STEERING AUTOREGRESSIVE RETROSYNTHESIS WITH CALIBRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Synthesis planning seeks an efficient sequence of chemical reactions building up to a target molecule. Such sequences are typically built by repeatedly invoking a pretrained autoregressive single-step model. Using classifier guidance, we can steer said model toward reactions with desired properties at inference time, without retraining. We show that cross-entropy trains a classifier to estimate a property's probability given the prefix, but to change an overconfident generator's ranking, that estimate must be extremely confident. For substructural properties like reaction type, where a specific, rare combination of tokens decides the label, no realistic corpus gives the classifier enough examples of each combination to reach that confidence. We overcome this issue with a novel method called Calibration for Autoregressive Guidance (CAG), which employs contrastive augmentation and a margin-based loss to calibrate the classifier so that it can meaningfully discriminate between continuations during decoding. Empirically, on USPTO-190 with chemist-specified guidance targets at every node, CAG returns a route meeting all the specified reaction types for 27.4% of targets, against 22.6% for an identically trained cross-entropy classifier, and 9.5% for an unguided model. Guidance also more than doubles the number of distinct routes found per target, from 2.1 for the unguided baseline to 4.4 with CAG, outperforming most template-based methods.

1 INTRODUCTION

Retrosynthesis, the task of predicting which precursor molecules can produce a given target, is a cornerstone of synthetic chemistry and drug discovery. Data-driven single-step models have made remarkable progress (Segler & Waller, 2017; Segler et al., 2018; Schwaller et al., 2019; Sacha et al., 2021; Zhong et al., 2022; Chen & Jung, 2021; Seidl et al., 2022), yet chemists rarely seek *any* viable precursors: a medicinal chemist may favor robust C–C cross-couplings for late-stage diversification (Brown & Boström, 2016; Roughley & Jordan, 2011), and a process chemist may avoid reactions with hazardous intermediates or pyrophoric reagents (Constable et al., 2007). Current models are largely agnostic to such preferences. Existing approaches either retrain the generator per property (expensive and brittle to constraint changes), or filter post-hoc from a fixed sample pool, leaving sequences outside the base model's beam unreachable. Classifier guidance (Yang & Klein, 2021; Meng et al., 2022) offers a lightweight alternative: steer at inference time using an auxiliary signal, leveraging the generative prior of an existing model.

We study *steerable single-step retrosynthesis* specifically for *substructural* properties (i.e. properties whose value depends on the presence of a specific rare combination of tokens) like reaction type, by applying classifier guidance to an autoregressive SMILES-based model. A token-level classifier predicts whether the completed reaction will satisfy the target property and reranks candidate completions during beam search (Eq. (2)). We notice that the classifier's training does not incentivize the core operation required during generation: the ability to rerank tokens proposed by the generator. The classifier only learns to assign each continuation a probability, so overriding the generator's preferred token can only happen if the classifier is extremely confident in its estimates. This problem is worse for substructural properties, where a specific, rare combination of tokens decides the label (Section 3.2). No realistic corpus supplies enough examples of each combination for a classifier to reach the necessary confidence. To overcome this, we propose *Calibration for Autoregressive Guidance* (CAG), combining *contrastive augmentation* (replacing the final token of partial sequences with random alternatives to

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

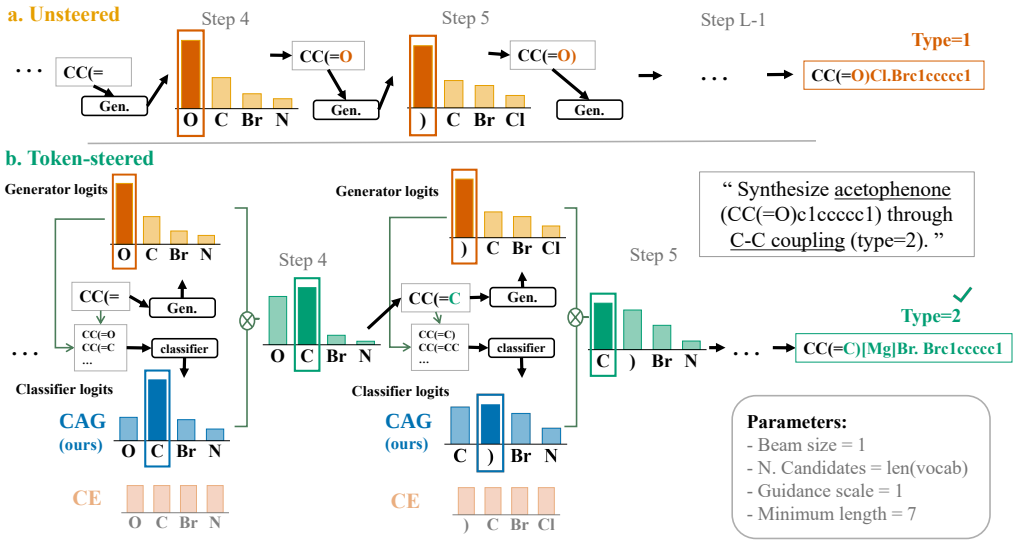


Figure 1: Autoregressive generation under three configurations: unguided generator, generator + CE-trained classifier, and generator + CAG-trained classifier. CE-guided decoding produces the same outcome as the unguided generator (Section 3.2), whereas CAG steers generation toward the target reaction type (C–C coupling).

force token-level sensitivity) with a *margin-based ranking loss* that directly enforces the discriminability gap required by guidance, instead of chasing down a confidence unachievable by correct estimators.

We evaluate CAG on reaction type steering. On USPTO-190-Steps (640 products), CAG reaches the highest achievable class diversity among all baselines including template-based methods, recovers valid steered precursors for 231 of the 287 products on which the unguided generator fails entirely, and produces samples with only 39% Jaccard overlap with unguided generation. In multi-step search on USPTO-190, CAG raises adherent solve from 0.14 (unguided generator) to 0.23 and more than doubles the routes found per target ($1.93 \rightarrow 4.40$). The mechanism behind this gain is direct: a cross-entropy classifier's error spread never reaches the generator's gap G_k (0% of decoding steps), while CAG's reaches it at 20.5% of steps, despite the two classifiers having nearly identical average confidence spread (3.7 vs. 3.6 nats).

- We prove that classifier guidance does not incentivize ranking under cross-entropy training, and that this is fatal for substructural properties like reaction type, where datasets naturally lack the rare-combination coverage needed to reach guidance-level confidence (Section 3.2).
- We propose *Calibration for Autoregressive Guidance* (CAG), pairing contrastive augmentation with a margin-based ranking loss that calibrates classifiers to clear the discriminability gap by construction (Section 3.3).
- We show the empirical benefits of CAG over CE-training (and filtering other baselines) in both single-step and multi-step generation, with measurements of the empirical behavior of both classifiers on our dataset corpus.

2 Preliminaries

2.1 Single-step retrosynthesis

Single-step retrosynthesis is the task of predicting plausible precursor molecules (reactants) for a given target product. Formally, we define $\mathcal{M}$ as the space of all valid molecules and represent a reaction as a pair $R = (\mathbf{r}, p)$, where $\mathbf{r} \subset \mathcal{M}$ is the set of reactants and $p \in \mathcal{M}$ is the product. A template-free single-step model parameterizes the conditional distribution $p_\theta(\mathbf{r} \mid p)$, generating reactant predictions directly without relying on predefined reaction templates.

Following the protocol of Zhong et al. (2022), we train our base generator with root-aligned SMILES, where products and reactants are aligned to minimize their edit distance so that the underlying autoregressive model focuses on learning local chemical transformations. Both products and reactants are represented as SMILES strings, and the generator emits the reactant string token by token, conditioned on the product. At inference we follow the augmentation protocol of Zhong et al. (2022); details in Section D.5. Throughout the paper, we refer to this product-reactant-aligned autoregressive model simply as the *generator*.

In many practical settings, chemists seek not just *any* valid set of precursors, but precursors satisfying additional constraints, such as proceeding via a specific reaction type. We refer to this as *steerable* single-step retrosynthesis. Prior work on steerable synthesis planning operates primarily at the search level in multi-step settings, filtering single-step predictions (Segler et al., 2018) or modifying value functions (Armstrong et al., 2024). At the single-step level, recent work steers retrosynthesis language models through disconnection prompts (Thakkar et al., 2023) or LLM agents that wrap chemistry tools (Bran et al., 2024); both operate above the token level and inherit the base model's coverage.

2.2 Classifier guidance in autoregressive models

Let ϕ denote a property of a reaction, such as its reaction type, and let $p_\theta(\phi \mid \mathbf{r}, p)$ be the probability that the reaction $(\mathbf{r}, p)$ satisfies it. Applying Bayes rule with the exponential tilt of (Dhariwal & Nichol, 2021), which raises the property term to a power $\lambda \geq 0$, gives the guided target distribution

$$p_\theta^\lambda(\mathbf{r} \mid \phi, p) = \frac{1}{Z_\lambda} \left[p_\theta(\phi \mid \mathbf{r}, p)\right]^\lambda p_\theta(\mathbf{r} \mid p), \qquad Z_\lambda = \sum_{\mathbf{r}' \in \mathcal{V}^L} \left[p_\theta(\phi \mid \mathbf{r}', p)\right]^\lambda p_\theta(\mathbf{r}' \mid p) \tag{1}$$

with $\mathcal{V}$ the token vocabulary and L the sequence length. Since our generator emits one token at a time, what decoding needs is the distribution of a single token r_k given the prefix $\mathbf{r}_{<k}$. Marginalizing Eq. (1) over completions and taking the ratio of two consecutive prefixes (full derivation in Section B.1) reduces this to the generator's own next-token distribution, reweighted by a single prefix-dependent factor:

$$p_\theta^\lambda(r_k \mid \mathbf{r}_{<k}, \phi, p) \;\propto\; p_\theta(r_k \mid \mathbf{r}_{<k}, p)\, U_\lambda(\mathbf{r}_{\leq k}) \tag{2}$$

We call $U_\lambda(\mathbf{r}_{\leq k})$ the *completion probability* of the prefix: it averages the property term over every way the generator could finish that prefix. When ϕ is a deterministic function of the finished reaction (as reaction type is), this average collapses to the true posterior probability that the property holds given the prefix,

$$U_\lambda(\mathbf{r}_{\leq k}) = p_\theta(\phi \mid p, \mathbf{r}_{\leq k}), \tag{3}$$

the quantity a token-level classifier has to estimate. Writing Eq. (2) in log space, the form used throughout the rest of the paper,

$$\log p_\theta^\lambda(r_k \mid \mathbf{r}_{<k}, \phi, p) = \log p_\theta(r_k \mid \mathbf{r}_{<k}, p) + \log U_\lambda(\mathbf{r}_{\leq k}) + C, \qquad C = -\log U_\lambda(\mathbf{r}_{<k}) \tag{4}$$

C is shared by every candidate value of r_k at that prefix and therefore cancels in any comparison between them. The full derivation, including the general (non-deterministic) form of U_λ, is given in Section B.1.

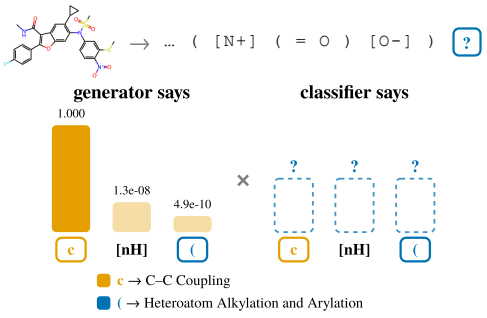
3 Methods

The point of applying guidance is to be able to steer the generator towards desired sequences, especially if they are otherwise unlikely for a given product. Figure 1 shows what it takes for classifier guidance to be able to steer: with the same generator and the same decoding budget, one trained classifier leaves the output untouched while another moves it onto the requested reaction type. In this section, we explore what is needed to achieve this effect, why it is specifically challenging for substructural properties, and how calibrating with a margin loss intervenes to enable the steering effect.

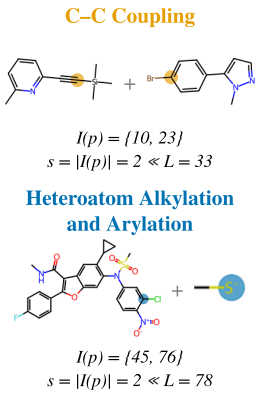
3.1 WHY CROSS-ENTROPY CLASSIFIERS ARE NOT BUILT FOR SUBSTRUCTURAL GUIDANCE

Classifier guidance usually trains its classifiers with a cross-entropy loss (Meng et al., 2022; Yang & Klein, 2021). This loss only teaches the classifier to assign the right property value to a training prefix, including prefixes sampled from the generator. It gives the classifier no incentive to rank one prefix above another, for instance, to say that a particular next token is more likely to satisfy the property than others. Yet ranking is exactly what the classifier's logits are used for at inference: they modify the generator's logits to make it prefer one next token over another.

This mismatch gets worse because the generator is overconfident to begin with (Holtzman et al., 2020; Kumar & Sarawagi, 2019). It typically favors one token over the second-best option by orders of magnitude. So no matter how it is trained, for a classifier to flip the generator's pick at a step, it needs to be at least as confident as the generator in the token it puts forward, relative to every other token. Figure 2a shows a real case: at one decoding step, the generator assigns probability 0.999 to a



(a) One decoding step



(b) Reaction type as a junta

Figure 2: Why guidance at a single decoding step is hard for substructural properties like reaction type. (a) The real decoding step of Section 3.1: the generator commits to one reaction type with near-certainty (C-C coupling), and the token needed ('(') for the requested class (HeteroAtom Alkylation and Arylation) sits far lower in its ranking. What the classifier reports at this step (right) is left open: this is the question Section 3.2 answers. (b) Reaction type is decided by a handful of positions in the generated token string, across otherwise unrelated reactions. $I(p)$ marks the decisive positions, coloured to match the class it decides; $s = |I(p)|$ stays small (≤ 2 here) relative to the sequence length L.

token that commits the reaction to C-C coupling, while the token needed for a different, verified type sits 21.4 nats lower in the generator's own ranking. Whatever the classifier reports at this step, it has to close a gap of the same size in the other direction, or the generator's choice remains unchallenged.

On top of that, some properties need one specific set of tokens switched for a certain value to be achieved. Figure 2b shows this across two unrelated reactions: in each case, reaction type is decided by only $s = 2$ or fewer positions in the generated token string (marked $I(p)$), while every other position can vary freely without changing the label. The rarer a reaction type is for a product under the generator, the bigger the gap the classifier has to close in the generator's logits to force its token choice at the required step. For any property decided by matching a local substructure (including reaction type, leaving group choice, functional group avoidance), winning at these positions is what guidance success comes down to.

Simply increasing the guidance strength λ does not fix this either: reranking needs a gap between logits, not a larger overall magnitude. With infinite training data, a classifier is guaranteed this level of confidence (Gneiting & Raftery, 2007; Bartlett et al., 2006). With a realistic, finite dataset, though, we need a prohibitively high number of samples to reach the confidence guidance requires. The next section derives exactly how many samples are needed and under what conditions.

3.2 THE SAMPLE COST OF A SUBSTRUCTURAL PROPERTY

Formally, write r^* for the generator's favorite next token at prefix $\mathbf{r}_{<k}$, and a for the token value that favors the property. Substituting the guidance rule (Eq. (4)) into the condition for the guided decoder to prefer a over r^*, and cancelling the constant shared by both candidates, gives

$$\underbrace{\log U_\lambda(\mathbf{r}_{<k}, a) - \log U_\lambda(\mathbf{r}_{<k}, r^*)}_{\Delta_k} \;>\; \underbrace{\log p_\theta(r^* \mid \mathbf{r}_{<k}, p) - \log p_\theta(a \mid \mathbf{r}_{<k}, p)}_{G_k} \tag{5}$$

Δ_k is the *token discriminability* the classifier reports, and G_k the *guidance requirement* the generator has already set. When $a = r^*$ both sides are zero, and guidance changes nothing at that step. We note that our decoding uses beam search, so a property-satisfying hypothesis has to keep winning, step after step, to stay on the beam. We derive a cumulative survival condition in full in Section B.2, and show that it requires winning Eq. (5) at enough individual steps.

How many steps guidance has to win depends on how many positions decide the property. Formally, fix a product p and call $\phi(\cdot, p) : \mathcal{V}^L \to \{0, 1\}$ an *s-junta* if there is an index set $I(p) \subseteq \{1, \ldots, L\}$ with $|I(p)| \leq s$ such that $\phi(\mathbf{r}, p) = \phi(\mathbf{r}', p)$ whenever $\mathbf{r}$ and $\mathbf{r}'$ agree on the positions in $I(p)$. Reaction type is one such property, with $I(p)$ the handful of atoms and bonds at the disconnection site (Fig. 2b). What a classifier has to learn at those positions is therefore an s-junta over $\mathcal{V}^{|I(p)|}$: a lookup from each of the $|\mathcal{V}|^s$ token combinations at $I(p)$ to a label, with no two combinations assumed related. This rules out pooling as a shortcut: since U is unconstrained across combinations, one atom's difference at $I(p)$ can flip the label outright, and there is no notion of "nearby" combinations for an embedding to exploit, so a classifier can only be confident about a combination it has actually seen, repeatedly, in training. This limit applies to any estimator of U, not only one built by cross-entropy, and it affects how many samples an estimator needs to see of each combination to achieve a minimum confidence, as established by the proposition below.

