainable fraction ▷ Algorithm C.2, offline only
    $q_i \leftarrow v_i\, o_i\, \rho_i\, \mu_i$
  end for
  $S \leftarrow \text{quality} \cdot [\beta + (1-\beta)\,\text{quantity}]$
  update π by GRPO on S
end for

Phase 2: practice-feedback route refinement

for each GRPO step **do**
  sample a rollout of m routes per target from π
  for each route R_i **do**
    $a_i, h_i \leftarrow \textsc{Gate}(R_i)$ ▷ deterministic; settles before any judge call
    if $a_i = 0$ **then**
      $\phi_i \leftarrow 0$
    else
      $j_i, h_i \leftarrow \textsc{Judge}(R_i, T(f_t))$ ▷ only routes clearing the gate
      $\phi_i \leftarrow j_i$
    end if
  end for
  $S_\phi \leftarrow \frac{n_\phi}{m}\left[\beta + (1-\beta)(1 - e^{-n_\phi/\tau})\right]$
  for each position i and candidate $v \in S_i = \text{top-}k\{\pi_{\text{old}}(\cdot \mid s, y_{<i})\}$ **do**
    $w_v \leftarrow \text{softmax}_{v \in S_i}(l_{\text{old}}(v))$
    $\Delta_v \leftarrow \text{clip}\big(l_T(v) - l_{\text{old}}(v),\, -C, +C\big)$ ▷ teacher is π itself under $s \oplus h$, no gradient
    $A_v \leftarrow \Delta_v \cdot w_v$ ▷ advantage, detached
  end for
  update π on the GRPO loss from S_ϕ and the clipped surrogate L^{OPD} over $\{A_v\}$
end for
return π

Three things the objectives alone leave implicit are visible here. The gate runs on every route and the judge only on what survives it, so a route rejected on element balance never costs a judge call and never depends on one. The reason h_i is produced by whichever of the two rejected the route, and re-enters not as text shown to the model at inference but as the context under which the teacher forward pass is taken. And that teacher is the policy itself: the same weights, read without gradient under the augmented context, so the signal measures what the reason is worth to the model rather than importing a second model's opinion.

F.1 Executability Reward

One rollout proposes an ordered set of routes for one target, and the reward grades the set rather than any route within it. Grading proceeds at three levels. For route i, let o_i be the fraction of its precursors that are obtainable and $v_i \in \{0, 1\}$ its chemical validity, both decided as in Appendix C, and let $\rho_i \in [0, 1]$ and $\mu_i \in [0.3, 1]$ be the repetition and minimality penalties of Section 4.1. The

Table F.1: Precursor classes conventionally used by each target family, counted over 19,608 solid-state synthesis records. A class enters a family's set at 1% of that family's routes.

Target family	Routes	Conventional precursor classes
Oxide	14,745	carbonate, carboxylate, hydroxide, nitrate, oxide, metal
Phosphate	1,387	carbonate, carboxylate, halide, hydroxide, nitrate, oxide, phosphate, silicate
Silicate	1,139	carbonate, carboxylate, halide, hydroxide, nitrate, nitride, oxide, silicate
Borate	525	borate, carbonate, carboxylate, halide, hydroxide, nitrate, oxide, oxyhalide, phosphate, silicate
Chalcogenide	495	metal, carbonate, chalcogenide, halide, nonmetal, oxide
Oxyhalide	202	metal, carbonate, halide, hydroxide, intermetallic, nitrate, nonmetal, oxide, oxyhalide
Halide	129	halide, nitrate, nonmetal, oxide, oxyhalide, metal
Boride	16	metal, boride, nonmetal

route score is $q_i = v_i\, o_i\, \rho_i\, \mu_i$, so an invalid route scores zero however purchasable its reagents are: it cannot produce the target, and partial credit for buying the wrong things would be credit for the wrong behaviour.

The rollout score combines how good the routes are with how many of them work,

$$S = \underbrace{\frac{1}{m}\sum_{i\in\mathcal{U}} q_i}_{\text{quality}} \cdot \Big[\beta + (1-\beta)\,\underbrace{\big(1 - e^{-n/\tau}\big)}_{\text{quantity}}\Big], \tag{6}$$

where m is the number of routes emitted, $\mathcal{U}$ indexes the distinct ones, and n counts the routes that are chemically valid and fully obtainable. We use $\tau = 8$ and $\beta = 0.2$.

