REACTPG: RETHINKING REACTION-TO-PROCEDURE GENERATION VIA PROCESS GRAPH DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Experimental procedure generation is a key task in automated chemical synthesis, requiring plausible operation sequences while maintaining global consistency across operation ordering, material-state dependencies, and parameter assignments. Conventional sequence-generation paradigms linearize these interdependent decisions into text, requiring the underlying relational structure to be recovered implicitly from token-level context. To address this limitation, we propose ReactPG, the first conditional discrete graph diffusion framework for chemical procedure generation. ReactPG reformulates procedure generation as conditional generation over typed process graphs and employs a categorical Markov diffusion process over a constrained state space, parameterized by a joint process graph Transformer. We further construct OpenExp-PG, an experimental process graph dataset derived from OpenExp, and evaluate ReactPG using both graph-level and text-level metrics. Experimental results show that ReactPG achieves strong performance across multiple metrics while attaining near-zero violation rates on the evaluated procedural consistency constraints, outperforming existing large language model-based approaches.

1 INTRODUCTION

A central problem in automated chemical synthesis is translating a target reaction and its material conditions into executable laboratory operations. AI-assisted synthesis planning and chemical robotics have advanced this goal, emphasizing the need to connect chemical transformations with concrete instructions (Coley et al., 2019; Steiner et al., 2019). A reaction specifies molecular changes but does not determine how materials are processed, intermediate states are transferred, or conditions are assigned to operations. Reaction-to-procedure generation bridges this gap by inferring experimental procedures from reactions and associated materials (Vaucher et al., 2021). Reliable generation therefore requires actions, objects, and conditions to form a coherent procedure.

Existing research largely relies on language models. Early work formulated procedure prediction as translation from reaction representations to operation sequences (Vaucher et al., 2021), while molecule-text pretraining and cross-modal alignment enhanced chemical representations (Christofidellis et al., 2023; Liu et al., 2023). ReactXT introduces reaction-contextualized pretraining and OpenExp (Liu et al., 2024); ReactGPT combines reaction retrieval with in-context tuning (Chen et al., 2025); QFANG incorporates chemical procedural reasoning, and Thoth adopts structured sketch-then-fill planning (Liu et al., 2025; Sun et al., 2026). Despite these advances, outputs remain primarily textual or serialized, leaving explicit modeling of cross-step dependencies underexplored.

A plausible action may still bind the wrong material, reference an unavailable intermediate state, or use a condition belonging to another operation. Sequence models can in principle encode these relations through lossless serialization, but related information is often dispersed across textual descriptions and must be recovered by the generator or downstream software. This motivates a central question: can the entities and dependencies governing procedural consistency be represented explicitly as part of the generation state?

Motivated by relational inductive biases (Battaglia et al., 2018) and prior graph representations of operations, materials, and conditions (Mysore et al., 2019; Yang et al., 2022), we propose ReactPG, a conditional categorical diffusion model for ordered typed process graphs. The process graph is

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.

treated directly as the generation object, with a state space and Markov process designed for experimental procedures. We introduce two-level transitions that separately model material roles and quantity slots, together with the corresponding ELBO. A predicted step-count prior is fixed throughout diffusion, while actions, materials, states, conditions, and references are jointly reconstructed.

The graph representation also provides explicit interfaces for downstream experimental software: material edges identify operation objects, state pointers trace intermediate outputs, and condition slots specify numerical-value ownership. These relations support direct consistency checking and deterministic text compilation without recovering dependencies from free text, providing a structured basis for experimental automation (Steiner et al., 2019). We construct OpenExp-PG from OpenExp and evaluate ReactPG using graph- and text-level metrics together with consistency-constraint violation rates. Our main contributions are as follows:

- **Method.** We propose ReactPG, a conditional generation paradigm for experimental process graphs that jointly models procedural entities and dependencies through conditional discrete graph diffusion, combines constrained state-space design with hierarchical transitions for joint denoising, and provides the full ELBO calculation.
- **Dataset.** We construct OpenExp-PG from OpenExp to explicitly represent procedural entities and dependencies, supporting process graph generator training and graph-level and text-level evaluation.
- **Experiments.** ReactPG achieves competitive performance on multiple prediction metrics and near-zero violations of the evaluated consistency constraints, outperforming the evaluated language-model baselines in procedural consistency.

2 Related Work

2.1 Reaction-to-Procedure Language Modeling

Early work formulated reaction-to-procedure prediction as sequence generation (Vaucher et al., 2021). MolCA connects molecular graphs with language models through cross-modal alignment (Liu et al., 2023), while ReactXT extends molecule-text learning to reaction contexts and introduces OpenExp (Liu et al., 2024). ReactGPT combines reaction retrieval with in-context tuning (Chen et al., 2025); QFANG introduces chemistry-guided procedural reasoning with supervised and reinforcement learning (Liu et al., 2025); and Thoth adopts a sketch-and-fill strategy for structured procedure generation (Sun et al., 2026). In contrast, ReactPG directly generates typed procedural entities and their relations through a probabilistic graph diffusion process.

2.2 Graph Diffusion Models

GDSS jointly diffuses node features and adjacency relations (Jo et al., 2022), while DiGress introduces categorical diffusion over discrete graph structures with graph Transformer denoising (Vignac et al., 2023a). EDM and MiDi further extend diffusion to molecular geometry and atom-level attributes (Hoogeboom et al., 2022; Vignac et al., 2023b). ReactPG builds on categorical graph diffusion but targets ordered experimental process graphs with fixed material identities and typed relational semantics. Its constrained state space, transition process, and hierarchical reconstruction are designed specifically for procedural generation.

2.3 Categorical Transitions

D3PM establishes transition-matrix design for discrete diffusion and combines variational training with auxiliary cross-entropy supervision (Austin et al., 2021), while MaskGIT performs iterative refinement through bidirectional masked prediction (Chang et al., 2022). ReactPG adopts a finite-step Markov process with analytical posteriors, but introduces a grouped material-edge kernel that separately models role changes and within-role quantity-slot changes while preserving closed-form cumulative transitions.

3 TYPED PROCESS GRAPH REPRESENTATION

3.1 TASK DEFINITION

Given a reaction SMILES, an inventory of available reaction materials (e.g., reactants, products, and catalysts), and reference procedures retrieved only from the training set, the model must reconstruct the complete experimental procedure. The target operation sequence, step-wise material bindings, conditions, and numerical parameters are hidden, requiring the model to infer the reaction process and generate a structured experimental procedure.

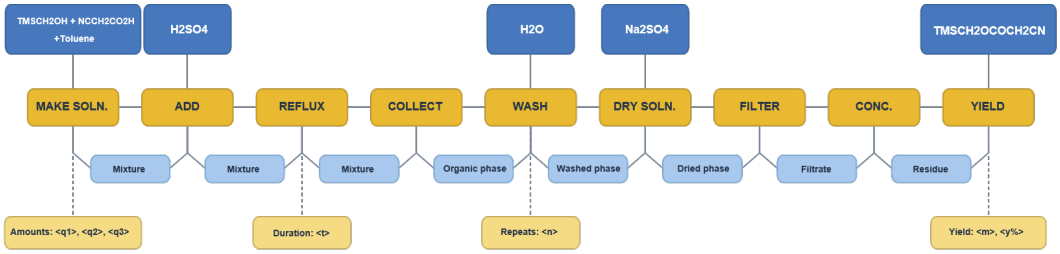


Figure 1: An example typed process graph. Operations (yellow), materials (blue), states (light blue), and condition or quantity slots (light yellow).

Table 1: Comparison of the original procedure and its compact structured representation. **BOLDED BLUE** indicates pre-defined action.

Representation	Content
Original procedure	In a reactor fitted with a Dean-Stark apparatus hydroxymethyltrimethylsilane (3.8 g, 0.036 mol) and cyanoacetic acid (3.1 g, 0.036 mol) in 50 ml of toluene are refluxed for 4 hours in the presence of one drop of concentrated sulfuric acid. The organic phase is washed twice with water and is dried over sodium sulfate. It is concentrated under reduced pressure. Distillation under vacuum gives 4.3 g of trimethylsilylmethyl cyanoacetate (69% yield) in the form of a colorless liquid which is used as it is in the following step.
Structured procedure	**MAKESOLUTION**: \$1\$ (3.8 g, 36 mmol); \$2\$ (3.1 g, 36 mmol); \$3\$ (50 mL). **ADD**: \$4\$. **REFLUX**: 4 h. **COLLECTLAYER**: organic phase. **WASH**: water; 2×. **DRYSOLUTION**: sodium sulfate. **FILTER**: filtrate. **CONCENTRATE**. **YIELD**: 4.3 g; 69%. \$1\$: hydroxymethyltrimethylsilane; \$2\$: cyanoacetic acid; \$3\$: toluene; \$4\$: sulfuric acid.

3.2 PROCESS GRAPH DEFINITION

To help the model understand the semantics of experimental steps and conditions and the links between them, we define a process graph G with two components: entity nodes and relation edges.

To maintain compatibility with the data and evaluation, we use the six-field graph annotation and export format in Table 2: actions, material bindings, operation conditions, quantity profiles, output states, and state consumption. Existing annotations use at most 64 operation positions and 23 pre-defined condition labels, and the existing format contains 23 valid operations; details are referenced in Appendix A. A material binding selects either no binding or one usage role for each operation-material pair. A quantity profile uses three bits to indicate whether amount, concentration, and equivalent-ratio slots are required; an amount slot can subsequently hold multiple measurements,

Table 2: Standard six-field graph annotation and export format.