Proposition 1 (Sample requirement for an s-junta posterior)**.** *Fix a combination $x \in \mathcal{V}^{|I(p)|}$ of tokens at the decisive positions $I(p)$, and let $\mathcal{F}_s$ be the family of functions $U : \mathcal{V}^{|I(p)|} \to [0, 1]$ under which $U(x)$ and $U(x')$ are unrelated for $x \neq x'$. Let $n(x)$ be the number of training examples whose decisive tokens equal x, and let $\hat{U}(x)$ be the output of* any *estimator required to satisfy $|\hat{U}(x) - U(x)| \leq \sigma$ with probability at least $1 - \delta$ for every $U \in \mathcal{F}_s$. Then*

$$n(x) \;\geq\; c \cdot \frac{\log(1/\delta)}{\sigma^2} \tag{6}$$

for a universal constant $c > 0$.

The proof is in Section B.3: it constructs two members of $\mathcal{F}_s$ that agree everywhere except at x, so any training row not landing at x is uninformative about which is true, reducing the question to a single Bernoulli test on the $n(x)$ rows that do. Concretely, the 50k-model vocabulary has $|\mathcal{V}| = 72$ tokens, and the 80%-concentration proxy m ranges up to 10, with median 1 (Section C.1). Even a middling $s = 4$ already gives $|\mathcal{V}|^s = 72^4 \approx 2.7 \times 10^7$ distinct combinations $I(p)$ can take, and at the proxy's upper end, $s = 10$, it is $72^{10} \approx 4 \times 10^{18}$. Most combinations a product can present are therefore guaranteed to appear zero or one times, no matter how the corpus is built, which is far below the $n(x) \geq 18$ examples Proposition 1 requires even for a modest accuracy target ($\sigma = 0.1$ at 95% confidence, $\delta = 0.05$; Eq. (43)).

Global properties Not all properties we could guide towards are substructural; there are also global properties like molecular similarity to a starting material, which is a function of the whole finished molecule that shifts continuously as the string is completed. In fact, whether pushing toward a given token helps or hurts depends on the tokens that follow it, so there is no fixed direction to push toward at a prefix, independent of the rest of the generation. Such properties have to be optimized globally instead, and this analysis does not apply to them.

3.3 ENFORCING THE GAP THROUGH A MARGIN CALIBRATION

If the gaps guidance needs cannot be learned from the data, a classifier that reports the completion probability cannot supply them, so it has to rely on something else to learn the needed gap. *Calibration*

for Autoregressive Guidance (CAG) does this with two changes to how the classifier is trained. One decides the sign of the difference between candidate tokens within a step, the other decides its size.

Contrastive augmentation. For each partial sequence $\mathbf{r}_{\leq k}$ we build a same-length negative example by replacing its final token r_k with a uniformly random vocabulary token. We label the modified prefix with a catch-all class $C+1$ distinct from the C reaction classes. This example differs from a real one in exactly the position guidance compares, so it lowers the classifier's score for the prefix with a wrong continuation. We draw the replacement token uniformly randomly to avoid producing a different prefix that is likely under the generator and therefore belongs to a true reaction class.

Ranking loss. Substituting the classifier's estimate p_α for U_λ in Eq. (4), and dropping the constant C that cancels in any comparison between candidates, gives the guided log-score the decoder actually compares,

$$\tilde{s}_\alpha(r) \;:=\; \log p_\theta(r \mid \mathbf{r}_{<k}, p) \;+\; \log p_\alpha(\phi \mid \mathbf{r}_{<k}, r, p), \tag{7}$$

and require a_k (the ground-truth next token) to beat r_k^* by a margin $\gamma > 0$: $\tilde{s}_\alpha(a_k) - \tilde{s}_\alpha(r_k^*) \geq \gamma$. This margin is exactly $\Delta_k - G_k$ from Eq. (5), so the loss enforces the per-step condition directly, instead of estimating Δ_k and G_k separately. Training p_α on the union $\mathcal{D}$ of ground-truth partial sequences, contrastive augmentations and on-policy generator samples, the objective is

$$\mathcal{L} \;=\; \mathbb{E}_{\mathcal{D}}\big[\mathcal{L}_{\mathrm{CE}}(p_\alpha, y)\big] \;+\; \mu\, \mathbb{E}_{\mathcal{D}}\big[\max\big(0,\; \gamma + \tilde{s}_\alpha(r_k^*) - \tilde{s}_\alpha(a_k)\big)\big], \tag{8}$$

with $\mu > 0$ weighting the two terms; the full procedure is in Algorithm 1.

The target distribution of CAG Since CAG uses an additional loss term and contrastive data augmentation, it is not a proper scoring rule (unlike cross-entropy), and therefore it does not converge to the true probability U. Here we consider what it converges to instead. While the cross-entropy term of Eq. (8) pulls the classifier toward U at each input on its own, the ranking term is a hinge on a difference between two inputs sharing a prefix, minimised by any function whose gap at (a_k, r_k^*) reaches γ on the guided score Eq. (7). The minimiser of the sum is U wherever the margin already holds, and is pushed off it, by as much as the hinge demands and no more, at the steps where it does not. This deliberate departure from U is a within-step error, and only its spread across candidate tokens, osc_k reaches the decoder. Raising γ inflates osc_k and, with it, the bound in Eq. (63) on how far the whole guided sequence distribution can drift from the one the ideal classifier would produce. We elaborate on this result with detailed derivations in Section B.4.

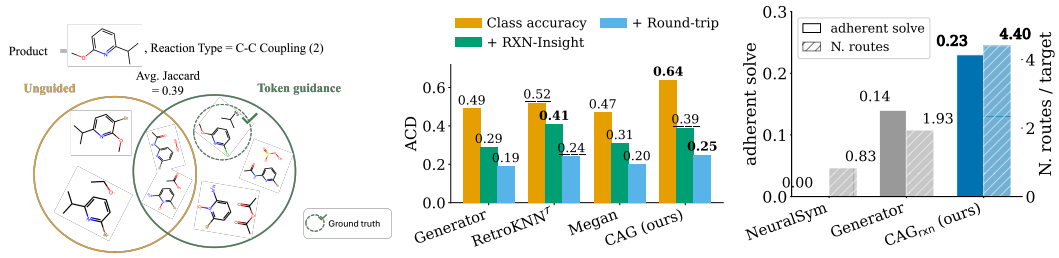
Relation to other guidance methods. The completion probability is what the other guidance methods in the literature aim at as well, by other routes. With an ideal conditional head and guidance strength 1, classifier-free guidance (Ho & Salimans, 2022) contributes $\log U(\mathbf{r}_{\leq k}) - \log U(\mathbf{r}_{<k})$ at a candidate token, whose second term is shared by the step. In twisted sequential Monte Carlo (Zhao et al., 2024) the optimal twist for a terminal reward is the expected reward under the base model's completions, again the completion probability, and the optimum of KL-regularised fine-tuning (Korbak et al., 2022) is the sequence-level tilt of Eq. (1), whose per-step factor is U_λ. Therefore, when ϕ is an s-junta, Proposition 1's per-cell sample requirement applies to whichever estimator of U_λ the method relies on, whether read off the base model's own completions or fit by a separate classifier. CAG does not remove that cost either. It instead asks the classifier to be accurate only where Eq. (5) is evaluated, at the pair (a, r^*) the decoder is choosing between, and enforces that with the margin γ on the discriminability gap Δ_k.

4 Experiments

We evaluate CAG along three axes: (1) whether token-level guidance reaches chemistry the unguided generator would not, at both the single-step and multi-step level (Section 4.1) and (2) how much of that gain comes from the training procedure itself, comparing cross-entropy and CAG trained on identical data (Section 4.2).

4.1 Why single-step guidance

Before asking which classifier steers best, we check whether guidance reaches candidates the generator would not otherwise produce at all.



(a) Guided vs. unguided samples (b) Single-step diversity (c) Multi-step search

Figure 3: Guidance broadens what the generator reaches, at both the single-step and multi-step level. **(a)** Effect of guidance on the generator's samples for one product (full-dataset numbers in Section E.1). **(b)** Achievable class diversity (ACD) on USPTO-190-STEPS for a subset of methods (full comparison with every baseline in Fig. 11); CAG reaches the highest diversity under every criterion. **(c)** Multi-step search on USPTO-190 (Retro*, USPTO-190 checkpoint): adherent solve rate and routes found per target for the unguided generator, the filtered baseline NeuralSym, and CAG (full comparison in Table 3).

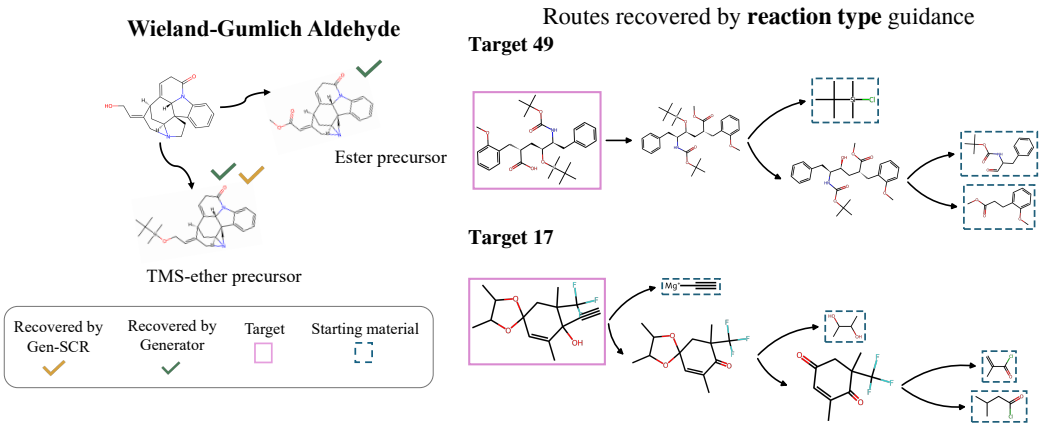
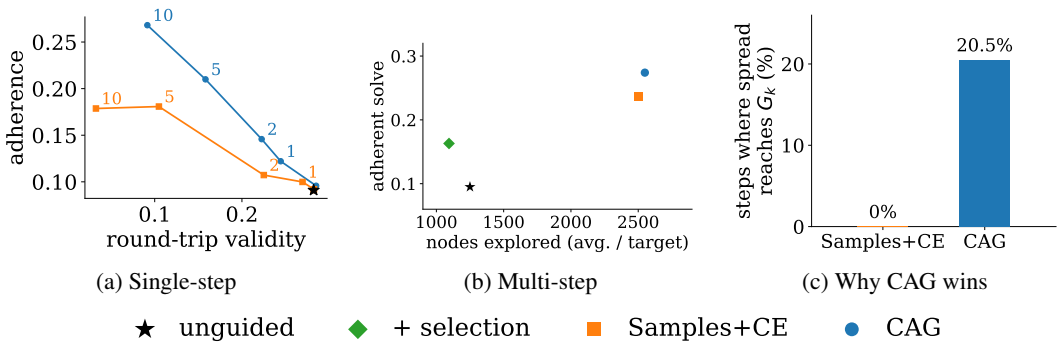


Figure 4: Case studies on USPTO-190. **(a)** Wieland-Gumlich Aldehyde admits two retrosynthetic disconnections: reduction to an ester precursor (recovered by the unguided generator) and deprotection to a TMS-ether precursor (recovered *only* by CAG with rxntype guidance). **(b)** Routes recovered by rxntype guidance. For Target 49, all eight baselines fail to produce any route. For Target 17, no other method returns a route in which every step keeps the atoms of its product.

Experimental setup. We evaluate single-step performance on USPTO-190-Steps (640 deduplicated test products of USPTO-190 (Chen et al., 2020)) and multi-step search on the full 190 USPTO-190 targets. We additionally report results on USPTO-50k (Schneider et al., 2016) in Section E. We use RSMILES (Zhong et al., 2022) as our base generator, and follow the original paper's parameters (augmentation with 20 canonical-SMILES randomizations, beam size=10, decoding with beam search), and use the CAG-trained reaction-type classifier. All our experiments are managed with Syntheseus (Maziarz et al., 2023) for all baselines, also in its default parameters (sampling 100 precursors, deduplicating, and discarding unparsable ones). In addition, we use RXN-Insight (Dobbelaere et al., 2024) to assign classes and reaction names to the sampled precursors, and checked for round-trip accuracy via the forward model of Zhong et al. (2022). Template-based methods are marked $^{\mathcal{T}}$. For single-step evaluation, we guide each product toward every one of the 10 reaction classes other than its ground-truth one and measure achievable class diversity: the share of those classes for which at least one sample is class-correct, matches its RXN-Insight reaction name, or round-trips through the forward model, averaged over products (full protocol in Section E.3). For multi-step search, we guide each step towards its ground-truth reaction class from the reference route, simulating a chemist who specifies the desired disconnection. For a fair comparison with other baselines, we filter their results with the same ground truth label post-hoc in Table 3.



(a) Single-step (b) Multi-step (c) Why CAG wins

Figure 5: Single-step and multi-step trade-offs on USPTO-190, and the mechanism behind them. Panels (a)-(b) share the same color and marker per classifier, given in the legend below. **(a)** Adherence against round-trip validity on USPTO-190-STEPS, one curve per classifier, each point one value of λ. Every curve starts at the unguided generator, higher / further right is better. **(b)** Adherent solve rate against search cost in multi-step search: + *selection* adds class-aware selection with no token-level steering. **(c)** Share of decoding steps at which each classifier's error spread reaches the generator gap G_k, the only steps at which it can change the decoder's choice (Section C.2).

Results. Figure 3 summarizes single-step and multi-step results together. On USPTO-190-STEPS (640 products), guidance reaches a higher achievable class diversity than every baseline, template-based or template-free (Fig. 3b; full comparison against every baseline in Fig. 11). Of the 287 products where the unguided generator fails outright, meaning none of its samples round-trip back to the product, guidance recovers a valid, round-trip-consistent precursor set for 231 of them, and the guided and unguided sample sets overlap by only 39% Jaccard (Fig. 3a; full numbers in Section E.1). In multi-step search on USPTO-190 (Fig. 3c), guidance raises adherent solve from 0.14 (unguided generator) to 0.23 and more than doubles the routes found per target ($1.93 \rightarrow 4.40$), well above the filtered baseline NeuralSym (0.00 adherent solve; full comparison in Table 3). All these results show that guiding at the single-step level indeed unlocks steerable candidates not achievable otherwise.

Case study. Reaction-type guidance recovers ground-truth precursors and complete routes unreachable by any baseline. These include a disconnection of Wieland-Gumlich Aldehyde, a key Strychnine intermediate, that the unguided generator misses entirely, and 3-step routes for two USPTO-190 targets that no baseline solves at all (Fig. 4). Across the full dataset, reaction-type guidance unlocks routes for 9 out of 190 targets (4.7%) that remain inaccessible to all baselines. Steering at the single-step unlocks routes that would otherwise be inaccessible to search-level steering alone, as further discussed in Section E.4.

Next, we look into how much CAG contributes to steering compared to a CE-trained classifier.

4.2 CE VS CAG: ISOLATING THE RANKING LOSS

We now compare two classifiers that guide the same frozen generator and differ only in how they were trained. *Samples+CE* is trained with cross-entropy on on-policy samples and the same contrastive augmentation that CAG uses. *CAG* adds the ranking loss of Eq. (8) on top of that. Comparing CAG to Samples+CE isolates what the ranking loss buys on its own, since the two classifiers are trained on identical data.

Single-step. As in Section 4.1, we guide every product toward every available reaction class, now to compare the two classifiers directly. We use the 640 USPTO-190-STEPS products with all 11 reaction classes, giving 7,040 (product, class) pairs. For each sampled precursor, we compute *adherence* (does its predicted class match the requested one) and *round-trip validity* (does the forward model map it back to the product). We then plot the pareto-front of adherence against validity to see how each classifier trades off the two metrics. Figure 5a plots one curve per classifier, each point a value of λ. CAG sits above Samples+CE along the whole curve: the ranking loss raises adherence at every level of validity the two classifiers can reach. Both classifiers break down at $\lambda = 10$, where

their round-trip validity collapses to 0.033 for CE+samples against 0.092 for CAG, past the usable range of guidance for this generator.

Multi-step. For multi-step synthesis, we put our generator, each time guided with a different classifier, inside Retro* search. We expand each node with different guidance strengths ($\lambda \in \{0, 1, 2, 3, 5\}$), and pool the candidates with class-aware selection at every node. We compute *adherent solve*, the share of targets with at least one route whose every step matches the requested class, with steps that carry no request skipped rather than counted as mismatches. We compare it against search cost (nodes explored per target, averaged over all 190 targets). Class-aware selection alone, with no token-level steering, raises adherent solve from 0.095 to 0.163 while search cost stays flat, in fact falling slightly from 1250 to 1094 nodes. Adding a trained classifier roughly doubles the search cost, to about 2500 nodes, and buys a further rise to 0.237 for Samples+CE or 0.274 for CAG. Between the two guided classifiers the search cost is matched (2503 against 2549 nodes, under 2% apart), so CAG's edge over Samples+CE comes from proposing a better adhering sample pool to expand from.