Two properties of this form are deliberate. Quality divides by every route emitted while summing only over distinct ones, so a repeated route costs exactly what an unusable one costs, and padding a rollout with copies is never profitable. Quantity saturates rather than caps, so proposing one more workable route always helps a little while the marginal gain falls away; a route is worth adding only while its chance of working is high enough to beat the dilution it causes. The model therefore learns where to stop instead of padding to a fixed count. A rollout that cannot be parsed, or that yields no route at all, scores zero.

F.2 Practice Rubric: Family Conventions

The rubric of Section 4.2 asks whether a route uses the kinds of starting material its target family conventionally uses. That standard is counted from the literature rather than written by hand, which is what lets it be defended by the data and regenerated when the data changes.

Counting. Over 19,608 records labelled as solid-state syntheses, we count for each material family the share of its routes that use a precursor of each class. The unit is the route, not the precursor occurrence: a class used once per route in a third of a family's routes is a normal way to make that material, however many times it appears within any single route. Classes reaching 1% of a family's routes enter that family's set, together with the family's own class, which is always admitted. Table F.1 gives the resulting sets.

Two stages. A route is judged in two passes, and the first is deterministic. The gate of Appendix C.1 rejects routes with an unbalanced element count, a gas cited as a solid precursor, or an unobtainable precursor, and states which of the three applies. These are facts a script settles, and spending a judge call on them would invite disagreement where none is possible. Only routes clearing the gate reach the judge, which checks each precursor against its target family's set in Table F.1. The judge is given the set and asked to apply it, not asked for its own view of what is conventional. Both stages return a reason in the same form, so the two merge into one feedback block and the model never needs to know which stage caught a route.

Judge configuration. The judge is Claude-Sonnet-5, decoded at temperature 0 with a 4,096-token budget, and is called once per rollout rather than once per route: every route that clears the gate is submitted in one batch and returned as one JSON object. Verdicts are matched back to routes by

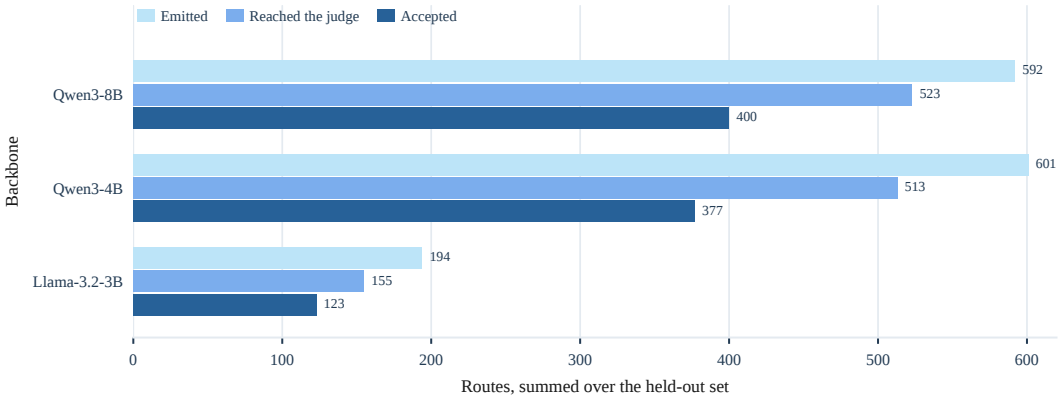


Figure F.1: Routes surviving each stage of the practice rubric, summed over the held-out targets. Emitted routes pass the deterministic gate before reaching the judge, which accepts a subset of those.

precursor content rather than by position, so a dropped or reordered entry cannot shift a verdict onto the wrong route, and a rollout that repeats a route consumes the judge's verdicts for it in order rather than collapsing them onto one. Greedy decoding is what makes the rubric an evaluable standard rather than a sampled opinion: the same route set submitted twice returns the same verdicts, so a score that moves over training reflects the policy rather than the judge. Three failure modes are distinguished and all score zero — an unparsable reply, a route absent from the returned verdict, and a failed call — and each is counted separately so an outage is visible as such instead of being mistaken for the policy getting worse.