Field	Shape	Prediction target
Operation types, **a**	N	Operation category
Material bindings, **b**	N × M	No binding or one material usage role
Operation conditions, **p**	N × P	Presence of an operation condition or fixed subtype
Quantity profiles, **q**	N × M	Amount/concentration/equivalent-ratio slot profile
Output states, **s**	N × S	Types of produced states
State consumption, **c**	N × 2N	Consumption of an earlier produced state

such as mass and amount of substance. The output-state field provides at most two state positions per step, and the consumption field points to previously produced states.

3.3 Process Graph Dataset

Table 3: Comparison of the source corpus and OpenExp-PG. HA denotes heavy-atoms.

Statistic	OpenExp	OpenExp-PG
Number of reactions	274,439	115,000
Median action count	11	11
Maximum material count	12	12
Maximum total HA	489	489
Mean material count	5.27	5.32
Mean action count	11.03	10.71
Mean product HA	24.15	24.72

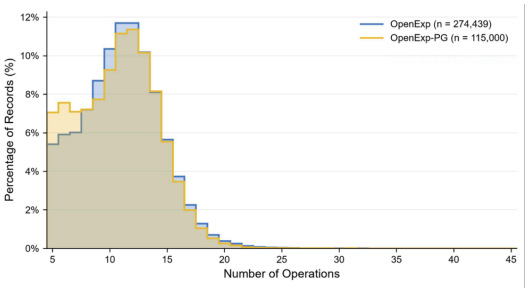


Figure 2: Operation-count distributions of OpenExp and OpenExp-PG.

Using this representation, we construct OpenExp-PG, a typed graph-annotated subset of OpenExp containing 115,000 records. Records are selected using explicit parsing and consistency criteria, and state annotations combine reference descriptions with a declared transition ontology.

Table 3 and Figure 2 show similar dataset statistics and operation-count distributions, suggesting comparable task complexity along these measured dimensions. Detailed analyses of dataset fields, annotation criteria, and split construction are provided in Appendix A.

4 Typed Process Graph Diffusion Generation

ReactPG connects length prediction, a typed graph distribution, and structure-preserving output compilation. The model first determines the number of operations, then completes the entire probabilistic graph generation chain on the canvas corresponding to that length. Figure 3 presents the overall framework; model computations and support details are given in Appendix B.

4.1 Forward Diffusion and Reverse Denoising

Generating an experimental process graph requires jointly modeling operation sequences, material assignments, condition slots, and state dependencies. We organize them into a constrained discrete state space and construct a conditional categorical diffusion model (Austin et al., 2021; Vignac et al., 2023a), incorporating process-graph semantics into the state representation, transition kernels, and reconstruction distribution.

Let a clean process graph in OpenExp-PG be $G_0 = (a_0, m_0, p_0, s_0, f_0)$, with diffusion state G_t. Here, a denotes operation types, p condition-slot presence, and s process states. m_{ij} jointly represents the binding role and quantity-slot profile between step i and material j, while f_i is either empty or points to an output-state slot of an earlier operation. The support excludes invalid combinations such as quantities on unbound materials and references to future states. Further details are given in Appendix B.

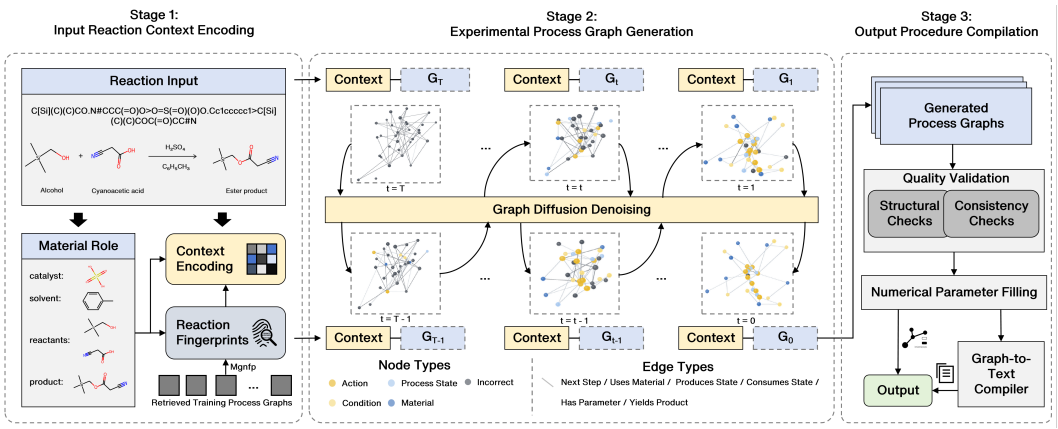


Figure 3: Overview of ReactPG: reaction context encoding, experimental process graph generation, and output procedure compilation. The encoder combines the role-annotated material catalog, reaction features, and retrieved training process graphs. The graph diffusion model generates discrete procedural structure, followed by structural and consistency checks, numerical filling, and graph-to-text compilation.

Given reaction-conditioning input C, the conditional encoder maps molecular context, fingerprints, reaction changes, and material-aligned reference graphs into memory $\mathcal{M}(C)$. We factorize length and graph generation as

$$p_{\theta,\phi}(\ell, G_0 \mid C) = p_\phi(\ell \mid C)\, p_\theta(G_0 \mid \ell, C), \qquad \mathcal{S} = \mathcal{S}(C, \ell). \tag{1}$$

$\mathcal{S}$ is the static support determined by length, step positions, material roles, and vocabulary. Training uses the true length, whereas generation determines a length from the predicted distribution and keeps it fixed throughout denoising.

For valid variable indices $\mathcal{U}(\mathcal{S})$ and category sets $\mathcal{K}_u$, we construct smoothed empirical priors π_u and define $\pi_{\mathcal{S}} = \prod_{u \in \mathcal{U}(\mathcal{S})} \pi_u$. The forward process independently corrupts variables:

$$\begin{aligned} q(G_{1:T} \mid G_0, \mathcal{S}) &= \prod_{t=1}^{T} q(G_t \mid G_{t-1}, \mathcal{S}), \\ q(G_t \mid G_{t-1}, \mathcal{S}) &= \prod_{u \in \mathcal{U}(\mathcal{S})} Q_{t,u}[x_{t-1,u}, x_{t,u}], \\ Q_{t,u} &= \alpha_{t,u} I + (\rho_{t,u} - \alpha_{t,u}) H_u + (1 - \rho_{t,u}) P_u, \quad P_u = \mathbf{1}\pi_u^\top. \end{aligned} \tag{2}$$

Here, $0 \le \alpha_{t,u} \le \rho_{t,u} \le 1$. For material relations, H_u resamples quantity profiles within the same role, while P_u resamples over the full support. The retention rates control corruption at the role and quantity-profile levels. For other variables, $\rho_{t,u} = \alpha_{t,u}$ gives a standard categorical reset kernel.

The reset operators admit the following closed-form cumulative transition:

$$\begin{aligned} \bar{Q}_{t,u} &= \prod_{v=1}^{t} Q_{v,u} = A_{t,u} I + (R_{t,u} - A_{t,u}) H_u + (1 - R_{t,u}) P_u, \\ A_{t,u} &= \prod_{v=1}^{t} \alpha_{v,u}, \qquad R_{t,u} = \prod_{v=1}^{t} \rho_{v,u}, \\ q(G_t \mid G_0, \mathcal{S}) &= \prod_{u \in \mathcal{U}(\mathcal{S})} \bar{Q}_{t,u}[x_{0,u}, x_{t,u}]. \end{aligned} \tag{3}$$

We choose $A_{T,u} = R_{T,u} = 0$, yielding $q(G_T \mid G_0, \mathcal{S}) = \pi_{\mathcal{S}}$. Equation (3) also enables direct sampling of G_t during training. Prior construction and hierarchical or shared noise schedules are detailed in Appendix B.

Reverse generation starts from $G_T \sim \pi_{\mathcal{S}}$. The denoising network predicts the clean-category distribution $\hat{p}_{\theta,u}(x_{0,u} \mid G_t, C, \ell)$. For $t > 1$, the analytical posterior gives

$$\begin{aligned} q_{t,u}(k \mid j, i, \mathcal{S}) &= \frac{\bar{Q}_{t-1,u}[i,k] Q_{t,u}[k,j]}{\bar{Q}_{t,u}[i,j]}, \\ p_{\theta,t,u}(k \mid G_t, C, \ell) &= \sum_{i \in \mathcal{K}_u} \hat{p}_{\theta,u}(i \mid G_t, C, \ell) q_{t,u}(k \mid x_{t,u}, i, \mathcal{S}), \\ p_{\theta,t}(G_{t-1} \mid G_t, C, \ell) &= \prod_{u \in \mathcal{U}(\mathcal{S})} p_{\theta,t,u}(x_{t-1,u} \mid G_t, C, \ell). \end{aligned} \tag{4}$$

A shared node-edge network predicts all variables, allowing graph-wide dependencies to inform iterative refinement of both operations and relations.

