RETROGEF: DYNAMIC GRAPH EDIT FLOW FOR SINGLE-STEP RETROSYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrosynthesis enables the discovery of viable synthetic routes to target molecules. It plays a central role in modern drug discovery and materials design. Retrosynthesis involves molecular graph transformations that can change both connectivity and graph size. These transformations may introduce reactant components absent from the target while revising the product-derived structure. To model these transformations, we propose RETROGEF, a flow-based generative model for single-step retrosynthesis. Starting from the target molecule, it constructs possible reactants by adding atoms and changing bonds in the molecular graph. RETROGEF models molecular transformations and changes in graph size within the same generative process, rather than relying on a fixed-size graph canvas. It learns this process directly from product–reactant pairs without requiring a prescribed edit order. Experiments on representative retrosynthesis benchmarks demonstrate that RETROGEF achieves state-of-the-art performance. Code is available at `https://anonymous.4open.science/r/retrogef-E7F8/`.

1 INTRODUCTION

Single-step retrosynthesis predicts the precursor molecules needed to obtain a desired product, providing the individual transformations used in synthesis planning (Segler et al., 2018; Chen et al., 2020; Song et al., 2026). Recovering these precursors requires more than changing bonds in the product: the reactant set may contain several molecules and additional atoms whose number and connectivity are not known in advance (Yao et al., 2023; Zhong et al., 2023). A model must both construct missing structure and revise the structure it inherits. The central challenge is to learn this variable-size construction process from reaction records that specify the product and reactants, but not how one graph is built from the other.

Existing representations organize this problem differently. Sequence models serialize reactants, graph-edit models expose structural transformations (Zhong et al., 2022; Sacha et al., 2021; Zhong et al., 2023; Deng et al., 2025), and diffusion or flow models learn stochastic paths between reaction graphs (Igashov et al., 2024; Yadav et al., 2025; Wang et al., 2026). A fixed graph canvas reserves capacity before generation, whereas a growing graph must decide how each new atom enters and connects to the existing structure. Atom creation and connectivity are coupled: a new atom becomes an attachment site, and its first bond changes the context for subsequent edits. This motivates a representation in which growth and structural revision act on the same evolving graph.

Learning these edits introduces a second ambiguity. Multiple construction paths can reach the same reactant set, yet reaction records provide no preferred order (Sacha et al., 2021; Zhong et al., 2023). Nor does a partial graph always identify which missing reactant atom a newly added atom represents: different correspondences can yield an identical structural update. Supervision should accommodate these alternatives rather than treat an arbitrary edit sequence or atom correspondence as the unique answer. The learning objective must therefore connect paired reaction graphs through the edits that the model will actually use during generation.

We propose RETROGEF, a continuous-time graph jump process built around *complete graph edits*. Each edit specifies a complete structural change: adding an attached atom determines its attachment site, attributes, and first bond together. The resulting graph immediately becomes the context for the next edit. Additional atoms can extend the new structure, while bond and attribute edits can revise

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.

both inherited and generated parts. Representing these changes relative to the product lets the state grow through construction rather than occupy a preallocated reactant canvas.

To learn this process, we construct a stochastic bridge from each product to its recorded reactant graph. At an intermediate graph, the bridge assigns rates to eligible edits, providing supervision for both which change to make and when to make it. Training samples different edit orders and aggregates rates when hidden atom correspondences yield the same observable update. Generator matching learns these rates from the sampled bridge states (Holderrieth et al., 2025), aligning paired-graph supervision with the complete edits used at inference.

Our contributions are threefold:

- A variable-size graph jump process that couples atom creation and initial connectivity through complete edits, allowing growth and revision within a single evolving state.
- A paired-graph bridge that supervises edit rates across multiple construction orders and combines equivalent atom correspondences into observable transitions.
- An empirical evaluation on USPTO-Full and USPTO-50K demonstrating improved top-k accuracy under predicted and reference reaction-center conditioning, with controlled ablations and analyses of graph growth, structural revision, and reaction classes.

2 Related Work

Representations for retrosynthesis. Retrosynthesis methods differ in the structural decisions they expose. Template-based models select transformations by similarity or learned representations (Coley et al., 2017; Dai et al., 2019; Chen & Jung, 2021). Sequence models generate SMILES (Weininger, 1988; Liu et al., 2017) with augmentation, alignment, or pretraining (Tetko et al., 2020; Zhong et al., 2022; Deng et al., 2025). Graph-to-sequence models retain structured inputs (Seo et al., 2021; Tu & Coley, 2022). Two-stage models identify centers or edits before completing synthons (Shi et al., 2020a; Yan et al., 2020; Somnath et al., 2021; Chen et al., 2023). Graph-to-graph translation (Lin et al., 2023; Yao et al., 2023) and autoregressive editing (Sacha et al., 2021; Zhong et al., 2023) expose molecular structure and intermediate states. We instead learn rates of complete structural edits across multiple product-to-reactant paths.

Graph generation and variable-size structure. Autoregressive generators make successive node and edge decisions (You et al., 2018; Shi et al., 2020b; Luo et al., 2021), whereas one-shot models generate graphs jointly (Martinkus et al., 2022). Score-based and discrete-state models evolve their representations (Niu et al., 2020; Jo et al., 2022; Vignac et al., 2023; Qin et al., 2025). Variable-size approaches include trans-dimensional jump diffusion (Campbell et al., 2023) and sequence insertion/deletion flows (Havasi et al., 2025). Retrosynthetic growth must additionally specify connectivity relative to the product (Yao et al., 2023; Laabid et al., 2025). Our attached atom edit addresses this requirement by coupling atom creation to its first bond. This keeps the evolving structure anchored to the product while allowing its size and component structure to be determined during generation.

Discrete dynamics and reaction-conditioned bridges. Discrete generation learns categorical transitions (Austin et al., 2021), continuous-time jump rates (Campbell et al., 2022), or reverse probability ratios (Lou et al., 2024). Flow matching learns from conditional paths in continuous (Lipman et al., 2023) and discrete spaces (Campbell et al., 2024; Gat et al., 2024); Generator Matching extends this principle to Markov generators (Holderrieth et al., 2025). For retrosynthesis, RetroBridge uses Markov bridges and RetroDiff uses staged distribution interpolation (Igashov et al., 2024; Wang et al., 2025). Retro SynFlow and RetroDiT apply discrete flow matching, the latter with reaction-center guidance (Yadav et al., 2025; Wang et al., 2026). Our bridge supervises complete edits, combining equivalent hidden atom correspondences while learning transition intensity and edit probabilities. The resulting construction order remains latent, while each observable edit is defined on the current graph and can be revised by later transitions. This perspective connects variable-size graph generation with the structural revisions required by retrosynthesis, which we develop in the next section.

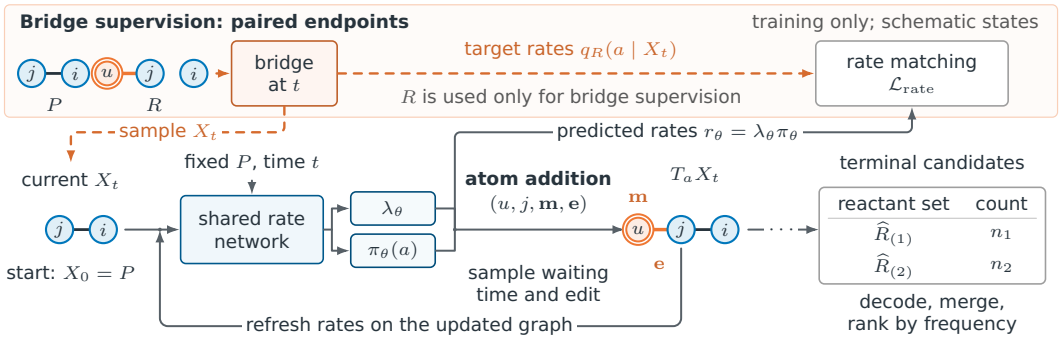


Figure 1: RETROGEF: product-relative complete edits, bridge supervision, and generation. Paired endpoints supervise edit rates (dashed arrows); the shared network observes (P, X_t, t). Each generated edit updates the graph before the next rate prediction. States are schematic.

3 RETROGEF

RETROGEF (Retrosynthetic Graph Edit Flow) learns a continuous-time process of complete graph edits from paired product and reactant graphs. Figure 1 summarizes training and generation.

3.1 DYNAMIC GRAPH STATES AND COMPLETE EDITS

Let $P = (V_P, E_P)$ be the product graph and $R = (V_R, E_R)$ the disjoint union of reactant graphs, with atom and bond attributes including stereochemistry. We learn $p_\theta(R \mid P)$, suppressing any supplied reaction-center condition throughout the notation. We describe R and each evolving graph relative to the fixed product: added atoms, revised attributes on inherited atoms, and added, removed, or relabelled bonds. This description is lossless; the network receives the evolving graph's changes relative to P. Atom mapping fixes the correspondence of inherited atoms to P (Schwaller et al., 2021). The state starts at $X_0 = P$, retaining the product atoms and any generated atoms still present.

A complete action a specifies the location and attributes needed to produce one successor graph $T_a X$. An *attached atom addition* creates a new atom and its first bond; an *isolated atom addition* starts a new component. Other edits add or remove bonds, update attributes, or remove generated atoms. Each update is completed before the next rate prediction, so newly created structure can be extended or revised.

The admissible action set $\mathcal{A}(X)$ preserves inherited product atoms and their atomic numbers, excludes self-loops and duplicate bonds, and restricts attribute choices to the supported vocabulary. Its masks are recomputed on the current graph. Intermediate states may be incomplete chemical structures; molecular validity is checked only after generation. The edit vocabulary and structural masks are specified in Appendices B and G.3.

3.2 BRIDGES FROM PRODUCTS TO REACTANTS

We construct a stochastic bridge from P to each recorded reactant graph R. The bridge creates missing atoms and corrects atoms or bonds that differ from R, producing intermediate graphs and local supervision without a prescribed edit sequence. This use of endpoint-conditioned bridges connects to bridge-based retrosynthesis and general Markov-generator matching (Igashov et al., 2024; Holderrieth et al., 2025).

The sum of edit rates controls when the graph changes, while their normalized values determine which edit occurs. We express both bridge and learned rates per unit of transformed time $\tau = -\log(1-t)$, so that $t \in [0, 1)$ corresponds to $\tau \in [0, \infty)$. The network receives t as its bounded time input. Let $\mathcal{C}_R(X)$ denote the corrections other than atom addition, $m_R(X)$ the number of missing reactant-only atoms, and $\rho_R(a \mid X)$ a normalized distribution over eligible atom-addition

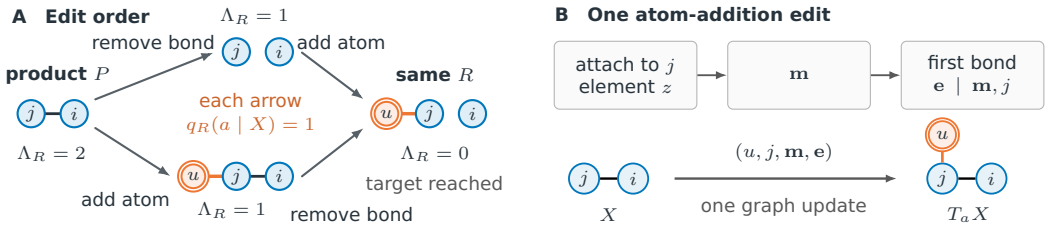


Figure 2: Edit order and complete graph updates. A: Two correction orders reach the same graph, with total bridge intensity $2 \to 1 \to 0$. B: Location and attributes jointly define one update. Blue: product atoms; orange: added atoms.