Reliability in use. Across the four backbones' full held-out evaluations, 740 judge calls covered 7,246 emitted routes, of which 1,578 were settled by the gate and 5,668 reached the judge. Every one of those returned a parsable verdict that matched back to its route: the three failure modes above account for none of the 7,246. On the training side, re-scoring the 171,666 routes of the objective ablation left 11 routes on a failed call, a rate of 0.006%. The rubric is therefore dependable enough to train against, which matters because a judge that failed intermittently would zero whole rollouts and enter the advantage as noise.

Staying on task. The judge is asked one question, and the division of labour holds in practice: of the 2,662 rejection reasons it wrote across those evaluations, not one cites element balance, purchasability, or a gas-phase reagent — the three grounds the gate has already settled. The gate's 1,578 rejections break down as 1,205 unobtainable precursors, 249 element-balance failures, and 124 gas-phase reagents. Every reason the judge returns is instead about which family a precursor belongs to, which is the only question the second stage is for.

Prompt design. The judge is asked for a judgement, not for a classification. An earlier prompt required it to name each precursor's class exactly before scoring, and a smaller judge mislabelled formulas under that instruction, calling B_2O_3 an oxide rather than a borate — a label that decides whether the precursor is conventional for a borate target. Supplying two worked examples per family and asking directly whether the precursors suit the target removed that failure mode without needing a larger model. The prompt also says to be generous with legitimate minority practice and to reserve a zero for a family genuinely out of place, since the counted sets of Table F.1 admit any class reaching 1% of a family's routes and a rubric applied more strictly than it was counted would reject routes the literature supports. Appendix I gives the prompt in full.

Figure F.1 shows what each stage removes. Both layers do work: the gate settles a substantial share before any judge call is made, and the judge declines a further share of what reaches it. Putting the deterministic checks first is therefore not only cheaper but load-bearing, since every route it settles is one the judge cannot decide differently on a later call.

F.3 Hyperparameters

Both stages optimize with GRPO. Table F.2 gives the settings. The optimization stage is run on four backbones; refinement is run on the three of them that carry a feedback term.

The refinement stage uses one OPD candidate by default, which is the sampled-token form the term reduces to at $k = 1$. The ablation varies it over $k \in \{4, 8, 16\}$. All reported results use this default, which costs one forward pass per update less than any $k > 1$ setting: the wider forms need a separate

Table F.2: Training settings for the two stages. Refinement resumes from the optimization checkpoint of the same backbone.

Setting	Optimization	Refinement
Advantage estimator	GRPO	GRPO
Learning rate	1×10^{-6}	1×10^{-6}
KL coefficient	0.001	not used
Train batch size	16	8
Mini-batch size	16	8
Rollouts per prompt	8	1
Training steps	100	20
Max prompt length	512	1,536
Max response length	8,192	8,192
OPD clip C	—	1.0
OPD candidates k	—	1
PPO clip $\epsilon_{\text{lo}}/\epsilon_{\text{hi}}$	0.2	0.2 / 0.28
Reward β / τ	0.2 / 8.0	0.2 / 8.0

pass to collect the student's candidate set before the teacher can be scored on it. Table 3 shows that width buys little beyond this point, so the reported configuration is also the cheapest one that reaches the level we report.

Two differences between the stages follow from what each is optimizing. The refinement prompt of Appendix I carries worked examples with their explanations, which is why its prompt budget is three times the optimization stage's. And refinement draws one rollout per prompt rather than eight, because its reward comes from a judge call: the cost per rollout is dominated by that call rather than by generation, and the verdict it returns already covers the whole route set.

G Extended Results

G.1 Retention Across Rank Depth

Figure 4 is measured in two ways. Panels (a), (b), (d) and (e) are read under the training protocol, at the step each run reached; panels (c) and (f) use the same step-100 checkpoints as Table 1, so every model compared in them was measured at one point in its own schedule.