Static support enforces type and temporal validity, while remaining dependencies between actions, material roles, states, and references are handled by a normalized hierarchical reconstruction distribution:

$$p_{\theta,1}(G_0) = p_\theta(a_0)\, p_\theta(m_0 \mid a_0)\, p_\theta(s_0 \mid a_0)\, p_\theta(p_0)\, p_\theta(f_0 \mid s_0). \tag{5}$$

Reconstruction first determines actions, then restricts material roles and output states, and finally reference targets. Each factor is normalized over candidates compatible with its generated parents. The conditional graph model is therefore

$$p_\theta(G_0 \mid \ell, C) = \sum_{G_{1:T}} \pi_{\mathcal{S}}(G_T) \prod_{t=1}^{T} p_{\theta,t}(G_{t-1} \mid G_t, C, \ell). \tag{6}$$

The diffusion model generates discrete operations, relations, and parameter slots; continuous numerical values are completed by a subsequent filling module. Further details are given in Appendix B.

4.2 Denoising Network Parameterization

To jointly model operations and relations throughout denoising, we use a node-edge graph Transformer maintaining step representations H, operation-material edges E^M, and state-flow edges E^F. Representations are initialized from G_t embeddings, with position, length, and diffusion-time embeddings added to step states:

$$(H^{(b+1)}, E^{M,(b+1)}, E^{F,(b+1)}) = F_\theta^{(b)}\big(H^{(b)}, E^{M,(b)}, E^{F,(b)}; \mathcal{M}(C), H_M, \mathcal{S}\big), \tag{7}$$

where H_M denotes encoded material representations. Each block combines step self-attention, material cross-attention, and conditional cross-attention. Step and edge representations are updated jointly, allowing relations to serve as both prediction targets and contextual features. Typed output heads produce the clean-category distributions in Equation (4) and reconstruction factors in Equation (5). Architectural details are given in Appendix B.

The reverse Markov process is trained with a variational objective. For one example, the negative evidence lower bound is

$$\begin{aligned} \mathcal{L}_{\mathrm{VLB}} &= \underbrace{-\log p_\phi(\ell \mid C)}_{\mathcal{L}_{\mathrm{len}}} + \underbrace{D_{\mathrm{KL}}(q(G_T \mid G_0, \mathcal{S}) \,\|\, \pi_{\mathcal{S}})}_{\mathcal{L}_T^{\mathrm{prior}} = 0} + \mathcal{L}_1 + \sum_{t=2}^{T} \mathcal{L}_t, \\ \mathcal{L}_1 &= \mathbb{E}_{q(G_1 \mid G_0, \mathcal{S})} \left[-\log p_{\theta,1}(G_0 \mid G_1, C, \ell) \right], \\ \mathcal{L}_t &= \mathbb{E}_{q(G_t \mid G_0, \mathcal{S})} D_{\mathrm{KL}}(q(G_{t-1} \mid G_t, G_0, \mathcal{S}) \,\|\, p_{\theta,t}(G_{t-1} \mid G_t, C, \ell)) \,. \end{aligned} \tag{8}$$

Thus, $\log p_{\theta,\phi}(\ell, G_0 \mid C) \geq -\mathcal{L}_{\mathrm{VLB}}$. The terminal KL is zero by construction; the remaining terms use the hierarchical reconstruction distribution and analytical forward posterior. We estimate the objective by random timestep sampling and add auxiliary supervision:

$$\mathcal{L}_{\mathrm{train}} = \mathbb{E}_{\mathrm{data}}\left[\frac{1}{\kappa}\mathcal{L}_{\mathrm{VLB}} + \lambda_0\mathcal{L}_{x_0} + \lambda_{\mathrm{c}}\mathcal{L}_{\mathrm{cond}} + \lambda_{\mathrm{a}}\mathcal{L}_{\mathrm{align}}\right]. \tag{9}$$

The auxiliary terms improve field-level supervision, condition discrimination, and reference alignment without modifying the forward or reverse diffusion distributions. Their definitions, coefficients, and timestep estimator are provided in Appendix C.

5 Experiments

To evaluate ReactPG, we conduct comprehensive experiments comparing our method with existing approaches on OpenExp-PG using structured-text and consistency metrics. We then perform ablation studies to examine the contributions of individual components.

5.1 Experimental Setup

5.1.1 Baselines

We select five representative classes of models as baselines, covering molecular language modeling, reaction-contextualized modeling, large language model adaptation, scientific reasoning, and explicit procedural structure generation through sketching followed by filling.

MolT5 (Edwards et al., 2022). MolT5 is a T5-based molecular language model pretrained on molecular strings and natural-language text. We fine-tune it on the experimental procedure texts corresponding to our training set.

ReactXT (Liu et al., 2024). ReactXT uses reaction-contextualized molecule-text pretraining to improve chemical reaction understanding and is adapted to our procedure texts.

ReactGPT (Chen et al., 2025). ReactGPT combines reaction retrieval, domain-specific prompting, and in-context tuning for reaction-to-text generation.

QFANG (Liu et al., 2025). QFANG generates structured synthesis procedures using chemistry-guided reasoning, supervised fine-tuning, and reinforcement learning.

Thoth (Sun et al., 2026). Thoth adopts a **Sketch-and-Fill** paradigm for biological protocol generation, separating high-level procedural planning from subsequent detail completion. This two-stage design is conceptually similar to ReactPG's separation of discrete procedural structure generation from numerical parameter filling, making Thoth a particularly relevant baseline when transferred to our reaction-to-procedure task.

5.1.2 Implementation Details

ReactPG uses a node hidden dimension of 512, 8 joint blocks, 8 attention heads, an edge hidden dimension of 64, and $T = 32$ diffusion steps. We use a hierarchical noise schedule and train for 50 epochs with AdamW; detailed settings are provided in Appendix D. All methods use the same data split and supervised examples: ReactPG is trained on process graphs, while language-model baselines use the corresponding procedure texts.

All trainable language-model baselines are adapted on our training corpus. Models above 4B parameters use LoRA (Hu et al., 2022), while smaller models are fully fine-tuned when feasible. Text models use a maximum input length of 4096 tokens and decode with temperature 0.95, top-p 0.95, and top-k 5. We preserve model-specific input or retrieval mechanisms while providing the corresponding reaction information whenever compatible.

All reported results are averaged over three independent runs with different random seeds, and all experiments run on two NVIDIA A100-80G GPUs. Detailed hyperparameter settings are provided in Appendix D.

5.1.3 EVALUATION METRICS

We evaluate both graph and text outputs to enable comparison with language-model baselines. Text outputs are assessed both as natural language and after parsing into structured action-object-condition representations.

Graph metrics include exact match, normalized action edit similarity, and F1 scores for material bindings and state flow. Exact match requires agreement across all six discrete graph fields.

Natural-language metrics include BLEU-2/4 (Papineni et al., 2002), ROUGE-1/2/L (Lin, 2004), and text validity. Numerical values and units are normalized to a unified representation before text evaluation. Structured text metrics include LEV50, LEV75, LEV90, and the Structural Consistency Violation Rate (SCVR). Text validity checks local action formats, including object counts and mandatory conditions. SCVR additionally audits schema violations (S_i), unresolved material references (M_i), action-object incompatibility (A_i), invalid parameters (P_i), conflicting values (C_i), and termination violations (T_i), where each symbol denotes the corresponding event for output i:

$$\mathrm{SCVR} = \frac{100}{N} \sum_{i=1}^{N} \mathbf{1}[S_i \cup M_i \cup A_i \cup P_i \cup C_i \cup T_i] \,.$$

Each output is counted once; lower is better.

5.2 PERFORMANCE COMPARISON

Table 4: Comparison on OpenExp-PG: (a) natural-language text metrics; (b) structured text metrics.

(a) Natural-language text

Method	Params	BLEU-2 ↑	BLEU-4 ↑	ROUGE-1 ↑	ROUGE-2 ↑	ROUGE-L ↑	Validity ↑
Nearest Neighbor	-	52.30	37.05	62.28	43.95	57.02	76.42
MolT5-large	0.77B	54.17	41.00	61.25	42.38	57.29	99.10
ReactXT	1.3B	59.65	44.62	69.33	52.55	60.99	99.80
ReactGPT	8B	57.89	43.73	66.10	42.95	59.98	99.72
QFANG	8B	58.87	43.14	69.50	51.81	66.04	100
Thoth	8B	59.56	43.81	68.83	51.53	65.41	100
ReactPG-R(ours)	1.5B	61.82	46.00	72.66	54.85	66.96	100
ReactPG-P(ours)	3B	**66.82**	**53.89**	**76.06**	**61.43**	**70.39**	**100**

(b) Parsed structured text

Method	Params	50%LEV ↑	75%LEV ↑	90%LEV ↑	SCVR ↓
Nearest Neighbor	-	88.64	21.36	5.78	3.36
MolT5-large	0.77B	89.82	23.28	8.74	4.16
ReactXT	1.3B	95.86	27.78	5.62	8.20
ReactGPT	8B	91.00	26.95	11.11	4.78
QFANG	8B	95.36	36.54	12.82	0.88
Thoth	8B	94.28	35.48	12.26	2.40
ReactPG-R(ours)	1.5B	97.48	36.61	11.23	0.03
ReactPG-P(ours)	3B	**97.77**	**39.64**	**13.05**	**0.03**

We construct two main ReactPG variants for performance comparison: ReactPG-Reference (ReactPG-R), which retrieves condition parameter values from similar training examples via nearest-neighbor search, and ReactPG-Prediction (ReactPG-P), which uses a pretrained large language model to predict the required condition parameter values. Details of numerical parameter filling are provided in Appendix B. As shown in Table 4, ReactPG achieves performance comparable to state-of-the-art large language model baselines on natural-language text metrics, while attaining 100% text validity. On structured-text metrics, ReactPG-P consistently outperforms the baselines. Specifically, it improves LEV75 by 3.10 percentage points and LEV50 by 1.91 percentage points over the best baseline. On SCVR, our primary consistency metric, ReactPG achieves a near-zero violation rate, 0.85 points lower than the best baseline, demonstrating strong structural consistency and constraint adherence.