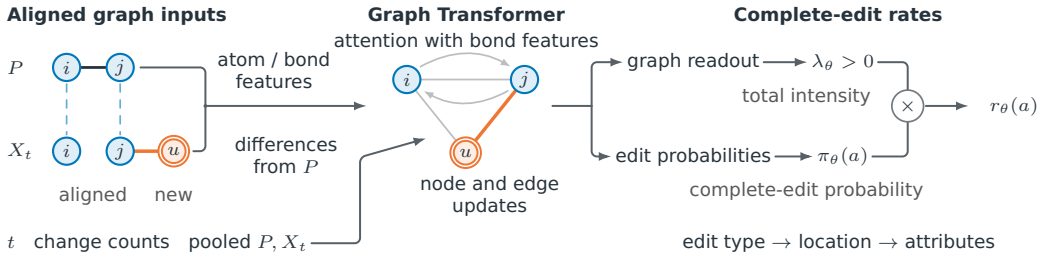


Figure 3: Edit-rate prediction. Intensity and complete-edit probability determine each rate. Dashed links align product atoms; gray links inside the Transformer denote feature interactions, not molecular bonds.

edits. The bridge assigns rate

$$q_R(a \mid X) = \mathbf{1}[a \in \mathcal{C}_R(X)] + m_R(X)\rho_R(a \mid X), \tag{1}$$

where the atom-addition term is zero when no atom is missing. Each other correction has unit rate, while the total atom-addition rate equals the number of atoms still to be created. This separates the amount of graph growth remaining from the number of possible attachment choices. The total bridge intensity is $\Lambda_R(X) = \sum_a q_R(a \mid X)$. Every bridge edit reduces the discrepancy with the target reactant graph. The bridge reaches a representative of the target reactant graph after finitely many edits and remains there as $t \to 1$.

The bridge admits different edit orders, such as exchanging bond deletion and atom addition (Figure 2A). A separate ambiguity is which missing target atom a new atom realizes. For example, two target atoms with the same attributes can propose the same attachment and first bond. We sum the rates of identical complete updates, defining the *quotient bridge*. These hidden correspondences guide training paths, while the network predicts observable edits. Appendix B specifies the atom-addition rules.

3.3 Parameterizing Graph Edit Rates

The rate of a complete edit factors as

$$r_\theta(a \mid X, P, t) = \lambda_\theta(X, P, t)\, \pi_\theta(a \mid X, P, t), \tag{2}$$

where $\lambda_\theta > 0$ is the total edit intensity (the sum of all edit rates) and $\sum_{a \in \mathcal{A}(X)} \pi_\theta(a \mid X, P, t) = 1$. The intensity controls the waiting time until the next edit, and π_θ determines which edit occurs.

A graph Transformer (Vaswani et al., 2017) encodes the current graph, the time, and atom and bond changes relative to the product. Bond features inform attention, while graph-level features summarize the amount of structural change. A graph readout predicts total intensity, and conditional distributions specify the edit location and attributes (Figure 3). Architecture and training settings appear in Appendix C.1.

Edit probabilities. The action distribution accounts for both the edit location and its attributes. To illustrate, let $Z = (P, X, t)$, let $\mathbf{m} = (z, m_2, \ldots, m_6)$ be a tuple of atom attributes with atomic

number z, and let $\mathbf{e}$ be a tuple of bond attributes. For an attached atom addition, denoted f_{A}, at existing atom j, the probability factors as

$$\pi_\theta(f_{\mathrm{A}}, j, \mathbf{m}, \mathbf{e} \mid Z) = \pi_F(f_{\mathrm{A}} \mid Z)\, \pi_{J,z}(j, z \mid f_{\mathrm{A}}, Z) \times \pi_M(m_{2:6} \mid j, z, Z)\, \pi_E(\mathbf{e} \mid \mathbf{m}, j, Z). \tag{3}$$

The attachment site and new element are selected jointly, followed by autoregressive prediction of the remaining atom attributes. Conditioned on the complete new atom, the bond decoder jointly predicts the first bond's type and stereochemistry. These factors define one update (Figure 2B); the remaining edit types use analogous factorizations. Training and sampling use the same complete-action probabilities (Appendix G.3).

3.4 Learning from endpoint bridges

For a sampled training time t, we simulate the bridge to obtain X_t and its target rates $q_R(a \mid X_t)$. We minimize generalized KL divergence between target and predicted rates. Omitting a target-only constant, the loss for one training state is

$$\mathcal{L}_{\mathrm{rate}} = \lambda_\theta - \Lambda_R \log \lambda_\theta - \sum_{a : q_R(a \mid X_t) > 0} q_R(a \mid X_t) \log \pi_\theta(a \mid X_t, P, t). \tag{4}$$

The first two terms fit total intensity, and the last fits the corrective-edit distribution. Only actions with nonzero target rates require explicit probabilities: normalization of π_θ makes the summed predicted rate over all admissible actions equal to λ_θ.

Unlike normalized next-edit classification, rate matching also fits the pace of graph change. At an endpoint, $\Lambda_R = 0$ and the loss reduces to λ_θ, supervising low outgoing intensity without a stop label. Appendix F.3 derives the decomposition into intensity and action-choice terms, including this zero-rate case.

For unperturbed bridges, the unrestricted population optimum over nonnegative rates is

$$r^*(a \mid X, P, t) = \mathbb{E}[q_R(a \mid X) \mid X_t = X, P, t]. \tag{5}$$

The expectation includes hidden endpoints and atom correspondences. Under the conditions of Proposition 3 in Appendix F.5, these rates define a product-initialized process with the conditional bridge's one-time graph marginals (Holderrieth et al., 2025). Since each bridge reaches its target, this idealized process recovers the conditional reactant distribution of the paired data as $t \to 1$.

Recovery augmentation. Unperturbed bridges add only target atoms. To supervise removal and repair, we perturb bridge states with unwanted bonds or superfluous generated leaf atoms and recompute corrective rates toward the same endpoint. This trains deletion and structural revision alongside graph growth, without a prescribed recovery sequence (Appendix C.1).

3.5 Reactant generation

At inference, RetroGEF starts from the product and samples edit times and actions from the learned rates. The time grid controls numerical integration, not the number of edits. Rates are recomputed after every edit, so several edits can occur within one interval and newly added atoms are immediately available for attachment or revision. At a finite terminal time, graphs are decoded into reactant sets. Because every action is applied to the current graph, the same representation can accommodate different numbers of components and edits without prescribing their order. Growth, disconnection, and scaffold revision consequently remain within one generation process. A newly introduced component can be extended by subsequent edits, and its bonds or attributes can be revised as the surrounding graph changes. Likewise, inherited product structure remains available as context for later actions rather than being frozen after the initial prediction. The terminal graph therefore records missing component structure and changes to the retained scaffold in one state description. This shared process does not require a separate stage for each transformation size or a predefined order of operations. The terminal distribution is consequently defined by the same edit space that is supervised during training. Appendix G.4 gives the sampling details.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate on USPTO-Full and USPTO-50K, derived from patent reaction records (Schneider et al., 2016; Lowe, 2017), using the benchmark splits distributed with RetroDiT (Wang et al., 2026). Models are trained separately on each dataset and evaluated with predicted (Pred.) or reference (Oracle) reaction-center information. We sample 100 trajectories per product using 50 sampling intervals, merge equivalent reactant sets, and rank them by frequency. Top-k accuracy follows the atom-and-bond topology matching of Wang et al. (2026). Appendix C specifies the architecture, training, decoding configurations, and matching rules.

Table 1: USPTO-Full test results. Top-k accuracy (%).

Method	Top-1	Top-3	Top-5	Top-10
RetroSim	32.8	–	–	56.1
LocalRetro	39.1	53.3	58.4	63.7
GLN	39.3	–	–	63.7
RetroPrime	44.1	59.1	62.8	68.5
R-SMILES	48.9	66.6	72.0	76.4
NAG2G	49.7	64.6	69.3	74.0
RetroDiT (Pred.)	51.3	67.8	72.3	75.8
RETROGEF (Pred.)	**54.3**	**70.1**	**74.1**	**77.1**
RSGPT	59.2	74.2	78.2	82.1
RetroDiT (Oracle)	63.4	77.6	80.9	83.6
RETROGEF (Oracle)	**68.6**	**80.7**	**82.8**	**84.1**

Table 2: USPTO-50K test results. Top-k accuracy (%).

Method	Top-1	Top-3	Top-5	Top-10
RetroSim	37.3	54.7	63.6	74.1
Retro ProdFlow (Yadav et al., 2025)	50.0	74.3	81.2	85.8
RetroBridge	50.8	74.1	80.6	85.6
RetroPrime	51.4	70.8	74.0	76.1
GLN	52.5	69.0	75.6	83.7
LocalRetro	53.4	77.5	85.9	92.4
NAG2G	55.1	76.9	83.4	89.9
Graph2Edits	55.1	77.3	83.4	89.4
R-SMILES	56.3	79.2	86.2	91.0
Retro SynFlow	60.0	77.9	82.7	85.3
RetroDiT (Pred.)	61.2	81.5	86.2	89.2
RETROGEF (Pred.)	**62.4**	**82.8**	**88.6**	**91.9**
RSGPT*	26.4	37.5	41.4	46.4
RSGPT	63.4	84.2	89.2	93.0
RetroDiT (Oracle)	71.1	90.8	94.5	96.0
RETROGEF (Oracle)	**72.3**	**92.9**	**96.0**	**97.0**

The comparisons cover complementary approaches to retrosynthesis: template-based prediction with RetroSim (Coley et al., 2017), GLN (Dai et al., 2019), and LocalRetro (Chen & Jung, 2021); sequence generation with RetroPrime (Wang et al., 2021), R-SMILES (Zhong et al., 2022), and RSGPT (Deng et al., 2025); graph translation and editing with NAG2G (Yao et al., 2023) and Graph2Edits (Zhong et al., 2023); and stochastic graph generation with RetroBridge (Igashov et al., 2024), Retro ProdFlow and Retro SynFlow (Yadav et al., 2025), and RetroDiT (Wang et al., 2026). We report the baseline scores from the cited sources, with Pred. and Oracle comparisons shown separately for center-conditioned generation. RSGPT includes its pretrained model and fine-tuning-only comparison (*). Appendix C.2 gives the benchmark conventions.

4.2 MAIN RESULTS

With predicted centers, RETROGEF reaches Top-1/Top-10 accuracies of 54.3/77.1% on USPTO-Full and 62.4/91.9% on USPTO-50K (Tables 1 and 2). Reference centers further improve these results to 68.6/84.1% and 72.3/97.0%, respectively. Within the center-conditioned comparisons, RETROGEF exceeds RetroDiT at all four cutoffs on both datasets. The USPTO-Full Top-1 gains are 3.0 points under Pred. and 5.2 under Oracle; on USPTO-50K, the corresponding gains are 1.2 points in both conditions, with Top-10 increasing by 2.7 and 1.0 points. The gains at both ends of the ranking indicate more accurate first predictions as well as better recovery of the recorded reactants among the ranked alternatives. Top-1 therefore measures the reliability of the first proposed reactant set, while Top-10 tests whether the ranking preserves complementary valid sets. The gains at both cutoffs show that the improvement is not only broader coverage: it also changes which candidate is ranked first. In both settings, the center signal identifies where a transformation may occur but leaves the missing reactant structure to be generated. Recovering complete candidates therefore still requires decisions about fragment size, connectivity, and revisions to the product graph. The two benchmarks show that learning these decisions through complete edits can support accurate ranking across distinct levels of reaction-center information. The ordering is consistent across the two datasets despite their different scales and product distributions: USPTO-Full stresses variable-size components, whereas USPTO-50K provides a more constrained class composition. The Pred.–Oracle gap therefore reflects the information supplied about the reaction center, while the decoder still has to choose component structure and connectivity in both cases. This separation motivates the structural analyses below.

