IMPLICIT LEARNING: REACTION OUTCOME PREDICTION FROM POSITIVE-ONLY DATA

Nick Wierich* **Niklas Dobberstein*** **Frank Glorius†**

Organisch-Chemisches-Institut
Universität Münster
48149 Münster, Germany
{nwierich, niklas.dobberstein, glorius}@uni-muenster.de

SUPPORTING INFORMATION

*These authors contributed equally to this work.
†Corresponding author.

1 Methods

1.1 Problem Formulation

We formulate forward reaction prediction as an iterative bond-pair ranking problem. Given a molecular graph $G = (V, E)$ with atoms V and bonds E, and optionally a history of previous reaction steps, TACIT predicts which pair of bonds $(b_i, b_j) \in E \times E$ will participate in the next elementary reaction step. Each elementary step corresponds to an operation where two bonds break and two new bonds form (i.e., a *bond swap*). Breaking the selected bonds creates four loose ends (u_1, u_2) and (v_1, v_2). These ends can then be recombined in two distinct ways (*bond swap types*), either as (u_1, v_1) and (u_2, v_2), or as (u_1, v_2) and (u_2, v_1). To account for steps where the valence of an atom bearing a lone pair (LP) changes, LPs are treated as nodes in the molecular graph. This enables a special type of bond swap effectively corresponding to breaking one bond, forming one, and shifting an LP (i.e., a *bond–lone-pair swap*). Both swap types are shown in Figure 1.

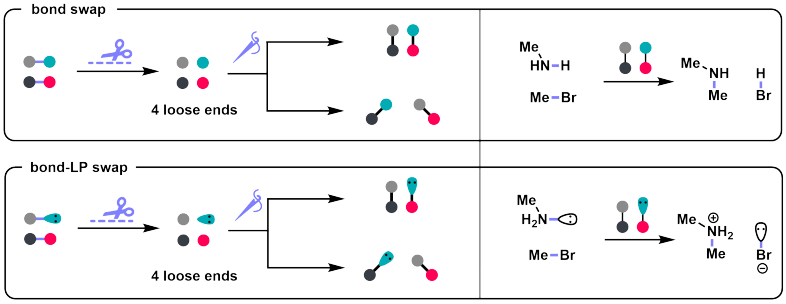


Figure 1: Formulation of the elementary step, shown for the nucleophilic substitution reaction between bromomethane and methylamine. Two bonds are selected to be broken, leaving four loose ends, which are then reconnected in pairs. When the C–Br and N–H bonds are selected, the elementary step leads to the final product dimethylamine directly, along with HBr. In contrast, when the C–Br and N–LP "bonds" are selected, the elementary step produces the protonated amine product as well as a bromide anion, aligning more closely with the actual reaction mechanism.

Simple reactions such as substitutions, amide couplings or cross couplings can be described using a single elementary step, while more complex cases such as Diels-Alder reactions require the application of multiple elementary steps. Therefore, the model is applied iteratively, with the product of step t becoming the input for step $t+1$, and all previous steps being provided as history context. In each step, the model performs a ranking between all combinatorially possible bond pairs (b_i, b_j) and a stop action S, thus deciding whether to apply another elementary step, or stop the reaction.

A direct consequence of this architecture is that TACIT implicitly learns to predict cases in which no reaction occurs. During the training process, the correct next action (either the next bond pair (b_i, b_j), or the stop action S) must be ranked above all other possible actions. The model thus sees every single unreactive bond as a negative example, without explicitly being trained on negative data.

1.2 The Blockspace Algorithm

We introduce the blockspace algorithm to go from an atom-mapped reaction SMILES to the elementary steps that the model is trained on. This is achieved by decomposing each reaction into a minimum-length sequence of elementary steps and, at the same time, filtering out reactions that are not chemically plausible. This is an important feature since available reaction datasets, even curated ones, contain a substantial amount of noise: Atom mappings can be wrong, products can be inconsistent with the reactants, and reactants can be missing entirely. Blockspace removes many of these cases by rejecting reactions that are not decomposable into elementary steps (Figure 2).

1.2.1 Bond-Electron Matrices and the Reaction Matrix

The core representation is the *Bond-Electron (BE) matrix*, originally introduced by Ugi and Dugundji [1, 2] and recently revisited by Joung et al. [3]. For a molecule with n atoms, the BE matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$ fully represents both the chemical bonds and the lone electron pairs of each atom. The off-diagonal entries encode bonds between atoms, with $B_{ij} = B_{ji} = 1, 2, 3$ corresponding to single, double, and triple bonds, respectively. The diagonal entries are

determined by the number of valence electrons present in each atom's LPs,

$$B_{ii} = v_i - c_i - \sum_{j \neq i} B_{ij}, \tag{1}$$

where v_i is the element's number of valence electrons and c_i is the formal charge of atom i. Since the sum over all entries of $\mathbf{B}$ gives the total number of valence electrons, electron conservation is encoded directly in the representation.

For each reaction, the substrate and product molecules are parsed into matrix pairs $(\mathbf{B}_0, \mathbf{A}_0)$ and $(\mathbf{B}_P, \mathbf{A}_P)$, where $\mathbf{A}$ stores atom-map identifiers and atomic numbers for each of the atoms in $\mathbf{B}$. The atom mapping tells us which substrate atom corresponds to which product atom. All unmapped atoms obtain a mapping number of 0, such that any two unmapped atoms of the same element are treated as interchangeable. This is important because the atom order in the substrate and product SMILES does not have to be the same, and because not every substrate atom has to appear in the product—in fact, side products are not included in most available datasets. Atoms that are present in the reactants but are absent from the product are treated as leaving groups.

We therefore apply a permutation π to $\mathbf{B}_P$ that aligns its atom order with that of $\mathbf{B}_0$. Substrate atoms with no product counterpart are thereby aligned to empty rows and columns. The reaction matrix $\mathbf{R}$ is then:

$$\mathbf{R} = \mathbf{B}_P^{\pi} - \mathbf{B}_0. \tag{2}$$

Positive off-diagonal entries in $\mathbf{R}$ are bonds that form, negative off-diagonal entries are bonds that break, and diagonal changes are lone-pair shifts. When multiple unmapped heavy atoms of the same element appear on both sides, π is not unique. We resolve this ambiguity by selecting the pairing that minimizes the number of non-zero off-diagonal entries in $\mathbf{R}$, thus minimizing the number of elementary steps required to describe the reaction. A reaction is rejected if a mapped atom changes its atomic number, or if $\sum_{ij} R_{ij} \neq 0$, since either case indicates an inconsistent atom mapping or electron balance (Figure 2).