5.3 Ablation Analysis

Table 5: Graph autoregressive generation versus graph diffusion.

Variant	Skeleton exact ↑	Edit similarity ↑	Graph exact ↑	SCVR ↓
ReactPG - graph autoregressive	15.34	70.57	2.86	3.76
ReactPG - graph diffusion	**16.65**	**74.78**	**6.77**	**0.03**

We compare ReactPG with an autoregressive graph baseline. The baseline aggregates node and edge information into tokens and autoregressively predicts the process graph in sequence. Both models use identical input information, graph representations, constraint-support mechanisms, numerical parameter filling module and compiler. As shown in Table 5, graph diffusion improves edit similarity from 70.57 to 74.78 and graph exact match from 2.86 to 6.77, while reducing SCVR from 3.76 to 0.03. These results suggest that graph diffusion benefits from joint relational modeling and iterative refinement, which allows earlier structural predictions to be revised during generation.

Table 6: Ablation of ReactPG components.

Variant	Edit similarity ↑	Binding F1 ↑	Flow F1 ↑	Graph exact ↑	SCVR ↓
ReactPG	**74.78**	**69.97**	**61.75**	**6.77**	**0.03**
No auxiliary losses	73.05	68.94	58.51	5.76	0.08
No joint constraints	72.45	68.09	59.45	4.82	0.08
No edge messages	72.40	66.47	57.79	2.54	0.16

Table 6 ablates auxiliary losses, joint constraints, and edge messages. Removing edge messages produces the largest drop in graph exact match, from 6.77 to 2.54, together with lower binding and flow F1, highlighting their role in modeling graph relations. The *No joint constraints* variant removes action-material compatibility, action-output-state compatibility, minimum material-binding requirements, and inventory-dependent action feasibility. Removing these constraints reduces graph exact match to 4.82 and increases SCVR from 0.03 to 0.08. Auxiliary losses mainly improve graph exact match, with little change in binding F1.

6 Limitations

ReactPG separates discrete procedural structure from continuous numerical values. A separate model is still required to fill the generated numerical slots before the system can produce a complete procedure. Consequently, accurate graph generation alone does not ensure accurate numerical completion, and the end-to-end output depends on both components. Our experiments are also limited to the OpenExp-derived dataset studied here. We have not yet evaluated larger datasets or datasets with substantially different distributions.

7 Conclusion and Future Works

We introduced ReactPG, a conditional discrete diffusion framework for generating chemical experimental procedures as typed process graphs. ReactPG jointly models procedural entities and dependencies under type and temporal constraints, with categorical Markov transitions and hierarchical reconstruction. Experiments on OpenExp-PG show competitive performance and near-zero violations of the evaluated consistency constraints. For future work, we plan to extend ReactPG toward discrete-continuous joint graph diffusion, jointly modeling procedural structures and continuous condition parameters, and to evaluate it on larger and more diverse datasets.

AI Use Statement

Generative AI tools were used only for language polishing, translation, and limited code assistance. They were not used for method design, theoretical development, mathematical claims or proofs,

experimental design, result interpretation, or scientific conclusions. All AI-assisted text and code were manually reviewed and revised by the authors.

Reproducibility Statement

Key hyperparameters and experimental settings are detailed in the main text; further implementation details and hyperparameter information can be found in the appendix. The supplementary materials include the source code, dataset processing scripts, and documentation files required to reproduce this experiment.

References

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2107.03006.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. URL https://arxiv.org/abs/1806.01261.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11315–11325, 2022. URL https://arxiv.org/abs/2202.04200.

Zhe Chen, Zhe Fang, Wenhao Tian, Zhaoguang Long, Changzhi Sun, Yuefeng Chen, Hao Yuan, Honglin Li, and Man Lan. ReactGPT: Understanding of chemical reactions via in-context tuning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(1):84–92, 2025. doi: 10.1609/aaai.v39i1.31983.

Dimitrios Christofidellis, Giorgio Giannone, Jannis Born, Ole Winther, Teodoro Laino, and Matteo Manica. Unifying molecular and textual representations via multi-task language modelling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 6140–6157, 2023. URL https://proceedings.mlr.press/v202/christofidellis23a.html.

Connor W. Coley, Dale A. Thomas, III, Justin A. M. Lummiss, Jonathan N. Jaworski, Christopher P. Breen, Victor Schultz, Travis Hart, Joshua S. Fishman, Luke Rogers, Hanyu Gao, Robert W. Hicklin, Pieter P. Plehiers, Joshua Byington, John S. Piotti, William H. Green, A. John Hart, Timothy F. Jamison, and Klavs F. Jensen. A robotic platform for flow synthesis of organic compounds informed by AI planning. *Science*, 365(6453):eaax1566, 2019. doi: 10.1126/science.aax1566.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. Translation between molecules and natural language. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 375–413, 2022. doi: 10.18653/v1/2022.emnlp-main.26. URL https://aclanthology.org/2022.emnlp-main.26/.

Emiel Hoogeboom, Víctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3D. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 8867–8887, 2022. URL https://proceedings.mlr.press/v162/hoogeboom22a.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 10362–10383, 2022. URL https://proceedings.mlr.press/v162/jo22a.html.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, 2004. URL https://aclanthology.org/W04-1013/.

Guoqing Liu, Junren Li, Zihan Zhao, Eray Inanc, Krzysztof Maziarz, Jose Garrido Torres, Victor Garcia Satorras, Shoko Ueda, Christopher M. Bishop, and Marwin Segler. A scientific reasoning model for organic synthesis procedure generation. *arXiv preprint arXiv:2512.13668*, 2025. URL https://arxiv.org/abs/2512.13668.

Zhiyuan Liu, Sihang Li, Yanchen Luo, Hao Fei, Yixin Cao, Kenji Kawaguchi, Xiang Wang, and Tat-Seng Chua. MolCA: Molecular graph-language modeling with cross-modal projector and uni-modal adapter. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15623–15638, 2023. doi: 10.18653/v1/2023.emnlp-main.966.

Zhiyuan Liu, Yaorui Shi, An Zhang, Sihang Li, Enzhi Zhang, Xiang Wang, Kenji Kawaguchi, and Tat-Seng Chua. ReactXT: Understanding molecular "reaction-ship" via reaction-contextualized molecule-text pretraining. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 5353–5377, 2024. doi: 10.18653/v1/2024.findings-acl.318.

Sheshera Mysore, Zachary Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures. In *Proceedings of the 13th Linguistic Annotation Workshop*, pp. 56–64, 2019. doi: 10.18653/v1/W19-4007. URL https://aclanthology.org/W19-4007/.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, 2002. doi: 10.3115/1073083.1073135.

Sebastian Steiner, Jakob Wolf, Stefan Glatzel, Anna Andreou, Jaroslaw M. Granda, Graham Keenan, Trevor Hinkley, Gerardo Aragon-Camarasa, Philip J. Kitson, Davide Angelone, and Leroy Cronin. Organic synthesis in a modular robotic system driven by a chemical programming language. *Science*, 363(6423):eaav2211, 2019. doi: 10.1126/science.aav2211.

Haoran Sun, Yankai Jiang, Zhenyu Tang, Yaning Pan, Shuang Gu, Zekai Lin, Lilong Wang, Wenjie Lou, Lei Liu, Lei Bai, and Xiaosong Wang. Unleashing scientific reasoning for bio-experimental protocol generation via structured component-based reward mechanism. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/7721f1fea280e9ffae528dc78c732576-Abstract-Conference.html.

Alain C. Vaucher, Philippe Schwaller, Joppe Geluykens, Vishnu H. Nair, Anna Iuliano, and Teodoro Laino. Inferring experimental procedures from text-based representations of chemical reactions. *Nature Communications*, 12(1):2573, 2021. doi: 10.1038/s41467-021-22951-1.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://arxiv.org/abs/2209.14734.

Clément Vignac, Nagham Osman, Laura Toni, and Pascal Frossard. MiDi: Mixed graph and 3D denoising diffusion for molecule generation. In *Machine Learning and Knowledge Discovery in Databases: Research Track*, pp. 560–576, 2023b. doi: 10.1007/978-3-031-43415-0_33.

Xianjun Yang, Ya Zhuo, Julia Zuo, Xinlu Zhang, Stephen Wilson, and Linda Petzold. PcMSP: A dataset for scientific action graphs extraction from polycrystalline materials synthesis procedure text. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 6033–6046, 2022. doi: 10.18653/v1/2022.findings-emnlp.446. URL https://aclanthology.org/2022.findings-emnlp.446/.

A Dataset Details

Dataset construction and splits. We construct OpenExp-PG from 274,439 OpenExp records using explicit parsing and annotation criteria. Records are excluded when material identities, operation-material roles, condition labels, or process-state references cannot be resolved within the declared schema (Table 7). These checks establish whether a record supports an unambiguous graph annotation; they do not certify chemical correctness or experimental feasibility. No model predictions or performance scores are used to select records.

After annotation filtering, we exclude reaction groups that cross the existing splits, retain one representative per normalized reactant-product group, and exclude historical training or validation exposure from the test candidates. Validation and test examples are sampled proportionally across strata defined by operation count, workup complexity, and material count, with coverage checks for available operation types. This produces 118,407 records. We then remove 3,407 five- or six-operation training records to obtain 100,000 training, 5,000 validation, and 10,000 test records. Removal quotas are proportional to the two length groups, and record selection uses deterministic hashing with seed 19. Validation and test membership remains unchanged.