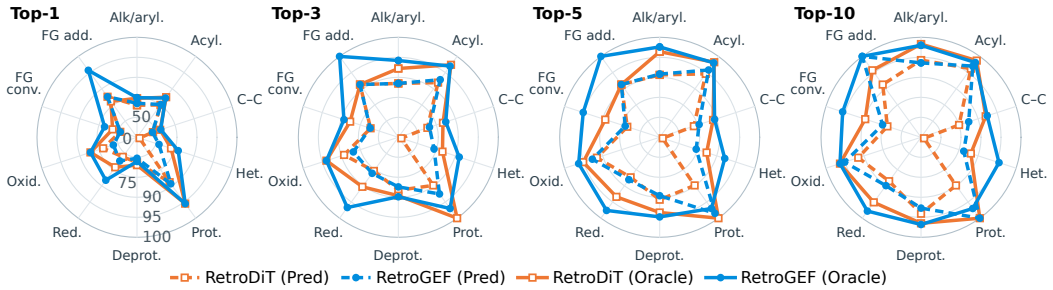


Figure 4: Accuracy across all ten USPTO-50K reaction classes. Shared nonuniform radial scale (%), labeled in the first panel. FG: functional group.

4.3 Graph growth and transformation complexity

Reactant reconstruction requires both introducing atoms absent from the product and revising inherited structure. We examine these demands separately on 90,598 shared USPTO-Full products, using paired predictions from RetroGEF and RetroDiT (Wang et al., 2026). Grouping by added atoms (Table 4), RetroGEF retains a Top-1 advantage in every group requiring graph growth under both center conditions. For 11–20 added atoms, Oracle Top-1 rises from 26.0% to 33.4% and Top-3 from 40.1% to 45.0%. Among these products, 823 have a correct first prediction only from RetroGEF, compared with 406 only from RetroDiT, giving 417 additional correct first predictions. At the other end of the range, reactions requiring no added atoms isolate reconstruction through revisions to existing structure: Oracle Top-1 increases from 71.9% to 84.8%, with gains at all four cutoffs in both conditions. The advantage therefore extends to both growth-intensive reactions and transformations confined to the inherited atoms.

The number of added atoms alone does not describe the extent of a transformation. Table 3 instead groups reactions by atom additions, updated product-atom attributes, and added, removed, or modified bonds, without assuming an edit order. RetroGEF improves Top-1 in every group under both conditions, and the Oracle advantage widens as the number of changes increases. For more than twenty changes, Oracle Top-1 reaches 42.5% versus 33.6%, and Pred. reaches 29.5% versus 25.7%, with gains at all four cutoffs. These results show that the improvement persists when reconstructing the reactants involves many structural changes, not just the introduction of additional atoms. The complete paths in Appendix E.2 illustrate the same behavior at the trajectory level, including examples that alternate growth with bond and attribute updates.

Table 3: Transformation complexity on USPTO-Full. Top-1 accuracy (%) on 90,598 shared test products.

	Pred.		Oracle	
Graph changes	RetroDiT	RetroGEF	RetroDiT	RetroGEF
≤ 5	65.1	**67.6**	78.7	**81.9**
6–10	45.1	**48.8**	57.3	**63.2**
11–20	32.6	**36.8**	43.8	**52.1**
> 20	25.7	**29.5**	33.6	**42.5**

4.4 Performance across reaction classes

Reaction classes provide a chemical perspective complementary to the structural groupings above. Figure 4 compares all ten USPTO-50K classes (Schneider et al., 2016) on 4,930 shared products, using the same edit vocabulary and generation process throughout. The improvements cover several distinct transformation families. Reduction improves at every cutoff in both conditions: Pred. Top-1/Top-10 rises from 55.2/85.7% to 61.6/89.8%, while Oracle rises from 71.3/95.1% to 84.8/97.8%. Heterocycle formation also improves throughout the Oracle ranking, reaching 77.5% Top-1 and 95.5% Top-10 versus 69.7% and 84.3%. Under Pred., C–C formation improves Top-3/5/10 from 63.2/69.9/75.0% to 65.9/76.1/82.6%, and functional-group interconversion improves Top-10 from 69.2% to 75.3%; its

Table 4: USPTO-Full by added atoms. Oracle test accuracy (%) under the common decoding budget.

Added atoms	RetroGEF				RetroDiT			
	1	3	5	10	1	3	5	10
0	**84.8**	**91.5**	**93.0**	**93.6**	71.9	85.4	88.2	90.5
1–2	**79.2**	**90.6**	**92.0**	**92.7**	76.4	88.6	90.8	92.4
3–5	**56.0**	**71.1**	**74.7**	**77.0**	49.4	67.7	72.5	76.6
6–10	**54.4**	**68.1**	**70.9**	**73.1**	47.1	63.8	68.6	72.4
11–20	**33.4**	**45.0**	**47.8**	50.0	26.0	40.1	45.6	**50.4**

Oracle Top-10 rises from 89.0% to 95.6%. These results place the gains across reduction, ring formation, carbon–carbon coupling, and functional-group changes, rather than restricting them to one reaction family. Table 11 provides the complete class-wise results.

Taken together, the structural and class-wise analyses separate two aspects of the task. Added-atom counts measure how much new material must enter the reactant set, whereas graph-change counts also include revisions to inherited atoms and bonds. The gains in both views, including the zero-added-atom group, show that the method is not relying on growth alone. A single complete-edit state can instead alternate between adding a component, changing its connectivity, and revising the product-derived scaffold. The complete class-wise values behind Figure 4 are given in Appendix D.3, allowing the same pattern to be inspected across chemically distinct reaction families.

4.5 Training and sampling ablations

Table 5 compares training and architecture variants with the reference results. Constant addition rate fixes the total growth rate to one while atoms remain missing, whereas uniform time sampling removes the late-state emphasis in the training distribution. The edge-feature variant removes Transformer edge-state updates while retaining bond inputs and bond-edit actions. The constant-rate variant reaches Top-10 of 79.3% under Oracle and 70.1% under Pred. on USPTO-Full, below the reference in both cases. The USPTO-50K variants likewise show lower Top-10 scores with constant-rate addition or uniform time sampling. Edge-feature updates have a smaller and less consistent effect across the two conditions. Thus, the rate controls change how the model allocates capacity over the construction path, whereas the edge-state control leaves the principal growth signal intact. The complete rankings are given in Appendix D.1.

Table 5: Training and architecture variants. Test accuracy (%) on USPTO-50K and USPTO-Full.

	Oracle		Pred.	
Variant	Top-1	Top-10	Top-1	Top-10
USPTO-50K				
Reference	72.3	**97.0**	**62.4**	**91.9**
Constant addition rate	72.1	93.0	61.4	88.1
Uniform time sampling	71.6	95.8	61.2	90.5
No edge-feature updates	**73.8**	96.4	**62.4**	91.5
USPTO-Full				
Reference	**68.6**	84.1	**54.3**	**77.1**
Constant addition rate	64.9	79.3	50.0	70.1
No edge-feature updates	68.4	**84.7**	52.1	74.8

Table 6: Supervision comparison on USPTO-50K. Accuracy (%): reference on test, training variants on validation. Column maxima are bold.

	Pred.		Oracle	
Training	Top-1	Top-10	Top-1	Top-10
Reference	62.4	**91.9**	72.3	**97.0**
+ Partial hints	62.8	90.4	72.9	94.1
+ Recovery	60.2	88.0	**73.8**	95.6
+ Both	**63.3**	90.2	73.2	95.1

Table 7: Sampler ablation. USPTO-50K Oracle test accuracy (%).

Sampling	Top-1	Top-3	Top-5	Top-10
Stratified	71.2	91.5	94.7	95.6
Independent	**72.3**	**92.9**	**96.0**	**97.0**

Complete hints supply the full reference center set; partial hints replace it with one reference-center atom on half the training examples. Recovery perturbs intermediate graphs while retaining the original reactant target. The training variants in Table 6 use a fixed training budget and one shared center predictor; validation uses 100 trajectories and 50 sampling intervals. Partial hints reach Pred. Top-1/Top-10 of 62.8/90.4%, while recovery reaches Oracle Top-10 of 95.6%. Combining both changes gives the highest Pred. Top-1 among these validation variants, 63.3%. The two training choices emphasize different parts of the conditional construction problem: partial hints improve localization under Pred., while recovery improves completion from a reference center. The complete rankings are provided in Appendix D.2.

Table 7 compares independent and stratified sampling, each using 100 trajectories and 50 sampling intervals. Independent sampling draws random numbers separately for each trajectory; stratified sampling partitions $[0, 1]$ into 100 equal intervals and assigns one to each trajectory for each edit-selection draw. Both procedures keep the model and candidate budget unchanged, so this comparison concerns the allocation of sampling variability rather than model capacity. Independent sampling remains the stronger configuration at the reported cutoffs. We therefore use it for the main comparisons and retain the stratified result as a controlled sampling check.

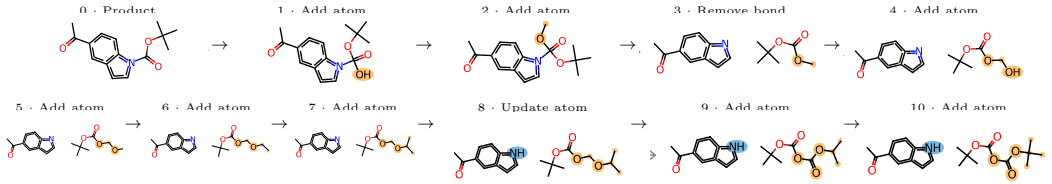


Figure 6: A sampled ten-edit construction path for a USPTO-50K protection reaction. The two rows list edits 0–4 and 5–10; the final panel is the complete reactant set.

4.6 Reactant construction examples

Complete reactant reconstruction can require recovering separate components alongside changes to the product-derived scaffold. Figure 5 shows two Oracle examples involving 15 and 10 additional atoms. In (a), RetroGEF recovers the Boc- and silyl-protected precursor together with an aryl isocyanate, whereas RetroDiT leaves the product unchanged. In (b), both methods recover the ring-opened amino alcohol, but only RetroGEF reconstructs triphosgene; RetroDiT predicts O=CCl. Both RetroGEF predictions match the complete reference topology. The second case separates scaffold recovery from reagent completion: the correct scaffold alone is insufficient. A growing graph represents both within one state, allowing generated structure to receive further edits alongside revisions to inherited atoms.

