MECHANISTIC REACTION PREDICTION VIA DISCRETE FLOW MATCHING ON GRAPH-STRUCTURED ELECTRON OCCUPATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Chemical reactions span diverse molecules and transformation types, but available data cover them unevenly, making prediction beyond the training distribution a central challenge. Chemists reason about reactivity through arrow-pushing: electrons move from a donor site to an acceptor site, and that direction helps determine which bonds form. Most reaction predictors instead generate products or edit molecular topology without explicitly representing these transfers. We introduce MAELLE, a discrete flow-matching model that predicts directional electron-pair edits on a graph of bonding, non-bonding, and hydrogen sites. Optimal transport infers an unordered set of edits from each reactant–product pair, allowing training without annotated elementary steps. The resulting stochastic trajectories retain the reactants' heavy atoms. MAELLE achieves 93.9% top-5 accuracy on USPTO-480K and 59.3% top-5 accuracy on the literature-derived AbSynth benchmark, the highest AbSynth result among the evaluated baselines. In a separate analysis of expert-annotated reactions, its electron sources and sinks agree with the annotations for 95.4% of recovered persistent bond-forming steps in trajectories reaching the recorded product. These results support endpoint-supervised directional electron flow as a useful representation for reaction prediction beyond pattern recognition.

1 INTRODUCTION

Reaction outcome prediction is central to computer-assisted synthesis (Corey & Wipke, 1969; Schwaller et al., 2019). Although large patent datasets have enabled accurate predictions on random test splits, chemical data are unevenly distributed: some transformations and molecular settings are well represented, while others have few recorded examples Thakkar et al. (2020). A model used to plan a new synthesis must therefore do more than recognize familiar reactant–product patterns. It must remain useful when the molecules or transformations differ from those in its training data (Gil et al., 2023; Bradshaw et al., 2025; Tran et al., 2026).

Chemists often reason about such transformations with arrow-pushing diagrams Grossman & Grossman (2003). An arrow begins at an electron source, such as a lone pair or bond, and ends at a site that accepts those electrons. This source–sink direction distinguishes an electron donor from an acceptor and, in many reactions, captures the nucleophile–electrophile relationship that determines which bonds can form. Predicting only the final product or the bonds that change leaves this direction implicit. Representing it explicitly could give a model a transferable description of reactivity, even when a particular overall transformation is uncommon in the training data.

Learning arrow-pushing behavior at scale is difficult because reaction databases usually record reactants and products, not the elementary steps between them. Existing mechanistic models can use annotated steps (Joung et al., 2025), while endpoint-supervised approaches either restrict electron paths to particular topologies (Bradshaw et al., 2018) or predict net electron changes without identifying the source and sink of each transfer (Bi et al., 2021). The challenge is to learn directional electron redistribution from abundant reaction endpoints without treating an inferred path as an observed mechanism.

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.

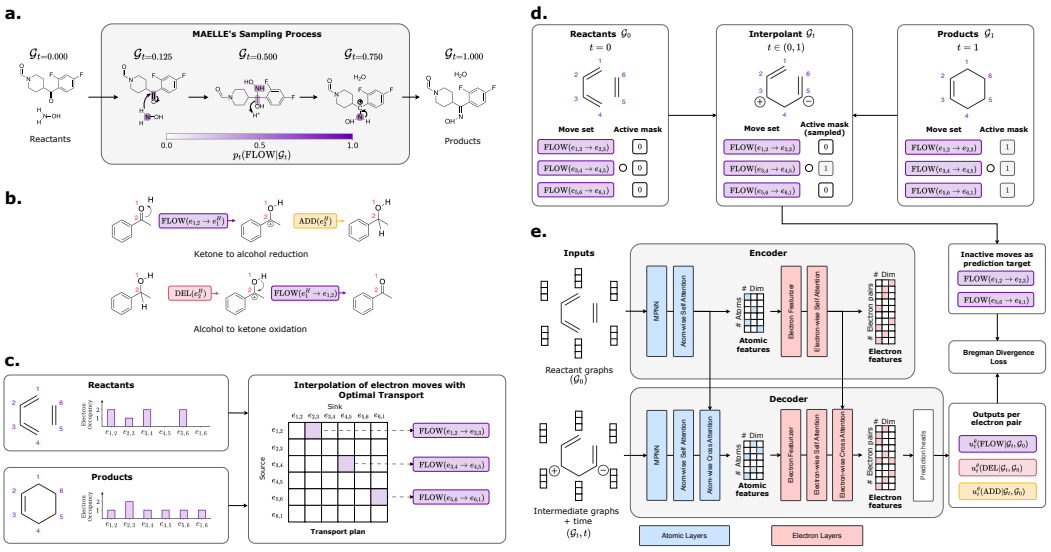


Figure 1: Conceptual overview of MAELLE: **a.** MAELLE's sampling process with mechanism-like trajectories that model the transition from reactants to products as a CTMC. **b.** The three elementary actions on electrons, FLOW, DEL, and ADD, demonstrated on an oxidation and a reduction reaction. At training time: **c.** Electron actions are interpolated using OT; **d.** interpolant graphs are sampled by sampling a subset of electron moves and applying them to the reactant graphs. **e.** MAELLE model architecture and learning objective.

We introduce MAELLE (**M**ech**A**nistic **E**dit f**L**ow-matching on e**L**ectron r**E**arrangements), which represents a reaction as edits to integer-valued electron occupations on a molecular graph. A FLOW edit transfers an electron pair from a source site to a sink site; DEL and ADD handle changes in the represented electron-pair count. Optimal transport constructs an unordered set of edits connecting each training pair, and discrete flow matching learns a continuous-time Markov chain that generates edit trajectories. This construction requires no elementary-step labels and retains the reactants' heavy atoms throughout generation.

Our contributions are:

1. We extend discrete flow matching to graph-structured electron occupations using directional electron-pair edits, allowing the model to generate heavy-atom-conserving reaction outcomes.
2. We derive electron moves from reactant–product pairs through optimal transport. These moves identify electron sources and sinks and resemble the symbolic arrow-pushing notation.
3. We evaluate prediction on USPTO-480K and on the literature-derived AbSynth benchmark (Jin et al., 2017; Czuba et al., 2026). MAELLE is competitive on the former and leads the evaluated baselines on the latter. For recovered persistent bond-forming steps, its predicted donor–acceptor directions also show high agreement with expert annotations.

2 Related Work

Products as sequences or graph edits. Sequence models translate reactant SMILES into product SMILES (Schwaller et al., 2019), sometimes using augmented strings to account for multiple representations of the same molecule (Irwin et al., 2022). Graph2SMILES instead encodes the reactants as a graph while retaining a SMILES decoder (Tu & Coley, 2022). These models generate products without explicitly tracking where atoms or electrons move; conservation and reactant–product correspondence are therefore not guaranteed by the representation. Generating an entire product can also be difficult when molecular complexity differs from the training data (Gil et al.,

2023; Bradshaw et al., 2025; Tran et al., 2026). Graph-edit methods such as MEGAN and GTPN retain the reactant structure and predict local changes (Sacha et al., 2021; Do et al., 2019). Their edits expose the changed bonds, but a bond edit does not specify the source of the electrons that form or break that bond.

Electron redistribution. Mechanistic models describe reactions through electron movement, often using expert-annotated elementary steps (Kayala et al., 2011; Chen et al., 2024). FlowER trains on a large mechanistic dataset and models changes to a bond–electron matrix (Joung et al., 2025). End-to-end reaction data are more abundant than step annotations: ELECTRO infers electron paths from reactant–product pairs, but restricts their topology to linear flows (Bradshaw et al., 2018). NERF predicts net changes in bond-electron counts from the same kind of endpoint data, without identifying the donor and acceptor of each transfer (Bi et al., 2021). MAELLE also learns from endpoints, but its FLOW edits identify electron sources and sinks. Optimal transport supplies an unordered set of moves between the endpoints without assuming a linear electron path (Section 3.3). These moves describe a model trajectory rather than a uniquely established elementary mechanism.

Discrete generative models. DiGress and DeFoG generate graphs through iterative changes to categorical node and edge states (Vignac et al., 2022; Qin et al., 2024); RetroBridge learns a bridge between coupled molecular endpoints for retrosynthesis (Igashov et al., 2024). MAELLE instead defines a discrete flow over integer-valued electron occupations. Its transitions move electron pairs among sites while retaining the input heavy atoms.

3 MAELLE: Reaction Prediction with Electron-based Edit Flow Matching

3.1 Electron occupation as the main source of change

We represent a molecule as an augmented graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathcal{E}, o)$, where $\mathcal{V}$ is the set of heavy atoms, $\mathcal{A}$ is the heavy-atom adjacency, and $\mathcal{E}$ is a fixed set of *electron sites* comprising bonding sites e_{ij} ($i \neq j$), per-atom lone-pair sites e_{ii} and hydrogen sites e_i^H. The *occupation vector* $o \in \mathbb{Z}^{|\mathcal{E}|}$ counts electron pairs at each site, including virtual ones with $o^e = 0$. See Appendix C for the formal definitions and an illustration on ethanol.

A chemical reaction maps reactants $\mathcal{G}_x$ to products $\mathcal{G}_y$ by redistributing electrons, i.e., changing o. The atom set $\mathcal{V}$ is preserved. Moreover, the heavy-atom adjacency can be recovered deterministically from the occupation through

$$a_{ij} = \mathbb{1}[\, o^{e_{ij}} > 0\,], \tag{1}$$

so $p(\mathcal{G}_y \mid \mathcal{G}_x) = p(o_y \mid \mathcal{G}_x)$. Thus, we could model a reaction by acting entirely on the vector o.

3.2 Editing electron occupation

In this section, we describe the electron edit operations that act on o, inspired by the arrow-pushing notation. Let $\mathbf{1}_e \in \{0,1\}^{|\mathcal{E}|}$ denote the one-hot vector with a single nonzero entry at the position corresponding to site $e \in \mathcal{E}$. We define the following actions on o (Figure 1b):

FLOW Transfer one electron pair from site e_{source} to site e_{sink}, requiring $o(e_{\text{source}}) > 0$:

$$\textsc{flow}(e_{\text{source}} \to e_{\text{sink}})(o) = o - \mathbf{1}_{e_{\text{source}}} + \mathbf{1}_{e_{\text{sink}}} \tag{2}$$

DEL Remove one electron pair from site e, requiring $o^e > 0$:

$$\textsc{del}(e)(o) = o - \mathbf{1}_e \tag{3}$$

ADD Add one electron pair to site e:

$$\textsc{add}(e)(o) = o + \mathbf{1}_e \tag{4}$$

The sparsity of these three operations enables us to define a tractable distribution of the intermediate states between the occupation vectors of the reactants and products o_x and o_y.

3.3 INTERPOLATION OF ARROW-PUSHING STEPS VIA OPTIMAL TRANSPORT

To obtain a set of electron moves that transform o_x into o_y, we solve a balanced integer optimal transport problem over the electron sites. To handle reactions where the total electron count is not conserved, we augment the site set with a virtual sink/source $e_\varnothing$, yielding $\bar{\mathcal{E}} = \mathcal{E} \cup \{e_\varnothing\}$ with augmented occupations $\bar{o}_x, \bar{o}_y$. The transport cost between two sites is the minimum shortest-path distance between their constituent atoms on either the reactant or product graph, with a large penalty for transport to or from $e_\varnothing$. We solve the optimal transport problem with the simplex algorithm, where the design of the objective, marginal constraints, and cost matrix can be found in Appendix D. This results in the optimal transport matrix $T^* \in \mathbb{Z}^{|\bar{\mathcal{E}}|\times|\bar{\mathcal{E}}|}$, whose columns and rows represent source and sink, respectively.