Table 7: Filtering and selection of OpenExp-PG. Annotation exclusions are counted by the first failed check for each record. Later rows report additional removals at successive stages. The final two removals adjust training-set size and are not annotation failures.

Exclusion or selection criterion	**Records removed**
Annotation filtering	
Unresolved phase selection	32,409
Unresolved material identity	29,046
Ambiguous filter retention	27,358
Unknown material annotation	21,438
Unresolved material role	21,080
Repeated material reference within an action	4,075
Source-action mismatch for reaction operations	3,383
Process action without a resolved material	2,427
Unrecognized condition	1,267
Other parsing, binding, or consistency failures	1,922
Annotation-filtering subtotal	144,405
Group isolation and split selection	
Reaction-group isolation, deduplication, and historical-exposure exclusion	1,000
Stratified validation and test subsampling	10,627
Training-size adjustment	
Five-operation training records	1,746
Six-operation training records	1,661
Total removed	**159,439**
Final retained records	**115,000**

Dataset characteristics and evaluation scope. The dataset comparison table and operation-count distributions summarize observable differences between OpenExp and OpenExp-PG. Table 8 reports operation, condition, and state occurrence counts in the final subset, including unused predefined categories. Repeated occurrences within a record are counted separately; EMPTY and NONE are excluded.

All compared methods are retrained and evaluated on the same final splits. Prior counts and action-role and action-state compatibility tables are estimated from training labels only.

Table 8: Category frequencies in the OpenExp-PG subset. Each category group is sorted by decreasing occurrence count, with unused categories listed last. Counts refer to operation instances, operation-level condition labels, and produced state nodes.

Operation	**Count**	**Condition**	**Count**	**Process state**	**Count**
ADD	315,053	Temperature	163,256	Mixture	735,246
CONCENTRATE	122,919	Duration	161,355	Unspecified	142,958
STIR	121,775	Yield percentage	58,969	Organic liquid	140,522
YIELD	115,000	Repeat count	17,059	Liquid	88,320
MAKESOLUTION	112,773	Atmosphere	11,565	Solid	87,994
WASH	101,977	Atmosphere: inert	10,371	Aqueous liquid	43,760
FILTER	83,201	pH	5,319	Oil	0
COLLECTLAYER	76,707	Acidity: acidic	2,705	-	-
DRYSOLUTION	70,126	Addition: dropwise	2,117	-	-
EXTRACT	27,727	Acidity: neutral	1,801	-	-
SETTEMPERATURE	18,277	Atmosphere: vacuum	1,194	-	-
WAIT	16,783	Acidity: basic	813	-	-
REFLUX	15,633	Concentration	624	-	-
QUENCH	8,338	Ratio	16	-	-
PHASESEPARATION	7,754	Pressure	0	-	-
PH	6,996	Flow rate	0	-	-
PARTITION	3,302	Stirring speed	0	-	-
RECRYSTALLIZE	3,220	Addition rate	0	-	-
DEGAS	1,324	Endpoint	0	-	-
DRYSOLID	1,313	Addition: portionwise	0	-	-
TRITURATE	1,099	Addition: slowly	0	-	-
MICROWAVE	403	Mixing: vigorous	0	-	-
SONICATE	116	Mixing: gentle	0	-	-
Total	**1,231,816**	**Total**	**437,164**	**Total**	**1,238,800**

B Method Details

B.1 Variables and Encoding

Let ℓ be the valid number of operations and M the size of the fixed input material catalog. The state contains the following variables.

Table 9: Diffusion variables and their encodings.

Field	Index and values	Meaning
a_i	$1 \le i \le \ell$; allowed operation category	Operation type
m_{ij}	$1 \le j \le M$; 0-144	Joint operation-material relation
p_{ih}	Condition vocabulary index h; 0 or 1	Condition/discrete-attribute slot presence
s_{ir}	$r \in \{1, 2\}$; state category or NONE	Two output-state slots per step
f_i	NONE or (j, r) with $j < i$	Categorical pointer to primary input state

For a nonempty binding, the material role index is $r_{ij} \in \{1, \ldots, 18\}$, and the quantity-slot profile index is $v_{ij} \in \{0, \ldots, 7\}$. The joint encoding is

$$m_{ij} = \begin{cases} 0, & \text{no material binding,} \\ 1 + 8(r_{ij} - 1) + v_{ij}, & \text{material binding present.} \end{cases} \tag{B1}$$

The total vocabulary size is $1 + 18 \times 8 = 145$. This is the model's global categorical vocabulary. Every diffusion state avoids combinations in which quantities exist without material ownership.

The state pointer is a lossless recoding within the scope of the existing labels: the first step has no primary state input, and each subsequent step has exactly one primary state input.

B.2 Statistical Priors

Let $n_u(k)$ denote the training count corresponding to variable u. Different fields share the relevant categorical statistics according to the implementation; for example, joint material categories use global statistics, whereas state-reference statistics are indexed by step position. The prior is

$$\pi_u(k) = \frac{\mathbf{1}[k \in \mathcal{K}_u]\sqrt{n_u(k) + 0.5}}{\sum_{j \in \mathcal{K}_u} \sqrt{n_u(j) + 0.5}}. \tag{B2}$$

Only training counts are used. The square-root transform reduces concentration in frequent categories, while smoothing gives every statically allowed category positive probability. Input support varies with the query, but the statistical counts themselves do not access validation or test targets.

For a material variable, let $g(k)$ denote the role group of joint category k, with the no-binding category forming its own group. The within-role reset matrix is

$$H_u[i, j] = \frac{\mathbf{1}[g(i) = g(j)]\pi_u(j)}{\sum_{k:g(k)=g(i)} \pi_u(k)}. \tag{B3}$$

The matrices are defined only on $\mathcal{K}_u$, so within-group denominators are nonzero. H_u preserves the role group while selecting another quantity profile within that group; $P_u[i, j] = \pi_u(j)$ ignores the original category. They satisfy

$$H_u^2 = H_u, \quad P_u^2 = P_u, \quad H_u P_u = P_u H_u = P_u. \tag{B4}$$

Thus, when two transition kernels are multiplied, the coefficients that retain the original category multiply, as do the coefficients that retain the role hierarchy. Induction over time gives Equation (3). Since $\pi_u^\top H_u = \pi_u^\top P_u = \pi_u^\top$, π_u is also stationary for every transition.

B.3 Hierarchical and Shared Noise Schedules

Both schedules are predefined and use a normalized cosine base with $z = t/T$ and $\epsilon = 0.008$:

$$c_t = \frac{\cos^2\left(\frac{z+\epsilon}{1+\epsilon}\frac{\pi}{2}\right)}{\cos^2\left(\frac{\epsilon}{1+\epsilon}\frac{\pi}{2}\right)}, \qquad c_0 = 1, \quad c_T = 0. \tag{B5}$$

The endpoints are set exactly to 1 and 0. The cumulative retention rates take either of two forms:

$$\begin{aligned} \text{Hierarchical:} \quad & A_{t,u} = c_t^{\gamma_u}, \qquad R_{t,u} = c_t^{\eta_u}, \quad \gamma_u \geq \eta_u > 0, \\ \text{Shared:} \quad & A_{t,u} = R_{t,u} = c_t \qquad \text{for every } u. \end{aligned} \tag{B6}$$

Our main experiments use the hierarchical schedule with $T = 32$ and the exponents below. Material roles retain information longer than their quantity-profile details during forward corruption.

Table 10: Exponents for the typed noise schedule.

Field	γ_u	η_u
Actions a	0.70	0.70
Joint material relations m	1.80	0.90
Condition slots p	1.50	1.50
Output states s	0.80	0.80
State references f	0.60	0.60

The shared alternative sets every exponent to 1, including both material exponents. Consequently, $\rho_{t,u} = \alpha_{t,u}$ and the within-role term in Equation (2) vanishes. It therefore removes both cross-field speed differences and within-material speed separation.

For either schedule, one-step rates are $\alpha_{t,u} = A_{t,u}/A_{t-1,u}$ and $\rho_{t,u} = R_{t,u}/R_{t-1,u}$ for $t = 1, \ldots, T$. The timestep and retention rates are embedded into the denoiser. Both choices preserve the closed-form posterior and yield $\bar{Q}_{T,u} = P_u$, so the terminal-prior KL remains zero.

B.4 Reverse Distributions and Hierarchical Reconstruction

For $t > 1$, the network logits undergo masked softmax over the static support $\mathcal{K}_u$ to obtain $\hat{p}_{\theta,u}$. The denominator in Equation (4) depends on the currently observed category and the candidate clean category. Generation performs categorical sampling from this mixture posterior.

For $t = 1$, the network performs only one forward pass. Final reconstruction uses the same logits, applying conditional masks and renormalizing after sampling parent variables:

1. Sample a_0 within the action support.
2. Sample m_0 and s_0 using the action-role and action-state compatibility tables for a_0, respectively; sample p_0 from its own allowed support.
3. Restrict the support of f_0 according to the existence of the sampled state slots. The first step uses NONE, and each subsequent step selects one valid preceding state slot. The primary output slot of the first step is restricted to non-NONE, ensuring nonempty reference support for subsequent steps.

Compatibility tables are initialized from explicit rules and expanded with acceptable combinations observed in training labels. When computing the final reconstruction probability during training, conditional masks use the true parent variables; inference uses sampled parents. Each factor is normalized, so Equation (5) defines a complete probability distribution.