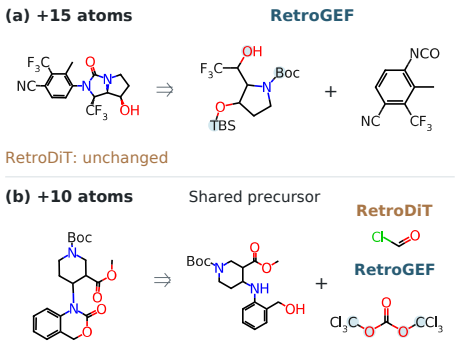


Figure 5: USPTO-Full Oracle Top-1 predictions.

Figure 6 follows this construction through ten edits, from a 19-atom protected indole to a two-component reactant set with eight additional atoms. Two additions precede cleavage of the N–carbonyl connection at edit three. Growth then continues on the separated fragment, completing di-tert-butyl dicarbonate, while an atom update restores the indole nitrogen. Disconnection therefore need not finish before growth begins: generated structure remains available for revision throughout the path. This is a graph-construction trace, not a chemical mechanism; the paired record specifies reactants but not this order. Appendix E.2 provides paths from all ten USPTO-50K reaction classes and three USPTO-Full graph-edit families, including a 14-edit construction from a 48-atom product to a 52-atom reactant set.

The path makes the variable-size representation concrete: new atoms enter only when requested, and the resulting component remains available for later edits. Pred. and Oracle change the initial center information, but both construct the same state; the gains across added-atom, graph-change, and reaction-class groups therefore reflect one shared edit space. The endpoint formulation supplies intermediate states without a recorded order, so the sampled path shows one consistent construction order rather than a chemical mechanism, alternating between component growth, connectivity change, and inherited-atom revision.

5 Conclusion

RetroGEF learns variable-size reactant construction through continuous-time complete graph edits, coupling atom creation with its first bond so generated structure can participate in later growth and revision. Paired-graph bridges provide supervision across alternative construction orders without requiring recorded edit sequences. Experiments on USPTO-Full and USPTO-50K support the formulation under predicted and reference reaction-center conditioning. Controlled ablations and structural analyses connect the gains to growth-aware editing, missing-fragment construction, and revision of inherited structure. The sampled paths make these operations explicit within individual predictions, linking endpoint supervision to the decisions required during generation.

Reproducibility Statement

Appendices A–G detail the representation, evaluation protocol, additional analyses, qualitative construction paths, and implementation details. The anonymized code link is provided in the abstract.

AI Use Statement

AI tools assisted with literature discovery, paper-structure review, code review, and language and LaTeX editing. The authors performed and verified the experiments, checked the reported numerical claims, reviewed all AI-assisted material, and remain responsible for the scientific content.

References

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-2049.

Andrew Campbell, William Harvey, Christian Weilbach, Valentin De Bortoli, Tom Rainforth, and Arnaud Doucet. Trans-dimensional generative modeling via jump diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5453–5512, 2024.

Binghong Chen, Chengtao Li, Hanjun Dai, and Le Song. Retro*: Learning retrosynthetic planning with neural guided A* search. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1608–1616, 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.

Ziqi Chen, Oluwatosin R. Ayinde, James R. Fuchs, Huan Sun, and Xia Ning. G^2Retro as a two-step graph generative models for retrosynthesis prediction. *Communications Chemistry*, 6:102, 2023. doi: 10.1038/s42004-023-00897-3.

Connor W. Coley, Luke Rogers, William H. Green, and Klavs F. Jensen. Computer-assisted retrosynthesis based on molecular similarity. *ACS Central Science*, 3(12):1237–1245, 2017. doi: 10.1021/acscentsci.7b00355.

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

Yafeng Deng, Xinda Zhao, Hanyu Sun, Yu Chen, Xiaorui Wang, Xi Xue, Liangning Li, Jianfei Song, Chang-Yu Hsieh, Tingjun Hou, Xiandao Pan, Taghrid Saad Alomar, Xiangyang Ji, and Xiaojian Wang. RSGPT: a generative transformer model for retrosynthesis planning pre-trained on ten billion datapoints. *Nature Communications*, 16:7012, 2025. doi: 10.1038/s41467-025-62308-6.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4239.

Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977. doi: 10.1021/j100540a008.

Marton Havasi, Brian Karrer, Itai Gat, and Ricky T. Q. Chen. Edit flows: Flow matching with edit operations. *arXiv preprint arXiv:2506.09018*, 2025. doi: 10.48550/arXiv.2506.09018.

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky T. Q. Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary Markov processes. In *International Conference on Learning Representations*, 2025.

Ilia Igashov, Arne Schneuing, Marwin Segler, Michael M. Bronstein, and Bruno Correia. RetroBridge: Modeling retrosynthesis with Markov bridges. In *International Conference on Learning Representations*, 2024.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 10362–10383, 2022.

Najwa Laabid, Severi Rissanen, Markus Heinonen, Arno Solin, and Vikas Garg. Equivariant denoisers cannot copy graphs: Align your graph diffusion models. In *International Conference on Learning Representations*, 2025.

Zaiyun Lin, Shiqiu Yin, Lei Shi, Wenbiao Zhou, and Yingsheng John Zhang. G2GT: Retrosynthesis prediction with graph-to-graph attention neural network and self-training. *Journal of Chemical Information and Modeling*, 63(7):1894–1905, 2023. doi: 10.1021/acs.jcim.2c01302.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Bowen Liu, Bharath Ramsundar, Prasad Kawthekar, Jade Shi, Joseph Gomes, Quang Luu Nguyen, Stephen Ho, Jack Sloane, Paul Wender, and Vijay Pande. Retrosynthetic reaction prediction using neural sequence-to-sequence models. *ACS Central Science*, 3(10):1103–1113, 2017. doi: 10.1021/acscentsci.7b00303.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32819–32848, 2024.

Daniel Lowe. Chemical reactions from US patents (1976–Sep2016). Figshare dataset, 2017.

Youzhi Luo, Keqiang Yan, and Shuiwang Ji. GraphDF: A discrete flow model for molecular graph generation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7192–7203, 2021.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. SPECTRE: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15159–15179, 2022.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 4474–4484, 2020.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete flow matching for graph generation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50269–50326, 2025.

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.

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. doi: 10.1021/acs.jcim.6b00564.

Philippe Schwaller, Benjamin Hoover, Jean-Louis Reymond, Hendrik Strobelt, and Teodoro Laino. Extraction of organic chemistry grammar from unsupervised learning of chemical reactions. *Science Advances*, 7(15):eabe4166, 2021. doi: 10.1126/sciadv.abe4166.

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

Seung-Woo Seo, You Young Song, June Yong Yang, Seohui Bae, Hankook Lee, Jinwoo Shin, Sung Ju Hwang, and Eunho Yang. GTA: Graph truncated attention for retrosynthesis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 531–539, 2021. doi: 10.1609/aaai.v35i1.16131.

Chence Shi, Minkai Xu, Hongyu Guo, Ming Zhang, and Jian Tang. A graph to graphs framework for retrosynthesis prediction. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8818–8827, 2020a.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. GraphAF: A flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2020b.

Vignesh Ram Somnath, Charlotte Bunne, Connor W. Coley, Andreas Krause, and Regina Barzilay. Learning graph models for retrosynthesis prediction. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Xiaozhuang Song, Xuanhao Pan, Xinjian Zhao, Hangting Ye, Shufei Zhang, Jian Tang, and Tianshu Yu. AOT*: Efficient synthesis planning via LLM-empowered AND-OR tree search. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 34727–34758, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1734.

Igor V. Tetko, Pavel Karpov, Ruud Van Deursen, and Guillaume Godin. State-of-the-art augmented NLP transformer models for direct and single-step retrosynthesis. *Nature Communications*, 11: 5575, 2020. doi: 10.1038/s41467-020-19266-y.

Zhengkai Tu and Connor W. Coley. Permutation invariant graph-to-sequence model for template-free retrosynthesis and reaction prediction. *Journal of Chemical Information and Modeling*, 62 (15):3503–3513, 2022. doi: 10.1021/acs.jcim.2c00321.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations*, 2023.

Chenguang Wang, Zihan Zhou, Lei Bai, and Tianshu Yu. Order matters in retrosynthesis: Structure-aware generation via reaction-center-guided discrete flow matching. In *International Conference on Machine Learning*, 2026.

Xiaorui Wang, Yuquan Li, Jiezhong Qiu, Guangyong Chen, Huanxiang Liu, Benben Liao, Chang-Yu Hsieh, and Xiaojun Yao. Retroprime: A diverse, plausible and transformer-based method for single-step retrosynthesis predictions. *Chemical Engineering Journal*, 420:129845, 2021. doi: 10.1016/j.cej.2021.129845.

Yiming Wang, Yuxuan Song, Yiqun Wang, Minkai Xu, Rui Wang, Hao Zhou, and Wei-Ying Ma. RetroDiff: Retrosynthesis as multi-stage distribution interpolation. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 1918–1926. PMLR, 2025.

David Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988. doi: 10.1021/ci00057a005.

Robin Yadav, Qi Yan, Guy Wolf, Joey Bose, and Renjie Liao. Retro synflow: Discrete flow-matching for accurate and diverse single-step retrosynthesis. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Chaochao Yan, Qianggang Ding, Peilin Zhao, Shuangjia Zheng, Jinyu Yang, Yang Yu, and Junzhou Huang. RetroXpert: Decompose retrosynthesis prediction like a chemist. In *Advances in Neural Information Processing Systems*, volume 33, pp. 11248–11258, 2020.

Lin Yao, Wentao Guo, Zhen Wang, Shang Xiang, Wentan Liu, and Guolin Ke. Node-aligned graph-to-graph: Elevating template-free deep learning approaches in single-step retrosynthesis. *arXiv preprint arXiv:2309.15798*, 2023. doi: 10.48550/arXiv.2309.15798.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, volume 80, pp. 5708–5717, 2018.

Weihe Zhong, Ziduo Yang, and Calvin Yu-Chian Chen. Retrosynthesis prediction using an end-to-end graph generative architecture for molecular graph editing. *Nature Communications*, 14:3009, 2023. doi: 10.1038/s41467-023-38851-5.

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. doi: 10.1039/D2SC02763A.

A Representation and reconstruction

Atom and bond attributes. Atom attributes are represented by $(z, q, h, r, \iota, \chi)$: atomic number, formal charge, explicit-hydrogen count, radical-electron count, isotope, and atom chirality. Bond attributes are represented by (b, s): bond type and bond stereochemistry. Aromaticity, conjugation, hybridization, and implicit valence are not independent generated channels; RDKit derives them after the explicit graph has been reconstructed.

Atom-order equivalence. For a reactant graph R, let

$$\langle R \rangle = \{\gamma R : \gamma \in \mathfrak{S}(V_R)\} \tag{6}$$

denote all permutations of atom indices for the same attributed graph. Molecular matching compares this equivalence class after removing atom maps and canonicalizing the decoded components.

Atom indices and correspondence. Atom mapping identifies corresponding atoms across a reaction; RXNMapper, for example, infers these correspondences from learned attention (Schwaller et al., 2021). We use the provided atom maps to partition reactant atoms into inherited product atoms and reactant-only atoms. The indices of product atoms, $0, \ldots, |V_P| - 1$, are fixed. Let $\Sigma_{R|P}$ be the permutations that fix these product atoms and permute only the indices of reactant-only atoms. The target graph represents the orbit

$$[R]_P = \{\sigma R : \sigma \in \Sigma_{R|P}\}. \tag{7}$$

Terminal decoding removes atom indices and maps every representative of $[R]_P$ to the same molecular equivalence class $\langle R \rangle$. In the quotient bridge, a reactant-only atom receives the next available graph index when created; the corresponding permutation is applied consistently to the endpoint and all subsequent bridge targets.