Each non-zero off-diagonal entry $T^*_{e,e'} > 0$ is translated to an electron action $m \in \{\text{FLOW}, \text{DEL}, \text{ADD}\}$ (Eqs. 2–4):

$$m = \begin{cases} \text{FLOW}(e \to e') & \text{if} \quad e \neq e_\varnothing, e' \neq e_\varnothing \\ \text{DEL}(e) & \text{if} \quad e \neq e_\varnothing, e' = e_\varnothing \\ \text{ADD}(e') & \text{if} \quad e = e_\varnothing, e' \neq e_\varnothing \end{cases} \tag{5}$$

The transport plan specifies which electron sites lose and gain occupation, but not the temporal order of chemical elementary steps. We use it as an endpoint-consistent training alignment: applying all moves recovers o_y, whereas a subset provides a partially edited state. Because distinct move subsets may produce the same occupation, the factorized interpolation is defined on the auxiliary move mask and then mapped to electron occupations through $\mathrm{ApplyEdit}$:

$$\mathcal{M} = \{m_1, \ldots, m_K\}, \qquad K = \sum_{e \neq e'} T^*_{e,e'}, \tag{6}$$

$$\mathrm{ApplyEdit}(o_x, \mathcal{M}) = o_y \tag{7}$$

3.4 EDIT FLOW ON ELECTRONS

Discrete flow matching (DFM) learns a continuous-time Markov chain (CTMC) that connects a source distribution to a target distribution through intermediate states. In our setting, the source is the reactant electron occupation $\mathbf{o}_x$, and the target is the product electron occupation $\mathbf{o}_y$. The move set M derived in Section 3.3 connects these endpoints. At time t, we sample an intermediate state by applying a subset of M to $\mathbf{o}_x$; the probability of including each move increases from zero to one with t. Training then teaches the model to assign transition rates to the moves remaining at that state. At inference, the product and its move set are unknown, so the model predicts rates conditioned on the reactants and the current state. Appendix B reviews the general DFM and Edit Flow formulation.

Edit-based mixture path Let $\mathcal{S} \subseteq \mathcal{M}$ be a subset of the moves, interpreted as the moves that have been applied so far. We define the interpolant occupation corresponding to $\mathcal{S}$ as $o_\mathcal{S} = \mathrm{ApplyEdit}(o_x, \mathcal{S})$, which by equation 6 recovers o_x at $\mathcal{S} = \varnothing$ and o_y at $\mathcal{S} = \mathcal{M}$. To interpolate between the two endpoints, we place a distribution over which subset has been applied at time t and read the occupation off through $\mathrm{ApplyEdit}$. Following a monotone schedule $\kappa : [0,1] \to [0,1]$ with $\kappa(0) = 0$, $\kappa(1) = 1$, each move is applied independently with probability $\kappa(t)$, giving the edit-based mixture path

$$p_t(o \mid o_x, o_y) = \sum_{\mathcal{S} \subseteq \mathcal{M}} \underbrace{\kappa(t)^{|\mathcal{S}|} \big(1 - \kappa(t)\big)^{|\mathcal{M}|-|\mathcal{S}|}}_{p_t(\mathcal{S} \mid o_x, o_y)} \mathbb{1}[\,\mathrm{ApplyEdit}(o_x, \mathcal{S}) = o\,] \tag{8}$$

Equivalently, the path is the law of $\mathrm{ApplyEdit}(o_x, \mathcal{S}_t)$ where $\mathcal{S}_t$ is obtained by including each move $m_k \in \mathcal{M}$ independently with probability $\kappa(t)$: sampling an interpolant amounts to flipping one $\mathrm{Bernoulli}(\kappa(t))$ mask per move and applying the selected subset to the reactant occupation (Algorithm 1). Since each move is an independent binary variable, the weight $p_t(\mathcal{S} \mid o_x, o_y)$ factorizes over $\mathcal{M}$, and the endpoints follow directly: at $t = 0$ no moves are sampled so $p_0 = \delta_{o_x}$, and at $t = 1$ all moves are applied so $p_1 = \delta_{o_y}$.

The path equation 8 is the electron-space instance of the augmented-space mixture path of Edit Flows (Havasi et al., 2025). There, the path is defined over an auxiliary space of padded sequences and pushed to token sequences by removing padding; here it is defined over subsets of electron moves and pushed to occupations by $\mathrm{ApplyEdit}$. In both, an interpolant is obtained by sampling which edits to apply, and the two constructions yield the same conditional path. We state this equivalence in Proposition 1.

Conditional and marginal rates The mixture path equation 8 is generated by a CTMC whose conditional rate fires each unapplied move independently. Following the standard DFM derivation, the rate of switching a move m_k from unapplied to applied is the hazard rate of its $\mathrm{Bernoulli}(\kappa(t))$ flip,

$$u_t\big(m_k \mid o_t, o_x, o_y\big) \;=\; \frac{\kappa'(t)}{1-\kappa(t)}, \tag{9}$$

with zero reverse rate (moves fire once and cannot be undone). This conditional rate assumes the product o_y is known. At inference, it is not, so the training target is the *marginal* rate, obtained by taking the expectation over the posterior of the endpoints given the current state,

$$u_t(m \mid o_t, \mathcal{G}_x) \;=\; \mathbb{E}_{p_t(o_y \mid o_t, \mathcal{G}_x)}\; u_t\big(m \mid o_t, o_x, o_y\big), \tag{10}$$

which depends only on the current state and the reactant, and is therefore learnable by a neural network. By the auxiliary-variable result of Havasi et al. (2025), a model trained to match the conditional rate in the move space yields the correct marginal rate after marginalization.

Training We model the reactant-to-product mapping as a coupling $\pi(\mathcal{G}_x, \mathcal{G}_y)$ over reactant-product pairs. At training time, π is the empirical distribution of reactions in the training set, each of which provides one coupled pair. The training target is the marginal rate $u_t(m \mid o_t, \mathcal{G}_x)$ of Eq. 10, which we approximate with a graph neural network $u_t^\theta(m \mid \mathcal{G}_x, \mathcal{G}_t, t)$. We adopt an encoder–decoder architecture: the encoder processes the fixed reactant $\mathcal{G}_x$ into conditioning representations, and the decoder processes the interpolant $\mathcal{G}_t$ and cross-attends to them at every layer, so the predicted rates are conditioned on the reactant throughout, analogous to conditioning on the source in conditional flow matching. Unlike standard molecular GNNs, which typically featurize only atoms and bonds, our edits act on the electron occupation o over the full site set $\mathcal{E}$, including bonding, valence, and hydrogen-bound electrons, so an ElectronFeaturizer lifts atom representations into a vector for every electron site (Figure 1e). The per-move rates are read off these electron-site representations, matching the granularity at which the edits operate. Full architectural details can be found in Appendix F.

We fit u_t^θ with the Bregman-divergence objective of Holderrieth et al. (2024); Havasi et al. (2025), which for a CTMC rate reduces to a total-rate penalty minus a log-rate reward on the correct transitions. Let $\mathcal{M}_{\mathrm{target}} = \mathcal{M} \setminus \mathcal{S}$ be the moves not yet applied at the sampled interpolant. The loss is

$$\mathcal{L}(\theta) \;=\; \mathop{\mathbb{E}}_{\substack{(\mathcal{G}_x,\mathcal{G}_y)\sim\pi \\ t\sim\mathcal{U}(0,1) \\ \mathcal{S}\sim p_t(\cdot\mid o_x,o_y)}} \left[\underbrace{\sum_m u_t^\theta(m \mid \mathcal{G}_x, \mathcal{G}_t, t)}_{\Lambda_{\theta,t}} \;-\; \frac{\kappa'(t)}{1-\kappa(t)} \sum_{m\in\mathcal{M}_{\mathrm{target}}} \log u_t^\theta(m \mid \mathcal{G}_x, \mathcal{G}_t, t) \right] \tag{11}$$

Because a FLOW move acts on pairs of atoms, and the ADD move could act on virtual electron sites, calculating the rates of all possible moves would make the term $\Lambda_{\theta,t}$ scale as $O(|\mathcal{E}|^2)$. We therefore factorize the move's rate for these two operations into a per-site source rate and a sink distribution over destinations:

$$u_t^\theta(m \mid \cdot) = \begin{cases} \lambda_{\theta,t}^{\mathrm{flow}}(e_{\mathrm{source}} \mid \cdot)\, p_\theta^{\mathrm{sink,flow}}(e_{\mathrm{sink}} \mid e_{\mathrm{source}}, \cdot) & m = \textsc{flow}(e_{\mathrm{source}} \to e_{\mathrm{sink}}), \\ \lambda_{\theta,t}^{\mathrm{del}}(e_{\mathrm{source}} \mid \cdot) & m = \textsc{del}(e_{\mathrm{source}}), \\ \lambda_{\theta,t}^{\mathrm{add}}(v \mid \cdot)\, p_\theta^{\mathrm{sink,add}}(e_{\mathrm{sink}} \mid v, \cdot) & m = \textsc{add}(e_{\mathrm{sink}}), \end{cases} \tag{12}$$

Here $\lambda^{\text{flow}}_{\theta,t}$ and $\lambda^{\text{del}}_{\theta,t}$ are per-site firing rates over non-virtual electron sites $e_{\text{source}} \in \mathcal{E}$, $o^{e_{\text{source}}} > 0$; $\lambda^{\text{add}}_{\theta,t}$ is a per-atom rate over atoms $v \in \mathcal{V}$, and $p^{\text{sink,flow}}_{\theta}$, $p^{\text{sink,add}}_{\theta}$ are separately parameterized distributions over destination sites. For DEL no sink is needed, since the operation only removes an electron pair.

We parameterize ADD through an atom v rather than a site because a new electron pair is not associated with any existing site a priori. Intuitively, the model first decides to donate a pair of electrons to atom v, and v then decides where to place them – selecting a destination site e_{sink} among its associated sites through a dedicated sink head. In this sense ADD behaves as a pseudo-FLOW whose source is the atom itself rather than an occupied electron site, though the two use separate sink parameterizations. The total rate then factorizes as $\sum_m u^\theta_t(m \mid \cdot) = \sum_e (\lambda^{\text{flow}} + \lambda^{\text{del}}) + \sum_v \lambda^{\text{add}}$, since each p^{sink} sums to one.

The training loop can be found in Algorithm 2.

Inference At inference, the products are generated by the marginal CTMC starting from the reactants. We initialize $\mathcal{G}_t = \mathcal{G}_x$ at $t = 0$ and take N Euler steps of size $\Delta t = 1/N$. Since the reactant is fixed, the encoder is run once on $\mathcal{G}_x$ and its representations are reused at every step, while the decoder predicts the move rates on the current graph $\mathcal{G}_t$. At each step, every admissible move fires independently as a thinned Poisson process: a move m with predicted rate $u^\theta_t(m \mid \mathcal{G}_x, \mathcal{G}_t, t)$ is applied with probability

$$p_{\text{fire}}(m) \;=\; 1 - \exp\big(- u^\theta_t(m \mid \mathcal{G}_x, \mathcal{G}_t, t)\, \Delta t\big) \tag{13}$$

The sampled edit moves are applied to $\mathcal{G}_t$ via $\mathrm{ApplyEdit}$. Iterating to $t = 1$ yields a predicted product $\hat{\mathcal{G}}_1$. The full procedure is given in Algorithm 4.

Product ranking from sampled trajectories We run multiple independent CTMC trajectories from the same reactant $\mathcal{G}_x$. Because edit firing is stochastic, these trajectories can produce distinct outcomes. We canonicalize the resulting product graphs, group identical products, and rank them by their frequency among samples to obtain a top-k list. Sample frequency estimates the model's relative probability of each product; we do not assume it is calibrated confidence. Since the edits retain the input heavy atoms, every sampled outcome preserves that atom set. An example with the recorded product and a plausible alternative is shown in Section G.2.

4 RESULTS

4.1 EXPERIMENTS

Datasets and benchmark. USPTO-480K contains reactions extracted from US patents (Lowe, 2012b; Jin et al., 2017) and uses a standard training, validation, and test split widely adopted for reaction prediction (Jin et al., 2017; Schwaller et al., 2019; Sacha et al., 2021; Bi et al., 2021; Do et al., 2019). This split primarily evaluates performance within the benchmark's distribution. To examine transfer beyond it, we also evaluate models trained on USPTO-480K on approximately 51,000 literature-derived reactions from AbSynth (Czuba et al., 2026). Dataset preprocessing is described in Section I, and a comparison of the datasets' structural properties appears in Section J.

Baselines. We compare against representative methods spanning the three prediction modalities: SMILES generation – Molecular Transformer (MT) (Schwaller et al., 2019) and Graph2SMILES (G2S) (Tu & Coley, 2022); graph edits – MEGAN (Sacha et al., 2021), and electron changes – NERF (Bi et al., 2021). We exclude ELECTRO (Bradshaw et al., 2018) and FlowER (Joung et al., 2025), which by design cannot be trained on USPTO-480K: ELECTRO applies only to reactions with linear-electron-flow topology, and FlowER requires elementary-step annotations, which are unavailable for most USPTO-480K reactions. For MEGAN and NERF, model weights are not published; therefore, we reproduce these models following the protocol in their corresponding GitHub repositories (Section L). All numbers are obtained under the setting where the models do not know which molecules in the reactants are reactants or reagents.