Figure 1b shows the executable share falling as more candidates are requested, for a selection of models. Over all sixteen evaluated configurations the fall runs from 37.4% to 65.5% between the first candidate and the tenth. The four ExecuSyn backbones fall from 10.1% to 16.4% over the same span. The two ranges do not overlap, and neither do the levels they end at. The weakest ExecuSyn backbone holds 0.718 at the tenth candidate, above the 0.597 of the strongest evaluated model. A chemist reading ten candidates rather than one is therefore reading a different kind of list: among the evaluated models the tenth is roughly a coin flip, while here it is about as reliable as the first.

Figure G.1a draws every one of the twenty configurations, the full version of the selection in Figure 1b. The grey band is not a few weak models dragging an average down; the decline is the shape of the whole field.

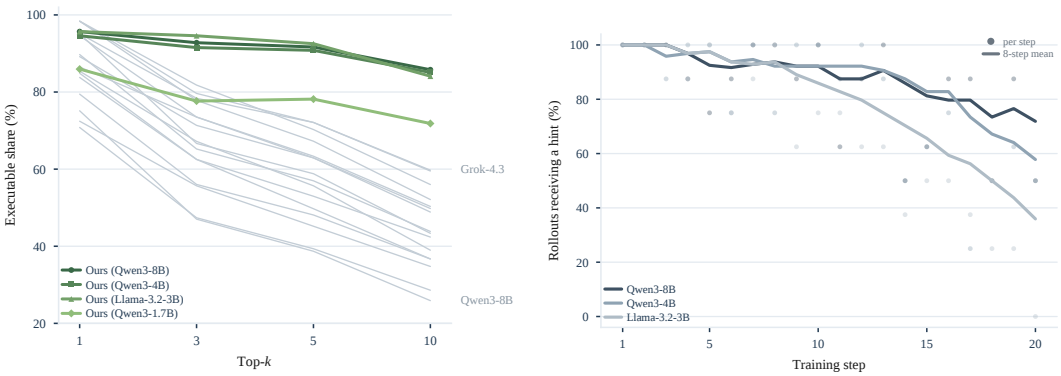
G.2 Hint Dependence Over Training

Figure 5b is a per-step mean over a trailing window of eight steps, with the raw per-step values drawn under it; Figure 5c is evaluated every four steps. The two panels plot the same quantity on different target sets, which is what lets the held-out panel separate transfer from memorization.

The rubric supplies a hint whenever a route departs from its family's conventions, so the share of rollouts receiving one measures how often the model still needs telling. Figure G.1b tracks it. All three backbones begin at saturation, every rollout drawing a hint, and all three fall away as training proceeds; by the last step Llama-3.2-3B-Instruct needs none. The convention is being taken on board rather than consulted afresh each time, which is what makes the feedback a signal worth optimizing against instead of a correction applied at inference.

G.3 Failure-Mode Decomposition

A route that is not executable failed one of two checks, and SynBench records which. This is what separates it from an evaluation scoring only formal correctness: a single number saying a route is unusable does not say whether the chemistry is wrong or the shopping is, and those call for different



(a) Executable share against Top-k

(b) Hint dependence over training

Figure G.1: **(a)** All twenty configurations. Grey is the sixteen evaluated models; green is the four ExecuSyn backbones, which stay above the field at every depth. **(b)** Share of refinement rollouts that receive a practice-feedback hint, by training step.