Why CAG wins. Figure 5c shows the mechanism behind both gaps. Guidance can only change the decoder's choice at a step where the classifier's error spread reaches the generator gap G_k (Section C.1). We estimate this spread by forcing each candidate token onto held-out decoding prefixes and comparing the classifier's reported gap against a rollout-based estimate of the true one, over 198 USPTO-50K test products and 780 measurable steps. The Samples+CE classifier's spread never reaches G_k (0% of steps), while CAG's reaches it at 20.5% of steps, despite the two classifiers having nearly the same median spread (3.7 vs. 3.6 nats). The margin loss does not inflate the spread everywhere; it inflates it at a minority of steps, and those are the only steps at which either classifier can steer at all. Full measurement details are in Section C.2.

5 Discussion and Limitations

CAG incorporates chemist-specified substructural properties (in this case reaction type) at decode time, expanding the reachable set of desired precursors beyond post-hoc filtering. The setup inherently requires property targets specified by a chemist, and in the absence of these defaults to unguided generation.

Other limitations include our evaluation metrics, and generator-dependent training procedure. Our evaluation depends on forward-model round-trip accuracy and template-based reaction-type classifiers, both of which struggle on novel chemistry. Additionally, CAG's ranking objective is itself generator-dependent (the negative anchor r_k^* is defined relative to p_θ), so the classifier has to be retrained whenever the generator is swapped. CAG's benefit is also property-dependent: the argument of Section 3.2 needs the property to be decided by a handful of positions, and on a property that is not (starting-material similarity via Tanimoto), the margin loss gives no measurable gain over plain cross-entropy.

Guidance also adds inference cost: scoring every candidate token with the classifier at each step makes CAG-guided single-step inference roughly 1.08–$1.14\times$ slower than the unguided baseline at the smallest candidate pools we evaluate ($N = 3$–5; Section F.1). Addressing these limitations, tighter coupling with search-level filters, multi-objective guidance, and cost-aware route optimization are natural next steps.

6 Conclusion

We introduced CAG, a training procedure that calibrates token-level classifiers for effective guidance in autoregressive retrosynthesis. Our analysis identifies a fundamental limit of cross-entropy training on retrosynthesis data: for substructural properties like reaction type, closing the gap between an overconfident generator and its classifier calls for more examples of each decisive token combination than any realistic corpus provides. CAG's contrastive augmentation and margin-based ranking loss sidestep this limit directly, calibrating the classifier to discriminate at the positions that decide the property rather than chasing a level of confidence the data cannot support. Empirically, we show that CAG clears the generator's gap in more steps than CE, that it explores a wider pool of the desired property value than other baselines, and that it combines well with search level metrics.

AI USE STATEMENT

In this work, we used generative AI tools for writing and editing assistance (polishing prose and restructuring exposition for clarity), verifying mathematical results (proof discussion and verification), and managing parts of our experiments (including monitoring runs of the baselines with different search-level guidance, and writing and executing scripts for the empirical measurements in Section C.1). We have not used generative AI tools to design the core method, to generate or select experimental results, or to produce scientific claims without author verification. The use of generative AI as an important, original, or non-standard component of the core methods is not applicable to this work: our base generator is a small, domain-specific autoregressive transformer trained on root-aligned SMILES, and the property classifiers are 4-layer transformer encoders (Section D.4), neither of which is a general-purpose LLM. We have reviewed all AI-assisted text and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Retrosynthesis planning tools, including ours, could in principle be used to generate hazardous or controlled chemistry. Our method targets reaction type as a guidance property and does not lower the barrier to synthesizing any specific hazardous compound beyond what unguided retrosynthesis models or the published chemistry literature already provide. All data used (USPTO-derived reaction records, Section D.2) are public and previously published; no human subjects were involved.

REPRODUCIBILITY STATEMENT

Data processing and property-predictor training are described in Sections D.2 to D.4; inference and baseline configurations in Sections D.5 and D.6; and the evaluation pipeline in Section D.7. All datasets (USPTO-50k, USPTO-190, Pistachio) are previously published and cited in Section 4.1. Proofs for the theoretical results are in Sections B and B.3. Code and trained checkpoints will be made available upon acceptance.

REFERENCES

Daniel Armstrong, Zlatko Joncev, Jeff Guo, and Philippe Schwaller. Tango*: Constrained synthesis planning using chemically informed value functions, 2024. URL https://arxiv.org/abs/2412.03424.

Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535, 2024. doi: 10.1038/s42256-024-00832-8.

Dean G. Brown and Jonas Boström. Analysis of past and present synthetic methodologies on medicinal chemistry: Where have all the new reactions gone? *Journal of Medicinal Chemistry*, 59, 2016. doi: 10.1021/acs.jmedchem.5b01409.

Benson Chen, Tianxiao Shen, Tommi S. Jaakkola, and Regina Barzilay. Learning to make generalizable and diverse predictions for retrosynthesis, 2019. URL https://arxiv.org/abs/1910.09688.

Binghong Chen, Chengtao Li, Hanjun Dai, and Le Song. Retro*: learning retrosynthetic planning with neural guided a* search. In *International conference on machine learning*, pp. 1608–1616. PMLR, 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.

Reuben Cohn-Gordon, Noah Goodman, and Christopher Potts. Pragmatically informative image captioning with character-level inference, 2018. URL https://arxiv.org/abs/1804.05417.

David J. C. Constable, Peter J. Dunn, John D. Hayler, Guy R. Humphrey, Johnnie L. Leazer, Jr., Russell J. Linderman, Kurt Lorenz, Julie Manley, Bruce A. Pearlman, Andrew Wells, Aleksey Zaks, and Tony Y. Zhang. Key green chemistry research areas—a perspective from pharmaceutical manufacturers. *Green Chemistry*, 9(5):411–420, 2007. doi: 10.1039/B703488C.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation, 2020. URL https://arxiv.org/abs/1912.02164.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

Maarten R Dobbelaere, István Lengyel, Christian V Stevens, and Kevin M Van Geem. Rxn-INSIGHT: fast chemical reaction analysis using bond-electron matrices. *J. Cheminform.*, 16(1), 2024.

Piotr Gaiński, Michał Koziarski, Krzysztof Maziarz, Marwin Segler, Jacek Tabor, and Marek Śmieja. Retrogfn: Diverse and feasible retrosynthesis using gflownets, 2024. URL https://arxiv.org/abs/2406.18739.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Jeff Guo and Philippe Schwaller. It takes two to tango: Directly optimizing for constrained synthesizability in generative molecular design, 2024. URL https://arxiv.org/abs/2410.11527.

Jiasheng Guo, Chenning Yu, Kenan Li, Yijian Zhang, Guoqiang Wang, Shuhua Li, and Hao Dong. Retrosynthesis zero: Self-improving global synthesis planning using reinforcement learning. *Journal of Chemical Theory and Computation*, 20(11):4921–4938, 2024. doi: 10.1021/acs.jctc.4c00071.

Huarui He, Jie Wang, Yunfei Liu, and Feng Wu. Modeling diverse chemical reactions for single-step retrosynthesis via discrete latent variables. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM)*, pp. 717–726, 2022. doi: 10.1145/3511808.3557397.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. Learning to write with cooperative discriminators, 2018.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

Tomasz Korbak, Ethan Perez, and Christopher L. Buckley. RL with KL penalties is better viewed as Bayesian inference. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 1083–1091, 2022. arXiv:2205.11275.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. Gedi: Generative discriminator guided sequence generation, 2020. URL https://arxiv.org/abs/2009.06367.

Aviral Kumar and Sunita Sarawagi. Calibration of encoder decoder models for neural machine translation, 2019.

Min Htoo Lin, Zhengkai Tu, and Connor W. Coley. Improving the performance of models for one-step retrosynthesis through re-ranking. *Journal of Cheminformatics*, 14(1):15, 2022. doi: 10.1186/s13321-022-00594-8.

Guoqing Liu, Di Xue, Shufang Xie, Yingce Xia, Austin Tripp, Krzysztof Maziarz, Marwin Segler, Tao Qin, Zongzhang Zhang, and Tie-Yan Liu. Retrosynthetic planning with dual value networks. In *Proceedings of the 40 th International Conference on Machine Learning*, 2023.

Yupeng Liu, Han Zhang, and Rui Hu. Molecular retrosynthesis top-k prediction based on the latent generation process. *CAAI Transactions on Intelligence Technology*, 10(3):902–911, 2025. doi: 10.1049/cit2.70005.

Krzysztof Maziarz, Austin Tripp, Guoqing Liu, Megan Stanley, Shufang Xie, Piotr Gaiński, Philipp Seidl, and Marwin Segler. Re-evaluating retrosynthesis algorithms with syntheseus. In *NeurIPS 2023 AI for Science Workshop*, 2023. URL https://openreview.net/forum?id=W5U18rgtpg.

Tao Meng, Sidi Lu, Nanyun Peng, and Kai-Wei Chang. Controllable text generation with neurally-decomposed oracle. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=yI7i9yc3Upr.

Anton Morgunov and Victor S. Batista. Project ariadne: Prompt-conditioned route generation for synthesis planning, 2026. URL https://arxiv.org/abs/2606.24184.

K. C. Nicolaou and E. J. Sorensen. *Classics in Total Synthesis: Targets, Strategies, Methods*. VCH Publishers, Weinheim, Germany, 1996. ISBN 3-527-29231-1.

Stephen D. Roughley and Allan M. Jordan. The medicinal chemist's toolbox: An analysis of reactions used in the pursuit of drug candidates. *Journal of Medicinal Chemistry*, 54(10):3451–3479, 2011. doi: 10.1021/jm200187y.

Mikołaj Sacha, Mikołaj Błaż, Piotr Byrski, Paweł Dąbrowski-Tumański, Mikołaj Chromiński, Rafał Loska, Paweł Włodarczyk-Pruszyński, and Stanisław Jastrzębski. Molecule edit graph attention network: Modeling chemical reactions as sequences of graph edits. *Journal of Chemical Information and Modeling*, 61(7):3273–3284, 2021. doi: 10.1021/acs.jcim.1c00537. PMID: 34251814.

Nadine Schneider, Nikolaus Stiefl, and Gregory A Landrum. What's what: The (nearly) definitive guide to reaction role assignment. *Journal of Chemical Information and Modeling*, 56(12): 2336–2346, 2016.

Philippe Schwaller, Teodoro Laino, Théophile 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.

Marwin H. S. Segler and Mark P. Waller. Neural-symbolic machine learning for retrosynthesis and reaction prediction. *Chemistry – A European Journal*, 23(25):5966–5971, 2017. doi: https://doi.org/10.1002/chem.201605499.

Marwin H. S. Segler, Mike Preuss, and Mark P. Waller. Planning chemical syntheses with deep neural networks and symbolic ai. *Nature*, 555(7698), March 2018. ISSN 1476–4687. doi: 10.1038/nature25978. URL http://dx.doi.org/10.1038/nature25978.

Philipp Seidl, Philipp Renz, Natalia Dyubankova, Paulo Neves, Jonas Verhoeven, Marwin Segler, Jörg K. Wegner, Sepp Hochreiter, and Günter Klambauer. Improving few- and zero-shot reaction template prediction using modern hopfield networks. *Journal of Chemical Information and Modeling*, 62:2111–2120, 2022. doi: 10.1021/acs.jcim.1c01065.

Yu Shee, Haote Li, Anton Morgunov, and Victor Batista. Directmultistep: Direct route generation for multi-step retrosynthesis, 2024. URL https://arxiv.org/abs/2405.13983.

Amol Thakkar, Alain C. Vaucher, Andrea Byekwaso, Philippe Schwaller, Alessandra Toniato, and Teodoro Laino. Unbiasing retrosynthesis language models with disconnection prompts. *ACS Central Science*, 9(7):1488–1498, 2023. doi: 10.1021/acscentsci.3c00372.

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

Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer, 2009.

Huibin Wang, Yueqing Zhang, Zehui Wang, Jiaxi Zhuang, Zixian Cheng, Ying Qian, Aimin Zhou, Sihua Peng, and Xiao He. Enhancing diversity of template-free retrosynthesis prediction via hierarchical latent variables. *Journal of Chemical Information and Modeling*, 66(6):3074–3090, 2026. doi: 10.1021/acs.jcim.5c03128.

Robin Yadav, Qi Yan, Guy Wolf, Avishek Joey Bose, and Renjie Liao. Retro synflow: Discrete flow matching for accurate and diverse single-step retrosynthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2506.04439.

Kevin Yang and Dan Klein. Fudge: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.276.

Kevin Yu, Jihye Roh, Ziang Li, Wenhao Gao, Runzhong Wang, and Connor W. Coley. Double-ended synthesis planning with goal-constrained bidirectional search, 2024. URL https://arxiv.org/abs/2407.06334.

Yemin Yu, Ying Wei, Kun Kuang, Zhengxing Huang, Huaxiu Yao, and Fei Wu. Grasp: Navigating retrosynthetic planning with goal-driven policy. In *36th Conference on Neural Information Processing Systems*, 2022.

Situo Zhang, Hanqi Li, Lu Chen, Zihan Zhao, Xuanze Lin, Zichen Zhu, Danyu Luo, Bo Chen, Xin Chen, and Kai Yu. Retrodfm-r: Reasoning-driven retrosynthesis prediction with large language models via reinforcement learning, 2025. URL https://arxiv.org/abs/2507.17448.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. Probabilistic inference in language models via twisted sequential monte carlo. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. arXiv:2404.17546.

Zipeng Zhong, Jie Song, Zunlei Feng, Tiantao Liu, Lingxiang Jia, Shaolun Yao, Min Wu, Tingjun Hou, and Mingli Song. Root-aligned smiles: a tight representation for chemical reaction prediction. *Chemical Science*, 13(31):9023—-9034, 2022. ISSN 2041-6539. doi: 10.1039/d2sc02763a. URL http://dx.doi.org/10.1039/D2SC02763A.

APPENDICES

A RELATED WORK **15**

B THEORETICAL ANALYSIS **16**

B.1 WHAT GUIDANCE IS IN AN AUTOREGRESSIVE REGIME 16

B.2 WHAT GUIDANCE NEEDS WITH BEAM DECODING 17

B.3 THE SAMPLE COMPLEXITY ARGUMENT THROUGH JUNTAS 19

B.4 CAG: FORCING A GAP THROUGH A MARGIN 20

B.5 HOW THE THEORETICAL RESULTS AND CAG RELATE TO METHODS IN THE LITERATURE 22

C MEASUREMENTS **22**

C.1 WHERE THE DISCRIMINABILITY GAP COMES FROM 22

C.2 THE STEERING EFFECT: SPREAD OF CLASSIFIER ERROR AGAINST THE GENERATOR GAP 26

D PROPERTIES, DATA AND IMPLEMENTATION **27**

D.1 GUIDED PROPERTY 27

D.2 REACTION AND ROUTE DATA 27

D.3 REACTION TYPE PARTIAL-SEQUENCE DATA 28

D.4 PROPERTY PREDICTOR TRAINING 28

D.5 INFERENCE 28

D.6 BASELINES 29

D.7 EVALUATION PIPELINE 29

D.8 COMPUTE RESOURCES 30

E COMPARISON WITH OTHER METHODS **30**

E.1 EFFECT OF GUIDANCE ON THE GENERATOR'S SAMPLES 30

E.2 ACHIEVABLE CLASS DIVERSITY: FULL COMPARISON 31

E.3 SINGLE-STEP RESULTS ON EVERY PAIR OF CLASSES 31

E.4 SEARCH-LEVEL RESULTS 32

F ABLATIONS AND COST **34**

F.1 GUIDANCE SCALE, ONSET LENGTH AND NUMBER OF CANDIDATE TOKENS . . . 34

F.2 MISSPECIFIED GUIDANCE TARGETS 36

A Related work

Steerable single-step synthesis. Several methods either steer directly at the single-step level, or pursue a different aim with an adjacent outcome. Early class-conditional evaluations give the ground-truth reaction class as additional input and let template-based methods pre-filter their template library to it before retrieval, which raises top-k metrics by $\sim 10\%$ over the unconditional task but assumes an oracle class unavailable at deployment. Other guiding methods introduce the guidance signal during training, either through prompt-disconnection or input conditioning (Thakkar et al., 2023; Shee et al., 2024; Morgunov & Batista, 2026), or distribution tilting through RL (Guo & Schwaller, 2024; Zhang et al., 2025). It is worth noting that none of these training-time approaches let a chemist request a property unseen during training, or switch guidance off to recover the unguided generator without retraining: once chosen, the property is baked into the weights.

More recent work seeks to diversify the model's predicted disconnections per product, by introducing latent variables (Wang et al., 2026; He et al., 2022; Chen et al., 2019; Liu et al., 2025), or by sampling w.r.t. a feasibility reward function, thus encouraging sampling beyond exact matches (Gaiński et al., 2024). In principle this could raise the diversity score enough for post-hoc filtering to recover the desired reactant set. The approach has two shortcomings, though: the property being diversified over must take a finite set of values, and there is no unguided baseline to fall back on, which Gaiński et al. (2024) report costs the model in multi-step planning. Our method in contrast can turn on guidance at critical or specified steps only, and benefit from a high-accuracy, low-diversity unguided baseline otherwise.

Lastly, the most similar line of work here is probably inference-time steering, currently occupied by (Yadav et al., 2025). RetroSynFlow proposes two models: RPF, translating from product to reactant directly, and RSF, which passes through synthons. To demonstrate the effect of steering in this single-step model, the authors apply Feynman-Kac steering with Sequential Monte Carlo based resampling, using a forward model as a reward function to increase the chemical plausibility of the predictions, and thus improve top-k performance. While this steering approach supports any reward function computable from a fully generated reactant set, including our reaction-type application, their method essentially redistributes mass within the samples of the unguided generator, and does not change what the generator produces otherwise.