Aromatic systems require one additional cleanup step, as different kekulizations of the same aromatic ring can introduce apparent alternating single- and double-bond changes, even though the aromatic system itself is unchanged. We detect such artifacts by recomputing $\mathbf{R}$ on the non-kekulized representation, where aromatic bonds carry a uniform order of 1.5, and then align the kekulization within each ring between $\mathbf{B}_P^{\pi}$ and $\mathbf{B}_0$ before computing $\mathbf{R}$.

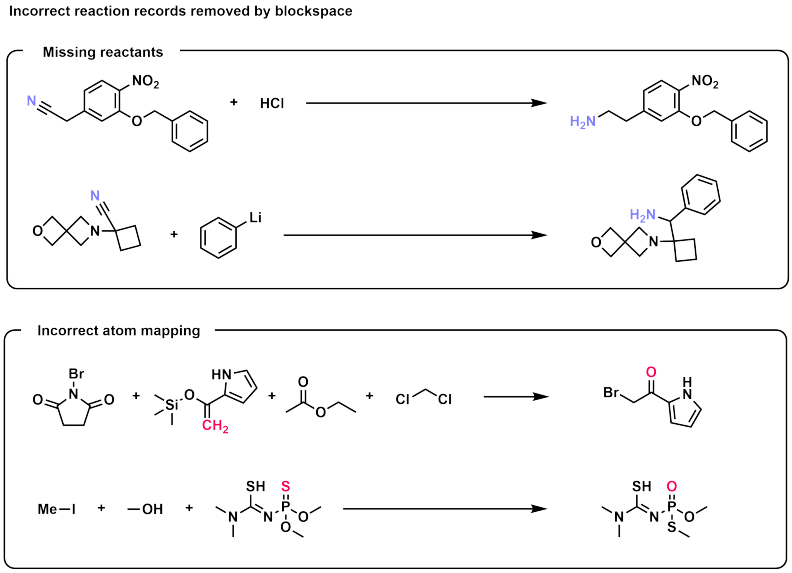


Figure 2: Examples of incorrect or incomplete reaction records from USPTO-MIT [4, 5] that were removed by the blockspace algorithm. Same-colored atoms within a reaction indicate an identical atom mapping number. The first two rows show cases where reactants are missing, rendering the BE matrix problem impossible to solve. The last two examples have incorrect atom mapping, with two atoms of different elements being assigned the same map number.

1.2.2 Decomposition as Shortest-Path Search

Finding an elementary-step decomposition of the reaction matrix can be formulated as a shortest-path search problem. Each state contains the current intermediate BE matrix $\mathbf{B}^{(t)}$ and the residual reaction matrix $\mathbf{R}^{(t)}$, where $\mathbf{B}^{(0)} = \mathbf{B}_0$

and $\mathbf{R}^{(0)} = \mathbf{R}$. Applying an elementary step $\mathbf{S}^{(t)}$ adds this step to the current molecule and subtracts it from the remaining unexplained reaction matrix,

$$\mathbf{B}^{(t+1)} = \mathbf{B}^{(t)} + \mathbf{S}^{(t)}, \qquad \mathbf{R}^{(t+1)} = \mathbf{R}^{(t)} - \mathbf{S}^{(t)}. \tag{3}$$

The goal is to reach a state with $\mathbf{R}^{(t)} = \mathbf{0}$. At the same time, every intermediate BE matrix must remain element-wise non-negative. This prevents the algorithm from breaking a bond or removing a lone pair that is not present in the current intermediate.

The reaction matrix formulation allows the decomposition to be treated as a local problem: Once the product is aligned to the substrate, the non-zero entries of $\mathbf{R}$ identify the reaction center directly, since only atoms in changed rows or columns can take part in the elementary steps. Therefore, instead of enumerating all possible elementary steps over the full molecule, we only need to consider steps affecting the remaining non-zero entries of the residual matrix $\mathbf{R}^{(t)}$ in each step. To construct the action set at a given state, the non-zero entries of $\mathbf{R}^{(t)}$ are split into unit changes. Bond changes are split into units of ± 1, corresponding to a bond order changing by one. Lone-pair changes are split into units of ± 2, corresponding to one electron pair being gained or lost. Each unit change contributes ± 2 to the matrix sum: an off-diagonal unit change of ± 1 appears both in R_{ij} and R_{ji}, while a diagonal unit change of ± 2 appears in R_{ii} once. An elementary step must therefore satisfy

$$\sum_{c \in C^-} e(c) = -4, \qquad \sum_{c \in C^+} e(c) = +4, \tag{4}$$

which enforces a four-electron rearrangement at every step. Here, C^- denotes the unit changes that remove electrons from the current BE matrix, and C^+ denotes the unit changes that add them elsewhere.

However, not every such combination is chemically valid. We only keep candidates that match one of the two elementary step templates as described in Section 1.1. In a bond swap, two bonds break and two bonds form, and the formed bonds must reconnect the same four atoms in one of the two possible pairings. In a bond–lone-pair swap, one bond breaks, one bond forms, and one atom loses a lone electron pair while another atom gains one. Finally, all atoms changed by the candidate step must be part of the same local rearrangement. To check this, we build a graph whose nodes are the atoms affected by the candidate and whose edges are the bonds that are broken or formed in that candidate. The candidate step is then kept only if this graph is connected. Thus, a bond swap such as breaking (a, b) and (c, d) while forming (a, c) and (b, d) is valid because all four atoms are linked by the changed bonds. In contrast, two unrelated bond changes in different parts of the molecule would give two disconnected components and are therefore rejected.