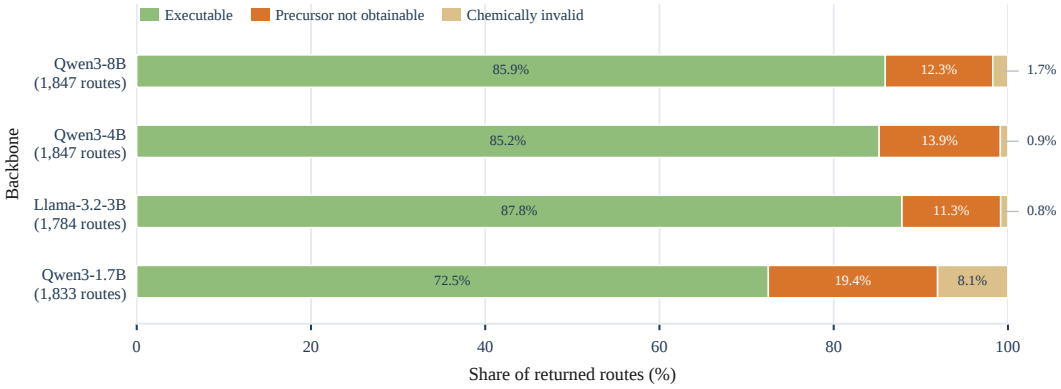


Figure G.2: Returned routes by outcome, over all routes within the first ten ranks. Route totals per backbone are given beside the labels.

fixes. The element sets returned by Algorithm C.1 carry the first case, and the obtainability label of Appendix C.3 the second.

Figure G.2 applies that decomposition to the trained backbones. What remains is a procurement problem, not a chemistry one: on the three larger backbones, chemically invalid routes account for under 2% of what is returned, and nearly all the shortfall is a precursor that cannot be bought. The diagnosis points somewhere specific, which a pass-or-fail score could not do.

G.4 Performance by Target Stratum

Figure G.3 splits the held-out targets two ways. Panel (a) groups them by how many literature routes they carry, which stands in for how well covered a target is: the single-route targets are the thinly documented ones, and every backbone scores lowest there and climbs as coverage grows. The test split therefore spans a real difficulty range, and the results are not an average over uniformly easy targets.

Panel (b) groups by compound family. The gain holds across families rather than resting on the oxides that dominate the split by count. Several of the smaller families carry few targets, marked beneath each group, and are read as indicative rather than as separable comparisons.

G.5 Agreement with Literature Routes

Optimizing for procurement raises a fair question: does a model trained this way drift away from the synthesis knowledge the literature records? Nothing in the reward asks for agreement with a recorded route, so agreement is free to fall.

It does not. Table G.1 reports LitHit@K, the share of held-out targets for which at least one of the first K returned routes matches a recorded one exactly, counting a match only when the scored

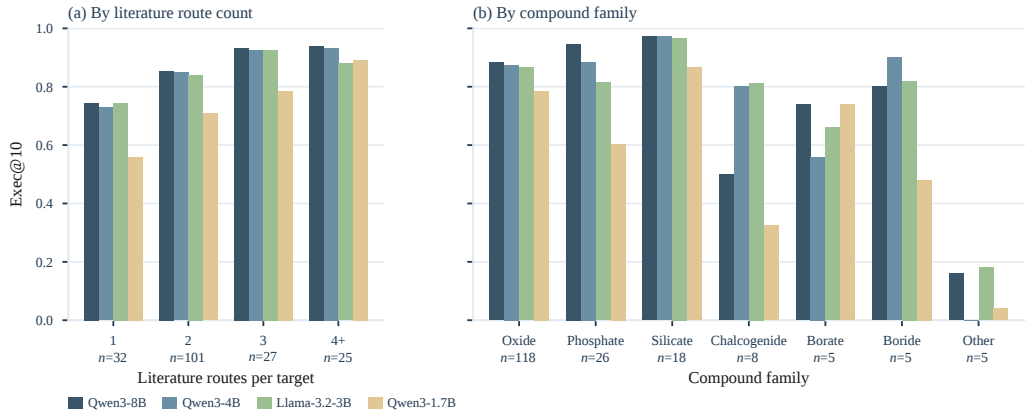


Figure G.3: Exec@10 by target stratum. **(a)** By recorded literature routes per target. **(b)** By compound family. Group sizes are given below each axis.

precursor sets are equal. This is an at-least-one rate over the first K routes, not an average over them as in Exec@K. At $K = 10$ the four backbones reach 0.751, 0.735, 0.730 and 0.654. Roughly seven targets in ten recover a route that was actually published, from an objective that never mentions publication.

The two constraints are therefore largely compatible. What a model learns under an executability reward is not a separate body of chemistry that happens to be purchasable, but the purchasable part of the chemistry the literature already describes.

Table G.1: LitHit@K: share of held-out targets with at least one of the first K returned routes matching a recorded literature route. An at-least-one rate, not an average over the K routes as in Exec@K.