Graph changes relative to the product. The representation records the atoms and bonds that differ from P: complete reactant-only atoms, changed product-atom attributes, missing product bonds, added bonds, and changed bond fields. Applying these changes to P creates a labelled representative of $[R]_P$. Because all explicitly represented atom and bond attributes are either inherited unchanged or specified by these changes, decoding and canonicalization recover the same chemical graph whenever RDKit accepts the explicit valence state. Preprocessing rejects records that fail this reconstruction check.

In the processed benchmark pairs, the full target reactant set retains the product atoms, so its total atom count is unchanged or larger. It is larger for 93.4% and 92.3% of evaluated USPTO-50K and USPTO-Full test products, respectively.

B GRAPH-EDIT VOCABULARY AND BRIDGE CONSTRUCTION

Table 8: Complete graph edits and their state effects. A selected edit contains all fields needed to construct its deterministic successor. Here $\mathbf{m}$ and $\mathbf{e}$ are complete atom and bond attribute tuples, and $d(u)$ is the current degree of a deleted atom. All edit types preserve unrelated graph records.

Edit type	Location	Attributes	State change	
			$\Delta\|V\|$	$\Delta\|E\|$
Attached atom addition	existing atom	$\mathbf{m}, \mathbf{e}$	$+1$	$+1$
Isolated atom addition	graph	$\mathbf{m}$	$+1$	0
Generated-atom deletion	most recent generated leaf/isolated u	none	-1	$-d(u)$
Bond addition	absent unordered pair	$\mathbf{e}$	0	$+1$
Bond deletion	existing bond	none	0	-1
Atom-attribute update	existing atom	replacement $\mathbf{m}$	0	0
Bond-attribute update	existing bond	replacement $\mathbf{e}$	0	0

Corrective edits. For a recorded reactant graph R, let $\mathcal{C}_R(X)$ contain one correction for each mismatched atom tuple shared by the two graphs, one deletion for each bond absent from the target graph, one attribute update for each mismatched bond tuple, and one addition for each missing target-graph bond whose atoms already exist. Attribute corrections write the complete target tuple. These edits have unit rate in τ. On bridge paths, created atoms always belong to the target graph. Deleting a generated atom has zero target rate. The training perturbations in Appendix C.1 can create superfluous atoms and thereby provide examples of this correction.

Atom-addition distribution. Let $m_R(X)$ be the number of missing reactant-only atoms. The total atom-addition rate is $m_R(X)$, distributed according to $\rho_R(a \mid X)$ in Equation 1. Write this rate measure as $q_R^{\mathrm{B}}(a \mid X) = m_R(X)\rho_R(a \mid X)$. For the quotient construction used by the conditioned generator, a missing atom with a target neighbour proposes an attached atom addition for each such neighbour. When at least one attached proposal exists, only attached additions are admitted; otherwise, every missing atom can be added as an isolated atom. Let $F_R(X)$ be the resulting eligible atoms and $\mathcal{B}_i(X)$ the complete addition actions for atom i. For $m_R(X) > 0$, the atom-addition rate is

$$q_R^{\mathrm{B}}(a \mid X) = \sum_{i \in F_R(X)} \mathbf{1}[a \in \mathcal{B}_i(X)] \frac{m_R(X)}{|F_R(X)|\,|\mathcal{B}_i(X)|}. \tag{8}$$

Each eligible atom receives total rate $m_R(X)/|F_R(X)|$, distributed uniformly among its possible attachment atoms. Identical observable actions are merged by summation. After sampling an atom-addition proposal, the bridge updates the target correspondence to associate the new node with the chosen target atom and applies the corresponding fresh-identity permutation to the remaining endpoint. This quotient operation is not supplied to the rate network.

For either atom-addition distribution, the complete target rate is

$$q_R(a \mid X) = \mathbf{1}[a \in \mathcal{C}_R(X)] + q_R^{\mathrm{B}}(a \mid X), \tag{9}$$

where the atom-addition term is zero when no atom is missing. Rates are defined per unit of $\tau = -\log(1-t)$; per unit of t, they are multiplied by $d\tau/dt = 1/(1-t)$. Let $D_R(X, \kappa)$ count disagreeing atom and bond tuples, including missing records, under the current correspondence κ, as formalized in Equation 13. Every bridge edit reduces this discrepancy. The target graph is reached after finitely many jumps and is absorbing for the bridge.

C EXPERIMENTAL SETUP

C.1 MODEL AND TRAINING

The generator uses a 16-layer graph Transformer with 768-dimensional node features and 384-dimensional edge features. We implement it in PyTorch with BF16 mixed-precision training on a

single compute node, using distributed data parallelism for multi-GPU runs. RDKit handles molecular parsing and reconstruction.

All comparisons use the same graph-Transformer family, complete-edit representation, and rate-matching objective within each dataset. Models are trained separately for the two center-information conditions, and the checkpoint used for testing is selected on validation before the test split is consulted. The same decoder is used across the center-information conditions.

Bridge states and recovery augmentation. Training covers both early and late construction states in transformed time $\tau = -\log(1-t)$. When enabled, recovery augmentation perturbs intermediate states while retaining supervision toward the same reactant endpoint.

C.2 Benchmarks and center inputs

Tables 1 and 2 use the published baseline scores and benchmark conventions of RSGPT and RetroDiT (Deng et al., 2025; Wang et al., 2026). The USPTO-50K models use the original training split and evaluate 4,944 of 5,007 test products under the ten-new-atom cap. USPTO-Full evaluation contains 93,289 products.

Reaction-center inputs. Pred. supplies each conditioned trajectory with a predicted center, held fixed during generation, without reference annotations. Oracle supplies the reference center information used by the benchmark protocol.

C.3 Evaluation and decoding

Following RetroDiT (Wang et al., 2026), we compare reactant sets by atom-and-bond topology and rank distinct decoded candidates by sampling frequency. Top-k accuracy uses all evaluated products as its denominator.

The accumulated-hazard sampler is specified in Algorithm 2. The benchmark uses 100 candidate trajectories and 50 integration intervals per product, with a decoder budget of at most 128 edits and ten/twenty newly created atoms on USPTO-50K/Full.

D Additional experimental results

D.1 Training and architecture variants

Table 9: Training and architecture variants. Test Top-$1/3/5/10$ accuracy (%).

Dataset	Variant	Oracle				Pred.			
		1	3	5	10	1	3	5	10
USPTO-50K	Reference	72.3	**92.9**	**96.0**	**97.0**	**62.4**	82.8	**88.6**	**91.9**
	Constant addition rate	72.1	89.0	91.3	93.0	61.4	80.9	85.1	88.1
	Uniform time sampling	71.6	91.5	94.5	95.8	61.2	82.3	87.4	90.5
	No edge-feature updates	**73.8**	92.6	95.5	96.4	**62.4**	**83.8**	88.4	91.5
USPTO-Full	Reference	**68.6**	80.7	82.8	84.1	**54.3**	**70.1**	**74.1**	**77.1**
	Constant addition rate	64.9	74.9	77.2	79.3	50.0	63.6	67.1	70.1
	No edge-feature updates	68.4	**80.8**	**83.1**	**84.7**	52.1	67.6	71.6	74.8

D.2 Supervision ablation

Table 10 expands the supervision ablation in Table 6. The training variants use the same graph representation, split, and validation protocol. Pred. uses one shared center predictor; Oracle uses the reference center information.

Table 10: Supervision comparison on USPTO-50K. Top-k accuracy (%): reference on test, training variants on validation. Column maxima are bold.

Condition	Partial hints	Recovery	Top-k			
			1	3	5	10
Pred.	Reference		62.4	82.8	**88.6**	**91.9**
	✓	–	62.8	**83.6**	88.0	90.4
	–	✓	60.2	81.1	85.4	88.0
	✓	✓	**63.3**	82.8	88.2	90.2
Oracle	Reference		72.3	**92.9**	**96.0**	**97.0**
	✓	–	72.9	91.5	93.5	94.1
	–	✓	**73.8**	92.6	95.1	95.6
	✓	✓	73.2	91.3	94.4	95.1

D.3 REACTION-CLASS ANALYSIS

The reaction-class analysis uses 4,930 paired USPTO-50K products under the same matching definition and candidate budget as the main evaluation.

D.3.1 REACTION CLASSES ON USPTO-50K

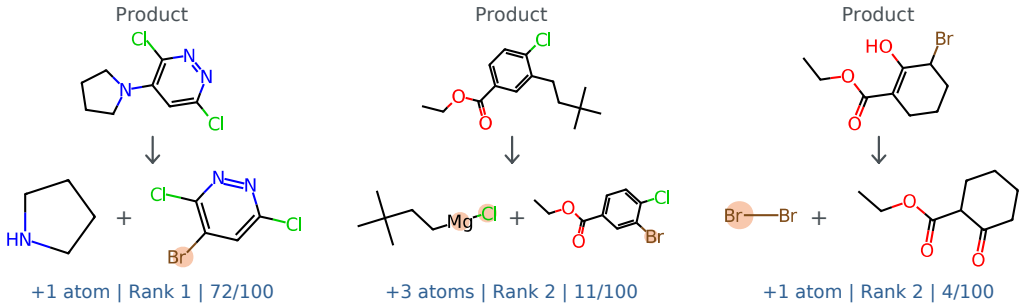
Table 11: USPTO-50K accuracy by reaction class. Top-k accuracy (%) on 4,930 paired products; bold denotes the larger value in each pair. FG denotes functional-group interconversion.

Reaction class	n	Method	Pred.				Oracle			
			Top-1	Top-3	Top-5	Top-10	Top-1	Top-3	Top-5	Top-10
Alkyl./aryl.	1,516	RetroDiT	65.1	**85.4**	90.6	**94.3**	73.7	92.2	96.4	**98.3**
		RETROGEF	**67.5**	**85.4**	**90.8**	93.6	**74.5**	**94.2**	**97.6**	98.0
Acylation	1,188	RetroDiT	74.9	91.9	94.6	96.2	**82.2**	**97.4**	**98.2**	**98.7**
		RETROGEF	**75.4**	**92.8**	**95.8**	**97.0**	81.4	97.0	98.0	98.1
C–C formation	511	RetroDiT	**41.9**	63.2	69.9	75.0	55.4	80.2	87.1	92.0
		RETROGEF	40.7	**65.9**	**76.1**	**82.6**	**56.2**	**82.4**	**88.3**	**92.4**
Heterocycles	89	RetroDiT	6.7	7.9	7.9	7.9	69.7	79.8	82.0	84.3
		RETROGEF	**53.9**	**71.9**	**73.0**	**78.7**	**77.5**	**91.0**	**92.1**	**95.5**
Protection	67	RetroDiT	86.6	89.6	89.6	89.6	**95.5**	**100.0**	**100.0**	**100.0**
		RETROGEF	**88.1**	**92.5**	**98.5**	**100.0**	**95.5**	97.0	97.0	97.0
Deprotection	819	RetroDiT	**56.4**	**84.7**	**90.6**	**94.1**	**59.8**	88.8	93.8	96.5
		RETROGEF	51.2	82.2	88.8	92.7	55.2	**89.6**	**94.9**	**96.8**
Reduction	453	RetroDiT	55.2	76.2	82.1	85.7	71.3	90.3	93.4	95.1
		RETROGEF	**61.6**	**78.4**	**84.3**	**89.8**	**84.8**	**96.7**	**97.6**	**97.8**
Oxidation	82	RetroDiT	**69.5**	**87.8**	91.5	91.5	**81.7**	**93.9**	95.1	**96.3**
		RETROGEF	56.1	80.5	**92.7**	**95.1**	**81.7**	**93.9**	**96.3**	**96.3**
FG interconv.	182	RetroDiT	**44.0**	61.5	67.6	69.2	57.1	83.0	87.9	89.0
		RETROGEF	**44.0**	**62.1**	**70.3**	**75.3**	**67.6**	**87.9**	**95.1**	**95.6**
FG addition	23	RetroDiT	**82.6**	**91.3**	**91.3**	91.3	78.3	91.3	91.3	95.7
		RETROGEF	**82.6**	**91.3**	**91.3**	**100.0**	**95.7**	**100.0**	**100.0**	**100.0**

E QUALITATIVE CONSTRUCTION PATHS

E.1 REACTANT CONSTRUCTION EXAMPLES

Figure 7 shows three further USPTO-Full Oracle construction examples.