We run breadth-first search (BFS) over the resulting state graph. For each state, the valid elementary steps are regenerated from the current residual matrix, applied to the current BE matrix, and discarded if the resulting intermediate contains a negative entry. Since all elementary steps have the same cost, the first time the search reaches $\mathbf{R}^{(t)} = \mathbf{0}$ gives the minimum number of steps T^*. We hash visited pairs $(\mathbf{B}^{(t)}, \mathbf{R}^{(t)})$ to avoid exploring the same partial mechanism multiple times through different step orderings.

Once T^* is known, we run a depth-limited depth-first search (DFS) to enumerate all decompositions of exactly this length. This second search uses the same step generator and the same non-negativity constraint, but explores all valid paths of length T^*. This is important because the BFS only needs to find one shortest path, while many reactions possess several equally short decompositions. We keep all of them as valid training trajectories.

1.2.3 Intermediate Bonds and Filtering

Some reactions cannot be decomposed directly using only the net changes in $\mathbf{R}$. A typical example is a reaction with three bonds forming and three bonds breaking, such as a Diels-Alder reaction, resulting in a $+6/-6$ net change. This cannot be split into four-electron steps without introducing a transient bond. In these cases, blockspace proposes intermediate bonds between atoms that participate in $\mathbf{R}$, and do not have a net bond change in $\mathbf{R}$. Such a bond can then be formed in one step and broken again later.

Overall, our blockspace algorithm thus decomposes chemical reactions into shortest sequences of elementary steps. While this concept is analogous to reaction mechanisms formulated in organic chemistry, it should be emphasized that both blockspace and our model are fully template-free and do not use any type of information not contained in the reaction SMILES. While the shortest-path solution does align with the actual chemical mechanism for many simple reactions, this is not necessarily the case. The same applies to the side products that are formed as a result of the elementary steps (see Figure 3).

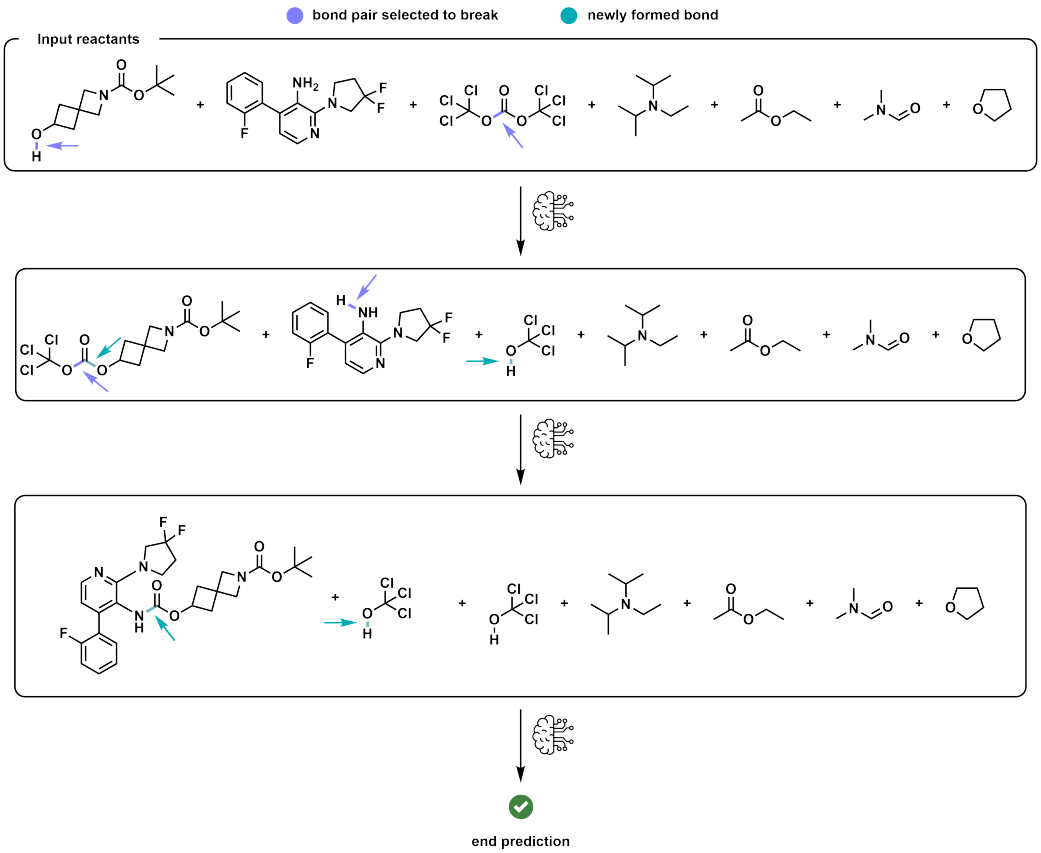


Figure 3: Example of TACIT predicting a multi-step reaction [6]. In this case, the prediction proceeds via a chemically reasonable mechanism, despite the model not being explicitly designed or trained to predict mechanisms.

1.3 Model Architecture

TACIT receives a molecular graph $G = (V, E)$ and the current step index t. An overview is shown in Figure 4.

Each atom i is described by its atomic number z_i, formal charge c_i, hydrogen count $n_{H,i}$, aromaticity flag $a_i \in \{0, 1\}$, and symmetry class s_i (calculated via RDKit's [7] `CanonicalRankAtoms()` function). The symmetry class is essential because molecules often contain symmetrically equivalent atoms, yet the blockspace algorithm selects only one of them. Without this feature, a graph neural network would produce identical representations for symmetrically equivalent atoms and could not distinguish between them. These per-atom features, together with the graph-level step index t, are each mapped to d-dimensional vectors via learned embedding tables and concatenated:

$$\mathbf{h}_i^{(0)} = \mathrm{Proj}([\mathbf{e}_{\text{atom}}(z_i) \,\|\, \mathbf{e}_{\text{charge}}(c_i) \,\|\, \mathbf{e}_{\text{H}}(n_{H,i}) \,\|\, \mathbf{e}_{\text{arom}}(a_i) \,\|\, \mathbf{e}_{\text{sym}}(s_i) \,\|\, \mathbf{e}_{\text{step}}(t)]) \tag{5}$$

where $\|$ denotes concatenation along the feature dimension and $\mathrm{Proj}(\cdot)$ is a linear projection from $6d$ to d, followed by LayerNorm [8] and GELU [9].