Metrics. We report top-k accuracy, which measures how often the correct product is found among the top-k predictions of the models.

Table 1: Top-k accuracy on the standard USPTO-480K benchmark and the out-of-distribution AbSynth benchmark. Best and second-best results among directly comparable methods are shown in **bold** and underline, respectively. Results marked with † are reported in the original papers; the rest are obtained using our evaluation.

Method	Output representation	USPTO-480K			AbSynth (OOD)		
		Top-1	Top-3	Top-5	Top-1	Top-3	Top-5
MT Schwaller et al. (2019)	SMILES	0.886†	0.935†	0.942†	0.354	0.419	0.441
G2S Tu & Coley (2022)	SMILES	**0.903**†	**0.940**†	**0.948**†	0.395	0.452	0.471
MEGAN Sacha et al. (2021)	Graph edits	0.858	0.919	0.934	0.456	0.544	0.579
NERF* Bi et al. (2021)	Electron changes	0.907†	0.933†	0.937†	—	—	—
NERF (joint reactant-reagent)	Electron changes	0.682	0.724	0.733	0.091	0.225	0.263
MAELLE (ours)	Electron flow	0.872	0.930	0.939	**0.466**	**0.567**	**0.593**

* *Original NERF treats reactants and reagents separately and is therefore not directly comparable to methods using joint reactant-reagent input.*

Electron donor-acceptor correctness To quantify how the arrow-pushing edit operations that MAELLE produces align with real reaction mechanisms, we train MAELLE on reactions from FlowER dataset Joung et al. (2025). The dataset contains ∼290K reactions extracted from USPTO-Full (Lowe, 2012b), each annotated with a sequence of expert-labeled elementary steps (∼2M steps in total). By design, MAELLE is trainable using only end-to-end reactions and does not consume the elementary steps. We use the FlowER elementary steps, however, as an anchor to perform the analysis on the arrow-pushing moves that MAELLE produces.

4.2 Reaction prediction beyond the training distribution

On the standard USPTO-480K split, MAELLE reaches 87.2%, 93.0%, and 93.9% accuracy at top-1, top-3, and top-5, respectively (Table 1). It outperforms the directly comparable graph-edit model MEGAN at each k, although G2S remains the strongest method on this in-distribution benchmark. The gap to G2S narrows from 3.1 percentage points at top-1 to 0.9 points at top-5.

We next evaluate the same models on AbSynth (Czuba et al., 2026), a literature-derived benchmark that differs from USPTO-480K in reaction types and molecular complexity (Section J). Accuracy decreases for every method, but MAELLE leads the evaluated baselines at each reported k: 46.6% at top-1, 56.7% at top-3, and 59.3% at top-5. MEGAN is the closest competitor, reaching 45.6%, 54.4%, and 57.9%, respectively. The advantage over MEGAN suggests that MAELLE's electron-edit representation may help beyond the shared benefit of editing a reactant structure.

Two controlled USPTO-480K splits probe this interpretation (Appendix K.2). On OOD-Mass, where products are larger than those in training, MAELLE and MEGAN remain more robust than the SMILES-based models. This is consistent with an advantage from editing reactants rather than generating products anew. OOD-Ester instead holds out non-methyl esterification reactions and separates the two edit-based models: MAELLE outperforms MEGAN, suggesting a possible benefit of directional electron edits when transferring to a held-out transformation class.

4.3 Donor–acceptor agreement in predicted electron flows

MAELLE learns from reaction endpoints without elementary-step supervision. We therefore ask whether its FLOW moves identify the electron donors and acceptors in expert-annotated bond-forming steps. We focus on persistent bonds, which remain in the product; transient bonds form and break along the annotated pathway (Figure 2).

Table 2: Coverage of persistent bond-forming steps and donor–acceptor agreement among recovered steps.

Mode	Coverage	Agreement
Correct	13,137/17,925 (73.3%)	12,527/13,137 (95.4%)
Top-1	14,236/17,925 (79.4%)	12,990/14,236 (91.2%)
All	17,847/17,925 (99.6%)	16,013/17,847 (89.7%)

We train MAELLE on the FlowER reactions using only their endpoints and sample 64 trajectories for each test reaction. For every annotated

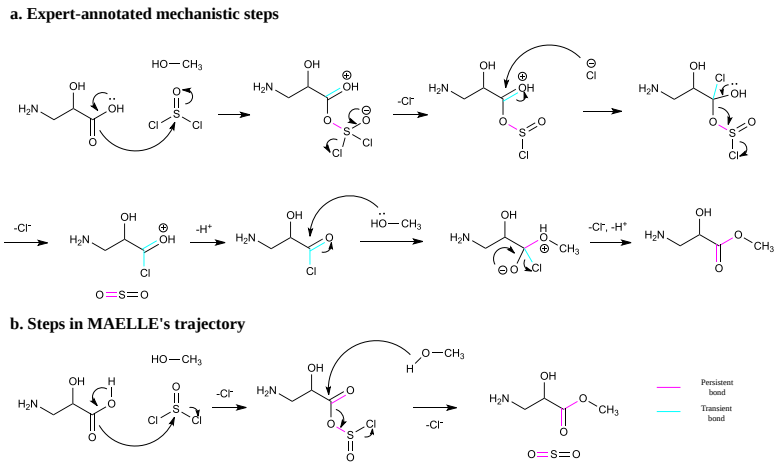


Figure 2: **a.** An example from FlowER's test set with expert-labeled elementary steps. **b.** Pseudo mechanistic steps in MAELLE's trajectory. Note that during training, MAELLE does not consume the labeled elementary steps.

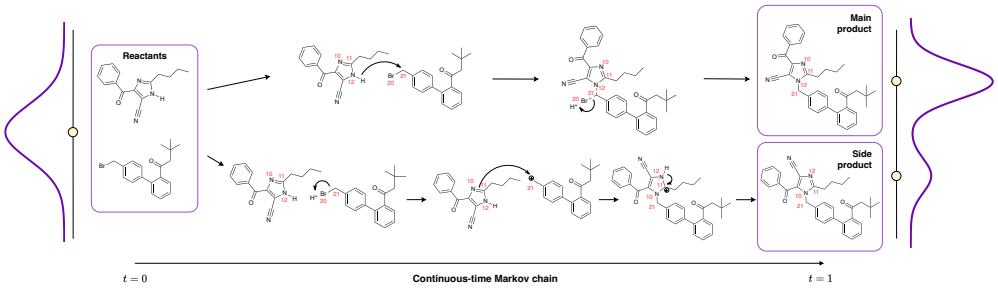


Figure 3: A case study of stochastic electron-edit trajectories from the same reactants yields two distinct product outcomes. The highest-frequency outcome matches the recorded product; another trajectory yields a plausible alternative. Both outcomes retain the reactants' heavy atoms, while differing in their electron redistribution and resulting bonds.

persistent bond formation, we measure whether a trajectory forms the same bond and, among recovered steps, whether its electron source and sink agree with the annotation. We report results for trajectories reaching the recorded product (*Correct*), those reaching the most frequently predicted product (*Top-1*), and all sampled trajectories (*All*).

Of 17,925 annotated persistent bond-forming steps, these selections recover 73.3%, 79.4%, and 99.6%, respectively (Table 2). Donor–acceptor agreement among the recovered steps is 95.4%, 91.2%, and 89.7%. Pooling trajectories improves coverage but lowers agreement. These results support directional fidelity for recovered, product-defining bond formations; they do not establish that the sampled trajectories reproduce transient intermediates. Appendix H evaluates transient bonds separately.

4.4 Distinct outcomes from stochastic electron-flow trajectories

MAELLE samples sequences of electron edits rather than predicting a single fixed product. Figure 3 follows two trajectories from the same reactants: the most frequent product matches the recorded outcome, while another trajectory produces a plausible alternative through a different sequence of edits. Both retain the input heavy atoms. The figure therefore shows how variation in predicted

electron flow can lead to different product structures. Appendix G.2 provides a systematic assessment of the chemical plausibility of the alternative products across a larger set of reactions.

5 Conclusion

MAELLE predicts reaction outcomes through directional electron-pair edits learned from reactant–product pairs, without elementary-step annotations. An optimal-transport-derived move set defines the training path, while discrete flow matching learns to generate electron-occupation trajectories that conserve heavy atoms. MAELLE is competitive on USPTO-480K and achieves the strongest results among the evaluated baselines on the literature-derived AbSynth benchmark. Its predicted donor-acceptor directions also agree with expert annotations for most recovered, persistent bond-forming steps. Together, these results suggest that endpoint-supervised electron redistribution is a useful representation for reaction prediction beyond the training distribution.

Limitations. The sampled trajectories are pseudo-mechanisms: agreement on persistent bond formations does not establish that transient steps follow the physical reaction pathway. The electron-pair representation also excludes radical reactions. Extending it to single-electron moves and incorporating energetic information are directions for improving mechanistic fidelity.

6 Reproducibility statement

The method is specified in Section 3, with formal derivations, training and inference pseudocode, architecture details, and hyperparameters provided in the appendix. The appendix also documents dataset preprocessing, evaluation protocols, and the reproduction of baselines whose model weights were unavailable. An anonymous implementation is available at `https://anonymous.4open.science/r/maelle-C087`.

7 AI use statement

The authors used AI tools to assist with writing code and polishing the language of the manuscript. The authors designed the model, training method, and experiments, and created the figures themselves. The authors reviewed and take responsibility for the code, analyses, and final manuscript.

References

Hangrui Bi, Hengyi Wang, Chence Shi, Connor Coley, Jian Tang, and Hongyu Guo. Non-autoregressive electron redistribution modeling for reaction prediction. In *International Conference on Machine Learning*, pp. 904–913. PMLR, 2021.

John Bradshaw, Matt J Kusner, Brooks Paige, Marwin HS Segler, and José Miguel Hernández-Lobato. A generative model for electron paths. *arXiv preprint arXiv:1805.10970*, 2018.

John Bradshaw, Anji Zhang, Babak Mahjour, David E Graff, Marwin HS Segler, and Connor W Coley. Challenging reaction prediction models to generalize to novel chemistry. *ACS Central Science*, 11(4):539–549, 2025.

Andres M Bran, Theo A Neukomm, Daniel P Armstrong, Zlatko Jončev, and Philippe Schwaller. Chemical reasoning in llms unlocks steerable synthesis planning and reaction mechanism elucidation. *arXiv preprint arXiv:2503.08537*, 2025.

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. *arXiv preprint arXiv:2402.04997*, 2024.

Shuan Chen, Ramil Babazade, Taewan Kim, Sunkyu Han, and Yousung Jung. A large-scale reaction dataset of mechanistic pathways of organic reactions. *Scientific Data*, 11(1):863, 2024.

Shuan Chen, Kye Sung Park, Taewan Kim, Sunkyu Han, and Yousung Jung. Predicting chemical reaction outcomes based on electron movements using machine learning. *arXiv preprint arXiv:2503.10197*, 2025.

Elias James Corey and W Todd Wipke. Computer-assisted design of complex organic syntheses: Pathways for molecular synthesis can be devised with a computer and equipment for graphical communication. *Science*, 166(3902):178–192, 1969.

Maciej Czuba, Konrad Barnowski, Karol Molga, Zofia Pławecka, Agnieszka Wołos, Szymon Stodołkiewicz, Sebastian Bas, Wojciech Petrykowski, Wiktor Beker, Evan Saillard, et al. Absynth: A century of total syntheses as a machine-readable resource for analyzing strategic logic in organic chemistry. *Journal of the American Chemical Society*, 148(36):38740–38753, 2026.

Kien Do, Truyen Tran, and Svetha Venkatesh. Graph transformation policy network for chemical reaction prediction. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 750–760, 2019.

Alexandre Duval, Simon V Mathis, Chaitanya K Joshi, Victor Schmidt, Santiago Miret, Fragkiskos D Malliaros, Taco Cohen, Pietro Lio, Yoshua Bengio, and Michael Bronstein. A hitchhiker's guide to geometric gnns for 3d atomic systems. *arXiv preprint arXiv:2312.07511*, 2023.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. Translation between molecules and natural language. *Proc. Conf. Empirical Methods Nat. Lang. Process.*, pp. 375–413, 2022.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

Victor Sabanza Gil, Andres M Bran, Malte Franke, Remi Schlama, Jeremy S Luterbacher, and Philippe Schwaller. Holistic chemical evaluation reveals pitfalls in reaction prediction models. *arXiv preprint arXiv:2312.09004*, 2023.