Given the length, the full chain is $G_T \to G_{T-1} \to \cdots \to G_1 \to G_0$. The implementation determines generation length as $\hat{\ell} = \arg\max_\ell p_\phi(\ell \mid C)$.

B.5 Joint Node-Edge Denoiser

Algorithm 1: Conditional sampling of a typed process graph

Input: Reaction context C, denoiser θ, length predictor ϕ, statistical priors, and schedule $\{Q_{t,u}, \bar{Q}_{t,u}\}_{t=1}^{T}$.

$\mathcal{M} \leftarrow \mathcal{M}(C)$
$\ell \leftarrow \arg\max_l p_\phi(l \mid C)$
Construct $\mathcal{S}(C, \ell)$ and support-restricted priors $\{\pi_u\}_{u \in \mathcal{U}(\mathcal{S})}$
Sample $G_T \sim \pi_{\mathcal{S}} = \prod_{u \in \mathcal{U}(\mathcal{S})} \pi_u$
for $t = T, \ldots, 2$ **do**
  Compute clean-category probabilities $\hat{p}_{\theta,u}(\cdot \mid G_t, C, \ell)$ using the joint node-edge denoiser
  foreach $u \in \mathcal{U}(\mathcal{S})$ **do**
    $j \leftarrow x_{t,u}$
    $r_u(k) \leftarrow \sum_{i \in \mathcal{K}_u} \hat{p}_{\theta,u}(i \mid G_t, C, \ell) \frac{\bar{Q}_{t-1,u}[i,k] Q_{t,u}[k,j]}{\bar{Q}_{t,u}[i,j]}$
    Sample $x_{t-1,u} \sim \mathrm{Cat}(r_u)$
  Assemble G_{t-1} from the sampled variables
Compute final reconstruction logits from $(G_1, \mathcal{M}, \ell)$
Sample $a_0 \sim p_\theta(a \mid G_1, C, \ell)$
Sample $m_0 \sim p_\theta(m \mid a_0, G_1, C, \ell)$ under role and minimum-binding constraints
Sample $s_0 \sim p_\theta(s \mid a_0, G_1, C, \ell)$ under action-state constraints
Sample $p_0 \sim p_\theta(p \mid G_1, C, \ell)$
Sample $f_0 \sim p_\theta(f \mid s_0, G_1, C, \ell)$ over valid preceding state outputs
Output: $G_0 = (a_0, m_0, p_0, s_0, f_0)$.

Conditional molecular features are produced by a graph-text encoder and form condition tokens together with reaction fingerprints and material roles. Reference graphs are retrieved only from the training set, excluding the query itself and records in the same normalized reaction group. Aligned reference steps are added to $\mathcal{M}(C)$. Neither target source text nor target numerical conditions are supplied to the denoiser.

Step representations comprise action, condition-presence, output-state, position, length, and time embeddings. Operation-material edges are initialized from joint-category embeddings of m_t. Inside the network, state references f_t are expanded into relation representations pointing to candidate pro-

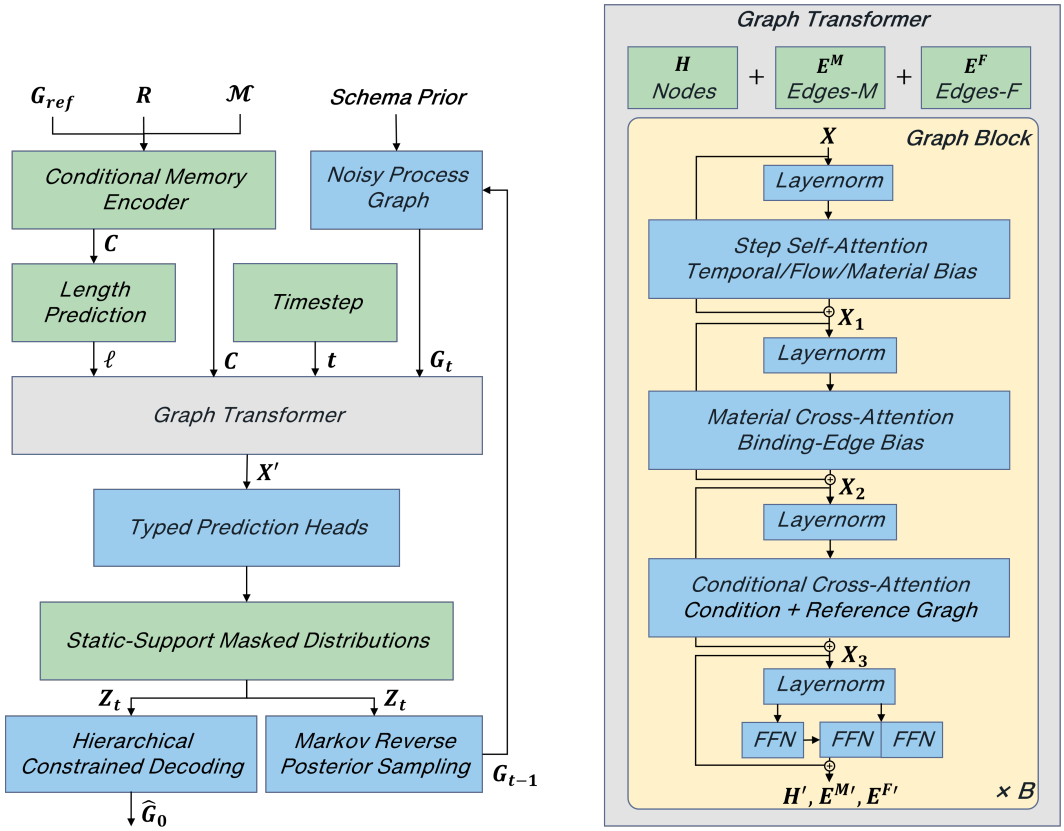


Figure 4: Detailed model architecture. Left: conditional memory, length and time conditioning, joint graph Transformer, typed prediction heads, static-support masking, hierarchical constrained reconstruction, and Markov reverse-posterior sampling. Right: joint updates of step and edge representations within a graph block.

ducer slots, with the state type and slot identity added. The probabilistic output remains a categorical pointer rather than independent binary edges.

Step self-attention. Q, K, and V are linear projections of the LayerNorm-normalized step representations. For each attention head, the weights take the form

$$\omega_{ij} = \mathrm{softmax}_j \left(\frac{Q_i K_j^\top}{\sqrt{d_h}} + B_{ij}^{\mathrm{seq}} + B_{ij}^{\mathrm{flow}} + B_{ij}^{\mathrm{shared}} \right). \tag{B7}$$

The sequential bias comes from adjacent step positions; the flow bias distinguishes state-reference direction; and the shared bias reflects binding and quantity overlap estimated from the current material-edge hidden representations. In addition to aggregating node value vectors, attention aggregates state-flow edge value messages and updates step representations through a residual connection. Step attention is bidirectional; temporal validity constraints are imposed on the candidate support of state references.

Material and conditional cross-attention. Updated step representations serve as Q, while material representations H_M provide K and V. Current material-edge representations generate both attention biases and edge value messages. The step representations then again serve as Q to read K and V from $\mathcal{M}(C)$; a residual FFN completes the node update.

Edge updates. Material edges combine the current edge representation, updated step representation, and material representation through a residual MLP. State-flow edges combine the current edge

representation, consuming-step representation, and producer-step/state-slot context through another residual MLP. Both updated edge types enter the next layer, giving the joint iteration in Equation (7).

The final action, condition, and output-state heads read node representations; the joint material-category head reads a combination of node, material, and material-edge representations; and the state-reference head reads a combination of consuming-step, producer-step/slot, and state-flow edge representations.

B.6 Numerical Parameter Filling

Numerical parameters are filled as a post-processing step after graph generation, while the generated graph structure remains unchanged. We evaluate two filling strategies.

Reference-based filling retrieves the most similar complete procedure graph from the training set and copies the values of the corresponding parameter slots.

Prediction-based filling uses a pretrained language model to predict the values of the predefined slots from the generated process context.

In our experiments, the prediction-based setting uses a fine-tuned ReactXT model. Parameter counts include the graph generator and, for ReactPG-P, the numerical-filling model.

C ELBO Estimation and Auxiliary Objectives

C.1 Random Timestep Estimation

For each training example, independently sample $t \sim \mathrm{Uniform}\{1, \dots, T\}$, then sample G_t using Equation (3). Let

$$\widehat{d}_t = \begin{cases} -\log p_{\theta,1}(G_0 \mid G_1, C, \ell), & t = 1, \\ \displaystyle\sum_{u \in \mathcal{U}(\mathcal{S})} D_{\mathrm{KL}}(q_{t,u}(\cdot \mid x_{t,u}, x_{0,u}, \mathcal{S}) \,\|\, p_{\theta,t,u}(\cdot \mid G_t, C, \ell)), & t > 1. \end{cases} \tag{C1}$$

Then $\widehat{\mathcal{L}}_{\mathrm{VLB}} = \mathcal{L}_{\mathrm{len}} + \mathcal{L}_T^{\mathrm{prior}} + T\widehat{d}_t$ is an unbiased Monte Carlo estimator of Equation (8). The evidence lower bound in Equation (8) follows from the standard variational decomposition of the generative and forward chains; the node-edge network determines the reverse distributions within it.