(a) Leaving-group recovery. (b) Coupling-partner recovery. (c) Bond-order changes.

Figure 7: Reactant construction on USPTO-Full under Oracle conditioning. Added atoms are orange; the panels illustrate fragment recovery, coupling partner recovery, and bond-order revision.

Panel (a) combines one added bromine atom with separation of the cyclic amine from the heteroaromatic scaffold. Panel (b) requires three added atoms distributed across the two reactants, including an organomagnesium coupling partner. Panel (c) combines one added atom with ring bond-order changes.

E.2 SAMPLED CONSTRUCTION PATHS

Figures 8–10 show complete sampled construction paths under Oracle conditioning. Each example includes the product and the successive edits; waiting intervals with no graph change are omitted. These are construction states, not chemical reaction mechanisms.

Chemical reaction classes on USPTO-50K. Figures 8 and 9 cover the ten USPTO-50K reaction classes. The paths range from a single atom addition to 14 edits, including trajectories that interleave fragment growth with atom and bond updates.

Graph-edit families on USPTO-Full. The USPTO-Full records used here have no explicit chemical reaction-class labels. Figure 10 instead groups examples by their graph changes. The first path starts from a 48-atom product and reaches a 52-atom reactant set in 14 edits: four atom additions, eight bond-attribute updates, one atom-attribute update, and one bond removal. The other two retain the product atom count while updating atom attributes and bonds, or bond attributes alone.

F RATE MATCHING AND BRIDGE MARGINALS

The target bridge is Markov on a state that includes more information than the rate network observes. Besides the current graph X_t, it retains the recorded reactant graph R and a correspondence κ_t from current atoms to target atoms. Write $U_t = (R, \kappa_t)$ for this hidden information. The notation q_R in the main text suppresses the correspondence. Throughout this appendix, the product and any supplied reaction-center condition are fixed visible conditioning variables. We suppress the center argument in the formulas, including conditional expectations. Unadorned rates q and r are per unit of transformed time τ.

We state the results for a finite representable graph state space, obtained with finite attribute vocabularies, an atom cap, and a finite set of atom indices. Targets in the conditional distribution and their bridge edits are represented in this space; endpoint validity is checked during decoding.

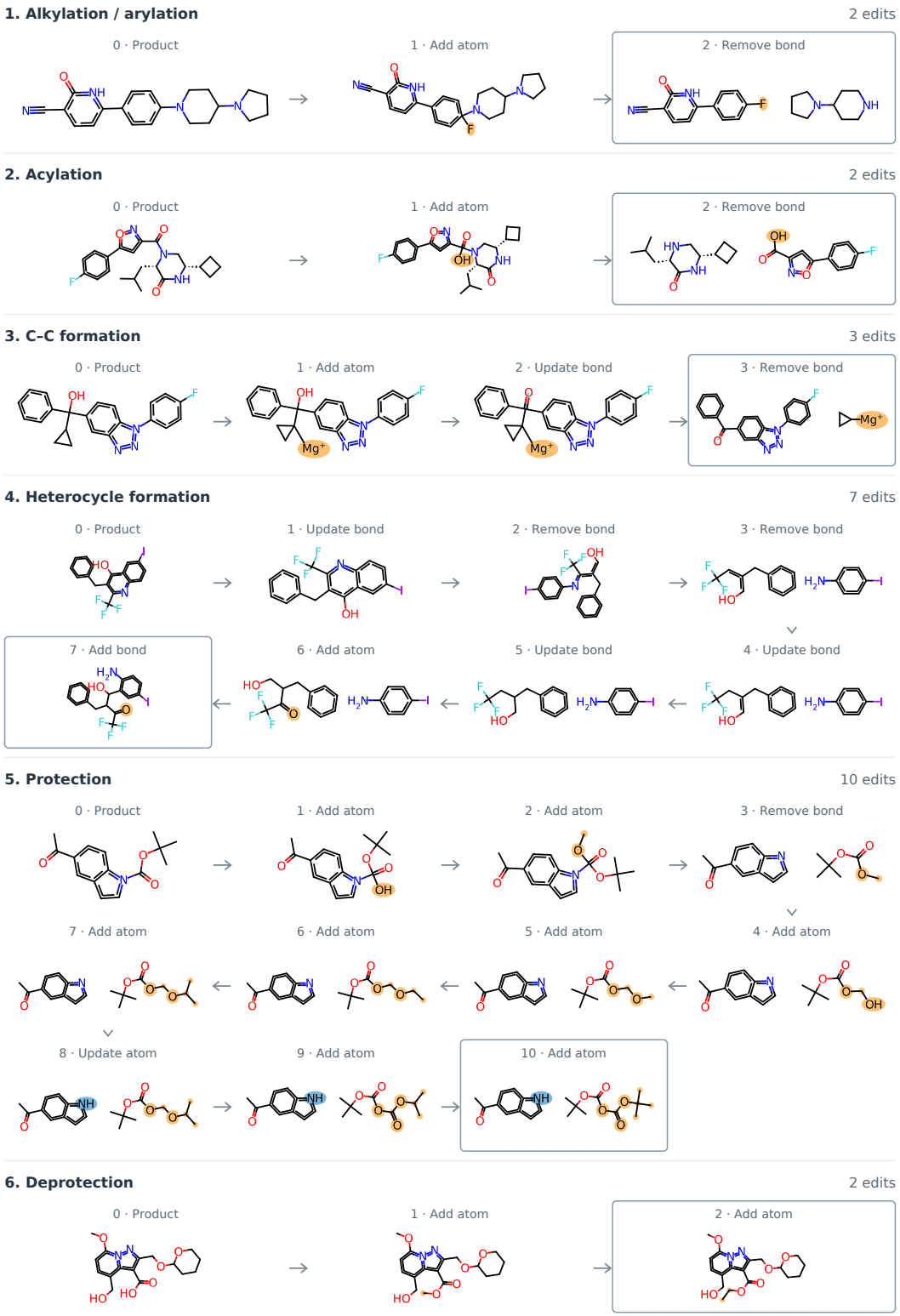


Figure 8: Complete trajectories for USPTO-50K classes 1–6. Orange marks added atoms and their bonds; blue marks changes to inherited atoms or bonds. Every arrow represents one edit.

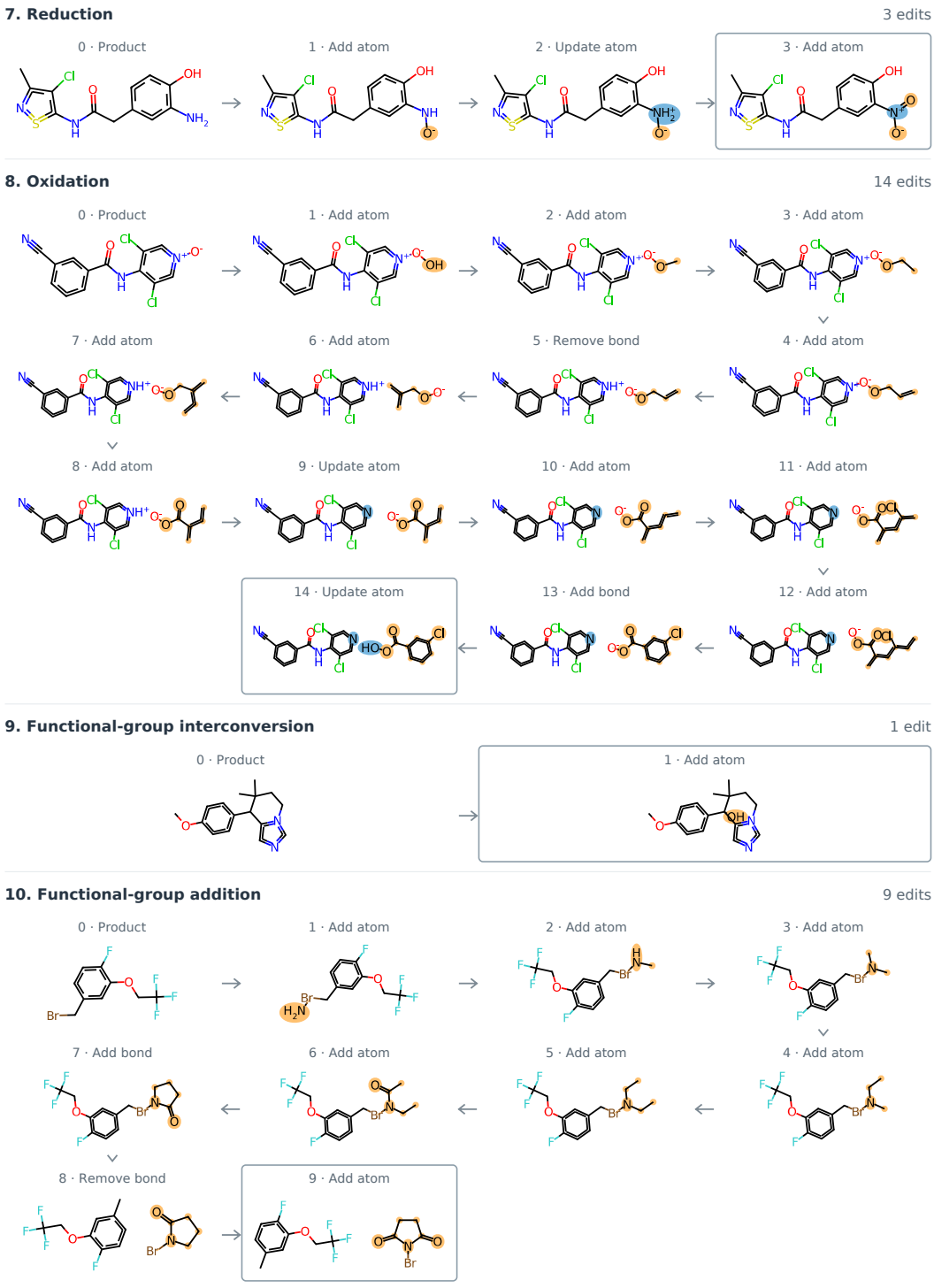


Figure 9: Complete trajectories for USPTO-50K classes 7–10. The oxidation example shows all 14 edits, including ten atom additions.