These initial representations are then refined by 4 layers of GATv2 [10] message passing with residual connections. Each bond (i, j) carries a learned bond-type embedding $\mathbf{e}_{ij} \in \mathbb{R}^d$ that encodes its bond order (single, double, triple, aromatic, or dative). We extend GATv2 with edge conditioning: the bond-type embedding is projected and added to the node pair features before the nonlinearity, so the attention coefficient for neighbour j at layer ℓ becomes

$$\alpha_{ij}^{(\ell)} = \frac{\exp\Big(\mathbf{a}^\top \,\mathrm{LeakyReLU}\Big(\mathbf{W}_l\,\mathbf{h}_i^{(\ell)} + \mathbf{W}_r\,\mathbf{h}_j^{(\ell)} + \mathbf{W}_e\,\mathbf{e}_{ij}\Big)\Big)}{\sum_{k\in\mathcal{N}(i)} \exp\Big(\mathbf{a}^\top \,\mathrm{LeakyReLU}\Big(\mathbf{W}_l\,\mathbf{h}_i^{(\ell)} + \mathbf{W}_r\,\mathbf{h}_k^{(\ell)} + \mathbf{W}_e\,\mathbf{e}_{ik}\Big)\Big)} \tag{6}$$

where $\mathbf{W}_l, \mathbf{W}_r, \mathbf{W}_e \in \mathbb{R}^{d'\times d}$ are learned weight matrices for the left node, right node, and edge respectively, $\mathbf{a} \in \mathbb{R}^{d'}$ is a learned attention vector, $\mathcal{N}(i)$ denotes the set of bonded neighbours of atom i, and ℓ is the GATv2 layer index.

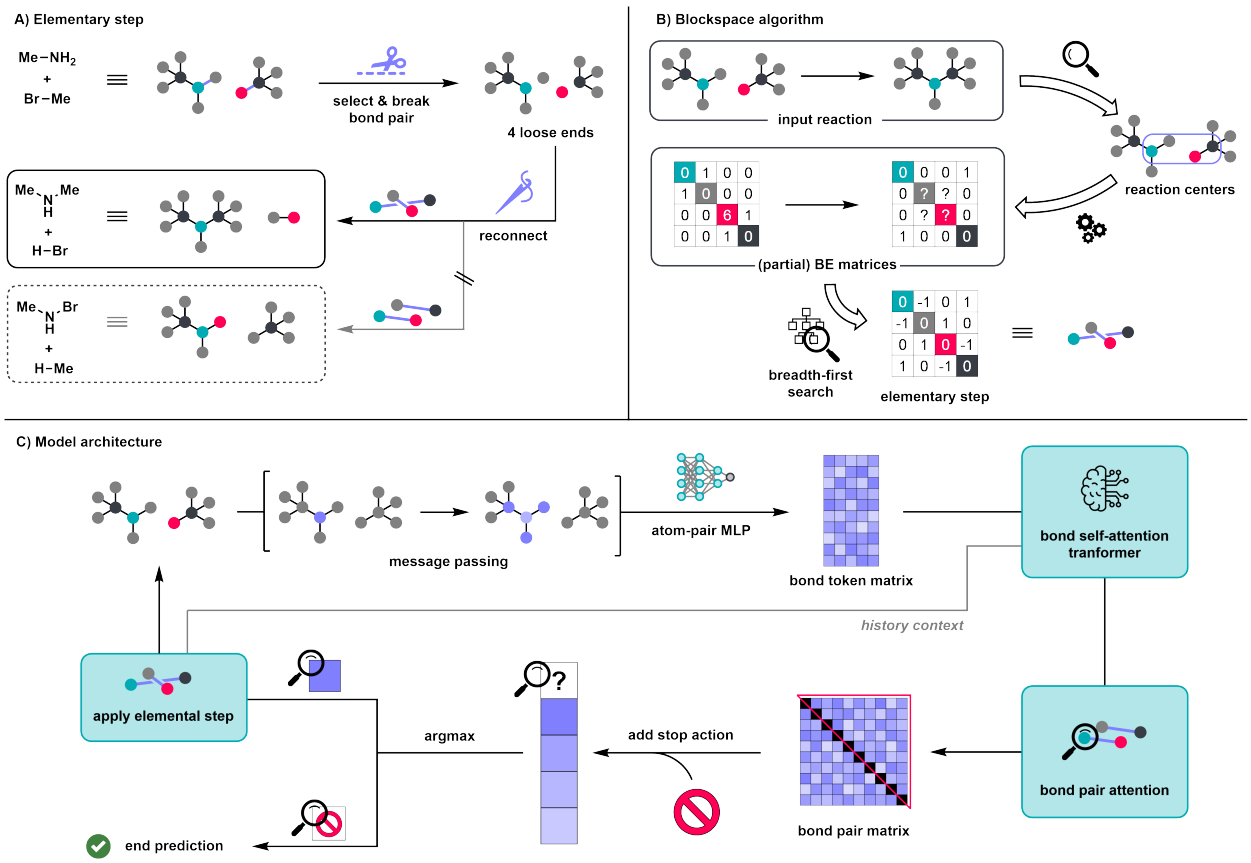


Figure 4: Overview of the TACIT workflow and architecture. A) Definition of the elementary step. Two bonds are selected and broken, resulting in four loose ends. This leaves two distinct possibilities for pairwise recombination (i.e., swap types). B) Blockspace algorithm for decomposing chemical reactions into sequences of elementary steps. First, the reaction centers are identified using the atom mapping provided in the dataset. The reactants and products are then expressed as bond electron (BE) matrices, where the product BE matrix may be a partial matrix if not all reactant atoms are given in the product. A breadth-first search (BFS) algorithm is then used to find the shortest sequence of elementary steps (expressed as BE change matrices) transferring the reactant BE matrix to the (partial) product BE matrix. C) Model architecture of TACIT. Message-passing is performed on the input molecular graph using GATv2 layers to yield refined per-atom representations, which are then converted to per-bond representations using an atom-pair MLP layer. A bond transformer is then used to perform bond self-attention to yield bond pair representations, which are subsequently fed into a bond pair attention block to give final scores. A separate stop action S is added, and an argmax is used on the result to select the next predicted action. In case this is the stop action, the prediction is terminated. Otherwise, the selected elementary step is executed, and the new molecular graph is fed back to the model with the accumulated step history.