Backbone	$K = 1$	$K = 3$	$K = 5$	$K = 10$
ExecuSyn (Qwen3-8B)	0.400	0.649	0.697	0.751
ExecuSyn (Qwen3-4B-Instruct)	0.595	0.622	0.670	0.735
ExecuSyn (Llama-3.2-3B-Instruct)	0.611	0.659	0.665	0.730
ExecuSyn (Qwen3-1.7B)	0.308	0.524	0.627	0.654

G.6 Optimization Beyond the Reported Backbones

Executability-oriented optimization is not tied to the scale of the backbones reported in the body. Two further stage-1 runs use Llama-3.1-8B-Instruct, a larger backbone than any of the four, under the configuration of Table F.2. Figure G.4 places them on the same axis as the reported runs.

Both runs reach the level the reported backbones settle at and hold it. Run a climbs from 0.140 to 0.830 by step 100 and holds above 0.79 from step 60 on; run b climbs from 0.174 to 0.836 over the same budget. Run b dips by 0.042 around step 45 and recovers within two validations, the only departure from a monotone climb in either run. Neither diverges, and neither needs a shorter or longer schedule than the smaller backbones: the climb is the same shape at a larger scale.

The two runs share a checkpoint and a configuration, so their agreement is a consistency check rather than a second result. What they establish is that the reward of Section 4.1 admits a stable optimum at 8B as it does below it, which is what a reader would otherwise have to assume from a panel that stops at 8B.

H Case Studies

H.1 Chemistry That Balances and Shopping That Does Not

Table H.1 lists the leading candidates one evaluated model returned for $Ca_2CuO_2Cl_2$. Every one of them balances: calcium, copper, oxygen, and chlorine each have a source, and nothing foreign is introduced. A criterion built on element balance passes all six, and a criterion built on agreement with the literature has nothing to say about them either way.

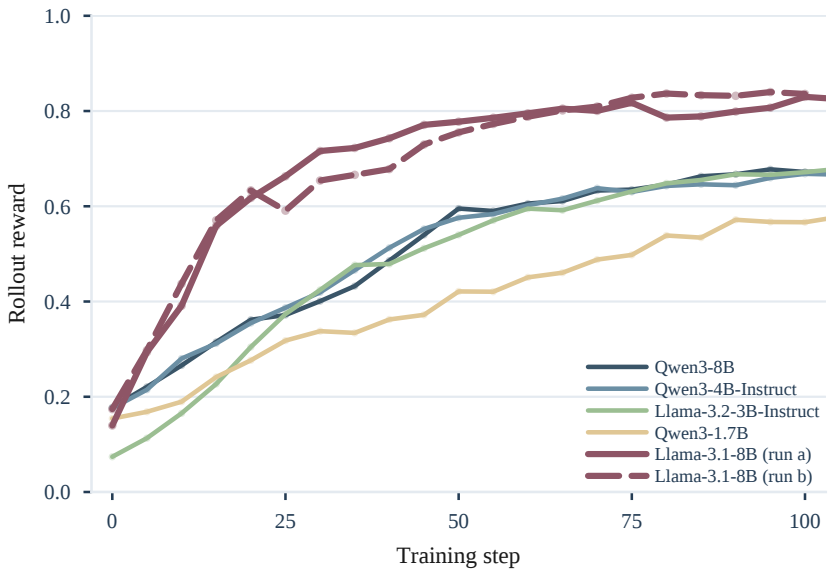


Figure G.4: Rollout reward over stage-1 training, to the 100 steps the two added runs share with the rest. The four reported backbones are drawn in their colour from Figure 4a; the two Llama-3.1-8B-Instruct runs are drawn in one colour and distinguished by line style.

None of the six can be run. Each supplies chlorine as Cl_2, which is a gas. A solid-state synthesis weighs out solids, and a gas cited as a starting material is not a route that a chemist can set up. Across the model's ten distinct candidates for this target, one was executable.

Table H.1: Leading candidates returned for $Ca_2CuO_2Cl_2$. All balance on elements; none can be run, and for the same reason each time.