F.1 From latent proposals to graph transitions

A complete action specifies an observable edit, whereas a bridge proposal can also specify which target atom it realizes. Let $\mathcal{H}_U(X)$ be the finite proposal set, $w_U(h \mid X)$ its nonnegative rates, and $o_X(h)$ the complete action obtained by omitting the proposal's hidden target assignment. The action

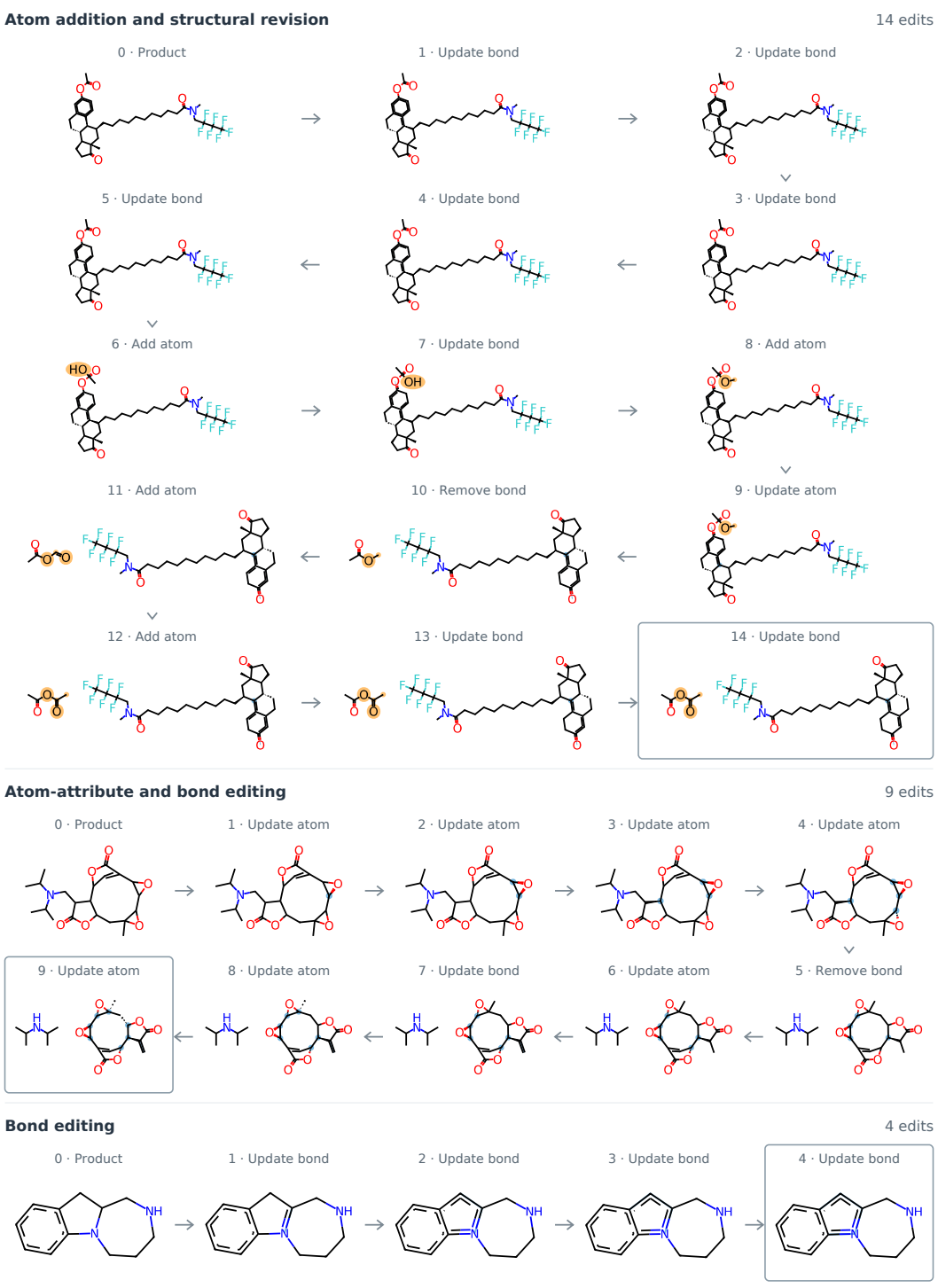


Figure 10: Complete trajectories across USPTO-Full graph-edit families. The three paths contain 14, nine, and four edits; each path is shown on its recorded inference grid.

rate is the pushforward of the proposal rates:

$$q_U(a \mid X) = \sum_{h \in \mathcal{H}_U(X):\, o_X(h)=a} w_U(h \mid X). \tag{10}$$

For quotient atom addition, different missing target atoms can therefore contribute to the same attached atom-addition action. If a simulator samples the merged action first, it recovers a compatible

latent proposal with probabilities $w_U(h \mid X)/q_U(a \mid X)$. This second draw updates the correspondence used by subsequent bridge steps.

Actions can be aggregated once more when several complete edits have the same successor. For $Y \neq X$, the graph transition rate and its diagonal entry are

$$Q_U^\tau(X,Y) = \sum_{a:T_aX=Y} q_U(a \mid X), \qquad Q_U^\tau(X,X) = -\sum_{Y\neq X} Q_U^\tau(X,Y). \tag{11}$$

We take actions to be genuine edits, so $T_aX \neq X$. Null actions, if present in a proposal representation, contribute nothing to a graph generator and can be omitted. The distinction between proposals, actions, and successors avoids treating an unobserved target assignment as an extra graph transition.

Replacing target rates by learned rates gives the generator acting on any real-valued function f of the graph:

$$(\mathcal{Q}_\theta^\tau f)(X;P) = \sum_{a\in\mathcal{A}(X)} r_\theta(a \mid X, P, t(\tau))\big[f(T_aX) - f(X)\big]. \tag{12}$$

The action distribution is useful computationally, but the sums in Equation 11 determine the evolution of graph probabilities. Learning all action rates is sufficient to learn these sums.

F.2 A DECREASING DISCREPANCY AND FINITE ABSORPTION

Fix a target R. On an unperturbed bridge, every current atom corresponds to a distinct target atom, with product-atom correspondences fixed. Existing assignments are never revised. An atom addition extends κ to one previously missing target atom. Equivalently, each realized path can be compared with its target using one consistent set of target atom indices.

These shared indices make discrepancies involving uncreated atoms well-defined. Let $\nu_X^\kappa(i)$ be the current atom tuple at target index i, or a missing-node symbol if that atom does not yet exist. Let $\beta_X^\kappa(i,j)$ be the current bond tuple, with a no-bond symbol when a bond or either endpoint is absent. Define

$$D_R(X,\kappa) = \sum_{i\in V_R} \mathbf{1}[\nu_X^\kappa(i) \neq \nu_R(i)] + \sum_{\{i,j\}\subset V_R} \mathbf{1}[\beta_X^\kappa(i,j) \neq \beta_R(i,j)]. \tag{13}$$

The second sum counts missing target bonds even before their endpoints are created. Counting only discrepancies among currently existing atoms would not give the required monotonicity: an atom addition could expose additional missing bonds.

Proposition 1 (Absorption of a target bridge)**.** *Suppose the target contains all product atoms, agrees on their immutable attributes, and the atom-addition and other corrective actions described in Appendix B are admissible. Starting at P, every bridge jump strictly decreases D_R. The bridge reaches a representative of $[R]_P$ after at most $D_R(P,\kappa_0)$ jumps and in finite transformed time almost surely. That terminal state is absorbing for the target bridge.*

Proof. A corrective edit writes a disagreeing record to its target value; an attached atom addition also writes its incident target bond. No correct record is changed, so the nonnegative integer D_R decreases at every jump. Before absorption, the bridge rate $\Lambda_U(X) = m_R(X) + |\mathcal{C}_R(X)|$ is finite and at least one. Hence at most $D_R(P,\kappa_0)$ exponential holding times are needed, their sum is finite almost surely, and both rate terms vanish at the terminal representative. □

The argument permits different construction orders, including the two in Figure 2, without requiring intermediate graphs to satisfy molecular valence rules.

F.3 GENERALIZED KL SEPARATES INTENSITY FROM EDIT CHOICE

For finite nonnegative action-rate measures q and r, define

$$D_{\mathrm{GKL}}(q\|r) = \sum_a \left[q(a)\log\frac{q(a)}{r(a)} - q(a) + r(a)\right]. \tag{14}$$

We use the conventions $0 \log(0/r) = 0$ and infinite divergence when $q(a) > 0 = r(a)$. Let $\Lambda = \sum_a q(a)$ and $\lambda = \sum_a r(a)$. When both are positive, set $\nu = q/\Lambda$ and $\pi = r/\lambda$. Substituting these factorizations and using $\sum_a \nu(a) = 1$ gives

$$D_{\mathrm{GKL}}(q\|r) = \underbrace{\Lambda \log \frac{\Lambda}{\lambda} - \Lambda + \lambda}_{\text{intensity divergence}} + \Lambda D_{\mathrm{KL}}(\nu\|\pi). \tag{15}$$

Thus action-choice errors are weighted by the target edit intensity. Removing terms independent of the model leaves $\lambda - \Lambda \log \lambda - \sum_a q(a) \log \pi(a)$, exactly the loss in Equation 4. When $\Lambda = 0$, the divergence is simply λ: an absorbed target supplies intensity supervision but no preferred next action. The optimum lies at zero intensity, a boundary approached rather than attained by a finite softplus output.

Scaling both rate measures by $c > 0$ scales their divergence by c. Consequently using rates per unit of t would multiply the statewise loss by $1/(1-t)$. The manuscript instead matches transformed-time rates. Different positive time-sampling weights preserve the unrestricted statewise optimum, but need not produce the same finite-capacity fit.

F.4 Optimal prediction with a hidden target and correspondence

Let $Z = (X_t, P, t)$ denote the visible conditioning, including any supplied center as above. Define $\bar{q}_Z(a) = \mathbb{E}[q_{U_t}(a \mid X_t) \mid Z]$. All latent alternatives are expressed on the same visible action set, with zero rate assigned to actions absent from a particular target's support.

Proposition 2 (Conditional-mean rate)**.** *For a finite action set and finite conditional expectations of $q(a)$ and $q(a) \log q(a)$, the minimizer of conditional expected generalized KL over nonnegative rate measures is $r^*(a \mid Z) = \bar{q}_Z(a)$.*

Proof. For each action, the model-dependent conditional risk is $r(a) - \bar{q}_Z(a) \log r(a)$. If $\bar{q}_Z(a) > 0$, its derivative is $1 - \bar{q}_Z(a)/r(a)$, with a unique minimum at $r(a) = \bar{q}_Z(a)$. If the mean is zero, the risk is $r(a)$, minimized at zero. Equivalently, direct expansion gives the excess-risk identity

$$\mathbb{E}[D_{\mathrm{GKL}}(q_{U_t}\|r) \mid Z] = \mathbb{E}[D_{\mathrm{GKL}}(q_{U_t}\|\bar{q}_Z) \mid Z] + D_{\mathrm{GKL}}(\bar{q}_Z\|r). \tag{16}$$

□

The first term in Equation 16 reflects uncertainty about the hidden target and correspondence even when the visible-state rate is predicted optimally. When $\bar{\Lambda}_Z = \mathbb{E}[\Lambda_{U_t} \mid Z] > 0$, the optimal factorization is

$$\lambda^*(Z) = \bar{\Lambda}_Z, \qquad \pi^*(a \mid Z) = \frac{\mathbb{E}[q_{U_t}(a \mid X_t) \mid Z]}{\bar{\Lambda}_Z}. \tag{17}$$

This is an intensity-weighted average of target action distributions, not their unweighted average.