Each atom then aggregates its neighbours' features weighted by these attention coefficients, and a residual connection with LayerNorm (LN) produces the updated representation:

$$\mathbf{h}_i^{(\ell+1)} = \mathrm{LN}\left(\mathbf{h}_i^{(\ell)} + \sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(\ell)}\, \mathbf{W}_r\, \mathbf{h}_j^{(\ell)}\right) \tag{7}$$

This allows each atom to selectively attend to its chemical environment, weighting more relevant neighbours higher. Unlike the original GAT (Graph ATtention Network) [11], which applies the attention vector before the nonlinearity, GATv2 applies it after, making the ranking of neighbours input-dependent and strictly more expressive [10].

After the GATv2 layers, we construct a bond embedding for each bond (i, j) by concatenating the two atom representations and their element-wise product, then projecting from $3d$ to d dimensions:

$$\mathbf{z}_{ij} = \mathrm{Proj}([\mathbf{h}_i \,\|\, \mathbf{h}_j \,\|\, \mathbf{h}_i \odot \mathbf{h}_j]) \tag{8}$$

The Hadamard product $\mathbf{h}_i \odot \mathbf{h}_j$ captures pairwise interaction while the individual embeddings preserve atom identity. Crucially, the bond embedding is asymmetric ($\mathbf{z}_{ij} \neq \mathbf{z}_{ji}$) because the concatenation order of $\mathbf{h}_i$ and $\mathbf{h}_j$ is fixed. This asymmetry is necessary for the swap-type prediction head, which must distinguish the two possible reconnection patterns given atoms (u_1, u_2) and (v_1, v_2) from the two breaking bonds. The atom ordering within each bond embedding is determined by the dataset construction (Section 1.4).

Since the GATv2 layers only capture local topological effects within each atom's neighbourhood, they cannot reason about interactions between distant bonds in the molecule. We therefore pass the bond embeddings through a transformer encoder that computes self-attention over all bonds simultaneously; we refer to this as bond attention. Each bond transformer block comprises two sublayers, namely multi-head self-attention across all bond embeddings and history information, and a feed-forward network.

To increase the model's capacity without simply widening all layers, we adopted two techniques from the natural language processing (NLP) community that add parameters efficiently. The first is Mixture-of-Experts (MoE) [12–14], which replaces the single dense feed-forward sublayer with K independent expert networks. For each bond embedding, a learned router projects the embedding to K gate logits, applies a softmax to obtain routing probabilities, and selects the top-N experts. The embedding is dispatched only to the selected experts, and their outputs are weighted by the normalized gate values and combined:

$$\mathrm{MoE}(\mathbf{x}) = \sum_{e \in \text{top-}N} \bar{g}_e \cdot \mathrm{Expert}_e(\mathbf{x}), \quad g_e = \frac{\exp(\mathbf{w}_e^\top \mathbf{x})}{\sum_{e'} \exp(\mathbf{w}_{e'}^\top \mathbf{x})}, \quad \bar{g}_e = \frac{g_e}{\sum_{e' \in \text{top-}N} g_{e'}} \tag{9}$$

Each expert follows the GEGLU architecture [15] with a learned multiplicative bias $\mathbf{m}^{(e)} \in \mathbb{R}^{d_{\text{ff}}}$ applied after the gating, as proposed by Zoph et al. [14]:

$$\mathrm{Expert}_e(\mathbf{x}) = \mathbf{W}_2^{(e)} \Big(\mathrm{GELU}(\mathbf{W}_{1g}^{(e)} \mathbf{x}) \odot \mathbf{W}_{1v}^{(e)} \mathbf{x} \odot \mathbf{m}^{(e)} \Big) \tag{10}$$

Here $\mathbf{x} \in \mathbb{R}^d$ is the input bond embedding, K is the total number of expert networks, and N is how many of them are selected per embedding. The router computes a raw gate value g_e for each expert e using a learned weight vector $\mathbf{w}_e \in \mathbb{R}^d$, and $\bar{g}_e$ is the renormalized weight after top-N selection. In Equation 10, $\mathbf{W}_{1g}^{(e)}, \mathbf{W}_{1v}^{(e)} \in \mathbb{R}^{d_{\text{ff}} \times d}$ are the gate and value projection weight matrices of expert e, $\mathbf{W}_2^{(e)} \in \mathbb{R}^{d \times d_{\text{ff}}}$ is its output projection weight matrix, $\mathbf{m}^{(e)}$ is a per-expert multiplicative bias initialized to ones, and $\odot$ denotes element-wise multiplication. In our configuration we use K=8 experts with N=2 active per bond, which increases the feed-forward parameter count eightfold while only roughly doubling the per-bond compute.

The second technique is attention residual connections [16], which replace the standard additive residual $\mathbf{h}_\ell = \mathbf{h}_{\ell-1} + f(\mathbf{h}_{\ell-1})$ with a learned, input-dependent aggregation over previous representations. Sublayers are grouped into blocks of size S. Within each block, sublayer outputs are accumulated into a block representation. At block boundaries, the model attends over all completed block representations $\mathbf{c}_k$ using a learned query $\mathbf{q}_\ell$:

$$\mathbf{h}_\ell = \sum_k \alpha_k \, \mathbf{c}_k, \quad \alpha_k = \frac{\exp(\mathbf{q}_\ell \cdot \mathrm{RMSNorm}(\mathbf{c}_k))}{\sum_{k'} \exp(\mathbf{q}_\ell \cdot \mathrm{RMSNorm}(\mathbf{c}_{k'}))} \tag{11}$$

where $\mathbf{q}_\ell \in \mathbb{R}^d$ is a learned query vector for sublayer ℓ, $\mathbf{c}_k$ is the accumulated representation of the k-th completed block, α_k is the attention weight assigned to block k, and RMSNorm is root-mean-square normalization [17] applied before the dot product. The block size S controls the granularity of aggregation. In the extreme case S=1 (full attention), every sublayer attends over all previous sublayer outputs, which is maximally expressive but also slower to train because the aggregation cost grows with depth. In our experiments we found S=6 to give the best performance, outperforming both full attention and a smaller block size of S=4. The block variant is also faster because the number of representations to attend over grows more slowly, since within-block sublayers simply use standard residual connections rather than attention.