Robert B Grossman and Robert Grossman. *The art of writing reasonable organic reaction mechanisms*. Springer, 2003.

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

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky TQ Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary markov processes. *arXiv preprint arXiv:2410.20587*, 2024.

Ilia Igashov, Arne Schneuing, Marwin Segler, Michael Bronstein, and Bruno Correia. Retrobridge: Modeling retrosynthesis with markov bridges. In *International Conference on Learning Representations*, volume 2024, pp. 39622–39640, 2024.

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

Hyosoon Jang, Yunhui Jang, Jaehyung Kim, and Sungsoo Ahn. Can llms generate diverse molecules? towards alignment with structural diversity. *arXiv preprint arXiv:2410.03138*, 2024.

Wengong Jin, Connor Coley, Regina Barzilay, and Tommi Jaakkola. Predicting organic reaction outcomes with weisfeiler-lehman network. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2607–2616. Curran Associates, Inc., 2017.

Joonyoung F Joung, Mun Hong Fong, Nicholas Casetti, Jordan P Liles, Ne S Dassanayake, and Connor W Coley. Electron flow matching for generative reaction mechanism prediction. *Nature*, 645(8079):115–123, 2025.

Matthew A Kayala, Chloé-Agathe Azencott, Jonathan H Chen, and Pierre Baldi. Learning to predict chemical reactions. *J. Chem. Inf. Model.*, 51:2209–2222, September 2011.

Daniel Lowe. Chemical reactions from US patents (1976-Sep2016) `http://doi.org/10.6084/m9.figshare.5104873.v1`, 6 2017.

Daniel Mark Lowe. *Extraction of chemical structures and reactions from the literature*. PhD thesis, University of Cambridge, 2012a.

Daniel Mark Lowe. *Extraction of chemical structures and reactions from the literature*. PhD thesis, University of Cambridge, 2012b.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation. *arXiv preprint arXiv:2410.04263*, 2024.

RDKit, online. RDKit: Open-source cheminformatics. `http://www.rdkit.org`, 2023.

Mikołaj Sacha, Mikołaj Błaz, Piotr Byrski, Paweł Dabrowski-Tumanski, Mikołaj Chrominski, Rafał Loska, Paweł Włodarczyk-Pruszynski, and Stanisław Jastrzebski. Molecule edit graph attention network: modeling chemical reactions as sequences of graph edits. *Journal of Chemical Information and Modeling*, 61(7):3273–3284, 2021.

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.

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.

Amol Thakkar, Nidhal Selmi, Jean-Louis Reymond, Ola Engkvist, and Esben Jannik Bjerrum. "Ring Breaker": Neural network driven synthesis prediction of the ring system chemical space. *J. Med. Chem.*, 63:8791–8808, 2020.

Suong BA Tran, Jihye Roh, and Connor W Coley. Quantifying the failure modes of current one-step retrosynthesis models. *Chemical Science*, 2026.

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.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Varvara Voinarovska, Rocío Mercado, Mikhail Kabeshov, and Samuel Genheden. Do humans and large language models agree on the quality of synthesis plans? 2026.

Nguyen Xuan-Vu, Daniel Armstrong, Zlatko Joncev, and Philippe Schwaller. Tempre: Template generation for single and direct multi-step retrosynthesis. *arXiv preprint arXiv:2507.21762*, 2025.

A Summary of related work

Comparison between different reaction prediction models and their output modalities is shown in Table 3.

Table 3: Comparison of reaction prediction approaches.

Method	Input	Output	Mech. step label-free	Directional elec. re-dist.	Broad coverage	Atom conservative
Molecular Transformer Schwaller et al. (2019)	SMILES	SMILES	✓	✗	✓	✗
Graph2SMILES Tu & Coley (2022)	SMILES	SMILES	✓	✗	✓	✗
Chemformer Irwin et al. (2022)	SMILES	SMILES	✓	✗	✓	✗
MEGAN Sacha et al. (2021)	Graph	Graph edits	✓	✗	✓	✗
GTPN Do et al. (2019)	Graph	Graph edits	✓	✗	✓	✓
NERF Bi et al. (2021)	Graph	Electron re-dist.	✓	✗	✓	✓
FlowER Joung et al. (2025)	Graph	Electron re-dist.	✗[†]	✓	✗[†]	✓
ELECTRO Bradshaw et al. (2018)	Graph	Electron re-dist.	✓	✓	✗[‡]	✓
MAELLE (ours)	Graph	Electron re-dist.	✓	✓	✓	✓[§]

[†]FlowER covers 86 expert-curated reaction types and requires a mechanistic dataset with imputed elementary steps.
[‡]ELECTRO is restricted to reactions with linear electron flow topology.
[§]MAELLE is heavy-atom conservative because protonation and deprotonation are simplified.

B Background

B.1 Discrete Flow Matching

Let $\mathcal{T}$ be a vocabulary and $\mathcal{X} = \mathcal{T}^* = \bigcup_{n=0}^{N} \mathcal{T}^n$ the space of token sequences of length at most N. Discrete flow matching (DFM) (Campbell et al., 2024; Gat et al., 2024) learns a continuous-time Markov chain (CTMC) that transports a source distribution p_0 at $t = 0$ to a target p_1 at $t = 1$. As in the continuous case, the central construction is a *conditional mixture path* $p_t(x \mid x_0, x_1)$ with boundary conditions $p_0(x \mid x_0, x_1) = \delta_{x_0}(x)$ and $p_1(x \mid x_0, x_1) = \delta_{x_1}(x)$; the endpoints (x_0, x_1) are drawn from a coupling $\pi(x_0, x_1)$ whose marginals are the source and target, and marginalizing the conditional path over π gives the marginal path p_t that interpolates p_0 and p_1.

For sequences of a common length n, the standard choice is the factorized, token-wise mixture path

$$p_t(x^i \mid x_0, x_1) = (1-\kappa(t))\,\delta_{x_0^i}(x^i) + \kappa(t)\,\delta_{x_1^i}(x^i), \qquad p_t(x \mid x_0, x_1) = \prod_{i=1}^{n} p_t(x^i \mid x_0, x_1), \tag{14}$$

where the schedule $\kappa : [0, 1] \to [0, 1]$ is monotone with $\kappa(0) = 0$, $\kappa(1) = 1$. Each token independently switches from its source value x_0^i to its target value x_1^i, and does so by time t with probability $\kappa(t)$. The CTMC that generates equation 14 has the token-wise marginal rate

$$u_t(x^i \mid x_t) = \frac{\kappa'(t)}{1-\kappa(t)} \left[\, p_{1|t}(x^i \mid x_t) - \delta_{x_t^i}(x^i) \,\right], \tag{15}$$

where $p_{1|t}(x^i \mid x_t)$ is the model's posterior over the target token at position i. Training therefore reduces to predicting $p_{1|t}$ per position, and the coefficient $\kappa'(t)/(1-\kappa(t))$ is the rate at which a position that has not yet reached its target switches to it.

B.2 Edit Flows

The token-wise path equation 14 assumes a fixed, common length and a position-to-position correspondence between x_0 and x_1. Edit Flows (Havasi et al., 2025) remove this assumption, allowing sequences of differing length related by *edit operations* – insertion, deletion, and substitution – rather than position-wise substitution alone. To recover a factorized path in this setting, the construction is lifted to an auxiliary space of aligned (padded) sequences $\mathcal{Z} = (\mathcal{T} \cup \{\epsilon\})^N$, where ϵ is a padding symbol, together with the map $f_{\mathrm{rm-blanks}} : \mathcal{Z} \to \mathcal{X}$ that deletes all padding. Given endpoints x_0, x_1 and a chosen alignment $z_0, z_1 \in \mathcal{Z}$ with $f_{\mathrm{rm-blanks}}(z_j) = x_j$, the mixture path is defined on the augmented space $\mathcal{X} \times \mathcal{Z}$ as

$$p_t(x, z \mid z_0, z_1) = p_t(z \mid z_0, z_1)\,\delta_{f_{\mathrm{rm-blank}}(z)}(x), \tag{16}$$

where $p_t(z \mid z_0, z_1)$ is the token-wise path equation 14 applied on the aligned sequences. The data-space path is the pushforward of the auxiliary path through $f_{\mathrm{rm-blanks}}$: the factorization lives in

$\mathcal{Z}$, while the observed sequence $x = f_{\mathrm{rm-blanks}}(z)$ can have variable length. Because each aligned position independently takes its source or target value, this is equivalent to sampling edit operations to apply to x_0. We adopt this view for MAELLE. In Section 3.4 (Proposition 1), we show that it is equivalent to the token-wise sampling.

B.3 Equivalence of the edit-based and augmented-space mixture paths

We show, in a setting-agnostic form, that the augmented-space mixture path of Edit Flows (Havasi et al., 2025) is equivalent to directly sampling a subset of edit moves and applying them. The statement is independent of what the edits are: it applies verbatim to sequence edits (insertion, deletion, substitution) and to the electron-occupation edits (FLOW, DEL, ADD) used in the main text.

Setup. Let $\mathcal{D}$ be a state space (e.g. token sequences, or electron occupation vectors) and fix a source state $x_0 \in \mathcal{D}$. Let $\mathcal{M} = \{m_1, \ldots, m_K\}$ be a finite set of edit moves, each an operation on $\mathcal{D}$, and let

$$\mathrm{Apply} : \mathcal{D} \times 2^{\mathcal{M}} \to \mathcal{D} \tag{17}$$

apply a subset of moves to a state, assumed well defined (independent of the order in which the moves in the subset are applied). Write $x_1 = \mathrm{Apply}(x_0, \mathcal{M})$ for the state obtained by applying all moves. Let $\kappa : [0,1] \to [0,1]$ be a schedule with $\kappa_0 = 0$ and $\kappa_1 = 1$.

We compare two ways of building a conditional path $p_t(\cdot \mid x_0, x_1)$ on $\mathcal{D}$ that interpolates from x_0 at $t = 0$ to x_1 at $t = 1$.

(A) Augmented-space path. Introduce the auxiliary space $\mathcal{Z} = \{0,1\}^K$, one binary coordinate per move, with the readout

$$g : \mathcal{Z} \to \mathcal{D}, \qquad g(z) = \mathrm{Apply}\big(x_0, \{m_k : z_k = 1\}\big). \tag{18}$$

Treating each coordinate as an independent binary token that interpolates $z_k : 0 \to 1$ (Campbell et al., 2024; Gat et al., 2024) gives the factorized path on $\mathcal{Z}$

$$p_t(z \mid x_0, x_1) = \prod_{k=1}^{K} \Big[(1-\kappa_t)\,\delta_0(z_k) + \kappa_t\,\delta_1(z_k)\Big], \tag{19}$$

and the augmented-space path on $\mathcal{D}$ is its pushforward through g,

$$p_t^{\mathrm{A}}(x \mid x_0, x_1) = \sum_{z \in \mathcal{Z}} p_t(z \mid x_0, x_1)\, \mathbb{1}[g(z) = x]. \tag{20}$$

For sequence edits with $\mathcal{Z}$ realized as padded sequences and g as padding removal, equation 20 is the mixture path of Havasi et al. (2025).