C.2 Auxiliary Supervision

$\mathcal{L}_{x_0}$ reconstructs the true clean categories from the logits at sampled timestep t. It uses the same conditional support and true parent variables as final decoding, but normalizes by the number of valid variables within each field before summing:

$$\mathcal{L}_{x_0} = \mathbb{E}_{t,G_t}\left[\sum_{v \in \{a,m,p,s,f\}} \frac{1}{|\mathcal{U}_v|} \sum_{u \in \mathcal{U}_v} -\log p_{\theta,u}^{\mathrm{clean}}(x_{0,u} \mid G_t, C, \ell, \mathrm{pa}(G_0))\right]. \tag{C2}$$

This differs from the reconstruction NLL/KL accumulated over the graph in the ELBO. $\mathcal{L}_{\mathrm{len}}$ is already included in the ELBO.

The conditional auxiliary term is $\mathcal{L}_{\mathrm{cond}} = \mathcal{L}_{\mathrm{group}} + \gamma\mathcal{L}_{\mathrm{pair}}$. Within a batch, examples are grouped by true action type and condition type, and the mean binary cross-entropy of each nonempty group is averaged with equal group weights. The observed proportions of condition presence and absence are retained within each group. For groups with at least two positive and two negative examples, let z denote the condition logit difference, and additionally compute

$$\mathcal{L}_{\mathrm{pair}} = \frac{1}{|\mathcal{G}_{\pm}|} \sum_{g \in \mathcal{G}_{\pm}} \mathrm{softplus}\big(0.5 - (\bar{z}_g^{+} - \bar{z}_g^{-})\big). \tag{C3}$$

This term is zero when no group qualifies. True actions are used only to group the training loss and are not inputs to generation. $\mathcal{L}_{\text{align}}$ supervises correspondences between reference and query materials that have compatible roles and identical identities; it is zero when no applicable correspondence exists.

We use $\kappa = 256$, $\lambda_0 = 1$, $\lambda_c = 1$, $\gamma = 0.1$, and $\lambda_a = 0.05$. Accordingly, the two conditional-loss coefficients in the code are 1 and 0.1.

D Additional Experimental Results

D.1 Hyperparameter settings

Table 11: Hyperparameter settings

Hyperparameter	ReactPG
Node hidden dimension	512
Edge hidden dimension	64
Joint node-edge blocks	8
Attention heads	8
Optimizer	AdamW
Training epochs	50
Training batch size	64
Gradient accumulation steps	1
Effective batch size	64
Evaluation batch size	32
Peak learning rate	2×10^{-4}
Minimum learning rate	5×10^{-5}
Learning-rate schedule	Cosine
Warmup	100 updates
Weight decay	0.01
Gradient clipping norm	1.0
Mixed precision	BF16

D.2 Additional ablation experiments

Table 12: Ablation of the noise schedule and categorical prior. All variants use the same test set. Results are reported in percentages.

Variant	Skeleton exact ↑	Edit similarity ↑	Binding F1 ↑	Flow F1 ↑	Graph exact ↑
Full ReactPG	**16.65**	**74.78**	**69.97**	**61.75**	**6.77**
Shared cosine schedule	15.96	74.45	69.52	60.30	6.37
Uniform categorical prior	16.22	73.59	68.82	59.85	5.78

Table 12 evaluates the hierarchical noise schedule and empirical categorical prior. Replacing the hierarchical schedule with a shared cosine schedule reduces graph exact match from 6.77 to 6.37. Replacing the empirical prior with a uniform distribution over the same legal support reduces graph exact match to 5.78. The uniform prior is used consistently in the forward transitions, terminal distribution, and reverse posterior.

D.3 Statistical Analysis

ReactPG significantly outperforms MolT5-large across all evaluated text-generation metrics, with all $p < 0.001$. Across three training seeds, ReactPG also exhibits consistently low variability over graph, natural-language, and structured-text metrics, indicating strong training stability and reproducibility.

Table 13: Performance significance tests against MolT5-large and training-seed stability of ReactPG-P on the 10000-reaction test set. (a) reports paired significance tests for text-generation metrics; (b) reports the mean ± SD of ReactPG-P across training seeds.

(a) Statistical significance against MolT5-large

	BLEU-2	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L
t-statistic	52.215	52.902	55.150	56.360	44.765
p-value	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001

(b) ReactPG-P training-seed stability

Metric	Edit similarity	Binding F1	Condition F1	State F1	Flow F1	Graph exact
Mean ± SD	74.78 ± 0.02	69.97 ± 0.44	55.41 ± 0.68	77.96 ± 0.10	61.75 ± 0.06	6.77 ± 0.28
Metric	BLEU-2	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L	Validity
Mean ± SD	66.82 ± 0.41	53.89 ± 0.40	76.06 ± 0.12	61.43 ± 0.18	70.39 ± 0.17	100.00 ± 0.00
Metric	LEV50	LEV75	LEV90	SCVR ↓		
Mean ± SD	97.77 ± 0.06	39.64 ± 0.97	13.05 ± 0.22	0.03 ± 0.01		

D.4 Detailed Analysis

Tables 14–19 compare ReactPG with ReactGPT, MolT5, Thoth, and the QFANG adaptation on matched validation records. Blue boldface denotes actions; dark-yellow boldface marks discrepancies or unsupported actions.

In Tables 14 and 15, ReactPG avoids conflicting masses assigned to one product (18 g versus 38 mg) and conflicting amounts assigned to one reactant (0.54 versus 1.09 mmol), despite remaining numerical errors.

In Table 16, ReactPG matches all nine annotated actions, material bindings, and numerical parameters up to equivalent notation. ReactGPT alters quantities and temperature and adds unsupported dilution conditions.

In Table 17, ReactPG retains the source-supported sequence of concentration, toluene addition, and a second concentration, whereas Thoth omits the latter two operations.

Table 18 shows that ReactPG preserves the annotated 13-action sequence, including the final vacuum drying, while avoiding the repeated sodium-carbonate washes generated by MolT5.

Table 19 highlights correct staging: ReactPG places sodium bicarbonate addition after the 18-hour stir, avoiding the premature addition and unsupported work-up steps generated by the QFANG adaptation.

Table 14: Conflicting product masses (record 256639). Left: ReactGPT; right: ReactPG.

Field	Value
Reactants	1: `C[C@@H]1CCCC[C@@H]1Nc1c(C#N)cnn2cccc12` 3: `[NH4+].[OH-]` 4: `OO`
Solvents	2: `CCO`
Product	-1: `C[C@@H]1CCCC[C@@H]1Nc1c(C(N)=O)cnn2cccc12`
Source	To a solution of 4-((1S,2R)-2-methylcyclohexylamino)pyrrolo[1,2-b]pyridazine-3-carbonitrile (18h) (105 mg, 0.41 mmol) in EtOH (10 mL) was added conc. NH4OH (4 mL), followed by dropwise addition of H2O2 (0.18 mL, 1.76 mmol). The reaction mixture was stirred at room temperature for 19 h and concentrated in vacuum to dryness. [...] to furnish 4-((1S,2R)-2-methylcyclohexylamino)pyrrolo[1,2-b]pyridazine-3-carboxamide (18i) (50 mg, 45%) as a light blue solid.

Language-model prediction	**ReactPG prediction**
MAKESOLUTION: 1 (**83 mg, 0.33 mmol**); 2 (**8 mL**). **ADD**: 3 (**3 mL**). **ADD**: 4 (**0.14 mL, 1.37 mmol**). **STIR**: **13 h**; room temperature. **CONCENTRATE**. **YIELD**: -1 (**18 g, 38 mg**; **42%**).	**MAKESOLUTION**: 1 (105 mg, 0.41 mmol); 2 (10 mL). **ADD**: 3 (**6 mL**). **ADD**: 4 (**0.27 mL, 2.64 mmol**). **STIR**: 19 h; room temperature. **CONCENTRATE**. **YIELD**: -1 (50 mg; 45%).

Table 15: Conflicting amounts of the same reactant (record 247834). Left: ReactGPT; right: ReactPG.

Field	Value
Reactants	3: `CC(C)N` 1: `COc1ccc2c(=O)[nH]c(-c3ccc(OCCOS(C)(=O)=O)c(-c4ccc(S(C)=O)cc4)n3)nc2c1` 4: `[I-].[Na+]`
Solvents	2: `CS(C)=O`
Product	-1: `COc1ccc2c(=O)[nH]c(-c3ccc(OCCNC(C)C)c(-c4ccc(S(C)=O)cc4)n3)nc2c1`
Source	To a solution of 2-((6-(7-methoxy-4-oxo-3,4-dihydroquinazolin-2-yl)-2-(4-(methylsulfinyl)phenyl)pyridin-3-yl)oxy)ethyl methanesulfonate (0.58 g, 1.09 mmol) in DMSO (4 mL) was added isopropyl amine (3.22 g, 54.5 mmol) and sodium iodide (0.16 g, 1.09 mmol). The reaction was heated at 60° C. in a sealed tube for 4 h. After that time the reaction was cooled to rt, concentrated under reduced pressure and diluted with dichloromethane (100 mL). The mixture was washed with saturated NaHCO3 (10 mL), water (10 mL), dried over Na2SO4 and concentrated under reduced pressure. [...] to give 2-(5-(2-(Isopropylamino)ethoxy)-6-(4-(methylsulfinyl)phenyl)pyridin-2-yl)-7-methoxyquinazolin-4(3H)-one (0.416 g, 77%) as a white foam.