For multi-step reactions at step t, history is incorporated directly into the bond transformer through its self-attention. Bond embeddings from all intermediates along the trajectory, namely steps $0, \ldots, t$, are concatenated into a single sequence and passed jointly through the transformer encoder, where each intermediate is encoded by the same GATv2 and bond projection layers with shared weights. A causal attention mask restricts each bond embedding to attend to embeddings from its own step and from all preceding steps, while embeddings from future steps are masked out. Because the full trajectory is processed in a single forward pass, gradients flow through both the current and the historical bond embeddings, so the model learns representations that serve simultaneously for predicting the current step and as context for subsequent steps.

The bond transformer output feeds into four prediction heads. A binary head predicts per-bond reactivity $p_{\text{react}}(b_{ij}) = \sigma(\text{MLP}(\mathbf{z}_{ij}))$. An attention-based pair scorer computes compatibility between two bonds via symmetrized multi-head dot-product attention:

$$s(b_{ij}, b_{kl}) = \frac{1}{2}\left(\frac{1}{H}\sum_{h}\frac{(\mathbf{W}_Q^{(h)}\mathbf{z}_{ij})^\top \mathbf{W}_K^{(h)}\mathbf{z}_{kl}}{\sqrt{d_h}} + \frac{(\mathbf{W}_Q^{(h)}\mathbf{z}_{kl})^\top \mathbf{W}_K^{(h)}\mathbf{z}_{ij}}{\sqrt{d_h}}\right) \tag{12}$$

where $\mathbf{W}_Q^{(h)}, \mathbf{W}_K^{(h)} \in \mathbb{R}^{d_h \times d}$ are the query and key projection weight matrices for attention head h, H is the total number of heads, and $d_h = \frac{d}{H}$ is the per-head dimension. The $\frac{1}{2}$ factor and the two summed terms symmetrize the score so that $s(b_{ij}, b_{kl}) = s(b_{kl}, b_{ij})$, which is necessary because the order in which the two breaking bonds are presented should not affect the score. We use fused scoring where the final pair score is $s_{\text{fused}} = s(b_{ij}, b_{kl}) + p_{\text{react}}(b_{ij}) + p_{\text{react}}(b_{kl})$, so the pair scorer can build on the binary head's per-bond reactivity signals rather than learning reactivity from scratch.

An attention-pooling termination head aggregates all bond embeddings conditioned on the current step index t and produces a scalar logit for the special termination action. During both training and inference, the action set is the union of all real bond-pair actions and this single termination action. Finally, given the predicted pair of breaking bonds with atoms (u_1, u_2) and (v_1, v_2), an atom-level swap-type head extracts the four atom embeddings and classifies the reconnection pattern into swap-0 (parallel: $u_1{-}v_1$, $u_2{-}v_2$), or swap-1 (crossed: $u_1{-}v_2$, $u_2{-}v_1$).

1.4 Datasets

All models in this work are trained on the USPTO-MIT dataset [4, 5], using its published train, validation and test splits. Before training, the train and validation sets are passed through the blockspace algorithm (Section 1.2), which removes all reactions that cannot be decomposed into elementary steps (Table 1). The test set is not filtered.

Table 1: Size of the USPTO-MIT splits before and after blockspace filtering.

Split	Original	After blockspace	Removed
Train	409,035	334,510	74,525 (18.2%)
Validation	30,000	24,522	5,478 (18.3%)
Test	40,000	40,000	not filtered

For preprocessing, we first move all reported reaction components, including reagents, catalysts, and solvents to the reactant side of the reaction SMILES. Our model therefore receives the full, mixed set of molecules present in the reaction and must infer which components participate in the product formation and which serve as reagents, catalysts, or solvents. We then canonicalize all reaction SMILES.

For the HTE evaluation, we use the amide coupling datasets CHEMLEX [18] and AICHEMECO [19]. Each well is converted into a mixed reaction SMILES whose reactant side contains the acid, the amine, the coupling reagent, the base, any additive and the solvent, and whose product side contains the reported amide. For CHEMLEX, these components are taken from the acid, amine, reagent and solvent columns of the published data. For AICHEMECO, we use the reaction SMILES provided with the dataset, which already contain all components of the well. All SMILES are canonicalized, and no well is removed from either dataset. A well counts as successful when its conversion (CHEMLEX) or yield (AICHEMECO) reaches at least 20% [18, 20], and as failed otherwise.

1.5 Training

We formulate the training objective as a ranking problem over all bond pairs in the reactants, plus a special stop action. We represent the reactants as an undirected molecular graph $G = (V, E)$, where V denotes the atoms and E denotes the set of chemical bonds. The graph G is not required to be connected, so E contains all bonds across all reactant components.

The model ranks all unordered pairs of distinct bonds, together with a special stop action. The action space is therefore defined as

$$\mathcal{A}(G) = E_{\text{pair}} \cup \{a_{\text{stop}}\}, \tag{13}$$

where

$$E_{\text{pair}} = \big\{\{e_i, e_j\} \mid e_i, e_j \in E,\ e_i \neq e_j\big\} \tag{14}$$

denotes the set of all unordered bond pairs in the reactants.

The main loss the model optimizes is an energy-based contrastive loss $\mathcal{L}_{\text{EBM}}$ [21, 22]. For all reactants we compute a bond pair score $s(b_{ij}, b_{kl})$ from Eq. 12 in addition to the stop action score.