(B) Edit-based path. Include each move independently with probability κ_t, collect the applied subset $\mathcal{S}$, and apply it to x_0. The resulting law is

$$p_t^{\mathrm{B}}(x \mid x_0, x_1) = \sum_{\mathcal{S} \subseteq \mathcal{M}} \kappa_t^{|\mathcal{S}|} (1-\kappa_t)^{\,K-|\mathcal{S}|}\, \mathbb{1}[\mathrm{Apply}(x_0, \mathcal{S}) = x]. \tag{21}$$

Proposition 1 (Edit-sampling form of the augmented-space path)**.** *For every $t \in [0,1]$ and every $x \in \mathcal{D}$,*

$$p_t^{\mathrm{A}}(x \mid x_0, x_1) = p_t^{\mathrm{B}}(x \mid x_0, x_1). \tag{22}$$

Consequently sampling $z \sim p_t(\cdot \mid x_0, x_1)$ from equation 19 and returning $g(z)$ has the same law as sampling a subset $\mathcal{S}$ of moves by independent $\mathrm{Bernoulli}(\kappa_t)$ *inclusion and returning* $\mathrm{Apply}(x_0, \mathcal{S})$*. In particular $p_0^{\mathrm{A}} = p_0^{\mathrm{B}} = \delta_{x_0}$ and $p_1^{\mathrm{A}} = p_1^{\mathrm{B}} = \delta_{x_1}$, so both are valid conditional interpolants.*

Proof. The map $z \mapsto \mathcal{S}(z) := \{m_k : z_k = 1\}$ is a bijection between $\mathcal{Z} = \{0,1\}^K$ and the power set $2^{\mathcal{M}}$, with inverse $\mathcal{S} \mapsto z$ where $z_k = \mathbb{1}[m_k \in \mathcal{S}]$. We rewrite the augmented-space weight equation 19 under this bijection.

Each factor in equation 19 selects one of its two terms according to z_k: it contributes κ_t when $z_k = 1$ and $1 - \kappa_t$ when $z_k = 0$. Hence for a fixed z,

$$p_t(z \mid x_0, x_1) = \prod_{k=1}^{K} \kappa_t^{z_k} (1-\kappa_t)^{\,1-z_k} = \kappa_t^{\sum_k z_k} \, (1-\kappa_t)^{\,K-\sum_k z_k} = \kappa_t^{|\mathcal{S}(z)|} (1-\kappa_t)^{\,K-|\mathcal{S}(z)|}, \quad (23)$$

using $\sum_k z_k = |\mathcal{S}(z)|$. By definition of the readout equation 18, $g(z) = \mathrm{Apply}(x_0, \mathcal{S}(z))$. Substituting both identities into equation 20 and reindexing the sum from z to $\mathcal{S} = \mathcal{S}(z)$,

$$\begin{aligned} p_t^{\mathrm{A}}(x \mid x_0, x_1) &= \sum_{z \in \mathcal{Z}} \kappa_t^{|\mathcal{S}(z)|} (1-\kappa_t)^{\,K-|\mathcal{S}(z)|} \, \mathbb{1}[\mathrm{Apply}(x_0, \mathcal{S}(z)) = x] \\ &= \sum_{\mathcal{S} \subseteq \mathcal{M}} \kappa_t^{|\mathcal{S}|} (1-\kappa_t)^{\,K-|\mathcal{S}|} \, \mathbb{1}[\mathrm{Apply}(x_0, \mathcal{S}) = x] = p_t^{\mathrm{B}}(x \mid x_0, x_1). \end{aligned} \quad (24)$$

This proves the equality for all t and x.

For the boundary conditions, at $t = 0$ we have $\kappa_0 = 0$, so the weight $\kappa_0^{|\mathcal{S}|}(1-\kappa_0)^{K-|\mathcal{S}|}$ is nonzero only for $\mathcal{S} = \varnothing$, where it equals 1; since $\mathrm{Apply}(x_0, \varnothing) = x_0$, we get $p_0 = \delta_{x_0}$. At $t = 1$, $\kappa_1 = 1$, so the weight is nonzero only for $\mathcal{S} = \mathcal{M}$, where it equals 1; since $\mathrm{Apply}(x_0, \mathcal{M}) = x_1$, we get $p_1 = \delta_{x_1}$. □

Corollary 1 (Sequence edits)**.** *Taking $\mathcal{D}$ to be token sequences and $\mathcal{M}$ a set of insertion, deletion, and substitution moves aligning x_0 to x_1, Proposition 1 recovers the augmented-space mixture path of Edit Flows (Havasi et al., 2025), with $\mathcal{Z}$ realized as padded sequences and $g = f_{\mathrm{rm-blanks}}$ that strips all padding tokens.*

Corollary 2 (Electron edits)**.** *Taking $\mathcal{D} = \mathbb{Z}_{\geq 0}^{|\mathcal{E}|}$, $x_0 = o_x$, $\mathcal{M}$ the move set of Eq. equation 6, and* $\mathrm{Apply} = \mathrm{ApplyEdit}$*, Proposition 1 shows that the edit-based mixture path equation 8 equals the augmented-space pushforward, justifying the interpolant-sampling scheme of Algorithm 1.*

C Problem-statement details

Atoms, adjacency, and electron sites. $\mathcal{V} = \{v_i \mid 1 \leq i \leq |\mathcal{V}|\}$ is the set of heavy atoms; hydrogens are not nodes but are pooled into per-atom electron sites. The adjacency $\mathcal{A} = \{a_{ij} \in \{0,1\} \mid 1 \leq i < j \leq |\mathcal{V}|\}$ follows the standard molecular graph convention Duval et al. (2023). The electron-site set

$$\mathcal{E} = \{e_{ij} \mid 1 \leq i \leq j \leq |\mathcal{V}|\} \cup \{e_i^H \mid 1 \leq i \leq |\mathcal{V}|\}$$

has $|\mathcal{E}| = \binom{|\mathcal{V}|}{2} + 2|\mathcal{V}|$ entries: bonding sites e_{ij} ($i \neq j$), lone-pair sites e_{ii}, and per-atom hydrogen sites e_i^H. Hydrogens bonded to the same heavy atom are topologically equivalent so a single e_i^H suffices Chen et al. (2025). Each site is either *realized* (existing bond/lone pair/hydrogen attachment) or *virtual* ($o(e) = 0$, available to become realized).

Occupation ⇒ topology. The map $f_{o \to a} : (\mathcal{V}, \mathcal{E}, o) \mapsto \mathcal{A}$ deterministically recovers the heavy-atom adjacency from the occupation via Eq. equation 1. Combined with the fact that $\mathcal{V}$ (and hence $\mathcal{E}$) is preserved by a reaction, this yields the simplification

$$p(\mathcal{G}_y \mid \mathcal{G}_x) = p(\mathcal{V}_y, \mathcal{A}_y, \mathcal{E}_y, o_y \mid \mathcal{V}_x, \mathcal{A}_x, \mathcal{E}_x, o_x) = p(o_y \mid \mathcal{V}_x, \mathcal{A}_x, \mathcal{E}_x, o_x).$$

Ethanol example. Consider ethanol (CH_3CH_2OH) with heavy atoms $\mathcal{V} = \{\mathrm{C}_1, \mathrm{C}_2, \mathrm{O}_3\}$. The electron sites and their occupations are:

Site	Electron type	Occupation o	Description
$e_{1,2}$ (C_1–C_2)	Bond	1	Single bond
$e_{2,3}$ (C_2–O)	Bond	1	Single bond
$e_{1,3}$ (C_1–O)	Virtual	0	Virtual (no bond)
$e_{3,3}$ (O lone pairs)	Valence	2	Two lone pairs
$e_{1,1}$, $e_{2,2}$	Virtual	0	No lone pairs on carbons
e_1^H (C_1)	w/ Hydrogens	3	Three hydrogens
e_2^H (C_2)	w/ Hydrogens	2	Two hydrogens
e_3^H (O)	w/ Hydrogens	1	One hydrogen

This gives a total of 10 occupied electron pairs across 9 sites, consistent with ethanol's 20 valence electrons.

D Optimal Transport Details

To handle reactions where the total electron count is not conserved, we augment the electron site set with a virtual site $e_\varnothing$ acting as a source and sink for electron pairs gained or lost. Let $\bar{\mathcal{E}} = \mathcal{E} \cup \{e_\varnothing\}$ and define augmented occupation vectors $\bar{o}_x, \bar{o}_y \in \mathbb{Z}_{\geq 0}^{|\bar{\mathcal{E}}|}$:

$$\bar{o}_x(e) = \begin{cases} o_x(e) & \text{if } e \in \mathcal{E}, \\ \sum_{e' \in \mathcal{E}} \max(o_y(e') - o_x(e'),\, 0) & \text{if } e = e_\varnothing, \end{cases}$$

$$\bar{o}_y(e) = \begin{cases} o_y(e) & \text{if } e \in \mathcal{E}, \\ \sum_{e' \in \mathcal{E}} \max(o_x(e') - o_y(e'),\, 0) & \text{if } e = e_\varnothing, \end{cases}$$

so that $\sum_{e \in \bar{\mathcal{E}}} \bar{o}_x(e) = \sum_{e \in \bar{\mathcal{E}}} \bar{o}_y(e)$, ensuring balanced transport.

Cost matrix. We define the cost matrix $C \in \mathbb{Z}_{\geq 0}^{|\bar{\mathcal{E}}| \times |\bar{\mathcal{E}}|}$ based on shortest-path distances. Let $d_x(v, v')$ and $d_y(v, v')$ denote shortest-path distances on the reactant and product graphs, respectively. Since bonds may form or break, we take the minimum over both:

$$C_{e,e'} = \min \left(\min_{\substack{v \in \text{atoms}(e) \\ v' \in \text{atoms}(e')}} d_x(v, v'), \min_{\substack{v \in \text{atoms}(e) \\ v' \in \text{atoms}(e')}} d_y(v, v') \right), \qquad \forall\, e, e' \in \mathcal{E},$$

where $\text{atoms}(e) \subseteq \mathcal{V}$ denotes the atoms associated with site e:

$$\text{atoms}(e) = \begin{cases} \{v_i, v_j\} & \text{if } e = e_{i,j} \text{ with } i \neq j, \\ \{v_i\} & \text{if } e = e_{i,i} \text{ or } e = e_i^H. \end{cases}$$

The cost to or from the virtual site is set to a fixed large constant $C_{e,e_\varnothing} = C_{e_\varnothing,e} = c_\varnothing$, incentivizing the solver to use $e_\varnothing$ only when necessary.

E Algorithms

Algorithm 1 SampleIntermediateGraph($\mathcal{G}_0,\ \mathbf{M},\ \kappa$)

Require: Reactant graph $\mathcal{G}_0$, set of electron moves $\mathbf{M}$, interpolation level $\kappa \in [0, 1]$
Ensure: Intermediate graph $\mathcal{G}_t$, remaining target moves $\mathbf{M}_{\text{target}}$

for each move $(e^s, e^d) \in \mathbf{M}$ **do**
    Sample $z_{s,d} \sim \text{Bernoulli}(\kappa)$
end for
$\mathbf{M}_{\text{fired}} \leftarrow \{(e^s, e^d) : z_{s,d} = 1\}$ ▷ moves already applied
$\mathbf{M}_{\text{target}} \leftarrow \{(e^s, e^d) : z_{s,d} = 0\}$ ▷ moves still to predict
$\mathcal{G}_t \leftarrow \mathit{ApplyElectronEdits}(\mathcal{G}_0,\ \mathbf{M}_{\text{fired}})$
return $\mathcal{G}_t,\ \mathbf{M}_{\text{target}}$

Algorithm 2 MAELLE's Training Loop

Require: Data distribution $p(\mathbf{x}, \mathbf{y})$, model u_θ, kappa schedule $\kappa(\cdot)$

1: Sample $(\mathcal{G}_0,\ \mathcal{G}_1) \sim p(\mathbf{x}, \mathbf{y})$ ▷ reactant and product graphs
2: $\mathbf{M} \leftarrow SolveOptimalTransport(\mathcal{G}_0,\ \mathcal{G}_1)$ ▷ electron moves from OT plan
3: Sample $t \sim \mathrm{Uniform}(0, 1)$
4: $\mathcal{G}_t,\ \mathbf{M}_{\mathrm{target}} \leftarrow SampleIntermediateGraph(\mathcal{G}_0,\ \mathbf{M},\ \kappa(t))$ ▷ Alg. 1

5: **// Forward pass**
6: $\boldsymbol{\lambda}^{\mathrm{flow}},\ \boldsymbol{\lambda}^{\mathrm{del}},\ \boldsymbol{\lambda}^{\mathrm{add}},\ \mathcal{E} \leftarrow u_\theta(\mathcal{G}_t,\ \mathcal{G}_0,\ t)$
▷ per-site flow/del rates over $\mathcal{E}$; per-atom add rates over $\mathcal{V}$