Rank	Candidate route	Balances	Executable
1	CaO, CuO, Cl_2	yes	no
2	$CaCO_3$, CuO, Cl_2	yes	no
3	$Ca(OH)_2$, CuO, Cl_2	yes	no
4	CaO, $Cu(OH)_2$, Cl_2	yes	no
5	$CaCO_3$, $Cu(OH)_2$, Cl_2	yes	no
6	$Ca(OH)_2$, $Cu(OH)_2$, Cl_2	yes	no

The case also shows why a candidate set is scored as a set. The six routes vary their calcium and copper sources across the reagents a chemist would reach for, which reads as a well-spread list. They are nonetheless one route repeated, because the choice that decides all six is the chlorine source, and it is the same choice every time. Scoring only the first candidate would record one failure here; scoring ten records that the model had no second idea.

I Prompt Templates

Three prompts are used, one per stage of the work. The first asks a model for candidate routes: every evaluated configuration receives it, and executability-oriented optimization (Section 4.1) trains on it unchanged, so a trained model is asked at evaluation exactly what it was asked during training. The second is used by practice-feedback refinement (Section 4.2), which asks in addition for a one-sentence account of each route. The third is given to the judge. Fields in angle brackets are filled per call.

Route Proposal — Evaluation and Executability Optimization

SYSTEM
You are a computational chemistry expert specializing in solid-state synthesis and retrosynthesis. Return only valid JSON. Do not include any explanation, markdown, or extra text outside the JSON.

USER
Identify precursor combinations for solid-state synthesis of the target material: <TARGET_FORMULA>.

Requirements:

1. Propose 10–15 distinct, viable precursor combinations. Order from MOST to LEAST plausible.
2. Each combination must be chemically valid and able to produce the target material.
3. Do not repeat the same precursor within a single combination.
4. Every precursor must be a real compound available as a solid laboratory reagent. No gases.

Examples of valid route proposals:

```
'BaTiO3': [["BaCO3","TiO2"], ["Ba(NO3)2","TiO2"], ["BaO","TiO2"], ...]
'LiFePO4': [["Li2CO3","Fe2O3","(NH4)2HPO4"],
["Li2CO3","FeC2O4·2H2O","NH4H2PO4"], ["LiH2PO4","Fe2O3"], ...]
'Li2FeSiO4': [["Fe2O3","Li2SiO3"], ["FeO","Li2SiO3"],
["Li2CO3","Fe2O3","SiO2"], ...]
```

Output format: a single list of lists, nothing else.

```
[["precursor1a","precursor1b"],["precursor2a","precursor2b"], ...]
```

Every reagent named as an example is labelled obtainable in the benchmark, so the prompt never points a model towards something it would then be marked down for.

Route Proposal with Explanations — Practice-Feedback Refinement

SYSTEM
You are a computational chemistry expert specializing in solid-state synthesis and retrosynthesis. Return only valid JSON. Do not include any markdown or extra text outside the JSON.

USER
Identify precursor combinations for solid-state synthesis of the target material: <TARGET_FORMULA>.

Requirements:

1. Propose 5–10 distinct, viable precursor combinations. Order from MOST to LEAST plausible.
2. Each combination must be chemically valid and able to produce the target material.
3. Do not repeat the same precursor within a single combination.
4. Every precursor must be a real compound available as a solid laboratory reagent. No gases.
5. After the full list of combinations, give one explanation (1 sentence) per combination, in the same order: what each precursor supplies, and the general class of reaction it undergoes on heating.

Output format: a single JSON object with exactly two keys.
"routes": a list of lists — the precursor combinations, most to least plausible.
"explanations": a list of strings — one per route, in the same order as "routes".

Example output for target 'BaTiO3':

```
{"routes": [["BaCO3","TiO2"], ["Ba(NO3)2","TiO2"], ["BaO","TiO2"], ...],
 "explanations": [
   "BaCO3 supplies Ba and releases CO2 on heating; TiO2 supplies Ti; the resulting oxides
combine by solid-state diffusion to form BaTiO3.",
   "Ba(NO3)2 supplies Ba and releases NOx on heating; TiO2 supplies Ti; the oxides then
combine by solid-state diffusion.", ...]}
```

Generate the JSON object for the target <TARGET_FORMULA>.