Steerable synthesis planning. Single-step retrosynthesis models span template-based methods (Segler & Waller, 2017; Chen & Jung, 2021; Seidl et al., 2022) and template-free approaches (Schwaller et al., 2019; Zhong et al., 2022; Sacha et al., 2021). Reranking (Lin et al., 2022) and diversity-aware generation (Gaiński et al., 2024) improve candidate quality but remain constrained to the base model's output space. At the multi-step level, search algorithms such as Retro* (Chen et al., 2020), MCTS (Segler et al., 2018), PDVN (Liu et al., 2023), and GRASP (Yu et al., 2022) orchestrate single-step predictions into full routes. Constrained planning methods steer search via filtering (Segler et al., 2018), value function biasing (Armstrong et al., 2024), starting-material-aware expansion (Yu et al., 2024; Tripp et al., 2024), or reinforcement learning (Guo et al., 2024). However, all these search-level interventions operate over candidates already produced by the underlying single-step model, fundamentally limiting exploration to the chemical space the base generator can access. Our work intervenes directly at the token generation level, which is complementary to search-level steering and can expand the reachable chemical space beyond the candidates the base model already produces.

Classifier guidance in autoregressive models. Classifier guidance originated in diffusion models (Dhariwal & Nichol, 2021) and was extended to classifier-free variants (Ho & Salimans, 2022). In the autoregressive setting, FUDGE (Yang & Klein, 2021) applies Bayesian token-level reranking based on future attribute prediction; PPLM (Dathathri et al., 2020) perturbs hidden states via attribute gradients; GeDi (Krause et al., 2020) uses a generative discriminator to contrast class-conditional and unconditional distributions; and NADO (Meng et al., 2022) decomposes oracle constraints into per-token guidance signals. Several works study cooperative or pragmatic decoding (Holtzman et al., 2018; Cohn-Gordon et al., 2018). While these methods demonstrate effective control in natural language settings, none analyze why cross-entropy training itself fails to produce a guidance-ready classifier: for a substructural property, reaching the confidence guidance requires calls for a training corpus with exponentially many examples of each rare token combination at the decisive positions,

a sample complexity no realistic corpus meets (Section 3.2). Applications to chemical synthesis planning also remain largely unexplored. Our work addresses both gaps: we provide a theoretical characterization of this failure mode and a training procedure (CAG) that overcomes it.

B THEORETICAL ANALYSIS

This appendix asks what token-level guidance needs from a classifier, and what cross-entropy training and the CAG margin each supply. Section B.1 writes the guidance rule for an autoregressive generator and reduces it to the completion probability. Section B.2 turns that rule into a condition on a single decoding step, and derives a ceiling on the gap any classifier can report there. Section B.4 shows what the margin changes, and Section B.5 places other guidance methods against the same ceiling. Every measurement these subsections rely on is in Section C.

B.1 WHAT GUIDANCE IS IN AN AUTOREGRESSIVE REGIME

To obtain the classifier guidance formula for an autoregressive model, we start by applying Bayes rule:

$$p_\theta(\mathbf{r} \mid \phi, p) = \frac{p_\theta(\phi \mid \mathbf{r}, p)\, p_\theta(\mathbf{r} \mid p)}{p_\theta(\phi \mid p)} \tag{9}$$

Dhariwal & Nichol (2021) introduce an exponential tilt to increase the effect of guidance:

$$p_\theta^\lambda(\mathbf{r} \mid \phi, p) = \frac{1}{Z_\lambda} \left[p_\theta(\phi \mid \mathbf{r}, p)\right]^\lambda p_\theta(\mathbf{r} \mid p), \qquad Z_\lambda = \sum_{\mathbf{r}' \in \mathcal{V}^L} \left[p_\theta(\phi \mid \mathbf{r}', p)\right]^\lambda p_\theta(\mathbf{r}' \mid p) \tag{10}$$

Now we check how this guidance applies at the token level, since our generator is autoregressive. We define the guided distribution at a single token as:

$$p_\theta^\lambda(r_k \mid \mathbf{r}_{<k}, \phi, p) = \frac{p_\theta^\lambda(\mathbf{r}_{\le k} \mid \phi, p)}{p_\theta^\lambda(\mathbf{r}_{<k} \mid \phi, p)} \tag{11}$$

We know that we can define the guided distribution for a prefix as:

$$p_\theta^\lambda(\mathbf{r}_{\le k} \mid \phi, p) = \sum_{\mathbf{r}_{>k} \in \mathcal{V}^{L-k}} p_\theta^\lambda(\mathbf{r}_{\le k}, \mathbf{r}_{>k} \mid \phi, p) \tag{12}$$

$$= \frac{1}{Z_\lambda} \sum_{\mathbf{r}_{>k} \in \mathcal{V}^{L-k}} \Big[p_\theta(\phi \mid \mathbf{r}_{\le k}, \mathbf{r}_{>k}, p)^\lambda\, p_\theta(\mathbf{r}_{\le k}, \mathbf{r}_{>k} \mid p)\Big] \tag{13}$$

$$= \frac{1}{Z_\lambda} \sum_{\mathbf{r}_{>k} \in \mathcal{V}^{L-k}} \Big[p_\theta(\phi \mid \mathbf{r}_{\le k}, \mathbf{r}_{>k}, p)^\lambda\, p_\theta(\mathbf{r}_{>k} \mid p, \mathbf{r}_{\le k})\, p_\theta(\mathbf{r}_{\le k} \mid p)\Big] \tag{14}$$

$$= \frac{p_\theta(\mathbf{r}_{\le k} \mid p)}{Z_\lambda} \sum_{\mathbf{r}_{>k} \in \mathcal{V}^{L-k}} \Big[p_\theta(\phi \mid \mathbf{r}_{\le k}, \mathbf{r}_{>k}, p)^\lambda\, p_\theta(\mathbf{r}_{>k} \mid p, \mathbf{r}_{\le k})\Big] \tag{15}$$

We call $U_\lambda(\mathbf{r}_{\le k})$ a *completion probability*, the probability of a sequence satisfying the property under the distribution of the generator:

$$U_\lambda(\mathbf{r}_{\le k}) = \sum_{\mathbf{r}_{>k} \in \mathcal{V}^{L-k}} \Big[p_\theta(\phi \mid \mathbf{r}_{\le k}, \mathbf{r}_{>k}, p)^\lambda\, p_\theta(\mathbf{r}_{>k} \mid p, \mathbf{r}_{\le k})\Big] \tag{16}$$

$$= \mathbb{E}_{\mathbf{r}_{>k} \sim p_\theta(\mathbf{r}_{>k} \mid p, \mathbf{r}_{\le k})} \Big[p_\theta(\phi \mid \mathbf{r}_{\le k}, \mathbf{r}_{>k}, p)^\lambda\Big] \tag{17}$$

Replacing $U_\lambda(\mathbf{r}_{\le k})$ in Eq. (12), we get:

$$p_\theta^\lambda(\mathbf{r}_{\le k} \mid \phi, p) = \frac{p_\theta(\mathbf{r}_{\le k} \mid p)}{Z_\lambda}\, U_\lambda(\mathbf{r}_{\le k}) \tag{18}$$

Similarly we define:

$$p_\theta^\lambda(\mathbf{r}_{<k} \mid \phi, p) = \frac{p_\theta(\mathbf{r}_{<k} \mid p)}{Z_\lambda}\, U_\lambda(\mathbf{r}_{<k}) \tag{19}$$

we replace in Eq. (11), and get:

$$p_\theta^\lambda(r_k \mid \mathbf{r}_{<k}, \phi, p) = \frac{p_\theta^\lambda(\mathbf{r}_{\leq k} \mid \phi, p)}{p_\theta^\lambda(\mathbf{r}_{<k} \mid \phi, p)} \tag{20}$$

$$= \frac{\frac{p_\theta(\mathbf{r}_{\leq k} \mid p)}{Z_\lambda} U_\lambda(\mathbf{r}_{\leq k})}{\frac{p_\theta(\mathbf{r}_{<k} \mid p)}{Z_\lambda} U_\lambda(\mathbf{r}_{<k})} \tag{21}$$

$$= \frac{p_\theta(\mathbf{r}_{\leq k} \mid p)}{p_\theta(\mathbf{r}_{<k} \mid p)} \cdot \frac{U_\lambda(\mathbf{r}_{\leq k})}{U_\lambda(\mathbf{r}_{<k})} \tag{22}$$

$$= p_\theta(r_k \mid \mathbf{r}_{<k}, p) \frac{U_\lambda(\mathbf{r}_{\leq k})}{U_\lambda(\mathbf{r}_{<k})} \tag{23}$$

Notice that $U_\lambda(\mathbf{r}_{<k})$ is a constant w.r.t r_k. This finally leads to the following expression for guidance applied at the token level:

$$p_\theta^\lambda(r_k \mid \mathbf{r}_{<k}, \phi, p) \propto p_\theta(r_k \mid \mathbf{r}_{<k}, p)\, U_\lambda(\mathbf{r}_{\leq k}) \tag{24}$$

We note that $U_\lambda(\mathbf{r}_{\leq k})$ can be further simplified when the property of interest is deterministic (which is our case with reaction class):

$$p_\theta(\phi \mid \mathbf{r}, p) = p_\theta(\phi \mid \mathbf{r}, p)^\lambda = \phi(\mathbf{r}, p) \tag{25}$$

Removing λ, we get:

$$U_\lambda(\mathbf{r}_{\leq k}) = \sum_{\mathbf{r}_{>k} \in \mathcal{V}^{L-k}} p_\theta(\mathbf{r}_{>k} \mid p, \mathbf{r}_{\leq k})\, p_\theta(\phi \mid \mathbf{r}_{\leq k}, \mathbf{r}_{>k}, p) \tag{26}$$

$$= \sum_{\mathbf{r}_{>k} \in \mathcal{V}^{L-k}} p_\theta(\phi, \mathbf{r}_{>k} \mid p, \mathbf{r}_{\leq k}) \tag{27}$$

$$= p_\theta(\phi \mid p, \mathbf{r}_{\leq k}) \tag{28}$$

Therefore, when ϕ is deterministic, the guided distribution at the token level is the same irrespective of λ

$$p_\theta^\lambda(r_k \mid \mathbf{r}_{<k}, \phi, p) \propto p_\theta(r_k \mid \mathbf{r}_{<k}, p)\, p_\theta(\phi \mid p, \mathbf{r}_{\leq k}) \tag{29}$$

B.2 What guidance needs with beam decoding

Now we consider what needs to be satisfied for guidance to influence the generator. We define r^* as the generator's favorite next token, and a as the next token value favoring the property. We start by considering the log probability version of the guidance equation for a random token r_k:

$$\log p_\theta^\lambda(r_k \mid \mathbf{r}_{<k}, \phi, p) = \log p_\theta(r_k \mid \mathbf{r}_{<k}, p) + \log U_\lambda(\mathbf{r}_{\leq k}) + C, \qquad C = -\log U_\lambda(\mathbf{r}_{<k}) \tag{30}$$

where C does not depend on r_k, so it cancels in every comparison between two candidate values of r_k at the same prefix.

Since our decoding uses beam search, we compare the cumulative score of the generator's favorite sequence, to that of a sequence satisfying the property. Writing $a_{\leq l} = (a_1, \ldots, a_l)$ for a property-satisfying sequence and summing Eq. (30) over positions, we define the cumulative score as:

$$S_\lambda(a_{\leq l}) = \sum_{j=1}^{l} \Big[\log p_\theta(a_j \mid a_{<j}, p) + \log U_\lambda(a_{\leq j}) - \log U_\lambda(a_{<j}) \Big] \tag{31}$$

We note that the p_θ terms have a closed form, since the generator is autoregressive:

$$\sum_{j=1}^{l} \log p_\theta(a_j \mid a_{<j}, p) = \log p_\theta(a_{\leq l} \mid p) \tag{32}$$

while the U_λ terms telescope, since each step subtracts the quantity the previous step added:

$$\sum_{j=1}^{l} \Big[\log U_\lambda(a_{\leq j}) - \log U_\lambda(a_{<j}) \Big] = \Big[\log U_\lambda(a_{\leq 1}) - \log U_\lambda(\varnothing) \Big] \tag{33}$$

$$+ \Big[\log U_\lambda(a_{\leq 2}) - \log U_\lambda(a_{\leq 1}) \Big] + \cdots \tag{34}$$

$$= \log U_\lambda(a_{\leq l}) - \log U_\lambda(\varnothing) \tag{35}$$

with $\varnothing$ the empty prefix, so that $U_\lambda(\varnothing) = \mathbb{E}_{\mathbf{r}\sim p_\theta(\cdot|p)}[p_\theta(\phi \mid \mathbf{r}, p)^\lambda]$ is the same constant for every sequence generated for the product p. This leads to the following simplified expression for the cumulative log probability score of a sequence:

$$S_\lambda(a_{\leq l}) = \log p_\theta(a_{\leq l} \mid p) + \log U_\lambda(a_{\leq l}) - \log U_\lambda(\varnothing) \tag{36}$$

We notice two things. First, adding the classifier term at every step is not an extra accumulation of guidance: the intermediate U_λ terms cancel, and only the value at the current prefix survives. Second, the cumulative score of a hypothesis is exactly its unguided beam score plus a single guidance term $\log U_\lambda(a_{\leq l})$, up to a constant shared by all hypotheses.

Next we study the condition through which a property satisfying sequence stays in the beam. Let $\mathcal{H}_l$ be the set of hypotheses held by the beam at step l, and let

$$M_l = \max_{\mathbf{r}\in\mathcal{H}_l} S_\lambda(\mathbf{r}_{\leq l}) \tag{37}$$

be the leading hypothesis's cumulative score at step l, and B^l_ω the ω-th largest score in $\mathcal{H}_l$, i.e. the cheapest score the beam of width ω still keeps. We define the deficit to the leader as

$$D_l = M_l - S_\lambda(a_{\leq l}) \tag{38}$$

and the beam score spread as

$$\sigma_l = M_l - B^l_\omega \tag{39}$$

A property satisfying sequence survives in the beam if:

$$D_l \leq \sigma_l \tag{40}$$

Note that while D_l accumulates during generation, σ_l becomes cheaper as later updates are easier. Concretely, as long as the leader at step l is the continuation of the leader at step $l-1$, the deficit grows by exactly the per-step shortfall, $D_l - D_{l-1} = G_l - \Delta_l$, with G_l and Δ_l defined below. When guidance is applied as a weighted classifier term λ, the increment becomes $G_l - \lambda\Delta_l$. The spread σ_l is not an accumulation: it is an order statistic, fixed at each step by how cheap the cheapest few alternatives on the beam happen to be. This means that if guidance does not overcome the deficit in enough single steps, it is unlikely to overpower the generator and include a property-satisfying sequence in the final beam. Figure 6 works through this on a concrete example under CAG guidance.

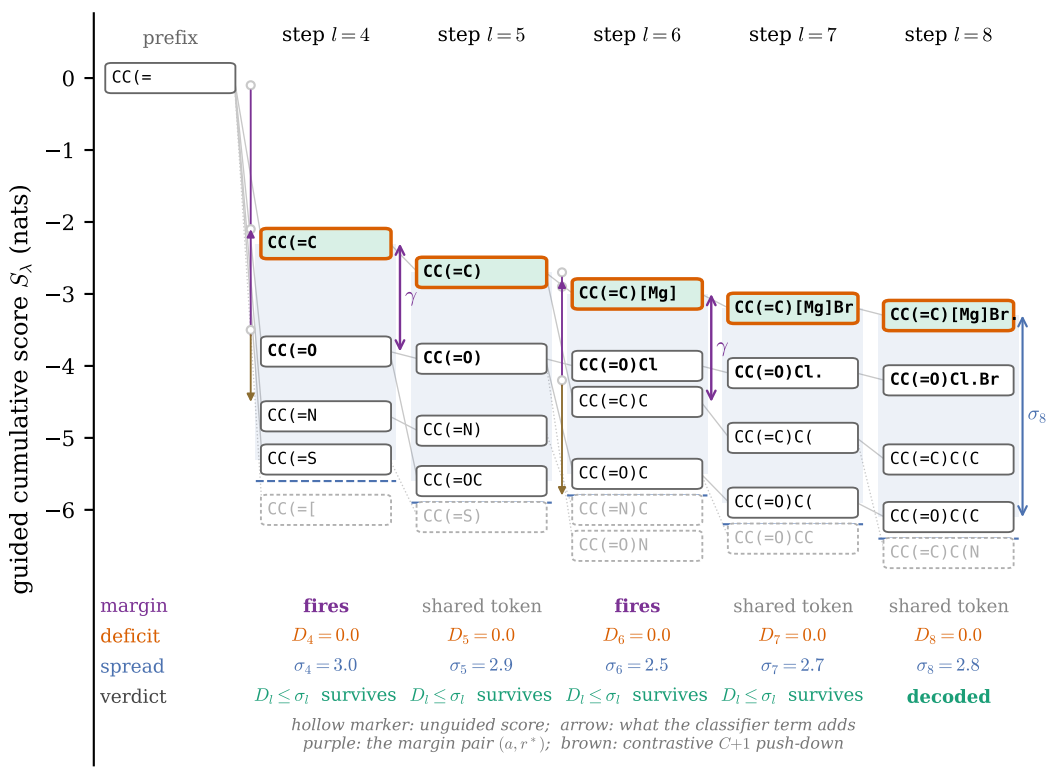


Figure 6: **The beam-survival condition on a worked beam under CAG guidance ($\lambda = 1$).** On the acetophenone example of Fig. 1: the generator favours the acyl-chloride path `CC(=O)Cl.Brc1ccccc1`, the requested property (C–C coupling) needs the Grignard path `CC(=C)[Mg]Br.Brc1ccccc1`. Scores are synthetic. Each column is one generation step; every box is a hypothesis placed at its cumulative guided score $\log p_\theta + \lambda \log p_\alpha$, so vertical distances are the deficit D_l (orange box to green box) and the spread σ_l (orange box to the dashed cutoff B^l_ω, beam width $\omega = 4$). Grey dashed boxes were dropped at that step. The classifier term does not accumulate (Eq. (33)), so each hypothesis is its unguided score plus its current log-score. At the two steps where the C–C coupling path diverges from the generator's favourite (`C` vs `O` at step 4, `[Mg]` vs `C` at step 6) the ranking term of Eq. (8) has installed a gap of at least γ between the guided scores of the two tokens (purple bracket), so the property token takes the lead by γ instead of trailing by G_l, and the deficit D_l is zero from then on. Hollow markers show where the two contested boxes and one off-target sibling would sit unguided; the arrows are what the classifier term adds. The contrastive $C{+}1$ samples (brown) push the off-target siblings that filled the unguided beam below the cutoff, so the spread σ_l is now measured among on-target continuations. At the shared-token steps (5, 7, 8) nothing changes, and the C–C coupling path is decoded.