7: **// Bregman divergence loss**
8: Compute $\mathcal{L}$ and update θ via back-propagation:

$$\mathcal{L} \;=\; \sum_{e \in \mathcal{E}} \big(\lambda_e^{\mathrm{flow}} + \lambda_e^{\mathrm{del}}\big) + \sum_{a \in \mathcal{V}} \lambda_a^{\mathrm{add}} \;-\; \frac{\kappa'(t)}{1-\kappa(t)} \sum_{(e^s,\, e^d) \,\in\, \mathbf{M}_{\mathrm{target}}} \begin{cases} \log \lambda_a^{\mathrm{add}} + \log p_\theta^{\mathrm{sink}}(e^d \mid a,\ \mathcal{E}) & \text{if } e^s = \varnothing \\ \log \lambda_{e^s}^{\mathrm{del}} & \text{if } e^d = \varnothing \\ \log \lambda_{e^s}^{\mathrm{flow}} + \log p_\theta^{\mathrm{sink}}(e^d \mid e^s,\ \mathcal{E}) & \text{otherwise} \end{cases}$$

Algorithm 3 SampleMoves($\mathcal{G}_t,\ \boldsymbol{\lambda}^{\text{flow}},\ \boldsymbol{\lambda}^{\text{del}},\ \boldsymbol{\lambda}^{\text{add}},\ \mathcal{E},\ \Delta t,\ \tau$)

Require: Rates $\boldsymbol{\lambda}^{\text{flow}}, \boldsymbol{\lambda}^{\text{del}}$ (per site), $\boldsymbol{\lambda}^{\text{add}}$ (per atom), electron sites $\mathcal{E}$, step size Δt, temperature τ
Ensure: Set of moves $\mathbf{M}_{\text{step}}$

// Fire sites via thinned Poisson process
for each site $e \in \mathcal{E}$ **do**
  $p_e^{\text{site}} \leftarrow 1 - \exp\big(-(\lambda_e^{\text{flow}} + \lambda_e^{\text{del}})\,\Delta t\,/\,\tau\big)$
  Sample $f_e \sim \text{Bernoulli}(p_e^{\text{site}})$
end for
$\mathcal{F} \leftarrow \{e : f_e = 1\}$ ▷ fired sites

// Fire addition atoms
for each atom $a \in \mathcal{V}$ **do**
  $p_a^{\text{add}} \leftarrow 1 - \exp\big(-\lambda_a^{\text{add}}\,\Delta t\,/\,\tau\big)$
  Sample $f_a \sim \text{Bernoulli}(p_a^{\text{add}})$
end for
$\mathcal{A}_{\text{add}} \leftarrow \{a : f_a = 1\}$ ▷ fired addition atoms

// Resolve actions for fired sites
$\mathbf{M}_{\text{flow}},\ \mathbf{M}_{\text{del}} \leftarrow \varnothing,\ \varnothing$
for each $e^{\text{fired}} \in \mathcal{F}$ **do**
  action $\leftarrow \begin{cases} \textit{flow} & \text{w.p. } \lambda_e^{\text{flow}}/(\lambda_e^{\text{flow}} + \lambda_e^{\text{del}}) \\ \textit{del} & \text{otherwise} \end{cases}$
  if action $=$ *flow* **then**
    Sample sink $e^d \sim p_\theta^{\text{sink}}(\,\cdot \mid e^{\text{fired}},\ \mathcal{E})$
    $\mathbf{M}_{\text{flow}} \leftarrow \mathbf{M}_{\text{flow}} \cup \{(e^{\text{fired}}, e^d)\}$
  else
    $\mathbf{M}_{\text{del}} \leftarrow \mathbf{M}_{\text{del}} \cup \{(e^{\text{fired}}, \varnothing)\}$
  end if
end for
Resolve conflicts in $\mathbf{M}_{\text{flow}}$: keep highest λ_e^{flow} per destination

// Resolve actions for fired additions
$\mathbf{M}_{\text{add}} \leftarrow \varnothing$
for each $a \in \mathcal{A}_{\text{add}}$ **do**
  Sample sink $e^d \sim p_\theta^{\text{sink}}(\,\cdot \mid a,\ \mathcal{E})$
  $\mathbf{M}_{\text{add}} \leftarrow \mathbf{M}_{\text{add}} \cup \{(\varnothing, e^d)\}$
end for

return $\mathbf{M}_{\text{step}} \leftarrow \mathbf{M}_{\text{flow}} \cup \mathbf{M}_{\text{del}} \cup \mathbf{M}_{\text{add}}$

Algorithm 4 MAELLE's Inference

Require: Reactant graph $\mathcal{G}_0$, trained model u_θ, number of steps N, temperature τ
Ensure: Predicted product graph $\hat{\mathcal{G}}_1$

$\mathcal{G}_t \leftarrow \mathcal{G}_0, \quad \Delta t \leftarrow 1/N$
Encode reactant: $\mathbf{z}_0 \leftarrow Encode_{u_\theta}(\mathcal{G}_0)$ ▷ computed once, reused at every step

for $n = 0, \ldots, N-1$ **do**
  $t \leftarrow n \,/\, N$
  $\Delta t_{\text{adapt}} \leftarrow AdaptiveStepSize(\Delta t,\ t,\ \kappa)$ ▷ adaptive h from schedule

  // Decode current state
  $\boldsymbol{\lambda}^{\text{flow}},\ \boldsymbol{\lambda}^{\text{del}},\ \boldsymbol{\lambda}^{\text{add}},\ \mathcal{E} \leftarrow Decode_{u_\theta}(\mathcal{G}_t,\ \mathbf{z}_0,\ t)$
    ▷ per-site flow/del rates; per-atom add rates; $\mathcal{E}$: electron sites in $\mathcal{G}_t$

  // Sample and apply moves
  $\mathbf{M}_{\text{step}} \leftarrow SampleMoves(\mathcal{G}_t,\ \boldsymbol{\lambda}^{\text{flow}},\ \boldsymbol{\lambda}^{\text{del}},\ \boldsymbol{\lambda}^{\text{add}},\ \mathcal{E},\ \Delta t_{\text{adapt}},\ \tau)$ ▷ Alg. 3
  $\mathcal{G}_t \leftarrow ApplyElectronEdits(\mathcal{G}_t,\ \mathbf{M}_{\text{step}})$
end for

return $\hat{\mathcal{G}}_1 \leftarrow \mathcal{G}_t$

F Neural Network Architecture

MAELLE uses a conditional encoder-decoder architecture. The *encoder* processes the fixed reactant graph $\mathcal{G}_0$ to produce conditioning representations; the *decoder* processes the time-dependent intermediate graph $\mathcal{G}_t$ and cross-attends to the encoder outputs. Both operate at two levels of granularity: atoms and electron sites (entries).

Notation. Let d denote the model dimension, N the number of heavy atoms in a graph, and $N_e = \binom{N}{2} + 2N$ the number of electron sites (valence, hydrogen, and bond entries). We write $\mathbf{h} \in \mathbb{R}^{N \times d}$ for atom embeddings and $\mathbf{e} \in \mathbb{R}^{N_e \times d}$ for entry embeddings.

F.1 Graph encoder (GNN backbone)

Both the encoder and decoder share the same GNN architecture but with separate weights. The backbone is a stack of L Graph Attention Network (GAT) layers Veličković et al. (2017). Intermediate layers use multi-head attention with 4 heads and concatenation (output dimension $4 \times d_{\text{hidden}}$), while the final layer uses a single head without concatenation (output dimension d). An ELU activation is applied between intermediate layers.

Input atom features are first projected via a linear layer $\mathbf{h}^{(0)} = W_{\text{node}}\mathbf{x} + \mathbf{b}_{\text{node}}$, where $\mathbf{x} \in \mathbb{R}^{N \times d_{\text{in}}}$ are the raw atom features.

Time embedding. The decoder's GNN incorporates the flow time $t \in [0, 1]$ via a sinusoidal positional embedding followed by a two-layer MLP with SiLU activation:

$$\mathbf{t}_{\text{emb}} = \text{MLP}_{\text{time}}(\text{SinEmb}(t)) \in \mathbb{R}^{d_{\text{hidden}}},$$

which is broadcast-added to the initial node embeddings $\mathbf{h}^{(0)}$ before the GAT layers. The encoder's GNN omits the time embedding, as the reactant graph $\mathcal{G}_0$ is fixed.

F.2 Two-level attention

After the GNN backbone, both encoder and decoder refine representations through two stages of attention operating at different granularities.

Stage 1: Atom-level attention. The atom embeddings $\mathbf{h} \in \mathbb{R}^{N\times d}$ from the GNN are passed through a Transformer. In the *encoder*, this is a TransformerEncoder (self-attention only). In the *decoder*, this is a TransformerDecoder: each layer applies self-attention over the decoder's atom embeddings, followed by cross-attention to the encoder's atom embeddings. This allows the decoder to condition on the reactant atom representations at every layer.

Stage 2: Entry-level feature construction and attention. Atom embeddings are lifted to electron-site (entry) embeddings via three learned projections:

- **Valence entries** (lone pairs): $\mathbf{e}_i^{\text{val}} = \mathrm{ReLU}(W_{\text{val}}[\mathbf{h}_i \,\|\, \phi(n_i^{\text{nb}})])$ for each atom i, where n_i^{nb} is the non-bonding electron count and $\phi(\cdot)$ is a 4-bin one-hot encoding of the electron pair count (0, 1, 2, $\geq$3 pairs).
- **Hydrogen entries**: $\mathbf{e}_i^H = \mathrm{ReLU}(W_H[\mathbf{h}_i \,\|\, \phi(n_i^H)])$ for each atom i, where n_i^H is the hydrogen electron count.
- **Bond entries**: $\mathbf{e}_{ij}^{\text{bond}} = \mathrm{ReLU}(W_{\text{bond}}[\mathrm{agg}(\mathbf{h}_i, \mathbf{h}_j) \,\|\, \rho(d_{ij}) \,\|\, \phi(n_{ij}^{\text{bond}})])$ for each realized bond (i, j), where $\mathrm{agg}(\cdot,\cdot)$ is a symmetric aggregation (sum, mean, Hadamard, or bilinear), $\rho(d_{ij})$ is a one-hot encoding of the topological distance (up to 10 hops), and n_{ij}^{bond} is the bond electron count.

These three sets of entry embeddings are interleaved into a single sequence of N_e entries per graph and passed through another Transformer block — self-attention in the encoder, self-attention plus cross-attention to the encoder's entry embeddings in the decoder.

F.3 Output heads

The decoder produces three types of predictions from the entry embeddings, corresponding to the source–sink factorization in Eq. 12:

Rate head. A 3-layer MLP (with ReLU activations and hidden dimension $2d$) maps each entry embedding to three non-negative scalars via softplus:

$$[\lambda_{\theta,t}^{\text{flow}}(e),\ \lambda_{\theta,t}^{\text{del}}(e),\ \lambda_{\theta,t}^{\text{add}}(e)] = \mathrm{softplus}(\mathrm{MLP}_{\text{rate}}(\mathbf{e})).$$

The addition rate $\lambda_{\theta,t}^{\text{add}}$ is only used for atom-level entries (valence sites); for bond entries it is masked out.

Sink heads. Two separate MLPs parameterize the sink distributions $p_\theta^{\text{sink,flow}}$ and $p_\theta^{\text{sink,add}}$. Given a fired source entry with embedding $\mathbf{e}^s$ and a candidate destination with embedding $\mathbf{e}^d$, the unnormalized score is:

$$s(e^s, e^d) = \mathrm{MLP}_{\text{sink}}([\mathbf{e}^s \,\|\, \mathbf{e}^d]),$$

where $[\cdot \,\|\, \cdot]$ denotes concatenation and the MLP maps $\mathbb{R}^{2d} \to \mathbb{R}$. The sink distribution is obtained by applying softmax over all valid candidates. The candidate sets are kept local: for FLOW, the destination must share at least one atom with the source, restricting the candidate set to $O(|\mathcal{V}|)$ per fired source; for ADD from atom a, the candidates are $\{e_{a,a}\} \cup \{e_a^H\} \cup \{e_{a,j} \mid j \neq a\}$, yielding at most $|\mathcal{V}| + 1$ candidates. For virtual bond entries (destinations that do not yet exist in $\mathcal{G}_t$), representations are constructed on-the-fly from the atomic embeddings of the endpoint atoms.

F.4 Training details

The full MAELLE model has approximately **16M** trainable parameters with hidden dimension $d = 256$ throughout. The GNN backbone uses **2** GAT layers; the encoder applies **4** fully-connected attention layers (**2** atom-level + **2** entry-level), and the decoder applies **6** fully-connected attention layers (**3** atom-level + **3** entry-level), with the latter two stages including cross-attention to the encoder. We train for $\sim$**50** epochs (about **72** hours on a single **NVIDIA H100** GPU) with an effective batch size of **64**, using gradient accumulation to fit memory.