Language-model prediction	**ReactPG prediction**
MAKESOLUTION: 1 (**0.54 mmol, 1.09 mmol**); 2 (4 mL). **ADD**: 3 (3.22 g, 54.5 mmol). **ADD**: 4 (0.16 g, 1.09 mmol). **ADD**: 3 (3.22 g, 54.5 mmol). **ADD**: 4 (0.16 g, 1.09 mmol). **STIR**: 4 h; 60 °C. **WAIT**: 4 h; 60 °C. **ADD**: dichloromethane (100 mL). **WASH**: NaHCO3 (10 mL). **WASH**: water (10 mL). **DRYSOLUTION**: Na2SO4. **FILTER**: keep filtrate. **CONCENTRATE**. **YIELD**: -1 (**0.38 g**; **76%**).	**MAKESOLUTION**: 1 (**1 g**); 2 (4 mL). **ADD**: 3 (3.22 g, 54.5 mmol). **ADD**: 4 (160 mg, 1.09 mmol). **STIR**: 4 h; 60 °C. **ADD**: dichloromethane (100 mL). **WASH**: NaHCO3 (10 mL). **WASH**: water (10 mL). **DRYSOLUTION**: sodium sulfate. **FILTER**: keep filtrate. **CONCENTRATE**. **YIELD**: -1 (416 mg; **83%**).

Table 16: Complete agreement with a nine-action reference (record 239013). Left: ReactGPT; right: ReactPG.

Field	Value
Reactants	1: `CN1C(=O)Cc2cc(N)cc(F)c21` 2: `CC(C)(C)OC(=O)NC[C@H]1CO1` 3: `O=S(=O)([O-])C(F)(F)F.[Li+]`
Solvents	4: `CC#N` 5: `CCOC(C)=O`
Product	-1: `CN1C(=O)Cc2cc(NC[C@@H](O)CNC(=O)OC(C)(C)C)cc(F)c21`
Source	5-Amino-7-fluoro-1-methyl-1,3-dihydro-indol-2-one (Preparation 2, 1.72 g, 9.54 mmol), (S)-oxiranylmethyl-carbamic acid tert-butyl ester (1.66 g, 9.54 mmol) and lithium trifluoromethane-sulfonate (1.46 g, 9.54 mmol) in acetonitrile (15 ml) are heated at 90° C. for 2 hours. The reaction mixture is diluted with ethyl acetate, washed with water and brine, dried (Na2SO4) and evaporated. Final purification by flash chromatography (70% Ethyl acetate/hexane) gives the title compound as a yellow foamy solid. HPLC R.T. 3.75 min; MS for C17H24FN3O4 m/z 354 (M+H)+.

Language-model prediction	ReactPG prediction
MAKESOLUTION: 1 (**1.00 g, 5.60 mmol**); 2 (**0.92 g, 5.60 mmol**); 3 (**0.83 g, 5.60 mmol**); 4 (**10 mL**). **STIR**: 2 h; **70 °C**. **ADD**: 5 **2 h; 70 °C**. **WASH**: water. **WASH**: brine. **DRYSOLUTION**: Na2SO4. **FILTER**: keep filtrate. **CONCENTRATE**. **YIELD**: -1.	**MAKESOLUTION**: 1 (1.72 g, 9.54 mmol); 2 (1.66 g, 9.54 mmol); 3 (1.46 g, 9.54 mmol); 4 (15 mL). **STIR**: 2 h; 90 °C. **ADD**: 5. **WASH**: water. **WASH**: brine. **DRYSOLUTION**: sodium sulfate. **FILTER**: keep filtrate. **CONCENTRATE**. **YIELD**: -1.

Table 17: Preserving an intervening solvent addition between two concentrations (record 227881). Left: Thoth adaptation; right: ReactPG.

Field	Value
Reactants	1: `CC(C)[C@@H](C(=O)O)N(C)C(=O)OC(C)(C)C` 3: `O=C([O-])[O-].[Cs+].[Cs+]`
Solvents	2: `CO` 4: `O`
Product	-1: `CC(C)[C@@H](C(=O)[O-])N(C)C(=O)OC(C)(C)C.[Cs+]`
Source	To a stirred solution of N-(tert-butoxycarbonyl)-N-methyl-L-valine (1.0 g) in methanol (18.0 mL) was added a solution of caesium carbonate in water until pH 7 was reached (approx. 704 mg caesium carbonate in 3.5 mL water) and the solution was stirred for 30 minutes. The solvent was removed in vacuum, toluene was added and the solvent was again removed in vacuum to give 1.55 g of the title compound.

Language-model prediction	ReactPG prediction
MAKESOLUTION: 1 (1.0 g, **4.4 mmol**); 2 (**10 mL**). **ADD**: 3 (**2.2 g, 6.6 mmol**). **ADD**: 4 (**1 mL**). **STIR**: **16 h; room temperature**. **CONCENTRATE**. **YIELD**: -1 (**1.2 g; 90%**).	**MAKESOLUTION**: 1 (**4 g**); 2 (**110 mL**). **MAKESOLUTION**: 3; 4. **STIR**: 30 min. **CONCENTRATE**. **ADD**: toluene. **CONCENTRATE**. **YIELD**: -1 (**10 g**).

Table 18: Preserving a multistage work-up without repetitive washing (record 206082). Left: MolT5; right: ReactPG.

Field	Value
Reactants	1: `CN(c1cc(-c2ccccc2)nc(Cl)n1)C(C)(C)C` 3: `COC(=O)C1(c2ccc(N)cc2)CCC1`
Solvents	2: `CCCCO` 4: `ClCCl`
Product	-1: `CNc1cc(-c2ccccc2)nc(Nc2ccc(C3(C(=O)OC)CCC3)cc2)n1`
Source	tert-Butyl-(2-chloro-6-phenyl-pyrimidin-4yl)-methylamine (1.8 g, 6.54 mmol) was dissolved in n-butanol (10 mL) followed by the addition of 1-(4-aminophenyl)-cyclobutanecarboxylic acid methyl ester (1.598 g, 7.84 mmol). The resulting mixture was stirred at reflux for about 10-19 h under nitrogen atmosphere. The mixture was diluted with dichloromethane, and was washed two times with water and one time with brine. The organic phase was dried over sodium sulfate, filtered, concentrated by rotary evaporation, and dried under vacuum. [...] gave the title compound as a pale yellow solid (800 mg, 31% yield).

Language-model prediction	**ReactPG prediction**
MAKESOLUTION: 1 (**5.0 mL, 75 mmol**); 2.	**ADD**: 1 (1.8 g, 6.54 mmol).
STIR: **6 h; 130 °C**.	**ADD**: 2 (**25 mL**).
WAIT: **6 h; 130 °C**.	**ADD**: 3 (**1.033 g, 5 mmol**).
STIR: **6 h; 130 °C**.	**REFLUX**: 10–19 h; nitrogen.
ADD: **3 (20 mL); room temperature**.	**ADD**: 4.
WASH: **sodium carbonate (20 mL)**.	**WASH**: water.
WASH: water (**20 mL**).	**WASH**: brine.
WASH: **sodium carbonate (20 mL)**.	**COLLECTLAYER**: **mixture**.
WASH: **sodium carbonate**.	**DRYSOLUTION**: sodium sulfate.
DRYSOLUTION: sodium sulfate.	**FILTER**: keep filtrate.
FILTER: keep filtrate.	**CONCENTRATE**.
CONCENTRATE.	**DRYSOLID**: under vacuum.
YIELD: -1 (**0.92 g; 89%**).	**YIELD**: -1 (800 mg; 31%).

Table 19: Preserving the timing of a work-up reagent (record 271308). Left: QFANG adaptation; right: ReactPG.

Field	Value
Reactants	3: `CC[SiH](CC)CC` 2: `COC(=O)c1cccc(C(O)c2c(-c3ccc(C)cc3)nn3cc(OC)ccc23)n1` 4: `O=C(O)C(F)(F)F` 5: `O=C([O-])O.[Na+]`
Solvents	1: `ClCCl`
Product	-1: `COC(=O)c1cccc(Cc2c(-c3ccc(C)cc3)nn3cc(OC)ccc23)n1`
Source	A dichloromethane (1.7 mL) solution of methyl 6-[hydroxy[6-methoxy-2-(4-methylphenyl)pyrazolo[1,5-a]pyridin-3-yl]methyl]pyridine-2-carboxylate (136 mg) was ice-cooled and triethylsilane (0.323 mL) and trifluoroacetic acid (0.300 mL) were added. The mixture was heated up to room temperature, stirred for 18 hours, and then ice-cooled again. Next, a saturated sodium bicarbonate aqueous solution was slowly added to achieve pH=9. The organic solvent was evaporated under vacuum [...] to obtain a title compound as a pale yellow oil (116 mg).

Language-model prediction

ADD: 1 (**1.2 mL**).
ADD: 2 (**102 mg**).
ADD: 3 (**0.233 mL**).
ADD: 4 (**0.216 mL**).
SETTEMPERATURE: room temperature.
ADD: **5**.
STIR: 18 h.
ADD: 5.
EXTRACT: **ethyl acetate**.
COLLECTLAYER: **organic**.
WASH: **sodium chloride**.
DRYSOLUTION: **magnesium sulfate**.
FILTER: **keep filtrate**.
CONCENTRATE.
YIELD: -1 (**78.5 mg**).

ReactPG prediction

ADD: 1 (**1.2 mL**).
ADD: 2 (**202 mg**).
STIR.
ADD: 3 (**0.233 mL**).
ADD: 4 (**0.216 mL**).
SETTEMPERATURE: room temperature.
STIR: 18 h.
ADD: 5.
CONCENTRATE.
YIELD: -1 (**156 mg**).