B.3 The sample complexity argument through juntas

This appendix proves Proposition 1: any estimator of an s-junta posterior needs $n(x) \geq c \log(1/\delta)/\sigma^2$ training examples landing exactly at the decisive combination x, regardless of how the estimator is built.

Proof. Fix a combination $x \in \mathcal{V}^{|I(p)|}$ and construct two members of $\mathcal{F}_s$, U_0 and U_1, that agree everywhere except at x: $U_0(x) = \frac{1}{2} - \sigma$, $U_1(x) = \frac{1}{2} + \sigma$, and $U_0(x') = U_1(x')$ for every $x' \neq x$. Membership in $\mathcal{F}_s$ places no constraint linking $U(x)$ to $U(x')$, so this pair is a valid instance of the family for every $\sigma \in (0, \frac{1}{2})$: the two worlds are as hard to tell apart as possible, since they disagree only at x, and there only by 2σ.

A training row whose decisive tokens equal some $x' \neq x$ is generated from $U_0(x') = U_1(x')$: the two worlds produce that row with exactly the same distribution, so observing it tells us nothing

about which world we are in. Only the $n(x)$ rows landing at x can be informative. Each one is an independent draw from $\mathrm{Bernoulli}(\frac{1}{2}-\sigma)$ under U_0 or $\mathrm{Bernoulli}(\frac{1}{2}+\sigma)$ under U_1.

Suppose $\hat{U}$ satisfies $|\hat{U}(x)-U(x)|\le\sigma$ with probability at least $1-\delta$, whichever world is true. Turn it into a test of U_0 against U_1 by thresholding at $\frac{1}{2}$: guess U_0 if $\hat{U}(x)\le\frac{1}{2}$, guess U_1 otherwise. Since $U_0(x)=\frac{1}{2}-\sigma$ and the estimate is within σ of it, under U_0 the estimate is at most $\frac{1}{2}$ whenever the accuracy guarantee holds, so the test guesses correctly; symmetrically it guesses correctly under U_1. So this test is wrong with probability at most δ under either world, exactly the estimator's own failure probability.

Write $P_0^{n(x)}$ and $P_1^{n(x)}$ for the joint distributions of the $n(x)$ informative rows under U_0 and U_1. The sum of the two error types of *any* test is lower-bounded by the Bretagnolle–Huber inequality (Tsybakov, 2009),

$$P_0^{n(x)}(\hat{U}(x)>\tfrac{1}{2})+P_1^{n(x)}(\hat{U}(x)\le\tfrac{1}{2})\;\ge\;\tfrac{1}{2}\exp\big(-\mathrm{KL}(P_0^{n(x)}\,\|\,P_1^{n(x)})\big). \tag{41}$$

Applying this to the thresholding test just constructed, whose two error probabilities are each at most δ, the left side is at most 2δ.

The $n(x)$ rows are i.i.d., so the KL divergence tensorizes, $\mathrm{KL}(P_0^{n(x)}\|P_1^{n(x)})=n(x)\,\mathrm{KL}(\mathrm{Ber}(\frac{1}{2}-\sigma)\,\|\,\mathrm{Ber}(\frac{1}{2}+\sigma))$, and for $\sigma\in(0,\frac{1}{4}]$ this per-example divergence is bounded by $c'\sigma^2$ for a universal constant c' (a second-order Taylor expansion of the Bernoulli KL around $p=\frac{1}{2}$). Combining the two bounds,

$$2\delta\;\ge\;\tfrac{1}{2}\exp\big(-n(x)\,c'\sigma^2\big)\quad\Longrightarrow\quad n(x)\;\ge\;\frac{\log(1/(4\delta))}{c'\sigma^2}. \tag{42}$$

For $\delta\le\frac{1}{4}$, $\log(1/(4\delta))\ge c''\log(1/\delta)$ for a universal constant $c''>0$, which gives Eq. (6) with $c=c''/c'$. Since the argument only used $\hat{U}$ through its accuracy guarantee, it applies to any estimator of U, cross-entropy-trained or otherwise. □

Remark 1 (Concrete constant). *For $\sigma\in(0,\frac{1}{4}]$, $\mathrm{KL}(\mathrm{Ber}(\frac{1}{2}-\sigma)\,\|\,\mathrm{Ber}(\frac{1}{2}+\sigma))\le 9\sigma^2$ (the ratio rises monotonically from 8 as $\sigma\to 0$ to 8.78 at $\sigma=\frac{1}{4}$). Using this bound directly in Eq.* (41)*, without the further loosening that gives Eq.* (6) *its unspecified c, yields the explicit*

$$n(x)\;\ge\;\frac{\log(1/(4\delta))}{9\sigma^2}. \tag{43}$$

At $\sigma=0.1$, $\delta=0.05$, this gives $n(x)\ge 18$.

B.4 CAG: FORCING A GAP THROUGH A MARGIN

This appendix makes precise the two claims Section 3.3 states about Eq. (8): what its minimizer is, and how far a classifier that departs from U is allowed to move the decoder.

What the loss minimizes. Fix a training prefix $\mathbf{r}_{<k}$ and hold every other input's contribution to Eq. (8) fixed; the two terms of the loss interact only through the pair of candidate tokens (a_k, r_k^*) that the ranking term compares there. Write $u_a=\log p_\alpha(\phi\mid\mathbf{r}_{<k},a_k,p)$ and $u_r=\log p_\alpha(\phi\mid\mathbf{r}_{<k},r_k^*,p)$ for the two log-scores p_α assigns, and $u_a^\star=\log U(\mathbf{r}_{<k},a_k)$, $u_r^\star=\log U(\mathbf{r}_{<k},r_k^*)$ for the values the completion probability itself takes there. Using G_k from Eq. (5), the local objective is

$$f(u_a,u_r)=\mathrm{CE}_a(u_a)+\mathrm{CE}_r(u_r)+\mu\max\big(0,\;\gamma+G_k+u_r-u_a\big), \tag{44}$$

where CE_a and CE_r are the (convex) cross-entropy losses at the two inputs, each minimized at $u_a^\star$ and $u_r^\star$ respectively since cross-entropy is a proper scoring rule.

Proposition 2 (Pointwise minimizer of the CAG loss). *f is convex in (u_a,u_r). If $u_a^\star-u_r^\star\ge\gamma+G_k$, i.e. the margin already holds under U, the minimizer of f is $(u_a^\star,u_r^\star)$ and the hinge is 0 there. Otherwise the minimizer lies on the boundary $u_a-u_r=\gamma+G_k$, at the point on that boundary closest to $(u_a^\star,u_r^\star)$ in the geometry induced by CE_a and CE_r's curvature.*

Proof. f is a sum of two convex functions of one variable each and a convex, piecewise-linear, non-decreasing function of u_r-u_a, hence convex.

If $u_a^\star - u_r^\star \geq \gamma + G_k$, the margin already holds under U: the hinge is 0 at $(u_a^\star, u_r^\star)$ and stays 0 on a neighborhood of it in that half-space. Since $(u_a^\star, u_r^\star)$ is the unconstrained minimizer of the first two terms of f, it is also a minimizer of f itself.

Otherwise $(u_a^\star, u_r^\star)$ lies strictly outside the half-space

$$\{(u_a, u_r) : u_a - u_r \geq \gamma + G_k\}, \tag{45}$$

where f is strictly increasing as (u_a, u_r) moves further from it, since both terms are minimized inside the half-space and the hinge is 0 only there. The minimizer of f over the whole plane therefore coincides with the minimizer of $\mathrm{CE}_a(u_a) + \mathrm{CE}_r(u_r)$ restricted to the boundary

$$u_a - u_r = \gamma + G_k. \tag{46}$$

The first-order condition of this restricted problem is

$$\mathrm{CE}_a'(u_a) = \nu, \qquad \mathrm{CE}_r'(u_r) = -\nu, \tag{47}$$

for the constraint's Lagrange multiplier $\nu \geq 0$. This is exactly the projection of $(u_a^\star, u_r^\star)$ onto the boundary under the metric

$$\mathrm{CE}_a''\, du_a^2 + \mathrm{CE}_r''\, du_r^2. \tag{48}$$

□

In words, the classifier that minimizes Eq. (8) equals the completion probability everywhere the margin already holds under U, and departs from it only at the pairs $(\mathbf{r}_{<k}, a_k, r_k^*)$ where Eq. (5) would otherwise fail, by exactly enough to reach the margin and no further. These are precisely the pairs Section B.3 shows a classifier cannot pin down to the accuracy Eq. (5) demands from a realistic sample size.

How far the departure travels. The previous paragraph bounds p_α against U only at the training pairs the margin touches; the decoder, however, sees p_α at every candidate token of every prefix it visits, most of which the margin never constrains. The next lemma bounds how much a bounded, direction-free change to a classifier's log-scores can move the decoder's next-token distribution, regardless of where that change comes from.

Lemma 1 (Guided-distribution deviation from a bounded log-score shift)**.** *Let* $q(r) \propto p_\theta(r \mid \mathbf{r}_{<k}, p) \exp(\lambda \log U(\mathbf{r}_{<k}, r))$ *and* $q'(r) \propto p_\theta(r \mid \mathbf{r}_{<k}, p) \exp(\lambda \log p_\alpha(\phi \mid \mathbf{r}_{<k}, r, p))$ *be the next-token distributions guided by the ideal classifier and by* p_α*, and write* $\delta(\mathbf{r}_{<k}, r) = \log p_\alpha(\phi \mid \mathbf{r}_{<k}, r, p) - \log U(\mathbf{r}_{<k}, r)$ *for the classifier's error. Then for every candidate token* $r \in \mathcal{V}$*,*

$$\Big|\log q'(r) - \log q(r)\Big| \;\leq\; \lambda\Big(\max_{v\in\mathcal{V}} \delta(\mathbf{r}_{<k}, v) - \min_{v\in\mathcal{V}} \delta(\mathbf{r}_{<k}, v)\Big) \;=\; \lambda \operatorname{osc}_k(\mathbf{r}_{<k}), \tag{49}$$

matching Eq. (62)*. Summing over decoding positions, the guided sequence distributions satisfy*

$$\Big|\log q'(\mathbf{r} \mid p) - \log q(\mathbf{r} \mid p)\Big| \;\leq\; \lambda \sum_{k=1}^{L} \operatorname{osc}_k(\mathbf{r}_{<k}), \tag{50}$$

matching Eq. (63)*.*

Proof. Write $g(r) = \lambda\, \delta(\mathbf{r}_{<k}, r)$. Substituting the two unnormalized densities gives

$$q'(r) = \frac{q(r) \exp(g(r))}{\mathbb{E}_{v\sim q}[\exp(g(v))]}. \tag{51}$$

Taking logs,

$$\log q'(r) - \log q(r) = g(r) - \log \mathbb{E}_{v\sim q}[\exp(g(v))]. \tag{52}$$

The second term of Eq. (52) is the log of a q-weighted average of $\exp(g(v))$ over $v \in \mathcal{V}$, a quantity sandwiched between $\exp(\min_v g(v))$ and $\exp(\max_v g(v))$, so it lies in $[\min_v g(v), \max_v g(v)]$. Hence

$$\log q'(r) - \log q(r) \;\in\; \big[\, g(r) - \max_v g(v),\; g(r) - \min_v g(v) \,\big], \tag{53}$$

an interval of width $\max_v g(v) - \min_v g(v) = \lambda \operatorname{osc}_k(\mathbf{r}_{<k})$ containing 0 (attained at r achieving either extreme), which gives Eq. (49).

Since the generator is autoregressive,

$$\log q(\mathbf{r} \mid p) = \sum_{k=1}^{L} \log q(r_k \mid \mathbf{r}_{<k}), \tag{54}$$

and likewise for q', so the sequence-level difference is a sum of L per-step differences, each bounded by Eq. (49); the triangle inequality gives Eq. (50). □

Lemma 1 does not depend on where δ comes from: it applies whether the classifier is cross-entropy-trained, CAG-trained, or hand-specified, and whether δ is large only at the s-junta positions Section B.3 identifies or spread evenly across the sequence. Raising γ in Eq. (8) can only widen osc_k at the training pairs it touches, so it trades a larger guaranteed deviation from the ideal guided sequence distribution for a better chance of clearing Eq. (5) at the steps that decide the property. Section C.2 measures osc_k directly for both a cross-entropy and a CAG classifier and reports how often it reaches G_k.

B.5 How the theoretical results and CAG relate to methods in the literature

Section 3.3 observes that classifier-free guidance, twisted sequential Monte Carlo, and KL-regularized fine-tuning all target the same completion probability U_λ that Section B.1 derives, by different routes. Of the three, only classifier-free guidance needs new algebra to see this; the other two reduce to U_λ by the same marginalization Section B.1 already carries out for the tilted target Eq. (1), so Proposition 1's sample-complexity bound binds on all three equally, not only on a classifier trained the way Section 3.3 trains one.

Classifier-free guidance. With an ideal conditional head $p_\theta(r_k \mid \mathbf{r}_{<k}, p, \phi)$ and guidance strength 1, classifier-free guidance (Ho & Salimans, 2022) scores a candidate token with the increment it adds over the unconditional model, $\log p_\theta(r_k \mid \mathbf{r}_{<k}, p, \phi) - \log p_\theta(r_k \mid \mathbf{r}_{<k}, p)$. Applying Bayes' rule to the conditional head,

$$p_\theta(r_k \mid \mathbf{r}_{<k}, p, \phi) = \frac{p_\theta(\phi \mid \mathbf{r}_{\le k}, p)\, p_\theta(r_k \mid \mathbf{r}_{<k}, p)}{p_\theta(\phi \mid \mathbf{r}_{<k}, p)}, \tag{55}$$

so that, using Eq. (3),

$$\begin{aligned} \log p_\theta(r_k \mid \mathbf{r}_{<k}, p, \phi) - \log p_\theta(r_k \mid \mathbf{r}_{<k}, p) &= \log p_\theta(\phi \mid \mathbf{r}_{\le k}, p) - \log p_\theta(\phi \mid \mathbf{r}_{<k}, p) \\ &= \log U(\mathbf{r}_{\le k}) - \log U(\mathbf{r}_{<k}). \end{aligned} \tag{56}$$

The increment classifier-free guidance adds at a candidate token is exactly $\log U(\mathbf{r}_{\le k})$ up to the second term of Eq. (56), which is shared by every candidate at that step and cancels in any comparison between them, matching Eq. (4). Twisted SMC's optimal twist and the KL-regularized optimum are both, by construction, the same expectation Eq. (16) already names U_λ; CAG does not avoid this shared cost, it only asks its estimator to be accurate at the pair (a, r^*) the decoder compares at each step (Proposition 2) rather than everywhere.

C Measurements

This appendix holds the measurements behind Section 3. Section C.1 measures the generator gap G_k, the largest gap any classifier could report against it, and whether pooling nearby cells can rescue a classifier's performance. Section C.2 measures the steering effect of the two classifiers.

C.1 Where the discriminability gap comes from

In this section, we show empirically the theoretical observations and analysis done in Section 3. We start by asking whether the true completion probability is ever certain enough to close a gap the generator has committed to. Then we show how, even when writing the corpus in terms of

grouped reaction centers (i.e. the expected s-coordinates of the true junta posterior), both classifiers barely correlate with the class ranking induced by a simple counting estimator. Finally, we show that the classifiers' pooling capacity does not rescue their performance, by showing that learning from neighboring groups of reaction center leads to a worse performance than seeing samples from the right group.