G QUANTIFYING PRODUCT PLAUSIBILITY

G.1 LLM AS A JUDGE

To evaluate the chemical plausibility of predicted products beyond exact-match top-k accuracy (Section G.2), we use a large language model as a chemistry judge. Concretely, we query `gemini-3-flash` with *minimal* thinking effort. For each reaction, the LLM is given the reactant SMILES and a *anonymized* list of candidate products pooled across MAELLE, Molecular Transformer, and Graph2SMILES (with the ground-truth product additionally injected as a sanity-check item). The judge has no information about which model produced which candidate. Each of the three independent runs uses a fresh shuffle and a non-zero sampling temperature, providing the standard deviation reported in Figure 4a.

Prompt structure. The query consists of a system prompt followed by a per-reaction user message assembled from three templated blocks (PREFIX, INPUTS, SUFFIX):

> **System prompt.** *You are an expert organic chemist. Your task is to evaluate whether predicted products are chemically plausible outcomes of a given reaction. A product is plausible if it could reasonably arise from the listed reactants under standard organic-chemistry conditions, considering valence, connectivity, common reactivity, and likely mechanisms. You do not need the product to be the major product — minor or side products are acceptable as long as they are mechanistically reasonable.*
>
> **Prefix.** *Reaction* `<reaction_id>`*. Reactants (SMILES):* `<reactant_smiles>`*.*
>
> **Inputs.** *Below is a list of candidate products. Evaluate each candidate independently.*
>
> ```
> [1] <product_smiles_1>
> [2] <product_smiles_2>
> ...
> [n] <product_smiles_n>
> ```
>
> **Suffix.** *For each candidate, provide a brief mechanistic reasoning and a plausibility verdict (*`true/false`*). Return your answer as a JSON object of the form:*
>
> ```
> {
>   "verdicts": [
>     {"index": 1, "reasoning": "...", "plausible": true},
>     {"index": 2, "reasoning": "...", "plausible": false},
>     ...
>   ]
> }
> ```

Aggregation. For each reaction and each model, the top-k plausibility rate is the fraction of the model's first k predictions that the judge marks `plausible`. We report the mean and standard deviation over three independent LLM runs (Figure 4a). To control for systematic over- or under-permissiveness of the judge, we additionally inject the ground-truth product into the candidate list and report its plausibility coverage as a sanity check; across runs we obtain $81.1 \pm 0.1\%$ ground-truth coverage, indicating a moderately conservative but stable judge.

Anonymization and de-duplication. Because MAELLE typically yields fewer unique candidates than beam-search SMILES models, for every test case we cap the per-model candidate list at the smallest number of unique predictions across the three models, ensuring a balanced comparison. Within a single query the candidate list is shuffled, and SMILES are canonicalized with RDKit before being passed to the judge so that surface-form differences do not bias the verdict.

G.2 CHEMICAL PLAUSIBILITY OF PREDICTED PRODUCT CANDIDATES

By modeling reaction as a generative problem, simulating multiple trajectories, and taking the most popular outputs, MAELLE somewhat mimics how a real chemistry process happens, where multiple pathways compete with others. To characterize this behavior, we sampled 5000 cases from the test

set of USPTO-480K where MAELLE has at least 2 high-confidence predictions with confidence no less than 7.8%, corresponding with 5 out of 64 samples. These examples are then matched with the predictions of Molecular Transformers and Graph2SMILES. As sequence-based methods, these model relies on beam-search for sampling different outcomes in the token space, in contrast with MAELLE which samples in a more chemistry-friendly electron space.

As recent LLMs have shown remarkable symbolic understanding and chemistry knowledge Bran et al. (2025); Voinarovska et al. (2026), we opted to use LLM-as-a-judge to address the plausibility of the proposed outcomes. Due to the inference nature, the number of unique outcomes from MAELLE is much less than what comes out of Molecular Transformers or Graph2SMILES and thus we capped the number of predictions for test case to be the smallest number of predictions among the models. Given a precursor set and a list of products, the LLM has to give a verdict of whether each of the products is plausible or not, without knowing which prediction comes from which model. As a sanity check, we also include the ground truth product into this list and measure how much proportion of the ground truth is deemed as plausible by the LLM (see Appendix G.1 for more details). The plausibility rate is then calculated as the number of predictions classified as plausible by the LLMs divided by the total number of predictions according to top-k.

Figure 4a displays side-by-side top-k accuracy and top-k plausibility rate. Agreeing with the USPTO-480K results in Table 1, top-k accuracy is ranked in the order of G2S $>$ MT $>$ MAELLE. However, top-1 plausibility rate of MAELLE (80.7%) is significant higher than its top-1 accuracy (60.3%), suggesting that incorrect predictions are not always implausible. Furthermore, we observe a reversed order when it comes to top-2 and top-3 plausibility rates, with MAELLE being the highest, followed by Molecular Transformers and Graph2SMILES. This hints at the fact that SMILES-based models are likely trained to overfit to one single outcome, which conflicts with the stochastic nature of chemical reactions.

For demonstration, Figure 4b shows an example where an incorrect prediction by MAELLE is still considered plausible by the LLM-judge. In the test dataset, this reaction is a bromide substitution reaction involving an imidazole ring with two nucleophilic nitrogen N:10 and N:12. The ground truth product features an attack of N:12 to C:21 as the main product, likely attributed to the fact that N:12 is less sterically hindered by the benzoyl group compared with N:10. All three models manage to predict the correct ground truth as top-1. However, intuitively speaking, the side product involved N:10 is not entirely impossible, and MAELLE was able to recover this minor outcome as its top-2 prediction. On the other hand, due to the fact that Graph2SMILES and Molecular Transformers sample in the token space, top-2 predictions of these models are just hallucinated versions of their top-1 prediction. This example is also aligned with the observation that AI systems have limited capability in writing SMILES and other symbolic sequences, which is one of the major bottlenecks in AI applications for chemistry Edwards et al. (2022); Jang et al. (2024).

Figure 4c showcases how the step-by-step CTMC of electron edits enables MAELLE to sample multiple outcomes that agree with chemical mechanisms. The confidence scores assigned to each outcome successfully differentiate the major and minor products, agreeing with the ground truth.

H Persistent and transient bond analysis

Table 4: MAELLE's coverage of bond-forming elementary steps categorized into *persistent* and *transient* bonds.

Bond formation type	**# ground-truth elementary steps**	**Recovered by MAELLE's trajectories**		
		Correct	**Top 1**	**All**
Persistent	17,925	13,137 (73.3%)	14,236 (79.4%)	17,847 (99.6%)
Transient	19,658	3,042 (15.5%)	747 (3.8%)	5,814 (29.6%)
Total	37,583	16,179 (43.0%)	14,983 (39.9%)	23,661 (63.0%)

Table 4 separates annotated bond-forming steps by whether the new bond survives in the recorded product. Persistent bonds are recovered more often: pooling all 64 sampled trajectories covers 17,847 of 17,925 persistent steps (99.6%), compared with 5,814 of 19,658 transient steps (29.6%). The difference is consistent with MAELLE's training signal. Its electron moves are inferred from

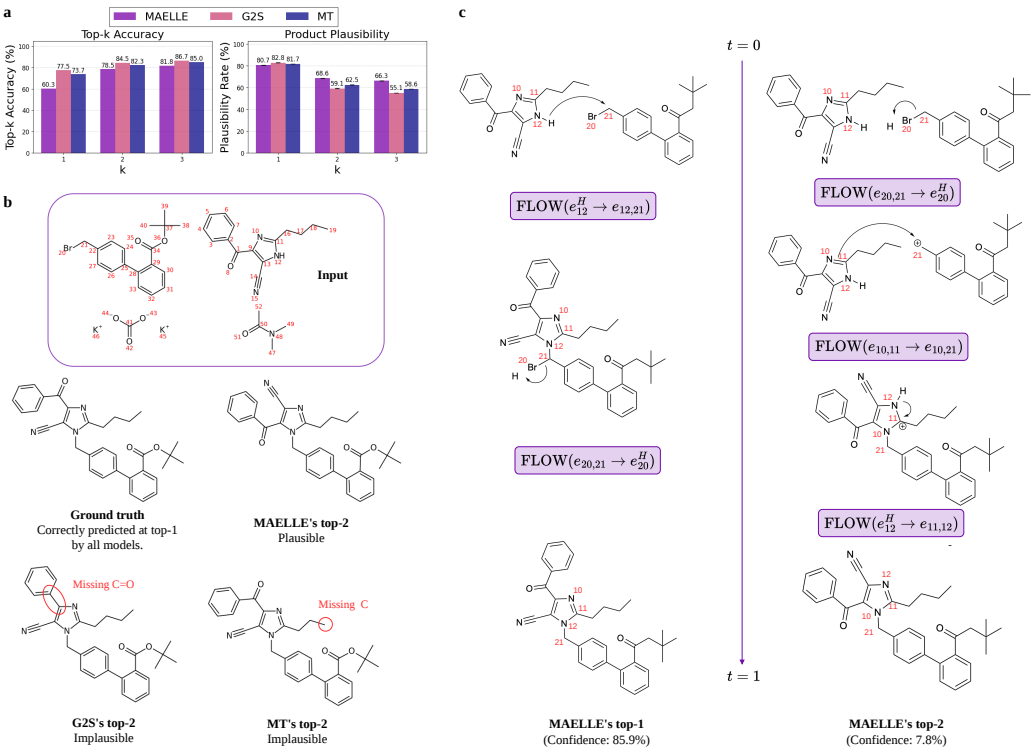


Figure 4: **a.** Top-k accuracy and plausibility rate, with the latter having mean and standard deviation reported for 3 runs of LLM evaluation. The LLM-judge has a ground truth coverage of $81.1 \pm 0.1\%$.**b.** Test case 39591 from the test set of USPTO-480K serves as an illustrative example where MAELLE successfully predicts the main product as top-1 and the side product as top-2, while Graph2SMILES and MolecularTransformers propose hallucinated outcomes. **c.** Two example trajectories that MAELLE samples leading to the two different outcomes.

reactant–product endpoints, which identify product-defining changes but generally do not reveal bonds formed and broken along the intervening pathway.

Coverage also depends on which trajectories are considered. For persistent bonds, it rises from 73.3% among trajectories reaching the recorded product to 79.4% for the most frequently predicted product and 99.6% when all trajectories are pooled. Transient-bond coverage remains lower in every mode (15.5%, 3.8%, and 29.6%, respectively).

I DATASETS

I.1 USPTO-480K

All experiments use the USPTO-480K reaction corpus (Lowe, 2012a; 2017). This appendix describes the preprocessing applied to the corpus (Appendix I.1), the standard and re-split partitionings used in the main text, and the construction of the out-of-distribution test axes (Appendix K.2).

The following steps are applied once to every reaction in the corpus, prior to any split.

Re-mapping. USPTO-480K ships with atom-mapped SMILES of the form `reactants>reagents>products`, but the provided maps are noisy and occasionally inconsistent. Because MAELLE's electron-site alignment relies on a correct atom correspondence between reactants and products (Section 3.1), we discard the original maps and re-map every reaction with RXNMapper (Schwaller et al., 2021), yielding a consistent atom mapping across the corpus.

Electron-move interpolation. For each re-mapped reaction, we solve the optimal-transport problem of Section 3.3 once, ahead of training, to obtain the move set $\mathcal{M}$ that transforms the reactant occupation o_x into the product occupation o_y. These precomputed moves are cached and reused across epochs; the per-step Bernoulli sampling of the mixture path (Section 3.4) is then applied at training time.

Atom maps and model inputs. The re-mapped correspondence is used to construct MAELLE's occupation targets and is therefore retained for our model. For the SMILES- and graph-based baselines, which do not consume atom maps, maps are stripped from the inputs so that all baselines receive the same map-free reactions they were designed for. This is a property of each model's input format, not an asymmetry in the data: every model sees the same reactions, and only MAELLE makes use of the atom correspondence.

Deduplication. Exact duplicates are identified by a canonical reaction key canon(reactant) $\gg$ canon_main(product), where canon_main selects the largest product fragment by heavy-atom count after stripping atom maps and canonicalizing with RDKit (RDKit, online). Pairwise overlap between all splits is verified to be zero on this key.