The route count is narrower here than in the first prompt, 5 to 10 against 10 to 15. The reward's quantity term rises with the number of workable routes, and a wider request leaves more room for a model to append weak routes to an otherwise sound list rather than improve the list. The parse cap in the refinement reward is set to match.

Practice Rubric — Judge

SYSTEM
You are an experienced experimentalist in inorganic solid-state synthesis.

Task. For each route, judge whether its precursors are the kinds of starting materials this target's family normally uses. Score 1 if yes, 0 if no. For each 0, name the offending precursor's family and what this family normally uses instead.

Substance families. Every compound belongs to one of these families. Use your own chemistry knowledge to recognise which family a formula belongs to.

Oxygen-bearing: oxide (BaTiO3, TiO2) · phosphate (LiFePO4, NH4H2PO4) · silicate (Li2FeSiO4, SiO2) · borate (NaBaB9O15, H3BO3) · sulfate (Na2SO4, FeSO4) · oxyhalide (BiOCl, Ca2CuO2Cl2) · carbonate (BaCO3, Li2CO3) · nitrate (Ba(NO3)2, LiNO3) · hydroxide (NaOH, Ba(OH)2) · carboxylate (FeC2O4, CH3COONa)

Oxygen-free: chalcogenide (Cu2ZnSnS4, Bi2Te3) · halide (BaF2, FeCl3) · nitride (Si3N4, AlN) · phosphide (FeP, Cu3P) · boride (MgB2, TiB2) · carbide (SiC, TiC) · silicide (Mg2Si, MoSi2) · hydride (LiH, CaH2) · intermetallic (TiNi, Mg2Cu)

Elements: alkali/alkaline-earth metal (Li, Na, K, Rb, Cs, Be, Mg, Ca, Sr, Ba — this is the complete list) · transition metal, which also covers the lanthanides/rare earths and the post-transition metals (Fe, Cu, Ti, Zr, W, La, Bi, Al, Sn) · nonmetal solid (B, C, P, S, Si, Te)

Note. The prompt is reproduced as it was run. Its second and third element classes are the two that Table C.4 calls *other elemental precursor* and *non-metal powder*; the prompt names the second after its largest membership, since that is the anchor most useful to the judge, while the table states the membership rule that defines it.

Rule 1: check for oxygen first. An oxygen-bearing compound is never in the oxygen-free group.
Rule 2: a compound with both a chalcogen (S/Se/Te) and phosphorus, no oxygen (e.g. Li3PS4), is a chalcogenide, not a phosphide.
Rule 3: a rare-earth or post-transition metal (La, Bi, Al, Sn) is a transition metal for this judgement — do not hesitate over which precise periodic-table group it belongs to.

What each target family normally uses. The eight lists of Table F.1, given in full.

Scoring.
`1` — every precursor's family is one this target normally uses.
`0` — at least one precursor's family is not.

Be generous with legitimate minority practice. Reserve 0 for a precursor family genuinely out of place for this target. For each 0, write exactly one settled sentence, the final answer only — no hedging, no "actually", no revising a family assignment mid-sentence. Decide the family first, then write:

`"{precursor} is a {family}; {target family} synthesis normally uses {a normal family} instead."`

Name families, never a specific formula to copy, and never the words "whitelist" or "paradigm" — say what a chemist would say, not what this rubric calls it internally.

Output. JSON only, one object per route, same order as given, nothing else:

```
{"routes": [
  {"precursors": ["BaCO3", "TiO2"], "score": 1, "reason": null},
  {"precursors": ["Ba", "TiO2"], "score": 0,
   "reason": "Ba metal is an alkaline-earth metal; oxide synthesis normally uses a
carbonate or oxide as the barium source instead."}]}
```

The judge is handed the family lists of Table F.1 and asked to apply them. It is not asked what it believes conventional practice to be, which is what keeps the standard traceable to the literature count rather than to the judge. The instruction to settle on one sentence, and to name families rather than formulas, keeps the reason usable as feedback: a route is told what kind of reagent was wrong, not handed a substitute to copy.