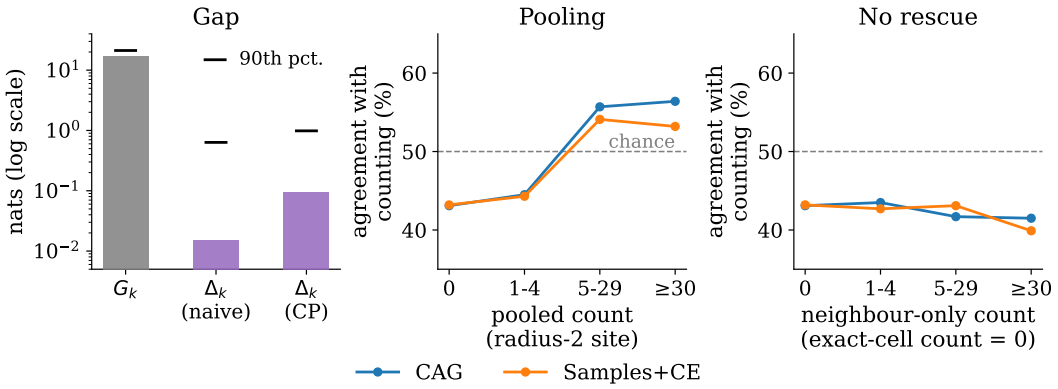


Figure 7: Three views of the discriminability shortfall (Section C.1). **Gap**: generator commitment G_k vs. the ceiling on Δ_k any classifier could report, three orders of magnitude apart at the median (bars: median; ticks: 90th percentile; log scale). **Pooling**: agreement between classifier and corpus-counting rankings, bucketed by pooled count under a radius-two chemical site; rises with count but stays near chance. **No rescue**: the same agreement measurement restricted to positions with zero exact-cell count and rebucketed by neighbour count alone; the rise does not survive isolating the exact-cell confound.

What we measure The step condition of Eq. (5) asks the classifier to report a gap Δ_k larger than the generator gap G_k. Since $U_\lambda \leq 1$, the gap any classifier could report is at most

$$\Delta_k \;=\; \log U_\lambda(\mathbf{r}_{<k}, a) - \log U_\lambda(\mathbf{r}_{<k}, r^*) \;\leq\; -\log U_\lambda(\mathbf{r}_{<k}, r^*), \tag{57}$$

which we call the *ceiling*. We estimate this ceiling at each decoding step. Since U_λ is an expectation over all possible completions of a sequence under the generator, it cannot be computed exactly. We instead estimate it using its definition: at a fixed prefix, finish the sequence with tokens sampled from the distribution of the generator, then evaluate every finished sequence and count the fraction that satisfies the property. This gives an estimate of the probability of satisfying the property under the generator's distribution. Formally, we draw N completions independently from the generator conditioned on the fixed prefix,

$$\mathbf{r}_{>k}^{(1)}, \ldots, \mathbf{r}_{>k}^{(N)} \overset{\text{iid}}{\sim} p_\theta(\mathbf{r}_{>k} \mid p, \mathbf{r}_{\leq k}), \tag{58}$$

and, since ϕ is deterministic and therefore binary (Eq. (25)), estimate U_λ by the fraction of them that satisfy the property:

$$\widehat{U}_\lambda(\mathbf{r}_{\leq k}) \;=\; \frac{1}{N} \sum_{i=1}^{N} \phi\big(\mathbf{r}_{\leq k}, \mathbf{r}_{>k}^{(i)}, p\big). \tag{59}$$

This is a sample proportion, so it is unbiased for the quantity in Eq. (16) and its sampling error follows the binomial rate:

$$\mathbb{E}\big[\widehat{U}_\lambda(\mathbf{r}_{\leq k})\big] = U_\lambda(\mathbf{r}_{\leq k}), \qquad \mathrm{sd}\big(\widehat{U}_\lambda(\mathbf{r}_{\leq k})\big) = \sqrt{\frac{U_\lambda(1 - U_\lambda)}{N}}. \tag{60}$$

Note that Monte Carlo estimation works at high probability quantities because we need on the order of $1/U_\lambda$ rollouts before a single completion satisfies the property. A fixed budget therefore resolves large values of U_λ cheaply, and leaves small ones indistinguishable from zero. Figure 8 summarises the procedure.

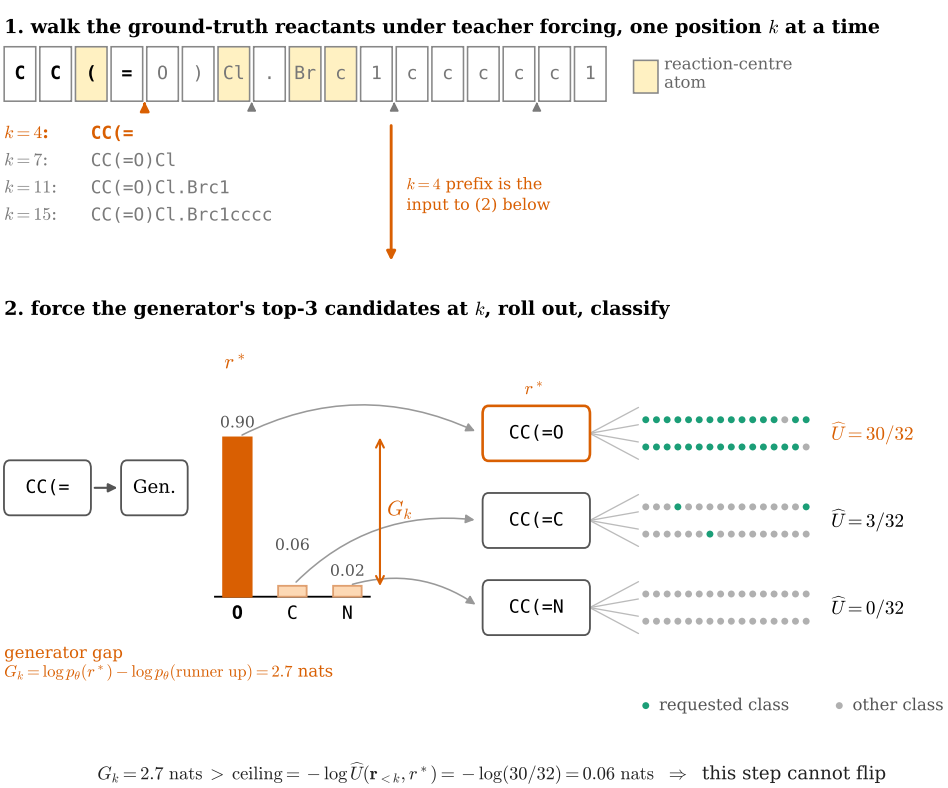


Figure 8: **Estimating the completion probability by rollouts.** Numbers are illustrative. (1) Positions are the ground-truth reactant tokens under teacher forcing; the shaded tokens sit on reaction-centre atoms; the prefix at a few example positions is listed out explicitly, and the highlighted one is the input to (2). (2) At position k the candidate set is the generator's top-3 next tokens, with r^* its favourite and generator gap G_k the log-probability gap to the runner-up; each candidate is forced onto the prefix, N completions are sampled from the generator, and every completion is classified once, with $\widehat{U}$ the fraction carrying the requested class (Eq. (59)). No classifier enters the measurement. The closing line compares G_k to the ceiling $-\log\widehat{U}(\mathbf{r}_{<k},r^*)$ of Eq. (57) directly: a step can only flip if a gap larger than G_k fits under the ceiling (Eq. (5)), which it does not here.

We compare the ceiling quantity to the score spread within the beam, i.e. how many nats it would take to go from the lowest ranked token to the highest. The idea is to see whether the best case gap in completion probability, which is set by how badly the generator's own favorite token fails the property, can ever be large enough to pull a candidate token right outside the beam back into it.

Experimental setup We run the measurement on the USPTO-50k test split, over the first 200 reactions, of which 197 yield a usable profile. The generator is the RSMILES product-to-reactant model trained on USPTO-50k. Positions are the ground-truth reactant tokens under teacher forcing, taken at every position up to a length of 120, which is the path along which the atom mapping and therefore the reaction centre are defined; each position records whether it sits on a reaction-centre atom. This gives 9,678 measured positions. We compare the gaps using two different estimations of the true completion probability **The ceiling bound** $\Delta_k \leq -\log U_\lambda(r_{<k}, r^*)$ (reported in Figure 9a) holds regardless of which token it is compared against, so it only needs the generator's own top-choice token r^*: we force r^* onto the prefix and draw 32 completions from there. This tells us whether a position has any room for guidance at all, before asking whether a real alternative token can occupy it. In **the realised gap** (used to create Figure 9b,c), we separately force each of the generator's top 3 next tokens (of which r^* is one) onto the prefix, again with 32 completions per forced token, and take

whichever of the other two beats r^* by the most. This gives the discriminability an actual candidate it could supply, which is what decides whether a step can flip in practice and where that ability concentrates.

Smoothing the result The ceiling $-\log \widehat{U}_\lambda(\mathbf{r}_{<k}, r^*)$ is built from only 32 completions per position, so we need to report it with care. First, the raw estimate $\widehat{U}_\lambda = c/32$, where c counts how many of the 32 completions reach the target class, is exactly 0 whenever none of them do, which makes $-\log \widehat{U}_\lambda$ infinite. This is because 32 draws cannot tell a true completion probability of, say, 10^{-2} apart from 10^{-6}, both read as $c = 0$. We smooth with the standard fix for a binomial proportion, a Jeffreys pseudo-count of 0.5 (the $\mathrm{Beta}(0.5, 0.5)$ reference prior), giving $\widehat{U}_\lambda = (c + 0.5)/(32 + 1)$. Second, because that floor is an artifact of the pseudo-count and the fixed budget of 32 rather than a statistically calibrated bound, we additionally report the ceiling under a one-sided 95% Clopper–Pearson lower confidence limit on the completion probability at r^*. The Clopper–Pearson limit asks a different question: what is the smallest completion probability still consistent with the observed count at 32 draws, at 95% confidence? Substituting this smaller, exact-coverage probability for $\widehat{U}_\lambda$ gives a strictly more conservative (larger) ceiling. We report both because they answer different questions: Jeffreys gives the best single guess at the ceiling from the experiment as run, while Clopper–Pearson asks how bad the ceiling could plausibly be given only 32 draws, and shows that the comparison in Section C.1 survives even that more cautious reading.

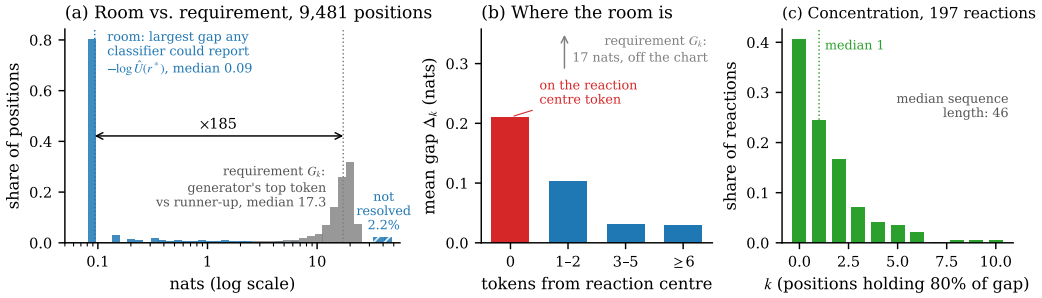


Figure 9: The per-step condition fails by two orders of magnitude, and clearing the generator's gap happens at the reaction center. Measured over 9,481 teacher-forced positions of 197 USPTO-50k test reactions, excluding each sequence's final end-of-sequence position. **(a)** Both sides of the condition on one log axis. In blue, the ceiling $-\log \widehat{U}_\lambda(\mathbf{r}_{<k}, r^*)$ of Eq. (57) under the one-sided 95% Clopper–Pearson limit, the largest token discriminability any classifier could report at that position: median 0.09 nats. In grey, the generator gap G_k between its top token and the runner-up, the smallest gap a competing token has to close: median 17.3 nats. The hatched bar counts the 2.2% of positions whose preferred token reached the target class in none of the 32 rollouts, where the ceiling does not resolve at this budget. **(b)** Mean realised discriminability Δ_k by distance in tokens to the nearest reaction-centre token. The gap is five times larger on the centre token than three or more tokens away, and every bar is two orders of magnitude below the requirement. **(c)** Per reaction, the smallest number of positions k whose gap mass covers 80% of that reaction's total positive gap: median 1, against a median sequence length of 46 positions.

Gap results Figure 9 shows the two numbers we are comparing (the generator's gap and the true completion probability ceiling) side by side. The generator gap has median 17.35 nats and 90th percentile 21.17. The ceiling has median 0.015 nats, 90th percentile 0.38. Under the Clopper–Pearson version the median is 0.094 and the 90th percentile 0.64. The realised discriminability Δ_k, the difference in log completion probability between the best candidate and the generator's own top token, is zero at the median and at the 90th percentile, with a maximum of 4.17. In other words, only 244 positions (2.6%) could clear the generator gap. The other result we have shows that the gap exists in very few positions (Figure 9c). Writing m for the smallest number of positions that together hold 80% of a reaction's total positive gap, m has median 1 and ranges from 0 to 10, against a median sequence length of 46 positions, and gaps are about five times as large on reaction-centre positions as elsewhere, mean 0.21 against 0.04 nats. The property is decided at a few chemically meaningful

tokens, as the localisation argument requires; what the measurement adds is that being decisive is not the same as being decisive enough. This shortfall is not a limit on any particular classifier's skill: no classifier, however well trained, could report a gap large enough to overturn the generator's own preference at these positions, because that confidence is not present in the completion probability it would be reading off.

Pooling experimental setup The natural rescue for this shortfall is to treat chemically similar corpus rows as extra evidence: even where the exact (product, prefix, class) cell is empty, pool the count over every cell sharing a two-bond neighbourhood around the reaction centre, and let the classifier's ranking draw on that pooled count instead. We build 'cells' of (product, prefix, class) units from a corpus mixing generator samples and ground-truth reactions, pool the exact-cell count over every cell sharing a radius-two chemical site around the reaction centre, then compare each classifier's ranking correlation with this pooled count, both overall and restricted to cells with zero exact-cell coverage, to isolate what pooling alone can supply. We use $29{,}196$ training products ($12{,}217$ site groups), sampling $6{,}000$ products of which $1{,}112$ show an extreme class-coverage contrast, giving $3{,}423$ positions.

Pooling results Agreement with a counting oracle does rise as the pooled count grows (Pooling), but pooled count and exact-cell count are correlated, so part of that rise could come from positions that also happen to carry their own exact-cell support rather than from the neighbours themselves. Restricting to positions with zero exact-cell support and rebucketing by neighbour count alone removes this confound, and the rise disappears entirely (No rescue): both classifiers are flat to slightly declining as pooled evidence grows. So chemically similar rows do not substitute for direct evidence at the exact cell; the small gain pooling appeared to offer came from positions that already had exact-cell coverage, not from generalizing across neighbours.

C.2 The steering effect: spread of classifier error against the generator gap

What we test By Eq. (5), guidance changes the decoder's choice at a step only when the gap the classifier reports exceeds G_k. A trained classifier does not report the true gap of U; it reports it up to an error. Write the error, in log-probability, as

$$\delta(\mathbf{r}_{\le k}) \;=\; \log p_\alpha(\phi \mid \mathbf{r}_{\le k}, p) - \log U(\mathbf{r}_{\le k}). \tag{61}$$

An error that is the same at every candidate token of a prefix cancels when the decoder compares candidates. Only its spread across candidates, which we call osc_k, reaches the decoder,

$$\mathrm{osc}_k(\mathbf{r}_{<k}) \;=\; \max_{v\in\mathcal{V}} \delta(\mathbf{r}_{<k}, v) - \min_{v\in\mathcal{V}} \delta(\mathbf{r}_{<k}, v), \tag{62}$$

and Lemma 1 bounds the guided next-token distribution's deviation from the ideal one, q_λ from q_λ^U, by $\lambda\,\mathrm{osc}_k(\mathbf{r}_{<k})$ at every candidate, and, summed over positions, bounds the whole guided sequence distribution's deviation by

$$\left|\log \frac{q_\lambda(\mathbf{r}\mid p)}{q_\lambda^U(\mathbf{r}\mid p)}\right| \;\le\; \lambda \sum_{k=1}^{L} \mathrm{osc}_k(\mathbf{r}_{<k}). \tag{63}$$

By Section C.1 the ideal classifier's own gap is far below G_k at almost every step, so at $\lambda = 1$ a classifier can move the decoder's choice only at steps where osc_k is at least about G_k. This is a necessary condition for steering, not a sufficient one: it does not say that the movement is toward the requested class.

Setup We measure the spread at decoding prefixes, using the rollout machinery of Section C.1 to supply $\widehat{U}$ and hence $\widehat{\delta}$. Forcing each candidate onto the prefix is necessary rather than convenient: sampled rollouts visit about 1.09 distinct first tokens per prefix, because the generator is near-deterministic, so an unforced estimate of the spread is structurally zero. The measurement covers 198 USPTO-50K test products, 990 prefixes and the 780 steps at which the spread is measurable, with the top 5 candidates per step and 40 rollouts per forced candidate. The two classifiers are the Samples+CE and CAG ablations of Section 4.2, both trained on the same data.

Results Table 1 compares G_k with $\widehat{\mathrm{osc}}_k$ for each classifier, in nats, and gives the share of steps at which the spread reaches G_k. The generator gap has mean 22.5 (median 23.2). The cross-entropy classifier's spread has mean 4.07 (maximum 14.19) and reaches G_k at 0.0% of steps. The CAG classifier's has mean 11.89 (median 3.55, 95th percentile 56.4, maximum 93.7) and reaches G_k at 20.5% of steps. The medians are close, so the margin's effect is a heavy tail rather than a uniform inflation: CAG does not spread every step, it spreads some steps far enough to matter.