I.2 AbSynth

AbSynth is given as a CSV with one column recording the reactants, one column recording the product, and reagents (solvents, catalyst) are put into a separate column, mixed with other information such as reaction conditions, First, we separate the reagents from the reaction condition column and map the 542 most popular reagents to SMILES strings using LLMs. Reagents that are not found in this mapping fell back to OPSIN, and if OPSIN also fails, then the whole reaction is skipped. The reagents are merged with reactants to be used as inputs to all models. After deduplication, in the end we have 51,456 unique reactions with stereo centers, and 50,927 unique reactions if stereo centers are stripped. The non-stereo centers test set is fed to all models.

J Comparison between AbSynth and USPTO-480K

As AbSynth contains data from total synthesis chemical literature Czuba et al. (2026), one of the most complex fields of organic chemistry, we expect it to be a more challenging dataset than USPTO-480K which is limited to pantented reactions. To prove this, we display in Figure 5 the distribution of molecular weights of the products and the corresponding number of rings as measures of molecular complexity, as well as the number of ring forming reactions as a measure of difficulty of the reaction prediction task. As AbSynth presents a higher fraction of both large molecular weight products and large number of rings, as well as a higher presence of ring forming reactions, we conclude that AbSynth is more complex than USPTO-480K from the point of view of all computed metrics.

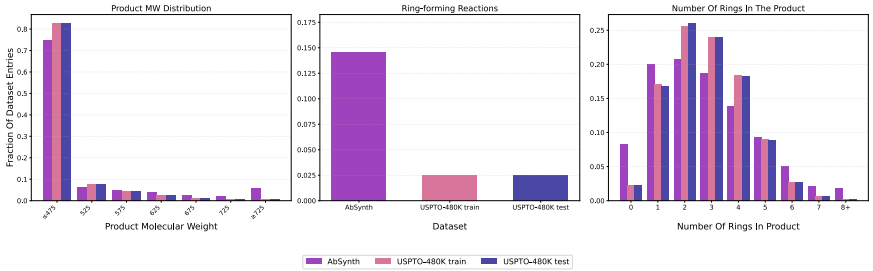


Figure 5: Comparison between the AbSynth dataset and the train and test shards of USPTO-480K. Having products of higher molecular weight than USPTO-480K and also a more complex molecular topology (as illustrated by the larger number of rings in the products), AbSynth proves to be a more complex dataset than USPTO-480K from a structural point of view. Moreover, the larger number of ring forming reactions increases the difficulty of the reaction prediction task, AbSynth being thus a more challenging benchmark that USPTO-480K.

K Diagnostic OOD Benchmark

K.1 Standard and low-MW splits

For the standard benchmark (Section 4.2) we use the corpus's original train/validation/test partition. For the out-of-distribution study (Section K.3) we re-partition the corpus as follows. The base train/validation/IID-test partition is constructed by Gaussian sampling over the product molecular weight (MW), with mean $250\,\mathrm{Da}$ and variance $1643.782\,\mathrm{Da}^2$ (std $\approx 40.5\,\mathrm{Da}$); we refer to this as the *low-MW split*. The deliberately low mean ensures the training distribution is well separated from the high-mass OOD axis below. The pooled train+validation set is partitioned 80/20, giving $304\,228$ training reactions after deduplication, a validation set of $37\,582$, and an IID test set of $37\,578$ drawn from the remaining half of the validation fold.

K.2 Out-of-distribution test axes

OOD Mass (high molecular weight). Similary to the training corpus, a test set was constructed using Gaussian sampling with mean 750 Da and variance of 6575.13 Da^2. The parameters of the Gaussian were chosen in order to ensure a clear separation between this set and the training-validation one. The test set described will be called hereon *test_ood_mass* ($38\,909$ reactions, median product MW 576 Da). Reactions that were present in *test_ood_mass* and also in the training-validation set were eliminated in the latter. Because the Molecular Transformer must generate the full product SMILES autoregressively, the length of the output sequence scales with molecular size; the mechanism model, which predicts only local bond edits, is invariant to this factor. This split therefore directly probes sensitivity to molecular size.

OOD Ester (non-methyl ester chemistry). An additional axis was constructed to isolate ester-forming reactions that produce linkages other than methyl esters – a transformation class under-represented in the training set relative to its synthetic importance. Reactions belonging to the following named reaction classes were excluded from all training and IID evaluation splits and instead pooled into a dedicated test set, *test_ood_ester* (1779 reactions): *Ethyl esterification*, *CO_2H-tBu protection*, *O-Acetylation*, *O-Piv protection*, *O-Formylation*, *Acetoxy thioether synthesis*, *Salol reaction*, *Yamaguchi esterification*, *Yamaguchi lactonization*, *Shiina macrolactonization*, *2-Benzofuranone synthesis*, and *Baeyer–Villiger oxidation*. In addition, eight reaction classes that produce ester linkages of ambiguous type (*Esterification*, *Ester Schotten–Baumann*, *Fischer–Speier esterification*, *Steglich esterification*, *Mitsunobu ester synthesis*, *Transesterification*, *Carboxylic anhydride alcoholysis*, and *Diazoalkane esterification*) were dropped from all splits entirely to prevent soft leakage of ester-chemistry signal into the training distribution.

Table 5: Split sizes, sampling MW (mean $\pm$ std of the Gaussian over product molecular weight), and OOD criteria for `uspto_unified_v3`.

Split	N	MW (mean $\pm$ std) [Da]	OOD criterion
train	$304\,228$	311.9 ± 90.6	–
val	$37\,582$	312.7 ± 90.6	–
test_iid	$37\,578$	311.3 ± 90.5	none (IID)
test_ood_mass	$38\,909$	585.2 ± 83.0	product MW $> 529\,\mathrm{Da}$
test_ood_ester	1779	321.1 ± 112.1	non-methyl ester chemistry

K.3 OOD Diagnostic Results

A major pitfall of reaction-prediction models is poor generalization outside their training domain Gil et al. (2023); Bradshaw et al. (2025); Tran et al. (2026). We re-split USPTO-480K along two axes – molecular complexity (*OOD Mass*) and reaction novelty (*OOD Ester*, with all non-methyl esterifications held out from training) – alongside an in-distribution control (IID); see Appendix K.2.

Top-k accuracies on these three test sets are shown for MAELLE, Graph2SMILES, MolecularTransformers, and MEGAN in Figure 6a. As expected, all models perform on par with each other on the IID test, with top-10 accuracies of approximately 90%, agreeing with the trend in the test set

of the original USPTO-480K benchmark. On the "OOD Mass", models that predict transformations - MAELLE and MEGAN - perform significantly better than the *de-novo* generative models like Molecular Transformers and Graph2SMILES. This is attributed to the former not having to reconstruct the products from scratch Xuan-Vu et al. (2025). Indeed, plotting the top-1 accuracy against the molecular weight reveals Molecular Transformers performance decays rapidly as the target becomes more complex, while Graph2SMILES shows a less severe drop. The better accuracy of Graph2SMILES compared with Molecular Transformers could be attributed to its graph encoder, which supports the permutation invariance. On the other hand, MAELLE and MEGAN maintain a stable accuracy across the bins.

While MAELLE and MEGAN perform similarly on OOD Mass, MAELLE shows a significant advantage over MEGAN on the OOD Ester, being the most performant model in this domain. This highlights the benefit of CTMC framework: Even though the model does not see the exact reaction type in the training data, it most likely has seen the intermediate graph states of reactions similar to esterification, such as amide coupling.

Two examples are shown in Figure 6b and 6c for the "OOD Mass" and "OOD Ester", respectively. The first case features a complex macrocycle structure but a very simple amide coupling reaction. Both MAELLE and MEGAN successfully recover the ground truth, while Graph2SMILES and Molecular Transformers struggle to reconstruct the molecular structure in SMILES format. The second case involves an esterification reaction with an anhydride. Out of the four models, only MAELLE correctly predicts the ester as the product

L BASELINE REPRODUCIBILITY

The authors of MEGAN Sacha et al. (2021) and NERF Bi et al. (2021) do not release the model weights along with the code. We report the details of our reproduction of these models in this section.

L.1 MEGAN

MEGAN is trained using the default hyperparameters as in `https://github.com/molecule-one/megan`.

L.2 NERF

Originally, NERF uses a feature called *segment embedding* which encodes whether molecules are reactants or reagents Bi et al. (2021). This feature is used both during training and inference. To ensure a fair comparison with the other models, we trained NERF without using the feature *segment embedding*. Moreover, due to the old version of the environment compared to modern GPUs, the learning rate needed to be decreased during training from $5 \cdot 10^{-4}$ to $2 \cdot 10^{-4}$ in order to stabilize the gradient.

As presented in Table 6, while the performance of the NERF model trained using the segment embedding feature is in agreement with the original results of Bi et al. (2021), removing the segment embedding feature drastically reduces the performance.

Source code of NERF can be found at: `https://github.com/20171130/NERF`.

Table 6: Top-k accuracy of the trained NERF model with and without segment embedding on USPTO-480K. The original NERF model is also displayed in the table

NERF Model	**Train**	**Test**		
	Top-1*	**Top-1**	**Top-3**	**Top-5**
With segment embedding	97.7%	91.4%	92.8%	92.9%
Without segment embedding	96.7%	68.2%	72.4%	73.3%
Original Bi et al. (2021)	—	90.7%	93.3%	93.7%

* *Train Top-1 accuracy is computed on 300 randomly sampled training reactions*

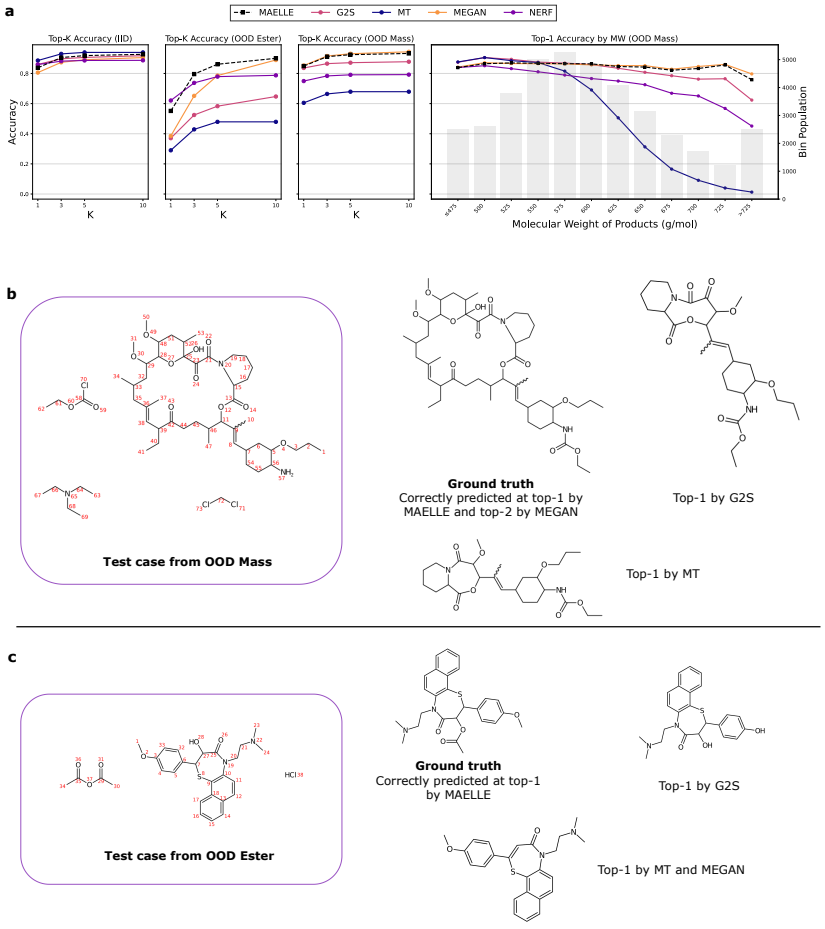


Figure 6: Comparative performance of MAELLE, Graph2SMILES, Molecular Transformers, and MEGAN on our OOD benchmark. **a.** From left to right, the first three plots show top-k accuracy on the in-distribution test set (IID), esterification-split test set (OOD Ester), and molecular-weight-split test set (OOD Mass). The last plot describes the top-1 accuracy divided into weight bins. **b.** An illustrative example from the OOD Mass split. **c.** An illustrative example from the OOD Ester split.