F.5 Consistency of one-time graph marginals

The generator-matching argument (Holderrieth et al., 2025) applies to the visible projection even when that projection is not itself Markov. Let $p_\tau(x)$ be the conditional bridge law given the visible inputs, with locally integrable rates on the finite state space.

Proposition 3 (Graph-edit marginal consistency)**.** *Fix the product and any supplied center condition. Assume the unperturbed bridges have a finite admissible state space containing their targets, consistently extended atom correspondences, and locally integrable rates defining non-explosive processes. Over unrestricted nonnegative rates, the population optimum of Equation 4, per unit τ, is the conditional mean in Equation 5. The expectation includes hidden atom correspondences. The time-inhomogeneous Markov process initialized at P with these rates has marginal law p_τ at every finite τ.*

Proof. Proposition 2 gives the conditional-mean optimum in Equation 5. For any graph function f, conditioning the bridge generator on the visible state replaces q_{U_t} by its conditional mean r^*. The conditional bridge law and the Markov process with rates r^* therefore satisfy the same finite forward equation and initial condition; uniqueness gives equal one-time marginals. □

Under Proposition 1, each bridge eventually reaches its own target representative. For targets accepted by the reconstruction check in Appendix A, decoding maps the limiting marginal to the recorded-target distribution used to construct the bridges. The learned sampler approximates this population process with a fitted rate network, a finite horizon, and numerical integration.

F.6 Perturbed training states and numerical generation

State perturbations augment the training population with recoverable graph errors. If $\widetilde{U}_t$ retains the target and its correspondence with a perturbed graph $\widetilde{X}_t$, recomputing corrective rates gives

$$\widetilde{r}^*(a \mid \widetilde{X}_t, P, t) = \mathbb{E}_{\mathrm{aug}}[q_{\widetilde{U}_t}(a \mid \widetilde{X}_t) \mid \widetilde{X}_t, P, t], \tag{18}$$

which supervises recovery without assigning a unique correction order.

G Algorithms and state evolution

G.1 Graph state and atom indices

The initial state is P. A new atom receives the largest current atom index plus one; subsequent edits can revise its attributes or incident bonds. Surviving atoms retain their indices after a deletion.

Attribute tuples and complete edits follow Appendices A and B. Bond absence is represented by a missing edge.

The state update can delete an isolated or leaf generated node; the rate network restricts this action to the most recently added generated node (the highest allocated atom index among generated atoms). A leaf deletion removes its incident bond. Atom-addition masks enforce the new-atom cap.

G.2 Constructing bridge states and sparse rate targets

During a quotient bridge, the endpoint indices assigned to uncreated atoms can change as atoms are added. The endpoint is relabelled to extend the correspondence with the newly created atom, while existing state identities and assignments remain unchanged. Denote the endpoint together with this correspondence by U. The model never receives U or a fixed slot assignment.

The bridge first forms weighted proposals (a, w, i), where a is a complete observable edit and i is the endpoint atom represented by an atom addition, or an empty tag for other corrections. Proposal weights follow Appendix B. An attached atom addition supplies only one first bond. Other endpoint bonds incident to the new node become eligible bond-addition corrections once both of their atoms exist. In perturbed states, corrections can also remove a superfluous newest isolated or leaf atom.

Distinct hidden endpoint atoms can propose the same observable action. Bridge simulation samples the tagged proposals, retaining the selected endpoint identity for the next correspondence update. The training target instead merges them:

$$q_U(a \mid X) = \sum_{(a', w, i):\, a' = a} w. \tag{19}$$

Thus the learner is not asked to distinguish endpoint assignments that yield the same current edit. The total rate is unchanged by this summation. Algorithm 1 uses exponential waiting times to simulate the bridge (Gillespie, 1977).

Optimization averages the loss in Algorithm 1 over the minibatch.

G.3 Factorization of edit probabilities

The probability of a complete edit is evaluated through a hierarchy of normalized conditional distributions. Let $Z = (P, X, t)$ denote the visible context. The graph readout predicts total intensity and edit-type probabilities. Type-specific distributions then select locations and attributes. For an attached atom addition at existing atom j, the model jointly selects j and the new element z. The

Algorithm 1 Bridge supervision for one training pair

Require: Product P, reactants R, rate network θ
Ensure: Rate-matching loss for one sampled state
Sample $t_\star$ from the training time distribution; $\tau_\star \leftarrow -\log(1 - t_\star)$
$X \leftarrow P$, $\tau \leftarrow 0$; initialize endpoint correspondence U from (P, R)
while $\tau < \tau_\star$ **do**
  $\mathcal{Q} \leftarrow \textsc{TaggedCorrections}(X, U)$; $\Lambda \leftarrow \sum_{(a,w,i)\in\mathcal{Q}} w$
  if $\Lambda = 0$ **then**
    break
  end if
  Draw $\Delta\tau \sim \mathrm{Exp}(\Lambda)$
  if $\tau + \Delta\tau > \tau_\star$ **then**
    break
  end if
  Draw $(a, w, i) \in \mathcal{Q}$ with probability w/Λ
  $X \leftarrow T_a X$; $\tau \leftarrow \tau + \Delta\tau$
  Update U using i if a adds an atom
end while
Optionally apply recoverable perturbations to X
Recompute $\mathcal{Q}$; merge identical actions into q_U using Eq. 19
$(\lambda_\theta, \pi_\theta) \leftarrow \textsc{RateNetwork}(P, X, t_\star)$
return $\mathcal{L}_{\mathrm{rate}}(\lambda_\theta, \pi_\theta; q_U)$ from Eq. 4

remaining atom attributes are generated autoregressively in tuple order. Writing f_{A} for this edit type as in the main text, the factorization is

$$\pi_\theta(f_{\mathrm{A}}, j, \mathbf{m}, \mathbf{e} \mid Z) = \pi_F(f_{\mathrm{A}} \mid Z)\pi_{J,z}(j, z \mid f_{\mathrm{A}}, Z) \times \prod_{k=2}^{6} \pi_k(m_k \mid m_{<k}, j, Z)\pi_E(\mathbf{e} \mid \mathbf{m}, j, Z). \tag{20}$$

The new atom's index is determined by the current state and adds no categorical choice. The autoregressive atom decoder updates its hidden state between attributes, so later attributes depend on the earlier choices. For attached atom additions, the first-bond prediction is conditioned on all attributes of the new atom. The bond decoder jointly predicts bond type and stereochemistry. This factorization couples attachment, atom attributes, and first-bond attributes within one complete edit.

Isolated atom additions use a graph-level element distribution followed by the same sequence of attribute predictions, without a bond prediction. They introduce only one atom; subsequent attached atom additions or bond edits determine the component's eventual structure. Bond additions select an absent unordered pair and then jointly predict its bond attributes. Bond-attribute updates select an existing bond and mask its current tuple out of the replacement distribution. For updates that keep the element fixed, a nonempty mask selects the atom attributes to change. Replacement values exclude the current values, while untouched fields are copied into the complete tuple. Deletions specify locations without new attributes.

Masks are applied before the corresponding normalization. An edit type with no eligible location is excluded from the type distribution. During atom addition, the allowed values of each attribute can depend on earlier choices, and the allowed bond attributes can depend on the new atom tuple. Training computes the log probability of the complete target action by adding these conditional log probabilities. Generation samples actions through the same hierarchy.

G.4 Generation by accumulated hazard

The numerical sampler partitions transformed time uniformly up to $\tau_{\mathrm{end}} = -\log(1 - t_{\mathrm{end}})$. The network still receives $t = 1 - e^{-\tau}$. At a rate evaluation, it freezes both total intensity $\widehat{\lambda}$ and action distribution $\widehat{\pi}$ until the next grid boundary or a graph edit. If the next boundary is β, the integrated hazard over an unchanged interval is approximated by

$$\int_\tau^\beta \lambda_\theta(X, P, 1 - e^{-s})\, ds \;\approx\; \widehat{\lambda}(\beta - \tau). \tag{21}$$

An exponential threshold is consumed across intervals. Crossing a grid boundary changes the rate approximation, but does not restart the event clock. When the threshold is crossed within an interval, the action is sampled from the same frozen distribution used for that event's waiting time.

Algorithm 2 Generation with accumulated hazard

Require: Product P, rates θ, grid ending at τ_{end}, caps $N_{\mathrm{edit}}, N_{\mathrm{new}}$
Ensure: Terminal graph X or a budget-limited terminal state
$X \leftarrow P, \tau \leftarrow 0, n \leftarrow 0$; draw $H \sim \mathrm{Exp}(1)$
while $\tau < \tau_{\mathrm{end}}$ **do**
  $\beta \leftarrow$ next grid boundary after τ
  $(\widehat{\lambda}, \widehat{\pi}) \leftarrow \textsc{RateNetwork}(P, X, 1 - e^{-\tau})$ with current masks and N_{new}
  $h \leftarrow \widehat{\lambda}(\beta - \tau)$
  if $\widehat{\lambda} = 0$ **or** $H > h$ **then**
    $H \leftarrow H - h; \tau \leftarrow \beta$ ▷ Keep the event clock
  else
    if $n = N_{\mathrm{edit}}$ **then**
      return BUDGETLIMIT
    end if
    $\tau \leftarrow \tau + H/\widehat{\lambda}$; draw $a \sim \widehat{\pi}$
    $X \leftarrow T_a X; n \leftarrow n + 1$
    Draw $H \sim \mathrm{Exp}(1)$ ▷ New clock after the edit
  end if
end while
return X

Algorithm 2 re-evaluates the rates after each edit and each grid crossing. The grid controls integration accuracy, not the edit count.

Updating the graph immediately matters for growth: a newly created atom can provide an attachment site, form another bond, or undergo an attribute update before the next grid boundary. Conversely, several boundaries can pass without an edit. The new-atom cap disables atom addition while retaining other admissible edit types. If the edit budget is reached before another edit can be applied, sampling returns the current state; otherwise it returns the graph at the terminal time.

G.5 From terminal states to canonical reactant sets

Decoding omits internal atom indices. Full reconstruction writes explicit fields and bond stereochemistry, then uses RDKit sanitization and stereochemistry assignment to obtain canonical isomeric SMILES (Weininger, 1988). For the primary topology-matching score, decoding retains elements and core bond types as non-isomeric SMILES. Each view decodes independently.

Sorted component strings form a canonical key, retaining repeated components. Evaluation follows Appendix C.3.

H Limitations and Future Work

RetroGEF studies single-step reactant construction from paired reaction records, with evaluation on patent-derived benchmarks. Extending this setting to more diverse reaction collections and chemical domains is a natural direction for examining how the learned construction strategies transfer across molecular families. Broader evaluation could also complement recovery of recorded reactants with assessments of alternative precursors under different synthesis objectives. Such studies would connect the standardized comparisons used here with a wider range of retrosynthetic choices, including transformations and applications beyond those represented in the current benchmarks.

A second direction is to connect dynamic graph construction with condition-aware generation and multistep synthesis planning. Reaction conditions, precursor availability, and route-level objectives could provide additional context for choosing both local edits and complete reactant sets. Incorporating these factors would allow the single-step generator to be studied as part of an overall synthesis strategy, where the value of a prediction depends on the route it enables. Computational assessment and experimental feedback could help evaluate these candidates in practical synthesis settings.