Table 1: The generator gap G_k against the within-step spread of each classifier's error, in nats, at decoding prefixes. Measured over 198 USPTO-50K test products, 990 prefixes and the 780 steps at which the spread is measurable, with the top 5 candidates per step and 40 rollouts per forced candidate. G_k is the generator's own log-probability gap between its favourite token and the property-favouring one, from Eq. (5). $\widehat{\mathrm{osc}}_k$ is the plug-in version of Eq. (62). The last column is the share of steps whose spread reaches G_k, the only steps at which the classifier can change the decoder's choice.

quantity (nats)	mean	median	95th pct.	max	steps reaching G_k
G_k	22.5	23.2	27.8	31.7	—
$\widehat{\mathrm{osc}}_k$, CE	4.1	3.7	9.1	14.2	0%
$\widehat{\mathrm{osc}}_k$, CAG	11.9	3.6	56.4	93.7	20.5%

D PROPERTIES, DATA AND IMPLEMENTATION

We present here the datasets, models, training and inference procedures, baselines, and evaluation pipeline used in this work. For the new datasets introduced, we explain the generation procedure, intended purpose, and key statistics.

D.1 GUIDED PROPERTY

We guide generation toward the reaction type, which controls the disconnection strategy. Chemists often prefer specific reaction mechanisms based on their expertise, laboratory capabilities, and available equipment (Nicolaou & Sorensen, 1996). Early computational work showed that conditioning on reaction type substantially improves retrosynthesis accuracy, with template-based methods gaining up to 11% in top-10 predictions when the reaction class is known a priori (Segler & Waller, 2017). We adopt the 12 superclass taxonomy of Schneider et al. (2016), available directly for USPTO-50k; for USPTO-190, we assign classes using RXN-Insight (Dobbelaere et al., 2024).

D.2 REACTION AND ROUTE DATA

We rely on two publicly available reaction datasets and derive a third for single-step evaluation.

USPTO-50k. The USPTO-50k dataset (Schneider et al., 2016) contains approximately 50,000 reactions extracted from the United States Patent and Trademark Office (USPTO) filings. Each reaction is annotated with one of 10 reaction superclasses (plus a miscellaneous class) defined by Schneider et al. (2016) based on functional group transformations. We use the standard train/validation/test splits and train both single-step models and reaction type classifiers on this data.

USPTO-190 (USPTO-Hard). USPTO-190, also known as USPTO-Hard, was introduced by Chen et al. (2020) as a challenging multi-step benchmark. It consists of 190 target molecules whose synthesis routes were curated from USPTO patent literature. Each target has a ground-truth synthesis route comprising multiple reaction steps. Unlike USPTO-50k, this dataset does not include reaction type annotations; we use RXN-Insight (Dobbelaere et al., 2024) to assign Schneider reaction classes to the individual steps.

USPTO-190-Steps (new). To evaluate single-step guidance on a diverse set of products, we extract the individual reactions composing the test routes of USPTO-190 and deduplicate them by product SMILES. This yields 640 unique products, each associated with its ground-truth precursors and reaction type. We refer to this dataset as USPTO-190-Steps throughout the paper. The distribution

of reaction types in this dataset reflects the composition of the original multi-step routes, with Heteroatom Alkylation (class 2), Acylation (class 9), and C–C Coupling (class 0) being the most common classes.

D.3 Reaction type partial-sequence data

We generate two datasets for training the reaction classifiers: one from USPTO-50k and one from USPTO-190. The procedure in both cases is the same. The procedure described below is applied to the training and validation subsets of both USPTO-190-steps and USPTO-50k.

- We augment each product with 3 root-aligned canonicalizations, obtained by first generating 20 canonicalizations then selecting the 3 most different ones.
- (Sample): We then sample for each product SMILES (including the root-aligned augmentations) 100 samples using the generator.
- We then split the samples dataset by reaction type, thus obtaining precursors for each reaction type from the products available in the dataset
- (Contrast): Next we generate the partial sequences by splitting each precursor SMILES at every position, and augmenting with contrastive samples by swapping out the last token with 3 random tokens sampled from the vocabulary.
- Lastly, we subsample the partial sequences dataset to obtain a training set of 2M sequences and a validation set of 200K sequences covering different completion ratios.

Table 2: Class-wise statistics of the reaction-type partial-sequence dataset built from USPTO-50k: number of original reactions, generator samples, and the (oversampled) per-class subsample used for training. The training and validation sets additionally contain 2.5M and 1.0M contrastive wrong-token sequences (class 11), giving totals of 4.84M training and 1.77M validation partial sequences.

Dataset	0	1	2	3	4	5	6	7	8	9	10
uspto50k-train											
original	12.4k	8.9k	4.3k	632	1.0k	4.0k	6.3k	632	106	649	853
samples	472.9k	241.6k	347.5k	336.3k	75.5k	69.1k	267.0k	63.4k	34.0k	329.9k	310.7k
subsamples	227k	227k	227k	227k	227k	227k	227k	227k	68k	227k	227k
uspto50k-val											
original	1.5k	1.1k	538	75	127	487	789	71	15	90	113
samples	256.6k	163.1k	141.1k	129.6k	17.9k	31.4k	122.4k	18.1k	12.6k	132.5k	116.8k
subsamples	91k	91k	91k	53k	81k	91k	91k	48k	14k	53k	70k

D.4 Property predictor training

All property classifiers are vanilla transformer encoders with 4 layers, 4 attention heads, 128 hidden dimensions, and 0.1 dropout. We train with Adam (learning rate 10^{-3}, weight decay 10^{-5}) on 4.84M partial sequences (Table 2). The ranking loss margin is $\gamma = 1.0$ and the ranking weight is $\mu = 1.0$ unless stated otherwise. Each classifier is validated on a held-out set of 1.77M partial sequences generated from ground-truth precursors and round-trip validated generator samples, augmented with contrastive wrong-token completions. The full training procedure is summarized in Algorithm 1.

D.5 Inference

Base generator. We use the product-reactant-aligned generator described in Section 2.1 with a beam size of 10. To obtain 100 samples per product despite a beam width of 10, we follow the RSMILES protocol (Zhong et al., 2022)—which is also the syntheseus default for template-free models—and augment each input product with 20 canonical-equivalent SMILES randomizations. Beam search is run independently per augmentation, and the resulting candidates are pooled and deduplicated to produce the per-product sample set (the same procedure used to construct training data, Section D.3). We evaluate two checkpoints: one trained on USPTO-50k and one on USPTO-190, to study how guidance interacts with generators of different training-set coverage.

Algorithm 1 Training a CAG Classifier

Require: Ground-truth dataset $\mathcal{D} = \{(\mathbf{r}^{(i)}, p^{(i)}, y^{(i)})\}_{i=1}^{N}$, generator p_θ, vocabulary $\mathcal{V}$
Require: Hyperparameters: margin γ, ranking weight μ, sampling ratio ρ
Ensure: Trained classifier p_α

Initialize classifier p_α
for each training iteration **do**
  Sample minibatch $\{(\mathbf{r}, p, y)\}$ from $\mathcal{D}$
  $\mathcal{B} \leftarrow \emptyset$ *// Training batch*
  for each $(\mathbf{r}, p, y)$ in minibatch **do**
    Sample position $k \sim \text{Uniform}(1, |\mathbf{r}|)$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\mathbf{r}_{\leq k}, p, y)\}$ *// Ground-truth partial sequence*
    Sample $\tilde{r}_k \sim \text{Uniform}(\mathcal{V})$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{((\mathbf{r}_{<k}, \tilde{r}_k), p, y_{\text{wrong}})\}$ *// Wrong-token augmentation*
    if $u < \rho$ where $u \sim \text{Uniform}(0, 1)$ **then**
      $\tilde{\mathbf{r}} \sim p_\theta(\cdot \mid p)$ *// Sample from generator*
      $\tilde{y} \leftarrow \textsc{GetReactionClass}(\tilde{\mathbf{r}}, p)$
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\tilde{\mathbf{r}}_{\leq k}, p, \tilde{y})\}$ *// On-policy sample*
    end if
  end for
  $\mathcal{L}_{\text{CE}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{(\mathbf{r}_{\leq k}, p, y) \in \mathcal{B}} -\log p_\alpha(y \mid \mathbf{r}_{\leq k}, p)$
  Let $\mathcal{B}' \subseteq \mathcal{B}$ denote the ground-truth partial sequences, with $r_k^* := \mathbf{r}_k$ *// Excludes wrong-token and on-policy samples*
  $\hat{r}_k \leftarrow \arg\max_{r \neq r_k^*} p_\theta(r \mid \mathbf{r}_{<k}, p)$ *// Generator's preferred alternative to ground truth*
  $\tilde{s}_\alpha(r) \leftarrow \log p_\theta(r \mid \mathbf{r}_{<k}, p) + \log p_\alpha(y \mid \mathbf{r}_{<k}, r, p)$ *// Guided log-score (generator + classifier), log-softmax-normalized over $\{r_k^*\} \cup$ top-$k(p_\theta)$*
  $\mathcal{L}_{\text{rank}} \leftarrow \frac{1}{|\mathcal{B}'|} \sum_{(\mathbf{r}_{\leq k}, p, y) \in \mathcal{B}'} \max\left(0,\, \gamma + \tilde{s}_\alpha(\hat{r}_k) - \tilde{s}_\alpha(r_k^*)\right)$
  Update p_α to minimize $\mathcal{L}_{\text{CE}} + \mu \mathcal{L}_{\text{rank}}$
end for
return p_α

Default inference parameters. Guided generation uses guidance scale $\lambda = 1.0$, guidance onset length $L_{\min} = 5$, and $N = 72$ candidate tokens per step, unless stated otherwise.

D.6 Baselines

For single-step evaluation, we compare against all baselines available in the Syntheseus library (Maziarz et al., 2023), including both template-free methods (MEGAN (Sacha et al., 2021), Chemformer, Graph2Edits) and template-based methods (LocalRetro (Chen & Jung, 2021), RetroKNN, GLN, MHNReact (Seidl et al., 2022)). We additionally include NeuralSym (Segler & Waller, 2017), a template-based method commonly used in multi-step planning. All methods generate 100 samples per product to ensure an equal computational budget. Template-based baselines are marked with † in result tables.

D.7 Evaluation pipeline

Chemical validity. Generated SMILES strings are validated using Syntheseus (Maziarz et al., 2023), which checks for parsability and chemical validity via RDKit. Invalid SMILES are discarded before computing any downstream metrics.

Reaction type classification. We use RXN-Insight (Dobbelaere et al., 2024) to classify each valid generated reaction into one of the Schneider reaction superclasses (Schneider et al., 2016). A sample is considered *class-correct* if the predicted reaction type matches the guidance target.

Round-trip accuracy. To assess whether a generated precursor set plausibly produces the target product, we apply the forward model of Zhong et al. (2022) to the generated reactants and check

whether the predicted product matches the original target (exact SMILES match after canonicalization). This round-trip check provides a proxy for chemical feasibility without requiring wet-lab validation.

Achievable class diversity. For each product in the dataset, we guide generation toward every available reaction type and record whether at least one sample is class-correct. The *guidance success* is the count of products achieving this for a given target class. The *achievable class diversity* is defined as the average number of reaction types for which guidance succeeds per product, normalized by the total number of classes. We also report stricter variants that additionally require RXN-Insight reaction identification (+RXN-I) or round-trip validity (+RT), measuring whether steered samples are not only class-correct but also chemically identified or feasible.

Jaccard overlap. To quantify whether guidance explores genuinely different chemical space, we compute the Jaccard similarity between the set of unique valid SMILES produced by guided and unguided generation for each product, then report the average across all products.

D.8 Compute resources

All experiments were run on the LUMI supercomputer using a single AMD Instinct MI250X GCD per job (`small-g` partition, 100 GB RAM); no multi-GPU training or inference was used.

Property predictor training. The reaction-type classifiers are small Transformers ($\sim$5 MB; 4 layers, hidden dim 128, 4 heads) trained on partial sequences sampled from the base generator. On USPTO-Full (1.36 M training prefixes), one training epoch takes approximately 1030 s; the reported reaction-type checkpoint converges by epoch 35, for roughly 10 GPU-hours of pure training time. On USPTO-50K, an epoch takes approximately 770 s and the reported checkpoint converges by epoch 45, for roughly 10–12 GPU-hours.

Single-step inference. With the R-SMILES generator at beam size 10 and 20 augmentations (100 candidates per target, $N = 72$ candidate tokens per step), the unguided baseline takes approximately 10–15 s per target on a single MI250X. CAG-guided inference (reaction type, λ=1.0, including round-trip evaluation) takes approximately 50–60 s per target, roughly a 4–5$\times$ overhead over the unguided baseline due to per-token classifier scoring of the $n_{\text{candidates}}$ continuations. This overhead scales with the candidate pool size N: at the smallest pools we evaluate ($N = 3$–5, Section F.1), the overhead drops to its lowest measured value, roughly 1.08–1.14$\times$.

Multi-step search. Retro* search with a 100-call model budget runs in approximately 600–700 s of wall-clock per target with CAG guidance; search jobs were dispatched in SLURM array tasks of 5 targets each.

E Comparison with other methods

This section places CAG against other retrosynthesis methods, at the single-step level and inside search. The classifier comparison of Section 4.2 holds the generator fixed; here the generator is one method among several.

E.1 Effect of guidance on the generator's samples

In this section, we show that guidance changes which samples the generator returns. Figure 10 shows that, of the 287 products where every unguided sample fails to round-trip back to the product, guidance recovers a valid, round-trip-consistent precursor set for 231. The guided and unguided sample sets overlap by 39%, measured as the size of the intersection over union of the samples obtained with and without guidance (Jaccard) averaged per product.

(1) Baseline failure rates

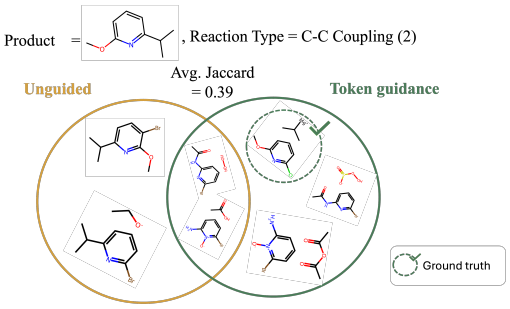
	CAG	Generator
Failure rate	-	287
Exact recovery	113	-
RT recovery	231	-

(2)

Detailed guidance effects

	CAG	Generator
Avg. unique samples	**46.4**	30.5
Avg. class correct	**13.4**	8.9
Avg. name correct	**19.1**	15.7
Avg. RT correct	18.6	**18.9**

(a) Effect of guidance on generator samples



(b) Samples with and without guidance

Figure 10: Effect of guidance on the samples of the generator. The tables show metrics computed over 100 samples per product, while the image visualizes samples for a single product.

E.2 Achievable class diversity: full comparison

Figure 3b in the main text shows achievable class diversity for a subset of methods (the unguided generator, the strongest template-based baseline, the strongest template-free baseline, and CAG). Figure 11 gives the full comparison against every baseline, template-based and template-free.

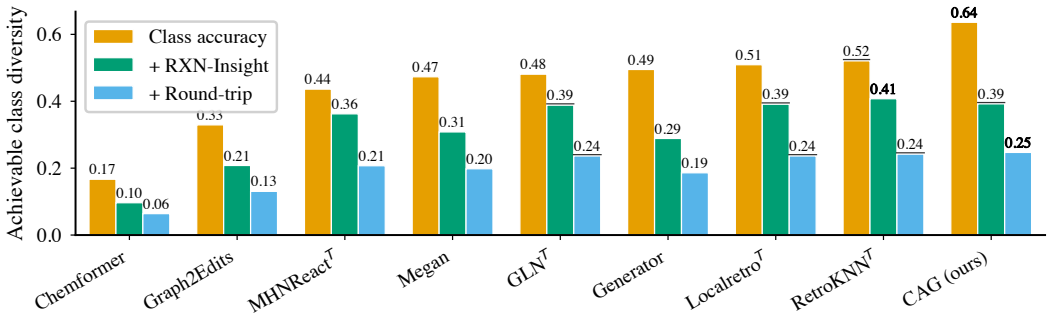


Figure 11: Achievable class diversity on USPTO-190-Steps for all baselines and guided generation, under three criteria: class match, RXN-Insight reaction naming, and round-trip validity. Guidance reaches the highest diversity of any baseline under every criterion.

E.3 Single-step results on every pair of classes

Experimental setup. For every product we guide the R-SMILES generator toward each of the 10 reaction classes other than its ground-truth one (of the 11 Schneider superclasses, Section D.1), at the default inference parameters of Section D.5 ($\lambda = 1.0$, $L_{\min} = 5$, $N = 72$); each (product, target class) pair gets the same 100-sample set as the single-step evaluation (beam width 10 over 20 canonical-equivalent augmentations, pooled and deduplicated, Section D.5). Every valid sample is classified with RXN-Insight (Dobbelaere et al., 2024) under three criteria of increasing strictness: whether its predicted class matches the requested one, whether its predicted reaction name matches (+RXN-I), and whether it round-trips back to the product under the forward model (+RT, Section D.7). *Achievable class diversity* is, per product, the number of non-ground-truth classes for which at least one sample clears a given criterion, averaged over products and normalised by the number of classes available to that product (Figs. 11 and 13). Only the R-SMILES generator supports token-level steering, so every baseline contributes one fixed 100-sample set per product, generated once with no guidance and reused as its own reference for every target class; the delta heatmaps below subtract,

cell by cell, the number of products class-correct under a reference method's unguided samples from the number under CAG_{rxn}, with the source class the product's ground truth and the target class the one guidance was asked for, so the diagonal is empty by construction.