Here the model learns to rank the reactive bond pair identified by the blockspace algorithm in first place, while all other actions in $\mathcal{A}(G)$ should be pushed down. However, the model also needs to learn predicting the stop action to mark that no (further) reaction occurs. Therefore, we also include the final product in the training, for which the model then learns to predict the stop action. Mathematically, the loss can be written as:

$$\mathcal{L}_{\text{EBM}} = -s_{a_c} + \log \sum_{a \in \mathcal{A}(G)} \exp(s_a), \tag{15}$$

which pushes the correct action a_c above all other actions. Since the EBM logits are unbounded, we add a regularization

$$\mathcal{L}_{\text{anc}} = \frac{1}{|\mathcal{A}(G)|} \sum_{a \in \mathcal{A}(G)} s_a^2 \tag{16}$$

to keep the logits centred near zero and prevent exploding values.

We found that with these two losses, the model already performs well at ordering the bond pairs correctly, but is weak at identifying cases in which no reaction is expected to occur. This is not surprising, since the model only ever learns to rank the stop action above all other actions in the final product state. To improve the model's ability to rank the stop action highest, we include an additional auxiliary stop loss. This loss is applied at every step whose target is a reactive bond pair and ranks the stop action above all other bond pairs, while the main loss keeps the reactive bond pair above the stop action. Overall, the model thus learns to rank $s_{\text{reactive}} > s_{\text{stop}} > s_{\text{unreactive}}$, mimicking the implicit learning mechanism used by humans: when no sufficiently reactive sites are present, the transformation is predicted to be unproductive. We implemented the auxiliary loss as follows:

$$\mathcal{L}_{\text{stop}} = -s_{a_{\text{stop}}} + \log \sum_{a \in \mathcal{A}(G) \setminus \{a_c\}} \exp(s_a), \tag{17}$$

where a_c is the reactive bond pair and $\mathcal{A}(G) \setminus \{a_c\}$ contains the stop action and all other bond pairs. Since a_c is excluded, this term does not push the stop action above the reactive bond pair. It does, however, treat every other bond pair as unreactive, including pairs that could also react. We therefore keep this term weak, so that it shapes the ordering of wrong actions without overpowering the main reactive-bond signal.

We further add a bond swap-type loss $\mathcal{L}_{\text{swap}}$, which is used to learn the swap type for a given bond pair, as described in Section 1.2. This is learned with a cross-entropy loss over the two possible swap options. Since the model uses an MoE architecture in the Bond Transformer, we also include an additional MoE regularization loss $\mathcal{L}_{\text{MoE}}$ [13, 14]. This loss combines a load-balancing term that encourages uniform expert utilization with a router z-loss that penalizes large gating logits.

The total loss is therefore

$$\mathcal{L} = \mathcal{L}_{\text{EBM}} + w_{\text{anc}}\,\mathcal{L}_{\text{anc}} + w_{\text{swap}}\,\mathcal{L}_{\text{swap}} + w_{\text{stop}}\,\mathcal{L}_{\text{stop}} + w_{\text{moe}}\,\mathcal{L}_{\text{MoE}}. \tag{18}$$

We optimize the model with AdamW [23] using a cosine learning-rate schedule after a short warmup. Training is performed on 4 Nvidia A40 GPUs with bfloat16 mixed precision using the PyTorch framework [24]. Multi-step reactions are oversampled to counterbalance the dominance of single blockspace-step reactions in the training data. Product states are included as training examples for which the model should predict the stop action. All evaluations use the final checkpoint after 50 epochs, without any checkpoint selection.

All models trained for this work are shown in Table 2, and their shared hyperparameters are listed in Table 3.

Table 2: Model variants trained for this work.

Model	**Training data**	**Stop-rank aux.**
TACIT	USPTO-MIT	$w_{\text{stop}} = 0.1$
$\text{TACIT}_{\text{no-stop}}$	USPTO-MIT	none

Table 3: Hyperparameters used for our model.

Architecture	
Embedding dimension d	384
Attention heads	8
GAT layers	4
Bond Transformer layers	4
MoE experts / top-N	8 / 2
Number of parameters	29.7 million
Loss weights	
Ancillary loss weight w_{anc}	0.05
Bond-swap loss weight w_{swap}	0.8
Stop-action loss weight w_{stop}	0.1
MoE loss weight w_{moe}	0.01
Label smoothing for bond-swap loss	0.1
Training	
Optimizer	AdamW (β_1=0.9, β_2=0.999)
Peak learning rate	2×10^{-4}
Minimum learning rate	5×10^{-5}
Learning-rate schedule	100 batch linear warmup, then cosine decay
Weight decay	0.1
Gradient clipping	1.0
Batch size / accumulation	16 / 1 per GPU
Effective batch size	64 across 4 GPUs
Precision	bfloat16
Epochs	50
Multi-step oversampling weight	3

2 Evaluation Details

2.1 Retraining of the Baseline Models

MEGAN, the Molecular Transformer, LocalTransform and ReactionT5 are retrained on the blockspace-filtered USPTO-MIT train and validation sets (Table 1), which are identical for all models. Each model is trained with the code and training settings published by its authors (Table 4). Changes to the published code are limited to compatibility with current library versions and to multi-GPU training, and do not alter the model, the loss or the optimization. For LocalTransform, the reaction templates are extracted again from the filtered train set using the authors' preprocessing, yielding 2,131 real and 1,252 virtual templates. ReactionT5 is trained on the authors' input format, which separates reactants and reagents. For evaluation, all species are given as reactants, in line with the mixed setting used for all models.

Table 4: Training settings of the retrained baseline models. All settings follow the authors' published code.

Model	Training settings	Selected model
MEGAN [25]	USPTO-MIT mixed configuration, learning rate reduced on plateau	Best validation accuracy (epoch 118)
Molecular Transformer [26]	Mixed setting, augmented data, 500,000 steps	Average of the last 20 checkpoints
LocalTransform [27]	At most 20 epochs, early stopping with patience of 3 epochs	Best validation loss (epoch 14)
ReactionT5 [28]	Fine-tuned from the ORD-pretrained model, at most 50 epochs	Best validation token accuracy (epoch 6)

2.2 Model Scores and Metrics

For TACIT and MEGAN, the score is $1 - P(\text{stop})$, where $P(\text{stop})$ is the softmax probability of the stop action at the first step. For the Molecular Transformer, the score is the log-probability of the greedy output sequence. For ReactionT5, it is the log-probability of the greedy output sequence divided by its length. For LocalTransform, it is the score of the top-1 template. Wells for which a model returns no score are ranked lowest. ROC-AUC and PR-AUC are computed with scikit-learn [29], where the PR-AUC is the average precision $\sum_n (R_n - R_{n-1}) P_n$ over all ranking thresholds n, with precision P_n and recall R_n.