On USPTO-190-STEPS (640 products, Section D.2) the reference is the unguided generator. On the full USPTO-50K test split we repeat the same protocol; the delta heatmaps there compare CAG_{rxn} against three references: the unguided generator ablation, RetroKNN (the strongest template-based baseline), and Megan (the strongest template-free baseline), while achievable class diversity is reported for every baseline of Section D.7.

USPTO-190-Steps. To evaluate steering beyond the ground-truth class, we guide each product toward every non-ground-truth reaction type and compare the number of class-correct products against the unguided baseline. Figure 12 shows the difference in success counts on USPTO-190-Steps, stratified by source class (row) and target class (column). Guidance improves steering rates for the vast majority of class pairs: most cells are positive, indicating that the guided model produces more class-correct samples than the unguided baseline for the corresponding (source, target) combination. The improvements are particularly strong for off-diagonal pairs where the target class differs substantially from the source class, confirming that guidance steers the model into genuinely different regions of the chemical space rather than reinforcing existing biases. A small number of cells show negative or near-zero deltas, typically involving rare classes with few training examples (e.g., class 8) or class pairs with high mechanistic overlap where the base model already produces some class-correct samples by chance.

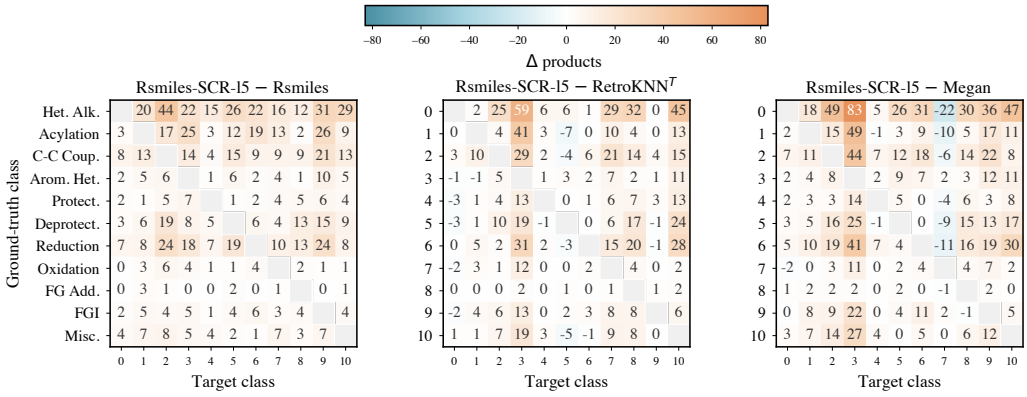


Figure 12: The difference between guided and baseline results for steering towards all non-ground-truth reaction types. The heatmaps show the difference in the number of products with precursors from non-ground-truth target classes, stratified per class. Our guidance scheme leads to improved rates for almost all class pairs, with exceptions discussed in the text.

USPTO-50k. We repeat the steering analysis on the full USPTO-50k test set to verify that the trends observed on USPTO-190-Steps generalize to a larger and more diverse benchmark. Figure 13 reports the achievable class diversity of all baselines and CAG_{rxn} on USPTO-50k, and Fig. 14 shows the difference in class-correct product counts between CAG_{rxn} and representative baselines (the unguided generator ablation, the strongest template-based method, and the strongest template-free method). The trends match Figure 12.

E.4 SEARCH-LEVEL RESULTS

Here we compare our guidance method to other single-step models guided at the search level.

Experimental setup. Every run is Retro* search (Chen et al., 2020) over the 190 USPTO-190 targets, capped by whichever comes first: 100 calls to the single-step model, 500 search iterations, or 600 s of wall-clock time, with 100 precursors sampled per expansion. A target is solved when search reaches a route whose leaves are all purchasable. We evaluate baselines trained on both USPTO-190 and USPTO-50k. In addition to the search metrics of nodes explored and number of model calls, we

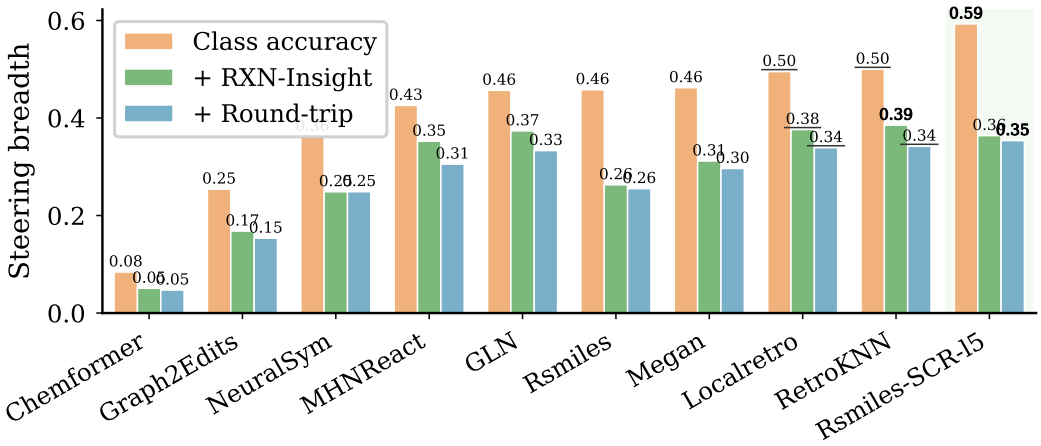


Figure 13: Achievable class diversity on USPTO-50k for all baselines and CAG_{rxn}. As on USPTO-190-Steps, CAG_{rxn} reaches the highest achievable class diversity across all three criteria (class accuracy, RXN-Insight name match, and round-trip accuracy).

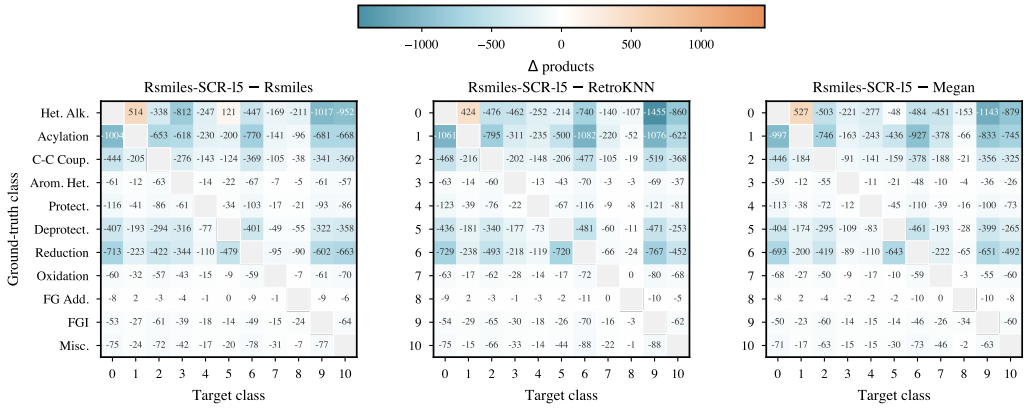


Figure 14: Difference between CAG_{rxn} and baseline results for steering towards all non-ground-truth reaction types on USPTO-50k. The heatmaps show the difference in the number of products with precursors from non-ground-truth target classes, stratified per class, against the unguided generator ablation, RetroKNN (template-based), and Megan (template-free). As on USPTO-190-Steps, our guidance scheme improves steering rates for the vast majority of class pairs.

report adherent solve, the number of products for which at least one route respecting the requested classes is found, and the average number of adhered routes per product.

Levels of guidance. We enforce guidance in this search process through 3 mechanisms, as standalone or in combination:

- **F (hard filtering).** At every expansion, candidates whose RXN-Insight class does not equal the requested one are dropped before they can enter the search tree.
- **G (search-level guidance).** The Retro* node cost, normally $-\log p_\theta$ of the reaction, gets a fixed additive penalty $\gamma = 8$ for candidates whose predicted class does not match the request. This changes what the value function prefers without removing any candidate, so it is a strict superset of what F can reach. A similar idea is explored for starting material constrained synthesis in (Armstrong et al., 2024).
- **S (single-step guidance, CAG_{rxn}).** The generator is steered at decoding time as in Section 2.2, so the candidate set that search receives is already shifted toward the requested class.

F and G are search-time interventions and apply to every single-step baseline. S is generator-specific and applies only to the R-SMILES generator.

Results. Table 3 puts every multi-step run on USPTO-190 in one table: the baselines, hard filtering applied to them, each guidance mechanism on the generator alone and in combination, and the USPTO-190 checkpoint. On both checkpoints, we notice that applying filtering leads to the highest combination of adherent solve rate and number of routes, with RSMILES clearly outperforming the other baselines. When combining single-step guidance on RSMILES with other levels of guidance, we notice that single-step guidance is necessary to unlock the highest number of adhered routes per product ($CAG_rxn = 2.41$, vs $G_rxn = 1.97$ and $F_rxn = 0.51$), with single-step guidance combined with value function guidance (SG_{rxn} in the table) leading to the best balance between adherent solve rate and number of adhered routes per product. We thus show empirically that guiding RSMILES at the single-step level unlocks previously unachievable results, and combines well with the search-level guidance mechanisms.

F Ablations and cost

F.1 Guidance scale, onset length and number of candidate tokens

We provide here the full details behind the ablation results. All experiments use reaction-type steering on USPTO-190-steps (640 products, 100 samples per product). Each ablation varies one hyperparameter while keeping the others at their default values (λ=1.0, N=72, $L_{\min}$=5).

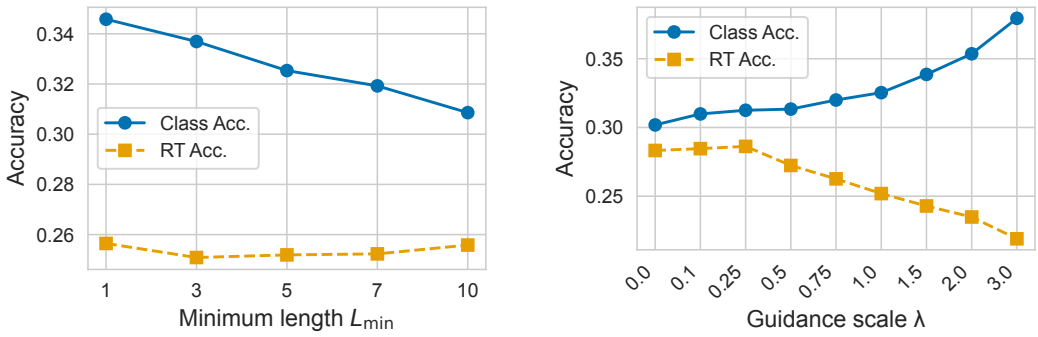
Number of candidate tokens (N) We evaluate $N \in \{10, 24, 48, 72\}$. All metrics are effectively constant across this range: class accuracy varies by less than 0.2 pp, and round-trip accuracy by less than 0.2 pp. This indicates that the ranking step successfully identifies the best tokens even from a small candidate pool, and that N can be reduced from 72 to 10 — yielding a proportional speedup in the classifier forward passes — with no measurable loss in quality. Pushing further to $N \in \{3, 5\}$, with measured generation cost, class accuracy stays within 1.2 points of $N = 72$ (32.15% at $N = 3$, 32.91% at $N = 5$, against 33.33% at $N = 72$), at a cost of 7.5% and 14.2% more generation wall-clock time respectively.

Guidance onset length ($L_{\min}$) We evaluate $L_{\min} \in \{1, 3, 5, 7, 10\}$. $L_{\min}$ controls the number of tokens generated by the unguided base model before classifier guidance is applied. Lower $L_{\min}$ means guidance starts earlier. As shown in Fig. 15(a), earlier guidance onset monotonically improves class accuracy (from 30.9% at $L_{\min}$=10 to 34.6% at $L_{\min}$=1, a $\sim$3.7 pp improvement), while round-trip accuracy remains essentially unchanged ($\Delta < 0.6$ pp). However, guiding from the very first token ($L_{\min}$=1) slightly reduces the number of surviving samples, likely because early-token guidance can occasionally push the model toward degenerate sequences.

Table 3: Multi-step search on USPTO-190 (Retro*, 100 model calls per target). *Adherent solve*: share of the 190 targets with at least one route in which every step that carries a requested class matches it. *N. routes*, *Nodes* and *Calls*: non-overlapping adhered routes, search nodes and model calls per target, averaged over covered targets. Single-step guidance (S_rxn) is necessary to unlock the highest adhered solve and number of adhered routes per target. It also combines well with the search-level guidance, with SG_rxn combination giving the highest number of adhered routes per product.

Method	Adherent solve (↑)	N. routes (↑)	Nodes (↓)	Calls (↓)
USPTO-50k checkpoint, no guidance				
LocalRetro	**0.11**	1.64	8075.65	39.08
GLN	0.05	0.39	6332.84	22.01
RetroKNN	0.10	1.58	8013.27	37.64
MHNreact	0.01	0.09	**551.09**	**5.03**
Chemformer	0.05	0.93	976.31	13.97
Graph2Edits	0.07	1.19	5487.67	22.53
Megan	0.08	1.37	8505.98	34.77
Generator	0.04	**2.03**	1254.00	9.89
USPTO-50k checkpoint, baselines with hard filtering (F_{rxn})				
LocalRetro	0.12	0.24	829.18	38.38
GLN	0.04	0.04	653.96	**0.00**
RetroKNN	0.13	0.31	845.06	**0.00**
Chemformer	0.05	0.15	207.56	6.16
Graph2Edits	0.14	0.28	393.13	18.84
Megan	0.00	0.00	245.00	57.00
NeuralSym	0.10	0.15	497.62	36.09
Generator (F_{rxn})	**0.19**	**0.50**	**143.82**	7.19
USPTO-50k checkpoint, Generator with each guidance mechanism				
G_{rxn}	0.15	1.97	923.07	8.07
FG_{rxn}	0.19	0.51	**151.64**	7.48
S_{rxn} (CAG_{rxn})	0.19	2.41	1023.66	7.00
SF_{rxn}	**0.26**	0.76	190.48	6.97
SG_{rxn}	0.20	**2.44**	1001.07	**6.96**
SFG_{rxn}	0.24	0.67	199.72	6.72
USPTO-190 checkpoint, no guidance				
NeuralSym	0.00	0.83	17622.49	87.96
Generator	**0.14**	**1.93**	**1071.63**	**9.14**
USPTO-190 checkpoint, baselines with hard filtering (F_{rxn})				
NeuralSym	0.18	0.29	554.72	38.96
Generator (F_{rxn})	**0.54**	**2.22**	**342.66**	**8.29**
USPTO-190 checkpoint, Generator with each guidance mechanism				
G_{rxn}	0.53	5.27	1713.43	8.15
FG_{rxn}	0.52	2.06	338.19	8.28
S_{rxn} (CAG_{rxn})	0.23	4.40	1436.35	**5.58**
SF_{rxn}	**0.56**	2.09	357.66	7.23
SG_{rxn}	0.39	**6.15**	1689.61	7.07
SFG_{rxn}	0.54	2.07	**334.09**	6.90

Guidance scale (λ) We evaluate $\lambda \in \{0.0, 0.1, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0, 3.0\}$. This is the only hyperparameter that produces a meaningful trade-off (Fig. 15(b)): as λ increases, class accuracy rises from 30.2% to 37.9% ($+7.7$ pp) while round-trip accuracy drops from 28.3% to 21.9% (-6.4 pp). The divergence between these two curves demonstrates that stronger guidance biases the beam toward class-correct but chemically less faithful sequences. This is a clear steering–validity trade-off, so λ sets where on that trade-off a run sits.



(a) Guidance onset length $L_{\min}$

(b) Guidance scale λ

Figure 15: Effect of the two main hyperparameters on class accuracy (steering, blue) and round-trip accuracy (validity, orange). **(a)** Guidance onset $L_{\min}$: earlier guidance improves class accuracy by $\sim$3.7 pp without degrading round-trip accuracy. **(b)** Guidance scale λ: class accuracy increases by 7.7 pp while round-trip accuracy drops by 6.4 pp.

F.2 Misspecified guidance targets

Guidance targets in the search experiments come from the reference route, which raises the question of how much the results depend on the target being right. We corrupt it node by node: with probability p, the requested class at an intermediate molecule is replaced by a class drawn uniformly from the classes other than its ground-truth one. Table 4 reports the sweep on 40 targets.

Solve rate does not depend on the guidance target being correct: 35, 36 and 38 of 40 targets are solved at $p = 0.25$, 0.5 and 1.0, against 36 to 39 for the uncorrupted reference. Adherence follows the requested label rather than the ground-truth one. Adherence to the ground-truth class falls from 0.53 to 0.12, but that fall is a mixture effect: nodes whose request was left alone hold a flat $\approx$0.65 wherever they exist, corrupted nodes reach their own wrong request at 0.27 to 0.35, and the pooled number declines only because the corrupted share grows with p.

Table 4: Misspecification sweep on 40 targets. Each intermediate molecule has its requested class replaced by a uniformly drawn wrong one with probability p. Adherence to a label is the share of returned precursors whose RXN-Insight class matches that label.

	$p = 0.25$	$p = 0.5$	$p = 1.0$
targets solved (of 40)	35	36	38
adherence to the ground-truth class, all nodes	0.53	0.43	0.12
uncorrupted nodes, adherence to their true request	0.67	0.65	—
corrupted nodes, adherence to their wrong request	0.27	0.35	0.31