FlowER is evaluated with its published checkpoint [3], using the authors' search with a beam size of 1 and at most 15 steps. At each step, FlowER samples several next states and ranks them by how often they are sampled. We draw 64 samples per step on the HTE datasets, which is the default setting of the published FlowER code [3]. The prediction ends when the most frequent next state is the unchanged current state. As the HTE score, we use the fraction of the 64 first-step samples that change the input, which corresponds to $1 - P(\text{stop})$ at the first step. Seven CHEMLEX wells cannot be encoded by FlowER and are ranked lowest. On USPTO-MIT, in contrast, we draw 8 samples per step to keep the run time feasible for 40,000 reactions.

References

[1] I. Ugi, J. Bauer, J. Brandt, J. Friedrich, J. Gasteiger, C. Jochum, W. Schubert, *Angew. Chem. Int. Ed. Engl.* **1979**, *18*, 111–123.

[2] J. Dugundji, I. Ugi in *Computers in Chemistry*, Springer-Verlag, **1973**, pp. 19–64.

[3] J. F. Joung, M. H. Fong, N. Casetti, J. P. Liles, N. S. Dassanayake, C. W. Coley, *Nature* **2025**, *645*, 115–123.

[4] D. M. Lowe, PhD thesis, University of Cambridge, **2012**.

[5] W. Jin, C. W. Coley, R. Barzilay, T. Jaakkola, *arXiv* **2017**, DOI 10.48550/ARXIV.1709.04555.

[6] V. Lofstrand, J. Y. Kim, H. Clement, P. Charifson, S. Johnstone, J. Sabbatani, J. F. Scholtes, W. Zhang, S. Sun, M. Clark, S. Wesolowski, R. Munuganti, R. Rajamani, K. Burford, Pyridinylacetamide derivatives as sodium channel activators, World Intellectual Property Organization Patent WO2023049367A1, **2023**.

[7] G. Landrum, P. Tosco, B. Kelley, R. Rodriguez, D. Cosgrove, et al., RDKit: Open-Source Cheminformatics, version 2025.09.3, **2025**, https://www.rdkit.org, DOI 10.5281/zenodo.17746401.

[8] J. L. Ba, J. R. Kiros, G. E. Hinton, *arXiv* **2016**, DOI 10.48550/ARXIV.1607.06450.

[9] D. Hendrycks, K. Gimpel, *arXiv* **2016**, DOI 10.48550/ARXIV.1606.08415.

[10] S. Brody, U. Alon, E. Yahav, *arXiv* **2021**, DOI 10.48550/ARXIV.2105.14491.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, *arXiv* **2017**, DOI 10.48550/ARXIV.1710.10903.

[12] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, J. Dean, *arXiv* **2017**, DOI 10.48550/ARXIV.1701.06538.

[13] W. Fedus, B. Zoph, N. Shazeer, *arXiv* **2021**, DOI 10.48550/ARXIV.2101.03961.

[14] B. Zoph, I. Bello, S. Kumar, N. Du, Y. Huang, J. Dean, N. Shazeer, W. Fedus, *arXiv* **2022**, DOI 10.48550/ARXIV.2202.08906.

[15] N. Shazeer, *arXiv* **2020**, DOI 10.48550/ARXIV.2002.05202.

[16] Kimi Team, G. Chen, Y. Zhang, J. Su, W. Xu, S. Pan, Y. Wang, Y. Wang, G. Chen, B. Yin, Y. Chen, J. Yan, M. Wei, Y. Zhang, F. Meng, C. Hong, X. Xie, S. Liu, E. Lu, Y. Tai, Y. Chen, X. Men, H. Guo, Y. Charles, H. Lu, L. Sui, J. Zhu, Z. Zhou, W. He, W. Huang, X. Xu, Y. Wang, G. Lai, Y. Du, Y. Wu, Z. Yang, X. Zhou, *arXiv* **2026**, DOI 10.48550/ARXIV.2603.15031.

[17] B. Zhang, R. Sennrich, *arXiv* **2019**, DOI 10.48550/ARXIV.1910.07467.

[18] H. Zhong, Y. Liu, H. Sun, Y. Liu, R. Zhang, B. Li, Y. Yang, Y. Huang, F. Yang, F. S. Mak, K. Foo, S. Lin, T. Yu, P. Wang, X. Wang, *Nat. Commun.* **2025**, *16*, 4522.

[19] C. Zhang, Q. Lin, C. Yang, Y. Kong, Z. Yu, K. Liao, *Chem. Sci.* **2025**, *16*, 11809–11822.

[20] F. Boser, J. C. Spies, F. Glorius, *J. Am. Chem. Soc.* **2026**, *148*, 15066–15075.

[21] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. J. Huang in *Predicting Structured Data*, (Eds.: G. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar), MIT Press, **2006**.

[22] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, H. Li, *Int. Conf. Mach. Learn.* **2007**, 129–136.

[23] I. Loshchilov, F. Hutter, *arXiv* **2017**, DOI 10.48550/ARXIV.1711.05101.

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, *arXiv* **2019**, DOI 10.48550/ARXIV.1912.01703.

[25] M. Sacha, M. Błaż, P. Byrski, P. Dąbrowski-Tumański, M. Chromiński, R. Loska, P. Włodarczyk-Pruszyński, S. Jastrzębski, *J. Chem. Inf. Model.* **2021**, *61*, 3273–3284.

[26] P. Schwaller, T. Laino, T. Gaudin, P. Bolgar, C. A. Hunter, C. Bekas, A. A. Lee, *ACS Cent. Sci.* **2019**, *5*, 1572–1583.

[27] S. Chen, Y. Jung, *Nat. Mach. Intell.* **2022**, *4*, 772–780.

[28] T. Sagawa, R. Kojima, *J. Cheminf.* **2025**, *17*, 126.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay, *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.